Dragan Pleskonjić
Borislav Đorđević
Nemanja Maček
Marko Carić

Sigurnost računarskih mreža

priručnik za laboratorijske vežbe

Autori: mr Dragan Pleskonjić, Nemanja Maček
dr Borislav Đorđević, Marko Carić

Recenzenti: prof. dr Borivoj Lazić, mr Verica Vasiljević

Izdavač: Viša elektrotehnička škola u Beogradu

Za izdavača: dr Dragoljub Martinović

Lektor: Milena Dorić

Tehnička obrada: Dragan Pleskonjić, Nemanja Maček,
Borislav Đorđević, Marko Carić

Dizajn: Katarina Carić

Štampa: Akademska izdanja
štampano u 200 primeraka

CIP – Каталогизација у публикацији
Народна библиотека Србије, Београд

004.7.056.5(075.8)(076)

SIGURNOST računarskih mreža : priručnik za laboratorijske vežbe / Dragan Pleskonjić ... [et al.]. - Beograd : Viša elektrotehnička škola, 2006 (Beograd : Akademska izdanja). - 432 str. : ilustr. ; 25 cm

Tiraž 200. - Rečnik termina: str. 423-430. - Bibliografija: str. 431-432.

ISBN 86-85081-49-1
1. Плескоњић, Драган
a) Рачунарске мреже – Заштита – Вежбе
COBISS.SR-ID 129437196

Predgovor

Računarske mreže su važan deo infrastrukture mnogih proizvodnih, upravnih, finansijskih i vojnih sistema. Ugrožavanje njihove sigurnosti, tj. neovlašćeni uvid, izmena ili oštećenje podataka, programa, servera, radnih stanica, prenosnih puteva ili drugih resursa je opasnost sa kojom se susreću gotovi svi koji su na bilo kakav način povezani sa razvojem, uvođenjem, upotrebom ili održavanjem ovakvih sistema. Problem sigurnosti postaje sve izraženiji i dodatno se podstiče zbog rasprostranjenosti Internet mreže i bežičnih mobilnih komunikacija.

Nastava iz predmeta Sigurnost računarskih mreža zasnovana je na savremenim nastavnim planovima i programima poznatih svetskih univerziteta. Ovaj priručnik za laboratorijske vežbe sadrži osnovne teorijske koncepte i praktične implementacije značajnijih sigurnosnih mehanizama i usklađen je sa nastavnim planom i programom predmeta.

U pripremi ove knjige učinjeni su svi napori da se izbegnu greške i omaške. Iako su autori proverili sve opisane postupke, algoritme i metode, moguće je da u knjizi postoje izvesni previdi, nepreciznosti i nepravilnosti. Izdavač i autori ne prihvataju bilo kakvu odgovornost za eventualne greške i omaške, kao ni za posledice do kojih može doći primenom saznanja iz ove knjige koja se kasnije mogu pokazati kao netačna.

Zaštitni softver, postupci i algoritmi ne mogu se jednostavno klasifikovati na zlonamerne i dobronamerne. Na primer, ukoliko korisitite napad rečnikom da proverite sigurnost svoje lozinke, postupak je dobronameran; ukoliko lozinku Vašeg prijatelja napadnete metodom rečnika, bez njegovog znanja, s namerom da neovlašćeno pristupite podacima na disku Vašeg prijatelja, postupak je zlonameran. Trudili smo se da priručnik koncipiramo tako da čitaoci u svoj zaštitni “arsenal” uvrste samo one postupke koje ne mogu da zloupotrebe. U svakom slučaju, autori ne odgovaraju za bilo kakvu zloupotrebu algoritama i postupaka opisanih u ovoj knjizi.

Za izradu ovog priručnika korišćeni su OpenOffice.org, GIMP i Inkscape programski paketi na SuSE Linux 10 operativnom sistemu.

Zahvaljujemo se svima koji su nam na bilo kakav način pomogli prilikom izrade priručnika.

Autori

Sadržaj

1. Matematičke osnove kriptografije i upoznavanje programskog jezika Java ... 13

1.1 Kombinatorika, verovatnoća i matematičko očekivanje ... 14
Pojam verovatnoće ... 15
Uslovna verovatnoća ... 16
Slučajna promenljiva i matematičko očekivanje ... 17
Pojam neodređenosti, entropije i informacije ... 18
1.2 Više o brojevima i deljivosti ... 19
NZD, NZS, Euklidov algoritam ... 20
Kongruencija ... 21
1.3 Zatvorenost skupova, grupe, podgrupe i polja ... 23
Pojam grupe ... 23
Mala Fermaova teorema ... 24
Ciklična grupa ... 25
Pojam podgrupe ... 26
Prsten i polje ... 26
Polinomi u konačnim poljima ... 27
Operacije nad polinomima u konačnim poljima ... 28
1.4 Uvod u programski jezik Java ... 30
Implementacija AES algoritma u Javi ... 30
Implementacija SHA-1 algoritma u Javi ... 44

2. Osnovni kriptografski pojmovi i klasična kriptografija ... 47

2.1 Osnovni kriptografski pojmovi ... 48
Napadi na šifrate ... 51
2.2 Klasična kriptografija i kriptoanaliza ... 52
Cezarova šifra ... 52
Afina šifra ... 54
Vigenèreova šifra ... 56
Jednostavan XOR algoritam ... 59
Jednokratna beležnica ... 60
Playfairova šifra ... 61
Hillova šifra ... 62
Šifrovanje transpozicijom ... 63
2.3 Pitanja i zadaci ... 64

3. Simetrični blokovski algoritmi 67

3.1 DES 68
Šifrovanje i dešifrovanje 69
Ključevi koji se ne koriste 71
Karakteristike i kriterijumi dizajna 72
Sigurnost DES-a 74
Režimi rada 76
3.2 AES/RIJNDAEL 79
Matematička osnova 81
Šifrovanje 83
Proširenje ključa 86
Dešifrovanje 86
Sigurnost AES algoritma 87
3.3 IDEA 88
Sigurnost IDEA algoritma 91
3.4 Pitanja i zadaci 91

4. Generatori pseudoslučajnih sekvenci i protočno šifrovanje 93

4.1 Pseudoslučajne sekvence i testovi slučajnosti 94
Testovi slučajnosti 97
4.2 Protočno šifrovanje 99
Linearni pomerački registar sa povratnom spregom 103
RC4 104
4.3 Pitanja i zadaci 105

5. Jednosmerne heš funkcije 107

5.1 Šta je heš, a šta jednosmerna funkcija ? 108
Više o jednosmernim heš funkcijama 110
Dužina heša 111
5.2 Značajnije heš funkcije 112
MD2 112
MD5 113
SHA 118
5.3 Primena heš funkcija 120
Heš funkcije i čuvanje lozinki na disku sistema 121
Heš funkcije i CHAP autentifikacija 123
Heš funkcije i digitalno potpisivanje 123
5.3 Pitanja i zadaci 124

6. Kriptografija sa javnim ključevima ... 125

6.1 Uvod u kriptografiju sa javnim ključevima ... 126
Diffie-Hellmanov protokol za razmenu ključeva ... 126
Kriptosistemi sa javnim ključem ... 127
6.2 RSA ... 129
Matematička podloga RSA algoritma ... 130
RSA problem ... 131
Generisanje ključeva, šifrovanje i dešifrovanje ... 133
Digitalno potpisivanje poruka ... 134
Kriptoanaliza RSA ... 135
6.3 El-Gamal ... 138
Generisanje ključeva, šifrovanje i dešifrovanje ... 138
Poređenje RSA i El-Gamal kriptosistema ... 139
6.4 Sertifikati ... 140
6.5 Infrastruktura javnih ključeva ... 141
Funkcije PKI ... 143
Primer upotrebe PKI u hibridnom kriptosistemu ... 146
Delegiranje odgovornosti ... 147
Sertifikati i LDAP direktorijum ... 147
6.6 X.509 sertifikati ... 148
X.500 ... 149
X.509 sertifikati – verzije 2 i 3 ... 150
Registracija objekata ... 152
6.7 Pitanja i zadaci ... 153

7. Kriptografski softver ... 155

7.1 Šifrovanje diskova i komunikacionih kanala ... 156
Šifrovanje komunikacionih kanala ... 156
Šifrovanje podataka na diskovima ... 157
7.2 Pretty Good Privacy ... 159
Šta je PGP i koliko je siguran? ... 161
PGP Desktop 8.x/9.x ... 163
OpenPGP ... 167
7.3 GNU Privacy Guard ... 168
Rad sa GnuPG paketom ... 169
GnuPG Explorer Extension ... 171
Win Privacy Tray (WinPT) ... 173
GPG Shell ... 173

KGpg..... 173
Enigmail..... 175
KMail..... 177
7.4 TrueCrypt..... 178
Ispitivanje performansi algoritama..... 179
Kreiranje standardnog volumena..... 179
Aktiviranje i deaktiviranje volumena..... 181
Ključ-datoteke..... 182
Skriveni volumeni..... 183
7.5 EFS..... 185
Kako se šifruje?..... 186
Oporavak podataka – klopka ili "za nedaj Bože" ?..... 186
7.6 Pitanja i zadaci..... 187
8. Kontrola pristupa i zaštita operativnog sistema Windows 2000/XP/2003 Server..... 189
8.1 Opšte o Windows 2000/XP/2003 sigurnosti..... 190
Koliko je sve to zaista sigurno?..... 191
8.2 Korisnički nalozi, grupe i prava korisnika..... 194
Korisnički nalozi (Windows XP)..... 195
Prava dodeljena korisnicima..... 196
Domenski korisnički nalozi i grupe..... 197
Grupne polise i dodela prava korisnicima domena..... 198
8.3 NTFS objekti i deljeni mrežni resursi..... 199
Deljeni mrežni resursi..... 203
8.4 Praćenje događaja i pristupa resursima..... 204
Praćenje pristupa resursima..... 206
8.5 Delegacija ovlašćenja u aktivnom direktorijumu..... 207
8.6 Pitanja i zadaci..... 210
9. Kontrola pristupa i zaštita operativnog sistema Linux..... 211
9.1 Opšte o sigurnosti Linux sistema..... 212
Počnite sa osnovnim merama zaštite..... 213
Prevođenje monolitnog jezgra sa odgovarajućim parametrima..... 214
Čišćenje sistema od nepotrebnog softvera..... 216
Sigurnost skriptova u /etc/init.d direktorijumu..... 216
9.2 Korisnički nalozi i lozinke..... 216
Lozinke korisnika i /etc/shadow..... 217
Root korisnički nalog..... 218

Sistemski korisnički nalozi ... 220
sudo ... 221
9.3 Sistemi datoteka i kontrola pristupa ... 222
Podrazumevani vlasnički odnosi i prava pristupa ... 224
Promena vlasničkih odnosa i prava pristupa ... 224
Datoteke bez vlasnika ... 226
SUID i SGID bitovi ... 227
ext2/ext3 – specijalni atributi i fleg nepromenljivosti ... 229
Datoteka /etc/fstab ... 229
9.4 Linux na mreži ... 231
Datoteke /etc/hosts.allow i /etc/hosts.deny ... 231
xinetd – praćenje aktivnosti i kontrola pristupa ... 231
Datoteka /etc/exports ... 235
Datoteke .rhosts ... 236
Zaštita Apache Web servera ... 237
“chroot jail” ... 239
9.5 Praćenje događaja i nadzor sistema ... 241
Praćenje događaja – syslog alat ... 242
Zaštita syslog servera ... 244
Nadzor sistema – pregledanje log datoteka ... 245
9.6 Pitanja i zadaci ... 247

10. Mrežne barijere ... 249

10.1 Osnovni pojmovi o računarskim mrežama ... 250
TCP/IP skup protokola ... 251
IP adresiranje i podmrežavanje ... 252
Rutiranje ... 255
10.2 Šta je mrežna barijera i koje su njene funkcije? ... 255
Filtriranje paketa ... 256
Mrežne barijere sa i bez uspostave stanja ... 257
Prevođenje mrežnih adresa ... 259
Mrežne barijere na aplikacionom sloju OSI modela ... 260
Demilitarizovane zone ... 260
10.3 iptables ... 261
Filtriranje paketa ... 262
Kreiranje novih lanaca ... 266
Prošireni skup pravila ... 267
Zaštita od čestih napada ... 270
NAT ... 271
Maskiranje ... 274

Mangle tabela..... 275
Put paketa kroz Netfilter.....275
Snimanje tekuće konfiguracije mrežne barijere.....276
Praćenje događaja.....277
10.4 Skeniranje portova - provera firewall konfiguracije.....279
nmap.....280
10.5 Squid proksi server.....281
Proxy-caching režim rada i kontrola pristupa.....282
10.6 Kućna rešenja – mrežne barijere za Windows XP.....284
Windows firewall..... 285
(Sunbelt) Kerio Personal Firewall.....286
Zone Alarm (Pro)..... 289
10.7 Pitanja i zadaci.....291

11. Kontrola pristupa na CISCO uređajima..... 293

11.1 Osnovna konfiguracija Cisco rutera.....294
Šta je Cisco IOS ?.....295
Postavljanje lozinki..... 296
Dodela IP adrese mrežnom interfejsu.....297
Konfigurisanje protokola za rutiranje..... 298
11.2 Liste za kontrolu pristupa (za IP protokol)..... 299
Standardne ACL..... 300
Proširene ACL.....302
Imenovane ACL..... 303
11.3 Pitanja i zadaci.....304

12. Sigurnost bežičnih mreža..... 305

12.1 Vrste bežičnih mreža..... 306
12.2 Šta trenutno “štiti” bežične mreže ?..... 307
Sigurnost na fizičkom nivou – ograničavanje propagacije signala....308
SSID..... 308
Autentifikacija korisnika mreže.....309
WEP.....309
Upravljanje ključevima.....311
12.3 Sigurnosni propusti i napadi na WEP.....311
Pasivni napadi na WEP.....312
Aktivni napadi na WEP.....313
12.4 Nadogradnje standarda 802.11 317
802.1x..... 317

EAP....317
Vrste EAP-a....319
Sigurnosni propusti i napadi na 802.1x....320
WEP2....321
12.5 Novi (i postojeći) standardi....321
802.11i....323
12.6 Pitanja i zadaci....324
13. Sistemi za detekciju i sprečavanje napada na sistem....325
13.1 Osnovni pojmovi i podele prema različitim kriterijumima....326
Kriterijum podele: šta se detektuje ?....328
Detekcija zloupotrebe....329
Detekcija anomalija....329
Kriterijum podele: gde je sistem smešten, tj. kako radi ?....330
Kriterijum podele: kada je napad otkriven ?....331
Kriterijum podele: reakcija na napad....332
Faze odgovora na napad i sistemi za sprečavanje upada....332
13.2 Postojeći sistemi za detekciju upada....333
13.3 IDS softver za Windows operativne sisteme....335
Fortego All-Seeing Eye....336
Arovax Shield....339
TeaTimer....340
13.4 Pitanja i zadaci....342
14. Antivirusna zaštita i zaštita od špijunskih programa....343
14.1 Šta je zlonamerni softver?....344
14.2 Crvi....345
E-mail crvi - MyDoom....346
IM crvi – CoolNow i Funner....350
Internet crvi – Sasser.b....351
File-sharing i P2P crvi – Benjamin....355
14.3 Trojanski konji....356
14.4 Virusi....359
Virusi u sistemu datoteka....359
Makro virusi....361
Skript virusi....364
14.5 Špijunski programi....364
Top lista špijunskih programa....369
14.6 Programi za zaštitu od virusa i špijunskih programa....373

14.7 Pitanja i zadaci....380

15. Programerske tehnike zaštite.... 381

15.1 Šta je prekoračenje bafera?.... 382
Kako dolazi do prekoračenja?....383
15.2 Prekoračenje na steku.... 386
Šta najčešće izaziva prekoračenje bafera na steku ?....387
Primer na Windowsu....387
Primer na Linuxu....389
15.3 Prekoračenje na heap-u....391
Dinamička dodela memorije.... 391
Primer prekoračenja.... 393
15.4 Sprečavanje prekoračenja.... 394
15.5 Pitanja i zadaci....396

A. Značajnije kriptografske tablice.... 397

B. Izvorni kod.... 403

B.1 Cezarova šifra (Java)....404
B.2 Jednostavan XOR algoritam (C++)....404
B.3 AES.... 405
B.4 SHA1 (Java)....413

C. Lozinke na nivou BIOS-a.... 419

Lozinke za oporavak.... 420
Upotreba "Clear CMOS" kratkospojnika na matičnoj ploči....421
Uklanjanje CMOS baterije....422

Rečnik termina.... 423

Literatura.... 431

1

Matematičke osnove kriptografije i upoznavanje programskog jezika Java

1.1 Kombinatorika, verovatnoća i matematičko očekivanje

Faktorijel broja *n* definiše se na sledeći način:

- $0! = 1! = 1$
- $n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1, n>1$

Na primer:

- $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$
- $5! / 3! = 5 \cdot 4 = 20$
- $n! / (n-2)! = n \cdot (n-1)$

Neka je $S = \{a_1, a_2, \ldots, a_n\}$. Varijacija bez ponavljanja klase *k* ($0 \leq k \leq n$) od *n* elemenata V(n,k) je bilo koja uređena k-torka različitih elemenata skupa *S*. Takvih različitih k-torki ima $n!/(n-k)!$. Dakle,

- $V(n,k) = n! / (n-k)!$

Specijalno, varijacija klase *n* od *n* elemenata (dakle, slučaj kada je k=n) je permutacija P(n) tih elemenata. Takvih n-torki ukupno ima n!:

- $P(n) = V(n,n) = n!$

Varijacija sa ponavljanjem klase *k* ($0 \leq k \leq n$) od *n* elemenata $V_p(n,k)$ je bilo koja uređena k-torka elemenata skupa *S*. Elementi u svakoj k-torki mogu se ponavljati, što znači da takvih različitih k-torki ima ukupno n^k:

- $V_p(n,k) = n^k$

Kombinacija bez ponavljanja klase *k* ($0 \leq k \leq n$) od *n* elemenata C(n,k) je neuređena k-torka različitih elemenata skupa S. Takvih različitih k-torki ukupno ima $n! / ((n-k)! \cdot k!)$. Drugim rečima, nije dozvoljeno da se, kao u slučaju varijacija, ponove dve mogućnosti koje imaju iste elemente a različit raspored. Dakle,

- $C(n,k) = n! / ((n-k)! \cdot k!)$

Neka je dat skup S = {1, 2, 3}. Odredićemo sve kombinacije, varijacije bez ponavljanja i varijacije sa ponavljanjem druge klase skupa *S*. Varijacije bez ponavljanja druge klase čine skup {12, 13, 23, 21, 31, 32} i ima ih ukupno 3!/(3-2)!=6. Varijacije sa ponavljanjem druge klase čine skup {11, 12, 13, 21, 22, 23, 31, 32, 33} i ima ih ukupno 3^2=9. Kombinacije druge klase čine skup {12, 13, 23} i ima ih ukupno 3!/(2!·1!)=3.

Binomna formula služi za izračunavanje vrednosti binoma proizvoljnog stepena:

- $(a+b)^n = a^n + C(n,1)a^{n-1}b^1 + C(n,2)a^{n-2}b^2 + \ldots + C(n,k)a^{n-k}b^k + \ldots + b^n$
- $(a+b)^n = \sum_{i=0,\ldots n} \{ C(n,0)\, a^{n-i}\, b^i \}$

Može se zapisati i kao:

- $(a+b)^n = \sum_{i=0,\ldots n} \binom{n}{k} a^{n-k}\, b^k$

Pojam verovatnoće

Bacamo kocku čije su strane numerisane brojevima od 1 do 6. Pojavljivanje jednog od tih brojeva nazivamo ishodom ili elementarnim događajem *A*. Količnik povoljnih i svih mogućih ishoda za događaj *A* nazivamo verovatnoćom događaja *A*, u oznaci *P(A)*. Pošto je broj povoljnih događaja podskup skupa svih mogućih, verovatnoća se kreće između 0 i 1, tj. $0 \le P(A) \le 1$.

Siguran događaj je onaj događaj koji omogućuje da ako jednom bacamo novčić onda padne ili pismo ili glava. Verovatnoća sigurnog događaja je 1. Nemoguć događaj je onaj događaj pri kojem ako jednom bacamo novčić padne i pismo i glava. Verovatnoća nemogućeg događaja je 0.

Na primer, koja je verovatnoća da će nakon jednog bacanja kocke pasti broj manji od 5? U ovom slučaju S = {1, 2, 3, 4, 5, 6}, a događaj "kocka pokazuje broj manji od 5" je A = {1, 2, 3, 4} . Dakle, verovatnoća događaja *A*

je 4/6 = 2/3.

Unija događaja *A* i *B* ($A \cup B$) je skup svih ishoda koji pripadaju bar jednom od događaja *A* i *B*. Presek događaja A i B ($A \cap B$) je skup svih ishoda koji pripadaju i događaju *A* i događaju *B*. Komplement događaja *A* ($\bar{A}$) u odnosu na skup *S* je skup svih ishoda koji ne pripadaju *A*. Dakle, *A* se ostvaruje onda kada se $\bar{A}$ ne ostvaruje. *A* i $\bar{A}$ su suprotni događaji.

Na primer, ako je *A* događaj da je prilikom bacanja novčića palo pismo, $\bar{A}$ je događaj da nije palo pismo, tj. da je pala glava. Ako je *A* događaj da je na kocki pao broj veći od 3 i ako je *B* događaj da je broj paran, $A \cap B$ obuhvata ishode {4, 6} dok ($A \cup B$) obuhvata ishode {2, 4, 5, 6}.

Važe sledeće jednakosti

- $P(A) + P(\bar{A}) = 1$
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

Ako se *A* i *B* međusobno isključuju tj. $P(A \cap B) = 0$, onda je verovatnoća unije događaja $P(A \cup B) = P(A) + P(B)$.

Uslovna verovatnoća

Neka su *A* i *B* događaji nekog eksperimenta. Ako ostvarivanje jednog od njh ne utiče na ostvarivanje drugog, kažemo da su događaji nezavisni. U suprotnom, zavisni su. Verovatnoća događaja *B* pod uslovom da se ostvario događaj *A* označava se $P(B|A)$ i naziva se uslovna verovatnoća.

- $P(B|A) = P(A \cap B)/P(A)$

Na primer, neka je *A* događaj da je na kocki pao broj veći od 3 i neka je *B* događaj da je pao paran broj. $P(B|A)$ tj. verovarnoća da je pao paran broj pod uslovom da je pao broj veći od 3 je 2/3, jer je $P(A \cap B)=2/6$, $P(A)=3/6$, a $P(B|A) = P(A \cap B)/P(A) = 2/3$.

Ako su *A* i *B* nezavisni događaji tj. $P(B|A) = P(B)$, onda je verovatnoća preseka događaja $P(A \cap B) = P(A) \cdot P(B)$.

Neka je S skup elementarnih događaja nekog eksperimenta i neka je A_1, A_2, ..., A_n potpun sistem događaja tog eksperimenta kod koga se svaka dva događaja uzajamno isključuju. Ovo znači da važi $A_1 \cup A_2 \cup \ldots \cup A_n = S$.

Za događaj $B \in S$ važi $B=S \cap B$ tj. $B = (A_1 \cup A_2 \cup .. \cup A_n) \cap B$, što se može predstaviti kao $B = (A_1 \cap B) \cup (A_2 \cap B) \cup \ldots \cup (A_n \cap B)$. Kako se događaji A_1, A_2, ..., A_n po pretpostavci isključuju, i događaji $A_1 \cap B$, $A_2 \cap B$, ..., $A_n \cap B$ se isključuju. Stoga je formula potpune verovatnoće sledeća:

- $P(B) = P(A_1 \cap B) + P(A_2 \cap B) + \ldots + P(A_n \cap B)$
- $P(B) = P(A_1)P(B|A_1) + P(A_2)P(B|A_2) + \ldots + P(A_n)P(B|A_n) = \sum P(A_k)P(B|A_k)$

Iz jednakosti $P(A_k \cap B) = P(A_k)P(B|A_k) = P(B)P(A_k|B)$ sledi:

- $P(A_k|B) = P(A_k)P(B|A_k) / P(B)$,

pa dobijamo Bajesovu formulu:

- $P(A_k|B) = P(A_k)P(B|A_k) / \sum P(A_k)P(B|A_k)$

Rođendanski paradoks: koliko učenika treba da bude u odeljenju da bi bilo verovatnije da su bar dvoje rođeni istog datuma nego da nisu? Neka ja *P(A)* verovatnoća da su svi rođeni različitog datuma, a $P(\bar{A})=1-P(A)$ verovatnoća koju tražimo:

- $P(A) = V(365,k) / V_p(365,k)$
- $P(\bar{A}) = 1 - P(A)$

Tražimo vrednost *k* za koju je Za k=23, $P(\bar{A})=0.507>0.5$. Pomalo iznenađujuće, odgovor je 23 učenika!

Slučajna promenljiva i matematičko očekivanje

Slučajna promenjiva X je funkcija koja nekom događaju dodeljuje neki realan broj. Raspodela slučajne promenljive pokazuje verovatnoću sa kojom slučajna promenjiva uzima određenu vrednost.

Na primer, neka se novčić baca tri puta. Svaki od ishoda {GGG, GGP, GPG, PGG, GPP, PGP, PPG, PPP} ima verovatnoću 1/8. Neka je X broj palih pisama. Na primer: X(GGP) = X(GPG) = X(PGG) = 1, X(PPP) = 3. Raspodela slučajne promenjive X je data sa:

- 0 1 2 3
  1/8 3/8 3/8 1/8

Ako je data raspodela slučajne promenjive X sa:

- x_1 x_2 ... x_m
  p_1 p_2 ... p_m

pri čemu je $p_1 + p_2 + \ldots + p_m = 1$, onda je matematičko očekivanje slučajne promenjive X broj $E(X) = x_1 \cdot p_1 + x_2 \cdot p_2 + \ldots + x_m \cdot p_m$.

Pojam neodređenosti, entropije i informacije

Ako posmatramo jedan slučajan događaj *A*, čija je verovatnoća p = P(A), onda se njegova "neodređenost" može meriti nekom monotono opadajućom funkcijom od *p*; ukoliko verovatnoća odstupa od "sredine", utoliko je neodređenost manja. Za takvu funkciju možemo uzeti $\log_2$ (logaritam sa osnovom 2). Na taj način neodređenost događaja čija je verovatnoća 0.5 iznosi 1.

"Razbijanje" događaja *S* preko događaja A_1, A_2, ..., A_n označićemo sa *A*. Neodređenost nekog događaja A_i iz *A* je $\log_2 p_i$, ali verovatnoća pojavljivanja te verovatnoće je p_i. To znači da na $-\log_2 p_i$ možemo gledati kao na moguće vrednosti jedne slučajne promenjive. Zato za meru neodređenosti uzimamo srednju ili očekivanu vrednost $H(A) = \sum p_i \cdot \log_2 p_i$.

Na primer, opit koji ima dva jednakoverovatna ishoda A_1 i A_2 (verovatnoće događaja su $p_1 = P(A_1) = p_2 = P(A_2) = 0.5$) ima entropiju:

- $H(A) = -0.5 \cdot \log 0.5 - 0.5 \cdot \log 0.5 = 1$.

To je jedinična entropija ili jedan bit. Entropija opita sa raspodelom:

- 0 1 2
  1/4 1/2 1/4

iznosi $H(A) = -0.25 \cdot \log 0.25 - 0.5 \cdot \log 0.5 - 0.25 \cdot \log 0.25 = 0.452$

Uslovnu entropiju $H(A|B_i)$ opita *A* u odnosu na događaj B_i definišemo tako što umesto verovatnoća pi = P(Ai) uvodimo uslovne verovatnoće $P(A_i|B_i)$, i=1,2,..,n.

- $H(A|B_i) = -\sum P(A_i|B_i) \cdot \log P(A_i|B_i)$.

Neka je $B_1, B_2, \ldots, B_m$ razbijanje opita *B*. Uslovnu entropiju opita *A* u odnosu na *B* definišemo kao matematičko očekivanje:

- $H(A|B) = -\sum P(Bi)H(A|Bi)$

Informacija I(A|B) koju o opitu *A* nosi opit *B* definiše se kao

- I(A|B) = H(A) – H(A|B)

Zaista, ako su opiti *A* i *B* nezavisni, onda je informacija koju o opitu *A* nosi *B* jednaka nuli jer je tada H(A) = H(A|B). Ako je H(A|B)=0, tj. opit *B* "iscrpljuje" svu neodređenost opita *A*, tada je informacija I(A|B) maksimalna i jednaka entropiji samog opita *A*.

1.2 Više o brojevima i deljivosti

Apsolutna vrednost ili moduo realnog broja je njegova numerička vrednost, ne uzimajući u obzir znak tog broja. Broju većem od nule dodeljujemo taj broj, broju nula dodeljujemo nulu, broju manjem od nule dodeljujemo njemu suprotan broj. Broj čija se apsolutna vrednost određuje stavlja se između dve paralelene crte |x|. Na primer:

- |4| = |–4| = 4.
- $\sqrt{x^2} = |x|$

Za izvođenje odgovarajućih kriptografskih zaključaka neophodno je koristiti deljivost brojeva:

- *a* deli *n* (u oznaci a|n) ukoliko postoji celobrojno *b* takvo da je $n = a \cdot b$,
- ako a|b i b|c sledi a|c,
- ako a|b tada $(a \cdot c)|(b \cdot c)$ za svako *c,*
- ako c|a i c|b sledi $c|(d \cdot a + e \cdot b)$ za svako *d*,*e,*
- ako a|b i $b \neq 0$ sledi $|a| \leq |b|$,
- ako a|b i b|a sledi $|a| = |b|$.

Ceo broj $p>1$ je prost ukoliko ima tačno dva pozitivna delioca: 1 i *p*. Jednostavnije rečeno, broj je prost ako je deljiv samo jedinicom i samim sobom. Prosti brojevi su: 1, 2, 3, 5, 7, 11, 13, 17, 19, ...

Svaki ceo broj $a>1$ se može napisati kao proizvod prostih brojeva. Taj proizvod je (ako izuzmete permutaciju činioca) jedinstveno određen. Na primer, $60 = 2 \cdot 2 \cdot 3 \cdot 5 = 3 \cdot 2 \cdot 5 \cdot 2$.

NZD, NZS, Euklidov algoritam

Najveći zajednički delilac NZD(a,b) brojeva *a* i *b* ($a \neq 0$, $b \neq 0$) je najveći broj koji deli *a* i *b*. Na primer: NZD(4,6)=2, NZD(4,8)=4, NZD(1,x)=1, gde je *x* bilo koji broj. Brojevi *a* i *b* su uzajamno prosti ako je NZD(a,b) = 1.

Ostatak pri deljenju a sa b obeležićemo sa a % b. Efikasan metod za izračunavanje najmanjeg zajedničkog delioca nudi Euklidov algoritam:

- ako je b=0, onda je NZD(a,b) = |a|,
- ako je $b \neq 0$, onda je NZD(a,b) = NZD(|b|, a%|b|).

Na primer:

- NZD(100,35) = NZD(35, 100%35) = NZD(35,30) =
  = NZD(30, 35%30) = NZD(30,5) = NZD(5, 30%5) = NZD(5,0) = 5

Prošireni Euklidov algoritam je vrlo značajan algoritam koji za zadata dva nenegativna cela broja *a* i *b* (a≥b) daje kao izlaz d=NZD(a,b) i cele brojeve *x* i *y* takve da zadovoljavaju jednačinu: ax + by = d.

[1.] Ako je b=0 tada[1] $d \leftarrow a$, $x \leftarrow 1$, $y \leftarrow 0$. Vrati (d, x, y)

[2.] Dodeli sledeće vrednosti: $x_2=1$, $x_1=0$, $y_2=0$ i $y_1=1$

[3.] Dok je b>0 radi sledeće[2]:

[3.1] $q \leftarrow [a/b]$, $r \leftarrow (a - q \cdot b)$, $x \leftarrow (x_2 - q \cdot x_1)$, $y \leftarrow (y_2 - q \cdot y)$

[3.2] $a \leftarrow b$, $b \leftarrow r$, $x_2 \leftarrow x_1$, $x_1 \leftarrow x$, $y_2 \leftarrow y1$, $y_1 \leftarrow y$

[4.] $d \leftarrow a$, $x \leftarrow x_2$, $y \leftarrow y_2$. Vrati (d, x, y)

Broj svih celih brojeva a u skupu {1, 2, ..., m} sa osobinom NZD(a,m) = 1 obeležavamo sa $\Phi(m)$. Ojlerova Φ funkcija ukazuje na broj brojeva iz pomenutog skupa koji su uzajamno prosti u odnosu na *m*. Ako je *m* prost broj, $\Phi(m) = m-1$. Ako je NZD(m,n) = 1, onda je $\Phi(m \cdot n) = \Phi(m) \cdot \Phi(n)$.

Najmanji zajednički sadržalac NZS(a,b) brojeva *a* i *b* (a≠0, b≠0) je najmanji broj deljiv brojevima *a* i *b*. Računa se prema sledećoj formuli:

- $NZS(a,b) = a \cdot b / NZD(a,b)$.

Na primer, $NZS(12,18) = 12 \cdot 18/6 = 36$.

Za svako *a*, *b*, i *n* jednačina $a \cdot x + b \cdot y = n$ je rešiva u skupu celih brojeva ako i samo ako NZD(a,b) deli *n*. Na primer, jednačina $3 \cdot x + 4 \cdot y = 123$ je rešiva u skupu celih brojeva jer NZD(3,4) = 1 deli 123.

Kongruencija

Neka su *a* i *b* celi brojevi. Kažemo da je *a* kongruentno sa *b* po modulu *n* ako *n* deli a–b. Pišemo $a \equiv b \pmod n$. Drugim rečima, ako *a* i *b* daju isti ostatak pri deljenju sa *n*, onda su oni kongruentni po modulu *n*. Na primer:

1 "$d \leftarrow a$" znači da *d* postaje *a*
2 "[x]" označava celobrojnu vrednost od *x*

- $24 \equiv 9 \pmod 5$ jer $24 - 9 = 3 \cdot 5$
- $-11 \equiv 17 \pmod 7$ jer $-11 - 17 = -4 \cdot 7$

Za svako celobrojno a, a_1, b, b_1, c važi:

- $a \equiv a \pmod n$,
- ako je $a \equiv b \pmod n$, onda je $b \equiv a \pmod n$,
- ako je $a \equiv b \pmod n$ i $b \equiv c \pmod n$, onda je $a \equiv c \pmod n$,
- ako je $a \equiv a_1 \pmod n$ i $b \equiv b_1 \pmod n$, onda je:
  $a + b \equiv a_1 + b_1 \pmod n$, i $a \cdot b \equiv a_1 \cdot b_1 \pmod n$.

Kineska teorema ostatka tvrdi sledeće: neka su n_1, n_2, ..., n_k uzajamno prosti brojevi. Sistem kongruencija:

- $x \equiv a_1 \pmod{n_1}$, $x \equiv a_2 \pmod{n_2}$, ... $x \equiv a_k \pmod{n_k}$

ima jedinstveno rešenje po modulu $n_1 \cdot n_2 \cdot \ldots \cdot n_k$.

Specijalno, ako je $NZD(n_1,n_2)=1$, tada par kongruencija $x \equiv a_1 \pmod{n_1}$ i $x \equiv a_2 \pmod{n_2}$ ima jedinstveno rešenje $x \equiv a \pmod{n_1 \cdot n_2}$. Na primer, par kongruencija $x \equiv 3 \pmod 7$ i $x \equiv 7 \pmod{13}$ ima jedinstveno rešenje $x \equiv 59 \pmod{91}$.

Klasa ekvivalencije u odnosu na ceo broj *a* je skup svih celih brojeva kongruentnih sa *a* po modulu *n*. Na primer, u odnosu na broj 2 postoje dve klase ekvivalencije: parni brojevi (daju ostatak 0) i neparni brojevi (daju ostatak 1). Svaki broj *a* je kongruentan po modulu *n* nekom broju od 1 do n–1. Brojevi iz tog skupa reprezentuju sve ostale brojeve i nazivaju se najmanjim ostacima broja *a* po modulu *n*.

Označimo sa $Z_n = \{0, 1, \ldots, n-1\}$. Sabiranje, oduzimanje, množenje u Z_n podrazumevamo po modulu *n*. Na primer, u Z_{25} $13+16 = 29 \equiv 4 \pmod{25}$. Inverzni element *a* iz Z_n po modulu *n* je ceo broj *x* iz Z_n takav da važi $a \cdot x \equiv 1 \pmod n$. Označavamo ga sa a^{-1}. Deljenje *a* i *b* po modulu *n* je proizvod *a* i inverznog elementa od *b* ako on postoji:

- $a/b \pmod n = a \cdot b^{-1} \pmod n$

Element *a* iz Z_n je invertibilan, tj. ima inverzni element ako i samo ako je NZD(a,n)=1. Invertibilni elementi u Z_9 su 1, 2, 4, 5, 7, 8. Na primer, $7^{-1}=4$ jer je $4 \cdot 7 \equiv 1 \pmod 9$.

Neka je d = NZD(a,n). Jednačina $a \cdot x = b \pmod n$ ima rešenje ako i samo ako *d* deli *b* i tada postoji tačno jedno rešenje za *d* između 0 i n-1.

1.3 Zatvorenost skupova, grupe, podgrupe i polja

Skup G je zatvoren za neku operaciju "●" ako za bilo koja dva elementa *a* i *b* iz *G* rezultat operacije a●b takođe pripada *G*. Na primer, skup prirodnih brojeva N je zatvoren za operaciju sabiranja, ali ne i za operaciju oduzimanja ($2 \in N$ i $3 \in N$, ali $2-3=-1 \notin N$).

Za operaciju "●" kažemo da je asocijativna ako za bilo koje brojeve *a*, *b*, i *c* iz nekog skupa *G* važi (a●b)●c = a●(b●c). Na primer, sabiranje u skupu *N* je asocijativno ali oduzimanje nije: $1-(2-3) \neq (1-2)-3$

Ako je data operacija "●", element *e* nazivamo neutralnim elementom ako za svako *a* iz nekog skupa *G* važi a●e = e●a = a. Na primer, ako je operacija "●" sabiranje, neutralni element je 0 jer je a+0 = 0+a = a. Ako je "●" množenje neutralni element je 1 jer je $a \cdot 1 = 1 \cdot a = a$.

Pojam grupe

Algebarska struktura (G,●) jeste grupa ukoliko važi sledeće:

- zatvorenost za binarnu operaciju: (a,b) -> $a \cdot b$,
- asocijativnost: (a●b)●c = a●(b●c),
- postojanje neutralnog elementa: a●e=e●a=a,
- postojanje inverznog elementa: $a \cdot a\text{-}1 = a\text{-}1 \cdot a = e$.

Grupa je Abelova ako još važi a●b=b●a (komutativnost).

Na primer, posmatrajmo skup celih brojeva i operaciju množenja $(Z,\cdot)$:

- Z je zatvoren za množenje – npr. za 3 i -7 važi $3 \cdot (-7)=21$,
- asocijativnost važi: $2 \cdot (3 \cdot (-7)) = (2 \cdot 3) \cdot (-7)$,
- neutralni element je 1 jer je, na primer, $3 \cdot 1 = 1 \cdot 3 = 3$,
- inverzni element za 3 je 1/3 a on ne pripada skupu Z.

Nepostojanje inverznog elementa ukazuje da $(Z,\cdot)$ nije grupa. Slično tome, može se pokazati da (Z,+) jeste grupa.

Multiplikativnu grupu Z_n^* čine svi brojevi od 1 do *n* koji su uzajamno prosti sa *n*. Dakle, $Z_n^* = \{a \in Z_n \mid NZD(a,n)=1\}$. Specijalno, ako je *n* prost broj, onda je $Z_n^* = \{a \mid 0<a<n\}$.

Mala Fermaova teorema

Red grupe Z_n^* definišemo kao broj elemenata grupe: $|Z_n| = \Phi(n)$, gde je Φ Ojlerova funkcija.

Navodimo Ojlerovu teoremu: ako je $a \in Z_n^*$, onda važi $a^{\Phi(n)} \equiv 1 \pmod n$.

Takođe, ako je $r \equiv s \pmod{\Phi(n)}$, onda za svako *a* važi $a^r = a^s \pmod n$.

Drugim rečima kad radimo po modulu *n*, stepen može biti redukovan po modulu $\Phi(n)$. Na primer, pošto je 2 iz Z_5 i pošto je $\Phi(Z_5)=4$, sledi $2^4 \equiv 1 \pmod 5$. Takođe, $2^4 \equiv 2^8 \pmod 5$.

Specijalan slučaj Ojlerove teoreme predstavlja Mala Fermaova teorema: ako je NZD(a,p)=1, tada važi: $a^{p-1} \equiv 1 \pmod p$.

Neka je $a \in Z_n$. Red elementa *a* predstavlja najmanji pozitivan broj *t* takav da važi: $a^{t-1} \equiv 1 \pmod t$.

Ako je *t* red elementa $a \in Z_n^*$ i ako je $a^s = 1 \pmod n$, onda *t* deli *s*. Specijalno, t deli $\Phi(n)$.

Na primer: $Z_{21}^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$.

$\Phi(21) = \Phi(7) \cdot \Phi(3) = 6 \cdot 2 = 12 = |Z_{21}^*|$.

Ako je $a \in Z_{21}^*$, sledeća tabela prikazuje red od *a*:

a	1	2	4	5	8	10	11	13	16	17	19	20
red	1	6	3	6	2	6	6	2	3	6	6	2

Na primer, $a^6 \equiv 1 \pmod{21}$ ako je a=2, 5, 10, 11, 17 ili 19.

Ciklična grupa

Grupa *G* je ciklična ako postoji element $a \in G$ takav da za svako $b \in G$ postoji celobrojno *i* tako da važi $b = a^i$. Element *a* nazivamo generatorom ciklične grupe *G*. Drugim rečima, ako se svi elementi nekog skupa mogu predstaviti različitim stepenima jednog istog elementa, onda taj element generiše celu grupu i naziva se generator, dok se pomenuta grupa naziva ciklična.

Neka je $a \in Z_n^*$. Ako je red elementa *a* jednak $\Phi(n)$ tada *a* zovemo generatorom grupe Z_n^*. Ako grupa Z_n^* ima generator, onda kažemo da je ciklična. Na primer, grupa Z_{21}^* nije ciklična jer ne sadrži element reda $\Phi(n)=21$. Grupa Z_{25}^* jeste ciklična jer ima generator a=2.

Navodimo neke značajnije osobine generatora:

- Z_n^* ima generator ako i samo ako je $n=2, 4, p^k, 2p^k$, gde je $k>0$, a *p* neparan prost broj.
- Ako je *a* generator Z_n^*, tada je: $Z_n^* = \{a^i \bmod n \mid 0 \le i \le \Phi(n)-1\}$.
- Neka je $a \in Z_n^*$; *a* je generator Z_n^* ako i samo ako za svaki prost delilac *p* broja $\Phi(n)$ ne važi: $a^{\Phi(n)/p} \equiv 1 \pmod{n}$.
- Neka je *a* generator Z_n^*. Element $b = a^i \pmod{n}$ je generator Z_n^* ako i samo ako je $NZD(i,\Phi(n)) = 1$. Sledi da grupa Z_n^* ima $\Phi(\Phi(n))$ generatora ukoliko je ciklična.

Na primer, grupa Z_5^* je ciklična jer je broj 3 njen generator. Iz $\Phi(5)=4$ i NZD(3,4)=1 sledi da je 33 (mod 5) =2 takođe generator. Dakle, grupa Z_5^* ima $\Phi(\Phi(5)) = 2$ generatora: 2 i 3.

Pojam podgrupe

Neprazan podskup *H* grupe *G* koji je takođe grupa nazivamo podgrupom grupe G. Na primer, podgrupe $Z_{19}^* = \{1, 2,..., 18\}$ su {1, 18} i {1, 7, 11}.

Prema Lagranžovoj teoremi, ako je *G* konačna grupa i *H* njena podgrupa, onda |H| deli |G|. Red grupe, u oznaci |G| ili |H|, predstavlja broj elemenata u njoj.

Svaka podgrupa ciklične grupe je ciklična. Ako je *G* ciklična grupa reda *n*, tada za svaki pozitivan delilac *d* broja *n* postoji tačno jedna podgrupa reda *d*. Tada takođe grupa *G* ima $\Phi(d)$ elemenata reda *d*. Pošto generatori moraju biti reda *n* i pošto n I n, sledi da *G* ima $\Phi(n)$ generatora.

Razmotrimo multiplikativnu grupu $Z_{19}^* = \{1, 2,..., 18\}$. Ona je ciklična sa, na primer, generatorom 2. Slede njene podgrupe sa mogućim generatorima:

Podgrupa	Generatori	Red
{1}	1	1
{1, 18}	18	2
{1, 7, 11}	7, 11	3
{1, 7, 8, 11, 12, 18}	8, 12	6
{1, 4, 5, 6, 7, 9, 11, 16, 17}	4, 5, 6, 9, 16, 17	9
{1, 2, 3, ..., 18}	2, 3, 10, 13, 14, 15	18

Prsten i polje

Prsten (R,+,*) je algebarska struktura sastavljena od skupa *R* i operacija "+" i "*" (uobičajeno sabiranja i množenja) koja ima sledeće osobine:

- (R,+) je Abelova grupa sa neutralnim elementom 0
- operacija "*" je asocijativna
- za svako *a* iz *R* postoji neutralni element 1 za "*"
- operacija "*" je distributivna u odnosu na "+": a*(b+c) = (a*b)+(a*c)

Ako za element *a* prstena *R* postoji b∈R tako da važi a*b = 1, onda b nazivamo inverznim elementom elementa *a*.

Prsten je komunikativan ukoliko za svako a,b∈R važi a*b = b*a. Na primer, strukture (Z,+, ·) i (Zn, +(mod n), · (mod n)) su komutativni prstenovi.

Komutativni prsten kod koga svaki element (osim nula elementa) ima multiplikativni inverz naziva se polje. Z_n je polje ako i samo ako je *n* prost broj. Na primer, strukture (Q,+, ·), (R,+, ·), (C,+, ·) su polja. Struktura (Z,+, ·) nije polje jer jedino elementi 1 i -1 imaju multiplikativni inverz.

Konačno polje *F* je polje koje sadrži konačan broj elemenata. Red konačnog polja F je broj elemenata polja *F*.

Za svaki stepen reda *m* prostog broja *p* postoji jedinstveno konačno polje reda p^m. To polje obeležavamo sa F_{p^m} ili $GF(p^m)$. Ako je *p* prost broj i F_q konačno polje reda $q = p^m$, F_q sadrži Z_p kao podpolje.

Polinomi u konačnim poljima

Z_p je polje ukoliko je *p* prost broj. Postoji konačno polje od $q=p^n$ elemenata gde je *p* prost broj i n>0 ceo broj. Definišimo $Z_p[x]$ kao skup svih polinoma čiji su koeficijenti redukovani po modulu *p*. Neka su f(x), g(x) i q(x) polinomi iz skupa Zp[x]. Kažemo:

- f(x) deli g(x), tj. f(x)|g(x), ako postoji q(x) tako da je g(x) = q(x) · f(x).

Za f(x) iz Zp[x] stepen polinoma u oznaci deg(f) predstavlja eksponent člana najvišeg stepena u polinomu. Neka su f(x), g(x) i h(x) iz Zp[x] i deg(f)=n>0. Definišimo:

- g(x) ≡ h(x) (mod f(x)), ako f(x)|(g(x)-h(x))

Konstrukcija prstena polinoma iz $Z_p[x]$ po modulu f(x), tj. $Z_p[x]/f(x)$, analogna je konstrukciji prstena Z_m iz Z. Ulogu modula *m* sada nosi f(x). To znači: ako je deg(f)=n i delimo g(x) sa f(x) dobijamo kao rezultat ostatak r(x) jer je g(x) = q(x)f(x) + r(x) i deg(r)<n.

Polinom f(x) iz $Z_p[x]$ je nesvodljiv ako ne postoje polinomi $f_1(x)$, $f_2(x)$ iz $Z_p[x]$ takvi da $f(x) = f_1(x) \cdot f_2(x)$, gde su $deg(f_1)>0$ i $deg(f_2)>0$. Drugim rečima, f(x) je nesvodljiv ako je deljiv samo sa sobom i sa 1. Nesvodljiv polinom među polinomima je isto što i prost broj među brojevima.

Z_m je polje ako i samo ako je *p* prost broj. Slično, $Z_p[x]/f(x)$ je polje ako i samo ako je f(x) nesvodljiv polinom. Traženje multiplikativnog inverza u $Z_p[x]/f(x)$, slično kao i u polju Z_p, obavlja se preko proširenog Euklidovog algoritma.

Na primer, konstruišimo polje od 8 elemenata. Najpre moramo naći nesvodljiv polinom stepena 3 u $Z_2[x]$. Ovde se nalaze samo polinomi sa koeficijentima 0 ili 1 i stepenom najviše 2. Kako su svi polinomi stepena 3 sa konstantnim članom nula deljivi sa *x*, kao kandidati za nesvodljiv polinom ostaju polinomi x^3+1, x^3+x+1, x^3+x^2+1, x^3+x^2+x+1. Posmatrajte najpre polinome x^3+1 i x^3+x^2+x+1:

- $x^3+1 = (x+1)(x^2+x+1)$, pa nije nesvodljiv,
- $x^3+x^2+x+1 = (x+1)(x^2+1)$, pa nije nesvodljiv.

Od preostala dva biramo na primer x^3+x+1 i na taj način dobijamo polje $Z_2[x]/(x^3+x+1)$ čiji su elementi: 0 , 1 , x , x+1, x^2 , x^2+1, x^2+x , x^2+x+1.

Operacije nad polinomima u konačnim poljima

Sabiranje polinoma se svodi na izvršavanje operacije ekskluzivno ILI (u oznaci " $\oplus$ ") nad odgovarajućim koeficijentima:

- $\{a_7\ a_6 \ldots a_1\ a_0\}+\{b_7\ b_6 \ldots b_1\ b_0\} = \{c_7\ c_6 \ldots c_1\ c_0\}$, gde je $c_i = a_i \oplus b_i$

Na primer:

- $(x^7+x^6+x^5+1) + (x^6+x^5+x+1) = (x^7+x)$

- {11100001} + {01100011} = {10000010}
- {E1} + {63} = {82}, zapisano heksadecimalno

Množenje u konačnom polju GF(2^8) odgovara množenju polinoma po modulu *m(x)*, gde je m(x) nedeljiv polinom osmog stepena. Operacija deljenja po modulu, tj. traženja ostatka pri deljenju sa *m(x)* obezbeđuje da će rezultat biti stepena manjeg od 8, tj. da se rezultat može prikazati kao polinom. Množenje definisano na ovakav način je asocijativno, a multiplikativna konstanta je 1. Množenje i sabiranje su distributivne operacije. Kao nesvodljiv polinom može se koristi, na primer, $m(x) = x^8+x^4+x^3+x+1$ (ovaj polinom koristi AES algoritam). Na primer:

- $(x^6+x^4+x^2+x+1) \cdot (x^7+x+1) \bmod (x^8+x^4+x^3+x+1) =$

  $= (x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1) \bmod (x^8+x^4+x^3+x+1) =$

  - $= x^7+x^6+1$

Navodimo prošireni Eulidov algoritam za polinome u polju $Z_p[x]$. Algoritam na osnovu ulaznih podataka, tj. na osnovu dva polinoma $g(x),h(x) \in Z_p(x)$ generiše $d(x) = NZD(g(x),h(x))$ i polinome $s(x),t(x) \in Zp(x)$ koji zadovoljavaju jednačinu $s(x) \cdot g(x) + t(x) \cdot h(x) = d(x)$

[1.] ako je $h(x)=0$, tada $d(x) \leftarrow g(x)$, $s(x) \leftarrow 1$, $t(x) \leftarrow 0$. Vrati $(d(x),s(x),t(x))$

[2.] $s_2(x) \leftarrow 1$, $s_1(x) \leftarrow 0$, $t_2(x) \leftarrow 0$, $t_1(x) \leftarrow 1$

[3.] dok je $h(x) \neq 0$ radi sledeće:

[3.1] $q(x) \leftarrow g(x)$ div $h(x)$, $r(x) \leftarrow g(x) - h(x)q(x)$

[3.2] $s(x) \leftarrow s_2(x) - q(x)\, s_1(x)$, $t(x) \leftarrow t_2(x) - q(x)t_1(x)$

[3.3] $g(x) \leftarrow h(x)$, $h(x) \leftarrow r(x)$

[3.4] $s_2(x) \leftarrow s_1(x)$, $s_1(x) \leftarrow (x)$, $t_2(x) \leftarrow t_1(x)$, $t_1(x) \leftarrow t(x)$

[4.] $d(x) \leftarrow g(x)$, $s(x) \leftarrow s_2(x)$, $t(x) \leftarrow t_2(x)$

[5.] vrati $(d(x), s(x), t(x))$

Algoritam je primenljiv na određivanje Inverznog elementa polinoma b(x) po modulu m(x). Proširenim Euklidovim algoritmom dobijamo a(x) i c(x), za koje važi: $b(x) \cdot a(x) + m(x) \cdot c(x) = 1$. Sledi $b(x) \cdot a(x) = 1 \bmod m(x)$, tj. a(x) je inverz za b(x) po modulu m(x).

1.4 Uvod u programski jezik Java

Sigurnost podataka u mnogome podrazumeva umeće programiranja. To može biti zaštita od prekoračenja bafera, pisanje antivirusnog softvera, itd. Kriptografija, kao specifičan deo zaštite podataka, takođe podrazumeva programiranje. Primeri koji slede u ovom priručniku, a koji će detaljnije biti predstavljeni u zbirci zadataka, napisani su pretežno u programskom jeziku Java i delimično u programskom jeziku C/C++.

Iako isuviše obiman jezik da bi se mogao predstaviti u uvodnom poglavlju, zbog lakšeg razumevanja koda koji predstoji, Java će biti opisana u kraćim crtama, i to kroz zvanično važeci kriptografski standard AES. Činjenica da je ovaj algoritam postao standard, povlači realnu pretpostavku da postoji relativno brz i lak način za njegovo razbijanje. Pa ipak, svako ko se bavi sigurnošću računarskih mreža trebalo bi da bude upoznat s njim. AES će u jednom od narednih poglavlja biti detaljnije opisan; cilj ovog uvoda je da čitalac stekne sliku o programskoj implementaciji kriptoalgoritma kao i da se sprijatelji s Javom.

Implementacija AES algoritma u Javi

Blokovi u kriptoalgoritmu predstavljaju parčiće koji se uglavnom zasebno šifruju. Na kraju se šifrat sastoji iz niza šifrovanih blokova. AES koristi blokove dužine 128 bitova. Ključ u kriptografiji predstavlja informaciju, najčešće izraženu u bitovima, koja se povezuje sa osnovnim tekstom kako bi se proizveo šifrat. AES koristi ključeve dužine 128, 192 ili 126 bitova. Osnovni tekst prolazi kroz više uzastopnih operacija koje se nazivaju runde. U skladu sa dužinom ključa, određen je i broj rundi N_r – 10, 12 ili 14. Svaka runda poseduje svoj ključ runde. Daćemo grub opis algoritma, a potom njegovu Java implementaciju, uz napomenu da je za njegovo razumevanje potrebno znanje iz poglavlja koja slede.

- Za dati osnovni tekst x inicijalizuj State sa x i primeni operaciju AddRoundKey koja primenjuje operaciju ekskluzivno ILI nad ključem runde (RoundKey) sa matricom State (kvadratna matrica od 16 bajtova).

- Za svaku od N_{r-1} rundi uradi sledeće operacije:
  - SubButes (state) – koja svaki bajt matrice State pretvara u novi bajt (zamena pomoću supstitucijske kutije),
  - ShiftRows (state) – rotira zasebno svaku vrstu za broj vrste mesta,
  - MixColumns (state) – menja sadržaj svake kolone matrice State,
  - AddRoundkey (state).
- U poslednoj, N_r-toj rundi uradi sledeće operacije:
  - SubButes (state),
  - ShiftRows (state),
  - AddRoundkey (state).

Matricu State dobijenu na ovakav način proglasi šifratom. Pogledajmo kako bi izgledao Java program:

```
// AESencrypt: AES šifrovanje
public class AESencrypt {
  private final int Nb = 4; // reči u bloku, uvek 4 (za sada)
  private int Nk; // dužina ključa u rečima (4,6 ili 8 reci)
  private int Nr; // broj rundi, = Nk + 6
  private int wCount; // pozicija u nizu w za ključ runde
  private AEStables tab; // sve tablice potrebne za AES
  private byte[] w; // konkatenacija svih ključeva rundi

// AESencrypt: konstruktor klase. Uglavnom proširuje ključ
  public AESencrypt(byte[] key, int NkIn) {
    Nk = NkIn; // reči u ključu, = 4, or 6, or 8
    Nr = Nk + 6; // odgovarajući broj rundi
    tab = new AEStables(); // klasa koja daje vrednost
                           // različitih tablica
    w = new byte[4*Nb*(Nr+1)]; //niz za expanded key
    KeyExpansion(key, w); // dužina za w zavisi od Nr
  }

// Cipher: pozivi funkcija za šifrovanje
  public void Cipher(byte[] in, byte[] out) {
    wCount = 0; // broji bajtove u expanded key
    byte[][] state = new byte[4][Nb]; // state matrica
    Copy.copy(state, in); // kopiranje iz in u state
```

```
    AddRoundKey(state); // xor sa expanded key
    for (int round = 1; round < Nr; round++) {
      Print.printArray("Runda " + round + ":", state);
      SubBytes(state);
      ShiftRows(state);
      MixColumns(state);
      AddRoundKey(state);
    }
    Print.printArray("Runda " + Nr + ":", state);
    SubBytes(state);
    ShiftRows(state);
    AddRoundKey(state);
    Copy.copy(out, state);
  }
```

Naš program će se sastojati iz više klasa (datoteka) koje ćemo, da bismo izbegli korišćenje paketa, staviti u isti direktorijum. Neka svaka klasa bude u posebnoj datoteci. Prvi klasa se zove AESencrypt i ona sadrži privatne atribute i privatne i javne funkcije članice – metode. Time je ispoštovan objektni princip enkapsulacije: pristup atributima pa i nekim metodama je dozvoljen samo unutar dotične klase. Reč final ukazuje na konstantu jer za sada AES koristi blokove od 128 bitova i u okviru njih 4 reci (words) od 4 bajta (32 bita). Klasa AEStables u sebi sadrži tablice iz kojih ćemo čitati potrebne podatke. Ona će biti predstavljena kasnije, a do tada ćemo samo napraviti referencu tab kojoj još nismo dodelili objekat. Takođe ćemo definisati niz bajtova w u koji ćemo jedan za drugim smestiti sve ključeve rundi i odakle ćemo ih po potrebi uzimati.

Kao posebnu metodu klase izdvojićemo konstruktor. Ona nema povratnu vrednost, skoro uvek je javna (public) i služi pre svega za inicijalizaciju atrubuta. Kada se pravi objekat, poziva se konstruktor, pa tako i sve što se nalazi u njegovom telu. U našem slučaju, u zavisnosti od ulaznih argumenata konstruktora, Nr će dobiti vrednost željenog broja rundi (10,12 ili 14) a samim tim će se i ogovarajuće tabele napraviti instanciranjem reference tab. Takođe, zauzimamo prostor (niz bajtova w) za niz ključeva rundi (16 puta (broj rundi+1) bajtova). Na kraju, funkcija KeyExpansion čiji su ulazni parametri polazni ključ i niz w, puni w dobijenim ključevima rundi.

U metodi Cipher dešava se algoritam koji je na pocetku bio grubo opisan. Na primer, u slucaju 128-bitnog ključa postoji ulazna iteracija sastavljena samo od kopiranja osnovnog teksta u matricu State (preko klase Copy i njene metode copy koje će nadalje biti predstavljene) i metode

AddRoundKey. Nadalje ide devet rundi u kojima se dešavaju metode SubBytes, ShiftRows, MixColumns i AddRoundKey. Poslednja, deseta runda, ne sadrži jedino metodu MixColumns. Dakle, zajedno sa početnim ključem ukupno imamo 11 ključeva rundi i samim tim niz w od 44 reči ili 176 bajtova.

Klasa Print sa svojim metodama printArray služi samo za ispis. Java dozvoljava da u slučaju statičke (static) metode ne pravimo objekat za njeno pozivanje već za to koristimo samo ime klase. Static metode su ekvivalent globalnim funkcijama koje Java ne podržava. Sintaksa Jave veoma liči na sintaksu jezika C i C++, pa je nećemo detaljno obrazlagati, uz napomenu da je ^ operator XOR, a % ostatak pri deljenju. Iznesimo sada Java kôd pomenutih metoda:

```
// KeyExpansion: pravi niz od konkatenacije svih ključeva
// radi sa bajtovima mada trazi reči (4 bajta)

private void KeyExpansion(byte[] key, byte[] w) {
  byte[] temp = new byte[4];
  // prvo se kopira ključ u w
  int j = 0;
  while (j < 4*Nk) w[j] = key[j++]; // ovde je j == 4*Nk;
  int i;
  while(j < 4*Nb*(Nr+1)) {
    i = j/4; // j je uvek deljivo sa 4
             // hvata svaku reč, (sukcesivno svaka 4 bajta)
    for (int iTemp = 0; iTemp < 4; iTemp++)
    temp[iTemp] = w[j-4+iTemp];
    if (i % Nk == 0) {
      byte ttemp, tRcon;
      byte oldtemp0 = temp[0];
      for (int iTemp = 0; iTemp < 4; iTemp++) {
        if (iTemp == 3) ttemp = oldtemp0;
        else ttemp = temp[iTemp+1];
        if (iTemp == 0) tRcon = tab.Rcon(i/Nk);
        else tRcon = 0;
        temp[iTemp] = (byte)(tab.SBox(ttemp) ^ tRcon);
      }
    }
    else if (Nk > 6 && (i%Nk) == 4) {
      for (int iTemp = 0; iTemp < 4; iTemp++)
      temp[iTemp] = tab.SBox(temp[iTemp]);
    }
    for (int iTemp = 0; iTemp < 4; iTemp++)
    w[j+iTemp] = (byte)(w[j - 4*Nk + iTemp] ^ temp[iTemp]);
```

```
    j = j + 4;
  }
}

// SybBytes: primenjuje Sbox zamenu na svaki bajt u state
private void SubBytes(byte[][] state) {
  for (int row = 0; row < 4; row++)
    for (int col = 0; col < Nb; col++)
      state[row][col] = tab.SBox(state[row][col]);
}

// ShiftRows: prosto pomeranje vrsta 1, 2, 3 za 1, 2, 3
private void ShiftRows(byte[][] state) {
  byte[] t = new byte[4];
  for (int r = 1; r < 4; r++) {
    for (int c = 0; c < Nb; c++) t[c] = state[r][(c+r)%Nb];
    for (int c = 0; c < Nb; c++) state[r][c] = t[c];
  }
}

// MixColumns: komplikovanije operacije nad svakom kolonom
private void MixColumns(byte[][] s) {
  int[] sp = new int[4];
  byte b02 = (byte)0x02, b03 = (byte)0x03;
  for (int c = 0; c < 4; c++) {
    sp[0] = tab.FFMul(b02, s[0][c]) ^
            tab.FFMul(b03, s[1][c]) ^ s[2][c] ^ s[3][c];
    sp[1] = s[0][c] ^ tab.FFMul(b02, s[1][c]) ^
            tab.FFMul(b03, s[2][c]) ^ s[3][c];
    sp[2] = s[0][c] ^ s[1][c] ^ tab.FFMul(b02, s[2][c]) ^
            tab.FFMul(b03, s[3][c]);
    sp[3] = tab.FFMul(b03, s[0][c]) ^ s[1][c] ^
            s[2][c] ^ tab.FFMul(b02, s[3][c]);
    for (int i = 0; i < 4; i++) s[i][c] = (byte)(sp[i]);
  }
}

// AddRoundKey: xor delova expanded key sa state
private void AddRoundKey(byte[][] state) {
  for (int c = 0; c < Nb; c++)
  for (int r = 0; r < 4; r++)
  state[r][c] = (byte)(state[r][c] ^ w[wCount++]);
  }
}
```

Jednostavnosti, radi podrazumevajmo nadalje da koristimo 128-bitni

ključ, odnosno 10 rundi, mada će program zadržati opštost upotrebe sve tri dužine ključa.

Sledeća metoda SubBytes koristi vrednosti iz odgovarajuće tablice koja je predstavljena nizom u klasi AEStables i to primenjuje na svaki bajt matrice State. Za sada je dovoljno reći da se nad svakim bajtom primenjuju neke transformacije koje za izlaz opet daju bajt.

Metoda KeyExpansion ima dva argumenta: ulazni ključ i niz w od 128 bitova. Pošto se niz kao argument funkcije prenosi po referenci, metoda može da bude void (nema povratnu vrednost) a sve promene na privatnom atributu (nizu w) će biti vidljive posle završetka metode. Posle kopiranja ulaznog ključa u prvih 16 bajtova niza w niz temp od 4 bajta (1 reč) puni se poslednjom reči stavljenoj u niz w. Pošto radimo sa 128-bitnim ključem on u sebi sadrzi 4 reči – pa je Nk (broj reči u ključu) jednako 4. Kada brojač " i" dođe do svake četvrte reči pomoćni niz temp postaje XOR rezultata rotiranja svojih bajtova za jedno mesto ulevo i primene operacije SybBytes nad svakim bajtom s jedne strane i niza od četiri promenljive Rcon s druge strane. Prva promenljiva Rcon, koja je veličine jednog bajta, uzima vrednost iz odgovarajuće tabele (niza) klase AEStables. Preciznije, to je bajtovni prikaz heksadecimalnog zapisa polinoma x+1 stepenovanog na broj runde. Ostale tri Rcon promenljive su nule. Na kraju, svaka reč se dobija kao xor prethodne i niza temp. Izlaz je konkatencija od 44 reči.

Metoda ShiftRows u matrici State ostavlja nultu vrstu nepromenjenu, dok prvu rotira za jedno mesto ulevo, drugu za dva mesta ulevo i treću za tri mesta ulevo.

Metoda MixColumns vrši određene transformacije nad svakom kolonom matrice State ponaosob. U njoj srećemo dva polinoma: x i x+1. Njihov heksadecimalni (binarni) zapis je 02(000000010) i 03(00000011). Prosto je izvršena korespodencija između bajta (8 bitova) i koeficijenata polinoma sedmog stepena (8 koeficijenata).

Heksadecimalni zapis u Javi se označava sa 0x02 i 0x03 i ako je potrebna konverzija u tip byte (byte)0x02. Svaki član (bajt) kolone se pretvara u polinom, te tako za svaku kolonu dobijamo četiri polinoma. Korišćenjem Funkcije FFMul koja množi polinome po modulu 0x11b i pomenutih polinoma ox02 i 0x03, uz odgovarajuće xor-ovanje dobijamo četiri nova polinoma koja ponovo vraćamo u bajtove i dobijamo promenjenu

kolonu. Važno je napomenuti da je u Javi rezultat xor-ovanja dve promenljive tipa byte promenljiva tipa int. Zato je za skladištenje rezultata xor-ovanja uzet niz sp tipa int. Alternativa je bila da se pre dodele u tip byte cela leva strana konvertuje u tip byte.

Metoda AddRoundKey xoruje svako novo stanje matrice State sa odgovarajućim ključem runde koji je došao na red. To se postiže povećavanjem privatnog atributa wCount koji je na početku inicijalizovan na nulu. Inače su Javini atributi u klasi podrazumevano postavljeni na nulu.

Postojanje tablica iz kojih možemo pročitati željene vrednosti značajno ubrzava i pojednostavljuje implementaciju AESa. Tablice ćemo predstavljati nizovima. U nizu E se nalaze rezultati stepenovanja polinoma x+1(0x03) sa 256 vrednosti počev od 0x00 do 0xff. U nizu L se nalaze inverzne vrednosti. Ideja je sledeća: pomnožimo npr 0xb6* 0x53 = E(L(0xb6))*E(L(0x53)) = E(L(0xb6) +L(0x53)) = E(0xb1 + 0x30) = E(0xe1) = 0x36. Ovo smo mogli uraditi budući da niz E predstavlja stepene polinoma 0x03, a niz L logaritme dobijenih vrednosti. Ovim smo množenje polinoma bitno pojednostavili. Jedino je još potrebno programski stvoriti potrebne tabele a za to će nam ipak biti potreban neki drugi način množenja. Niz inv sadrži inverzne polinome po polinomu modulu 0x11b koji je fiksno određen kao modul u AESu prilikom množenja polinoma u polju GF(256). Niz powX sadrži 16 vrednosti: stepene polinoma 0x02 od 0x0000 do 0x000f. U nizu S se nalaze vrednosti bajtova posle primene operacije SybBytes na svakom bajtu. Ta operacija ulazni bajt pretvara u polinom, nalazi njegov inverz koji ponovo vraća kao bajt i uz korišćenje bajta 01100011 u petlji daje izlazni bajt. Niz invS daje inverznu SybBytes tabli koju koristimo prilikom dešifrovanja.

Predstavimo sada klasu AEStables:

```
// AEStables: konstrukcija različitih 256-bajtnih
// tablica potrebnih za //AES
public class AEStables {
  public AEStables() {
    loadE(); loadL(); loadInv();
    loadS(); loadInvS(); loadPowX();
  }
  // "exp" tablica E (osnova 0x03)
  private byte[] E = new byte[256];
  // "Log" tablica L (osnova 0x03) – inverzna E tablici
  private byte[] L = new byte[256];
  // SubBytes tablica
```

```
  private byte[] S = new byte[256];
  // inverzna SubBytes tablica
  private byte[] invS = new byte[256];
  // tablica za multiplikativne inverze
  private byte[] inv = new byte[256];
  // stepeni od x = 0x02
  private byte[] powX = new byte[15];

  // Funkcije za dohvatanje sadrzaja tablica
  public byte SBox(byte b) { return S[b & 0xff]; }
  public byte invSBox(byte b) { return invS[b & 0xff]; }
  public byte Rcon(int i) { return powX[i-1]; }

// FFMulFast: brzo množenje polinoma koriscenjem tablica
  public byte FFMulFast(byte a, byte b){
    int t = 0;;
    if (a == 0 || b == 0) return 0;
    t = (L[(a & 0xff)] & 0xff) + (L[(b & 0xff)] & 0xff);
    if (t > 255) t = t - 255;
    return E[(t & 0xff)];
  }

// FFMul: sporo množenje, koriscenjem pomeraja bitova
  public byte FFMul(byte a, byte b) {
    byte aa = a, bb = b, r = 0, t;
    while (aa != 0) {
      if ((aa & 1) != 0) r = (byte)(r ^ bb);
      t = (byte)(bb & 0x80);
      bb = (byte)(bb << 1);
      if (t != 0) bb = (byte)(bb ^ 0x1b);
      aa = (byte)((aa & 0xff) >> 1);
    }
    return r;
  }

// loadE: kreiranje i punjenje E tablice (tablice stepena)
  private void loadE() {
    byte x = (byte)0x01;
    int index = 0;
    E[index++] = (byte)0x01;
    for (int i = 0; i < 255; i++) {
      byte y = FFMul(x, (byte)0x03);
      E[index++] = y;
      x = y;
    }
  }
```

```
// loadL: punjenje L tablice koriscenjem E tablice
  private void loadL() {
    // pazljivo: tablica L ima 254 vrednosti (nema za 00)
    int index;
    for (int i = 0; i < 255; i++) L[E[i] & 0xff] = (byte)i;
  }

// loadS: punjenje tablice S
  private void loadS() {
    int index;
    for (int i = 0; i < 256; i++)
    S[i] = (byte)(subBytes((byte)(i & 0xff)) & 0xff);
  }

// loadInv: punjenje tablice inv
  private void loadInv() {
    int index;
    for (int i = 0; i < 256; i++)
      inv[i] = (byte)(FFInv((byte)(i & 0xff)) & 0xff);
  }

// loadInvS: punjenje invS tablice koriscenjem S tablice
  private void loadInvS() {
    int index;
    for (int i = 0; i < 256; i++)
      invS[S[i] & 0xff] = (byte)i;
  }

// loadPowX: punjenje powX tablice množenjem polinoma
  private void loadPowX() {
    int index;
    byte x = (byte)0x02;
    byte xp = x;
    powX[0] = 1; powX[1] = x;
    for (int i = 2; i < 15; i++) {
      xp = FFMul(xp, x);
      powX[i] = xp;
    }
  }

// FFInv: multiplikativni inverz za bajt (novi bajt)
  public byte FFInv(byte b) {
  byte e = L[b & 0xff];
    return E[0xff - (e & 0xff)];
  }
```

```
// ithBit: vraćanje itog bita iz bajta
  public int ithBit(byte b, int i) {
    int m[] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
    return (b & m[i]) >> i;
  }

// subBytes: subBytes funkcija
  public int subBytes(byte b) {
    byte inB = b;
    int res = 0;
    if (b != 0)  // ako je b == 0, ne trazi inverz
      b = (byte)(FFInv(b) & 0xff);
    byte c = (byte)0x63;
    for (int i = 0; i < 8; i++) {
      int temp = 0;
      temp = ithBit(b, i) ^ ithBit(b, (i+4)%8) ^
             ithBit(b,(i+5)%8) ^ ithBit(b, (i+6)%8) ^
             ithBit(b, (i+7)%8) ^ ithBit(c, i);
      res = res | (temp << i);
    }
    return res;
  }
}
```

Operator temp>>i označava pomeranje promenljive temp udesno za "i" bitova. Slično za levu stranu funkcioniše operator <<.

Kao što smo rekli, prilikom kreiranja objekta se poziva metoda konstruktor i u nasem slučaju u okviru nje šest metoda: loadE, loadL, loadInv, loadS, loadInvS i loadPowX. Pomenute metode služe da napune šest nizova koji simuliraju tablice iz kojih čitamo vrednosti. U Javi ne postoji tip unsigned byte. To nam oduzima mogućnost da imamo bajtovne vrednosti u rasponu od 0 do 255. Zato koristimo tip int xor-ovan sa maskom 0xff. Pošto tip int zauzima 32 bita, ovo znači da samo njegovih 8 poslednjih bitova možda nisu nule, a upravo nam je tih 8 bitova potrebno za tip bajt. Dovoljno je na kraju uraditi konverziju u tip byte.

Metoda FFMul predstavlja sporiji način množenja dva polinoma u polju GF(256) korišćenjem pomeraja bitova. Polinom aa u svakoj iteraciji pomeramo u desno, što znači da će ukupno biti stepen od aa iteracija. Prilikom svakog pomeraja, ukoliko je poslednji bit od aa jedinica XOR-ovaćemo ostatak sa polinomom bb. Polinom bb množimo sukcesivno sa polinomom x(0x2), a to u stvari znači pomeraj za jedan bit u levo. Pre tog

pomeraja u promenjivoj temp pamtimo vrednost njegovog krajnjeg desnog bita. Ukoliko je on pre pomeraja bio jedan, polinom bb je bio sedmog stepena pa posle pomeraja postaje polinom osmog stepena, što onda moramo xorovati sa polinomom modulom 0x11b. Pošto radimo sa bajtovima, dovoljno je xorovati sa 0x1b. Operator & predstavlja operator konjukcije nad bitovima. Samo ako su oba bita 1 rezultat je 1. Kao sto je vec pomenuto Java ne podržava tip unsigned byte pa se za dobijanje vrednosti od 0 do 255 koristi maska 0xff primenjena na tip int i potom konvertovana u bajt. Inače, primena operacije & na tip bajt podrazumeva prethodno proširivanje u tip int.

Metoda FFMulFast služi za brzo množenje dva polinoma u polju GF(256) korišćenjem tablica za esponent i logaritam u vidu nizova E i L. Uz neprekidnu primenu maske 0xff na tip int i uz korišćenje osobine da se stepeni od 0x03 ponavljaju nakon 255 iteracija, dobija se traženi ostatak proizvoda po modulu 0x11b.

Metoda SBox prosto uzima odgovarajuću vrednost iz niza S. U njemu se nalaze rezultati operacija subBytes primenjeni na svakom bajtu.

Metoda invSBox uzima odgovarajuću vrednost iz niza invS u kome se nalaze inverzne vrednosti niza S.

Metoda Rcon sadrzi 16 članova – stepena polinoma 0x02.

Sledeća klasa, GetBytes služi za bajtovni prikaz parova heksadecimalnih cifara (heksadecimalna cifra zauzima četiri bita):

```
// GetBytes: stvaranje niza bajtova od heksadecimalnih cifara
import java.io.*;
public class GetBytes {
  private String fileName; // ulazno ime fajla
  private int arraySize; // broj bajtova za čitanje
  private Reader in;

  // GetBytes: konstruktor, otvara ulazni fajl
  public GetBytes(String file, int n) {
    fileName = file;
    arraySize = n;
    try {
      in = new FileReader(fileName);
    } catch (IOException e) {
```

```
      System.out.println("Exception opening " + fileName);
    }
  }

  // getNextChar: uzima sledeći karakter
  private char getNextChar() {
    char ch = ' ';
    // = ’ ’ da bi ch svakako bila inicijalizovana
    try {
      ch = (char)in.read();
    } catch (IOException e) {
      System.out.println("Exception reading character");
    }
    return ch;
  }

  // val: vraca int vrednost heksadecimalne cifre
  private int val(char ch) {
    if (ch >= '0' && ch <= '9') return ch - '0';
    if (ch >= 'a' && ch <= 'f') return ch - 'a' + 10;
    if (ch >= 'A' && ch <= 'F') return ch - 'A' + 10;
    return -1000000;
  }

  // getBytes: stvara niz bajtova od heksadecimalnih cifara
  public byte[] getBytes() {
    byte[] ret = new byte[arraySize];
    for (int i = 0; i < arraySize; i++) {
      char ch1 = getNextChar();
      char ch2 = getNextChar();
      ret[i] = (byte)(val(ch1)*16 + val(ch2));
    }
    return ret;
  }
}
```

Argumenti konstruktora klase su ime datoteke u kojoj se nalaze heksadecimalne cifre, bilo da je u pitanju osnovni tekst, šifrat ili ključ. Klasa Reader je abstraktna klasa, što znači da sve njene metode nemaju telo, pa se ne može praviti objekat ove klase, već je neophodno te metode definisati u nekoj od nasleđenih klasa. U nasleđenoj hijerarhiji se nalazi klasa FileReader koja služi za čitanje sadržaja datoteke. Ovde je, u stvari, demonstriran polimorfizam: referenca tipa osnovne klase pokazuje na objekat izvedene klase. Metoda getNextChar služi za uzimanje sledećeg karaktera dok metoda val vraća celobrojnu vrednost heksadecimalne cifre.

Metoda getBytes pretvara svaki par heksadecimalnih cifara u bajt, tako da sadržaj datoteke u heksadecimalnom zapisu postaje niz bajtova.

Sledi klasa Copy, namenjena kopiranju ulaznih podatka u matricu State i kopiranju izlaznih podataka iz nje:

```
// Copy: kopiranje niza bajtova
public class Copy {
  private static final int Nb = 4;

  // copy: kopiranje in u state
  public static void copy(byte[][] state, byte[] in) {
    int inLoc = 0;
    for (int c = 0; c < Nb; c++)
    for (int r = 0; r < 4; r++) state[r][c] = in[inLoc++];
  }

  // copy: kopiranje state u out
  public static void copy(byte[] out, byte[][] state) {
    int outLoc = 0;
    for (int c = 0; c < Nb; c++)
    for (int r = 0; r < 4; r++) out[outLoc++] = state[r][c];
  }
}
```

Sledi klasa Print namenjena za ispis bajtova u heksadecimalnom obliku:

```
// Print: ispis niza bajtova
public class Print {
  private static final int Nb = 4;
  private static String[] dig = {"0","1","2","3","4","5","6",
          "7","8","9","a","b","c","d","e","f"};

  // hex: ispis bajta u obliku dve heksadecimalne cifre
  public static String hex(byte a) {
    return dig[(a & 0xff) >> 4] + dig[a & 0x0f];
  }

  public static void printArray(String name, byte[] a) {
    System.out.print(name + " ");
    for (int i = 0; i < a.length; i++)
      System.out.print(hex(a[i]) + " ");
      System.out.println();
  }
```

```
  public static void printArray(String name, byte[][] s) {
    System.out.print(name + " ");
    for (int c = 0; c < Nb; c++)
    for (int r = 0; r < 4; r++)
      System.out.print(hex(s[r][c]) + " ");
    System.out.println();
  }
}
```

Metoda hex od leva četiri bita ulaznog bajta čita prvu heksadecimalnu cifru iz niza dig i slično za četiri desna bita drugu.

Svaka aplikacija u Javi mora posedovati javnu statičku funkciju main sa obaveznim argumentima: public static void main(String args[]). Može postojati više main funkcija, ali je neophodno da se, onaj koji želimo da bude važeći, nalazi u javnoj klasi koja se zove isto kao i ime programa – u našem slučaju AEStest.

```
// AEStest: test za AES šifrovanje
import java.io.*;
public class AEStest {
  public static void main(String[] args) {
  // za 128-bit ključ, koristi 16, 16 i 4
  // za 192-bit ključ, koristi 16, 24 i 6
  // za 256-bit ključ, koristi 16, 32 i 8
  System.out.println("Šifrovanje...");
  GetBytes getInput = new GetBytes("plaintext1.txt", 16);
  byte[] in = getInput.getBytes();
  GetBytes getKey = new GetBytes("key1.txt", 16);
  byte[] key = getKey.getBytes();
  AESencrypt aes = new AESencrypt(key, 4);
  Print.printArray("Plaintext: ", in);
  Print.printArray("Key: ", key);
  byte[] out = new byte[16];
  aes.Cipher(in, out);
  try {
    FileWriter upis = new FileWriter("ciphertext1.txt");
    for(inti=0;i<out.length;i++)
      upis.write(Print.hex(out[i]),0,2);
    upis.close();
  } catch(Exception e) {
    System.out.println("Neuspešan upis šifrata u datoteku");
  }
  Print.printArray("Ciphertext: ", out);
  System.out.println("\nDešifrovanje...");
  getInput = new GetBytes("ciphertext1.txt", 16);
```

```
    in = getInput.getBytes();
    AESdecrypt aesDec = new AESdecrypt(key, 4);
    Print.printArray("Ciphertext: ", in);
    Print.printArray("Key: ", key);
    aesDec.InvCipher(in, out);
    Print.printArray("Plaintext: ", out);
  }
```

Otvoreni tekst (engl. *plaintext*) u kriptografiji predstavlja ono što treba šifrovati i što se za neko vreme pretvara uz pomoć ključa (key) u šifrat (engl. *ciphertext*). Dobra praksa u programiranju podrazumeva odsustvo direktnih promena u programu dok je god to moguće: promene treba vršiti u spoljnim komponentama – u našem slučaju datotekama, u kojima se nalaze osnovni tekst, ključ i šifrat. Reč je o trima tekstualnim datotekama plaintext1.txt, ciphertext1.txt i key1.txt. Potrebno je da se nalaze u istom direktorijumu u kome se nalaze datoteke celog programa. U datoteku plaintext1.txt ćemo ručno upisati osnovni tekst koji želimo šifrovati. Takođe ćemo i ključ ručno upisati u datoteku key1.txt. Instanciraćemo klasu GetBytes, čiji je prvi argument ime datoteke, a drugi broj bajtova za čitanje. Respektivno ćemo uzeti heksadecimalne vrednosti iz pomenute dve datoteke i u obliku bajtova ih upisati u nizove bajtova in i key. U skladu sa željenom dužinom ključa unećemo drugi argument. Mi ćemo testirati slučaj 128-bitnog ključa. Dobijeni šifrat se nalazi u nizu out. Njega ćemo pomoću objekta upis klase FileWriter upisati u datoteku ciphertext1.txt. Upis se postiže funkcijom write koja ima tri parametra: String, koji upisujemo, startna pozicija sa koje upisujemo i broj karaktera koji upisujemo. Naš string se sastoji iz dva heksadecimalna podatka pa je startna pozicija – pocetak stringa, a broj karaktera dva. Naredbom close zatvaramo datoteku. Neophodno je da čitav ovaj odeljak upisivanja u ciphertext1.txt bude u try – catch bloku. U prevodu: ako se bilo koja greška desi u try bloku, obradi je u catch bloku adekvatnom radnjom ili porukom. Ovo zahteva Javin sistem izuzetaka.

Dešifovanje AESa nećemo posebno opisivati pošto bukvalno predstavlja inverznu operaciju šifrovanju. U dodatku B.4 naveden je ceo program, sa manjim izmenama (jedino je klasa u kojoj je main javna), preglednosti radi, u jednoj datoteci.

Implementacija SHA-1 algoritma u Javi

Svaku poruku, ma kolika ona bila, SHA-1 algoritam pretvara u 160

bitova. To je ekvivalentno konkatenaciji od pet 32-bitnih reči. Ideja SHA-1, kao i svakog heš algoritma, jeste da bilo koja promena na originalnoj poruci uzrokuje promenu i na sažetku, tj. hešu te poruke. Kod koji ćemo ukratko opisati naveden je u dodatku 2.4. SHA-1 je detaljno opisan u poglavlju 5.2. SHA1 fukcioniše tako što originalnu poruku deli u blokove od po 512 bitova. Ukoliko je poslednji blok manji od toga, dodaje mu se jedinica (bit), potreban broj nula bitova (padding) i poslednjih 64 bita koji binarno reprezentuju veličinu polazne poruke. Njena veličina zato ne sme prelaziti $2^{64} - 1$. Svaki blok se šifruje posebno. Blok ćemo podeliti na 16 novih blokova i time napuniti niz block čiji je jedan element reč (word) od 32 bita ($16 \cdot 32 = 512$).

Metoda update puni svaki niz block tako što dodaje bajtove u njega. Jedan poziv metode upisuje u niz jedan bajt. Dakle, da bi se napunio jedan član niza (32 bita) potrebna su 4 poziva funkcije update. Punjenje se dešava s desna na levo – što ne želimo: zato funkcija blk0 okreće redosled bajtova u reči – elementu niza block.

Za stvaranje sažetka SHA1 algoritam koristi dva niza od po pet reči (H0, H1, H2, H3, H4 i A, B, C, D, E) i jedan od 80 reči (W0, W1, ..., W79). U poslednjem nizu je moguće umesto 80 koristiti niz od samo 16 članova, ali je onda potrebno na indeks niza primeniti masku 0000000F koja će ograničiti njegovo kretanje od 0 do 15. Naš program podržava ovu drugu varijantu. U tom cilju metoda blk stvara nove reči kada brojač t uzima vrednosti od 16 do 79. Pre toga je funkcija blk0 stvorila niz W[0], W[1], ..., W[15] čime je stvorena osnova za dalje transformacije niza W.

```
public void update (byte b) {
  int mask = (8 * (blockIndex  & 3));
  count  += 8;
  block [blockIndex  >> 2] &= ~(0xff << mask);
  block [blockIndex  >> 2] |= (b & 0xff) << mask;
  blockIndex ++;
  if (blockIndex  == 64) {
    transform ();
    blockIndex  = 0;
  }
}
```

Za upotrebu finalnih metoda u Javi postoje dva razloga. Prvi je da se metoda "zaključa" da bi se sprečilo da bilo koja klasa naslednica promeni

njeno značenje. Drugi razlog je efikasnost: prevodiocu dozvoljavate da sve pozive toj metodi ugradi direktno u kod.

Finalne metode R0, R1, R2, R3, R4 menjaju niz W za nove vrednosti brojača i u skladu sa promenjivama K[i] koje za svakih 20 vrednosti brojača uzimaju novu vrednost. Njihovih 80 poziva sa različitim rasporedom argumenata u metodi transform ustvari odrađuju sve promene promenjivih A, B, C, D, E u algoritmu. Tome prethodi inicijalizacija niza dd nizom state koji je u metodi init inicijalizovan promenjivama H0, H1, H2, H3, H4. Na kraju, svaki član niza state se uvećava odgovarajućim članom niza dd.

Metoda init sem što inicijalizuje niz state, inicijalizuje i određene privatne atribute. Pošto funkcija update ubacuje po jedan bajt originalne poruke u niz block koji se potom s leva upisuje u niz W, brojač count beleži do kog smo bita u tome stigli. Drugim rečima on opisuje veličinu osnovne poruke u bitovima.

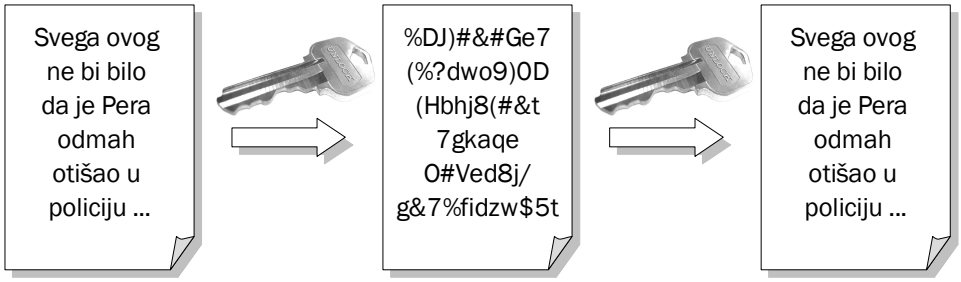
Niz digestBits je dužine 20 bajtova i upravo predstavlja 160-bitni sažetak. U njega upisujemo vrednosti niza state i to s leva na desno jer želimo da očuvamo težinsku vrednost bitova. Promenljiva blockIndex broji broj bajtova unetih u niz block (niz W) nad kojim se vrše dalje transformacije. Taj unos odrađuje metoda update čiji je argument jedan bajt. Preklopljena metoda update čiji je argument niz bajtova obrađuje taj niz višestrukim pozivanjem istoimene metode sa jednim argumentom.

2

Osnovni kriptografski pojmovi i klasična kriptografija

2.1 Osnovni kriptografski pojmovi

Šifrovanje (engl. *encryption*, slika 2.1) obuhvata matematičke postupke modifikacije podataka takve da šifrovane podatke mogu pročitati samo korisnici sa odgovarajućim ključem. Proces šifrovanja transformiše otvoreni tekst (engl. *plain text*) – originalnu poruku ili datoteku – pomoću ključa u zaštićen, šifrovan tekst, tj. šifrat (engl. *ciphertext*). Dešifrovanje (engl. *decryption*) je obrnut proces: šifrovani podaci se pomoću ključa transformišu u originalnu poruku ili datoteku. Šifrovani podaci su zaštićeni od neovlašćenog pristupa (korisnik bez odgovarajućeg ključa nema pristup šifrovanim podacima) i kao takvi se mogu preneti preko nesigurnog kanala ili čuvati na disku koji nije zaštićen od neovlašćenog pristupa.



Slika 2.1 *- Šifrovanje i dešifrovanje podataka*

Algoritam za šifrovanje može se smatrati sigurnim ukoliko sigurnost šifrata zavisi samo od tajnosti ključa, a ne i od tajnosti algoritma.

Kriptosistem se definiše kao uređena petorka (P, C, K, E, D), gde je:

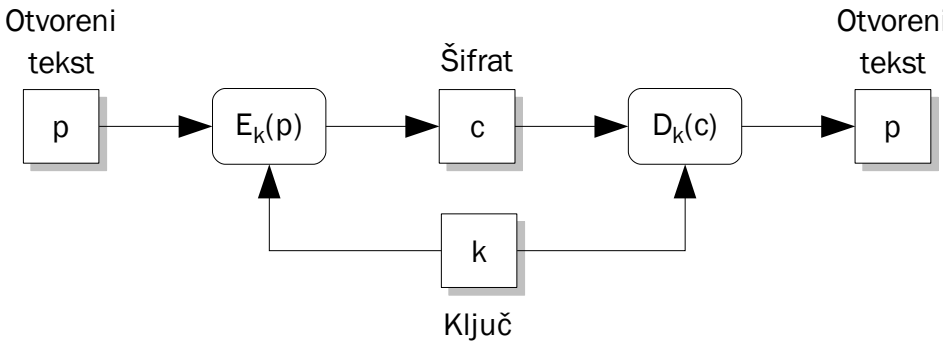
- P - skup poruka
- C - skup šifrata
- K - skup ključeva
- $E(P, K) \rightarrow C$ - funkcija šifrovanja
- $D(C,K) \rightarrow P$ - funkcija dešifrovanja

Algoritmi za šifrovanje se dele na simetrične (isti ključ se koristi i za šifrovanje i za dešifrovanje podataka) i algoritme sa javnim ključem (podaci se šifruju javnim ključem a dešifruju privatnim).

Funkcija šifrovanja simetričnim algoritmom *E* (slika 2.2) na osnovu ključa *k* i ulaznih podataka *p* proizvodi šifrat *c*. Funkcija dešifrovanja *D* na osnovu istog ključa *k* i šifrata *c* proizvodi originalnu poruku *p*:.

- $c = E(p,k)$
- $p = D(c,k)$

Simetrični algoritmi su brzi i kao takvi se mogu koristiti za šifrovanje većih datoteka ili implementaciju u kripto sisteme datoteka. Najpoznatiji simetrični kriptoalgoritmi su DES (*Data Encryption Standard*), AES (*Advanced Encryption Standard*), IDEA (*International Data Encryption Algorithm*), Blowfish i drugi.

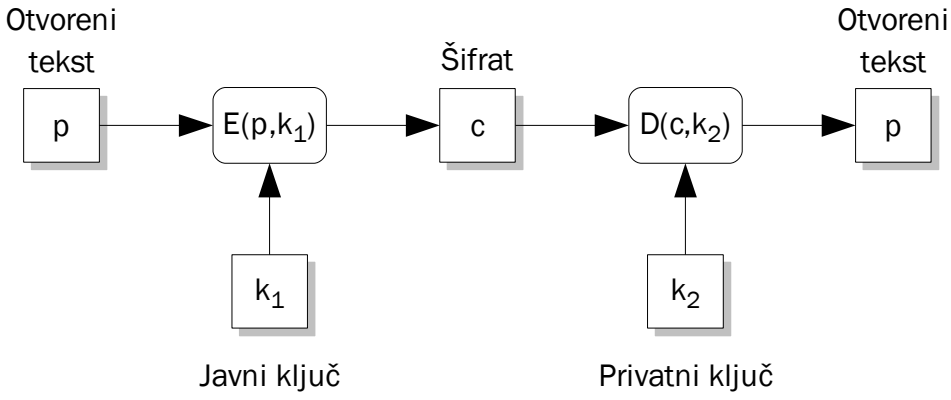


Slika 2.2 *– Simetrični algoritmi*

Funkcija šifrovanja algoritmom sa javnim ključem *E* na osnovu javnog ključa (engl. *public key*) k_1 i ulaznih podataka *p* proizvodi šifrat *c*. Funkcija dešifrovanja *D* na osnovu privatnog ključa (engl. *private key*) k_2 i šifrata *c* proizvodi originalnu poruku *p*.

- $c = E(p,k_1)$
- $p = D(c,k_2)$

Javni ključ je poznat onim osobama sa kojima korisnik želi da komunicira, dok je tajni ključ poznat samo korisniku koji je ovlašćen da dešifruje poruke. Privatni i javni ključ su matematički povezani, ali se privatni ključ ne može odrediti na osnovu javnog ključa. Asimetrični algoritmi su sporiji i primenjuju se za digitalno potpisivanje i šifrovanje ključeva simetričnih algoritama kojima su šifrovane datoteke. Najpoznatiji algoritam sa javnim ključem je RSA.



Slika 2.3 *– Algoritmi sa javnim ključem*

Digitalni potpis (engl. *digital signature*) je elektronska verziju potpisa, na osnovu kog se može identifikovati pošiljaoc i dokazati verodostojnost poruke. Digitalni potpisi usko su povezani sa pojmovima heš i jednosmerna heš funkcija. Jednosmerna heš funkcija na osnovu ulaznog podatka ma koje dužine proizvodi rezultujući niz tačno određene dužine – heš (engl. *hash*) koji, uslovno rečeno, jednoznačno identifikuje ulazni podatak. Pri tome se, zbog stroge jednosmernosti heš funkcije, originalni podaci ne mogu odrediti. Najčešće korišćene heš funkcije su MD5 (*Message Digest*) i SHA1 *(Secure Hash Algorithm)*. Prilikom potpisivanja, pošiljaoc najpre jednosmernom heš funkcijom računa heš h_1 poruke *p*, koju nakon toga potpisuje svojim privatnim ključem (uslovno se može shvatiti kao šifrovanje privatnim ključem).

- $h_1 = H(p)$, $s_1 = E(k_1,h_1)$

Pošiljaoc šalje originalnu poruku i digitalni potpis primaocu. Primalac

određuje heš h_2 primljene poruke i proverava primljeni potpis s_1 javnim ključem pošiljaoca (uslovno se može shvatiti kao dešifrovanje javnim ključem).

- $h_2 = H(p)$, $h_1 = D(k_2, s_1)$

Upoređivanjem vrednosti h_1 i h_2 proverava se identitiet pošiljaoca.

Napadi na šifrate

Cilj napada na šifrat je otkrivanje otvorenog teksta, ili, još češće ključa kojim je otvoreni tekst šifrovan. Osnovna pretpostavka kriptoanalize je da kriptoanalitičar zna koji se kriptosistem koristi (Kerckhoffsov princip). Naravno, ova pretpostavka u konkretnom slučaju ne mora biti tačna, ali se složenost procedure bitno ne menja čak i ako kriptoanalitičar treba proveriti nekoliko mogućih kriptosistema. Dakle, mi pretpostavljamo da tajnost šifrata u potpunosti leži u ključu. Napadi se mogu klasifikovati u sledeće kategorije:

- Samo šifrat (engl. *ciphertext-only attack*). Kriptoanalitičar poseduje samo šifrate nekoliko poruka šifrovanih pomoću istog algoritma. Njegov je zadatak da otkrije otvoreni tekst što većeg broja poruka ili u najboljem slučaju da otkrije ključ kojim su poruke šifrovane.

- Poznat otvoreni tekst (engl. *known-plaintext attack*). Kriptoanalitičar poseduje šifrat neke poruke i njemu odgovarajući otvoreni tekst. Njegov je zadatak da otkrije ključ ili neki algoritam za dešifrovanje poruka šifrovanih tim ključem.

- Odabran otvoreni tekst (engl. *chosen-plaintext attack*). Kriptoanalitičar je dobio privremeni pristup alatu za šifrovanje, tako da može dobiti šifrat odabranog otvorenog teksta. Ovaj napad je jači od prethodnog.

- Odabrani šifrat (engl. *chosen-ciphertext attack*). Kriptoanalitičar je dobio pristup alatu za dešifrovanje, tako da može dobiti otvoreni tekst odabranog šifrata Ovo je tipičan napad na kriptosisteme sa javnim ključem.

- Potkupljivanje, ucena, krađa i slične aktivnosti (engl. *rubber-hose attack*). Ovaj napad ne spada u matematičke oblike kriptoanalize, ali je vrlo efikasan i često se upotrebljava.

2.2 Klasična kriptografija i kriptoanaliza

Osnovu većine algoritama za šifrovanje čine matematički postupci supstitucije (zamene) i transpozicije (permutacije). Supstitucija je zamena delova originalne poruke (pojedinačnih karaktera ili grupa karaktera konstantne dužine) drugim karakterima ili rezultatom neke funkcije čiji su ulaz ti karakteri i ključ (npr. TAJNA $\rightarrow$ XIWOI). Permutacijom se originalna poruka preuređuje po nekom algoritmu (npr. TAJNA $\rightarrow$ JANAT). Kombinovanjem ovih postupaka ostvaruje se visok stepen zaštite podataka.

Šifrovanje supstitucijom obuhvata sledeće vrste zamena:

- Monoalfabetska zamena (Cezarova i Afina šifra). Bijektivno preslikavanje: svaki karakter poruke se preslikava u tačno jedan karakter šifrata.

- Polialfabetska zamena (Vigenèreova i Playfairova šifra). Preslikavanje 1-n: svaki karakter poruke može se preslikati u jedan od n dozvoljenih karaktera šifrata, zavisno od algoritma koji se koristi i od dužine ključa.

- Poligramska zamena (Playfairova i Hillova šifra). Bijektivno preslikavanje, pri čemu se kao osnovna jedinica nad kojom se izvršava supstitucija uzima poligram (niz od više karaktera).

Cezarova šifra

Svako slovo poruke se u šifratu menja trećim karakterom udesno po modulu *n* (moduo *n* je određen brojem karaktera u alfabetu, u primerima zbog jednostavnosti pretpostavljamo engleski alfabet, pa je n=26). Tako se, na primer, slovo A menja slovom D, slovo B slovom E, slovo V slovom Y, a slovo Z slovom C, kao što je prikazano sledećom tablicom. Poruke šifrovane

na ovaj način Cezar je svojim prijateljima i vojskovođama slao po kuriru u kog nije imao poverenja.

Otvoreni tekst:	A	B	C	D	...	W	X	Y	Z
Šifrat:	D	E	F	G	...	Z	A	B	C

Na primer, E (VENI VIDI VICI) = YHQL YLGL YLFL.

Danas se Cezarovom šifrom nazivaju i šifre istog oblika s pomerajem različitim od 3. Definisaćemo operacije šifrovanja i dešifrovanja nad alfabetom sa *n* karaktera. Neka je $P=C=K=Z_n$. Za ključ *k* takav da je $0 \leq k \leq n$ važi:

- $e_k(x) = (x+k) \bmod n$
- $d_k(y) = (y-k) \bmod n$.

Promenljive *x* i *y* su numerički ekvivalenti odgovarajućih slova. Korespondencija slova engleskog alfabeta (Z_{26}) i njihovih numeričkih vrednosti prikazana je sledećom tabelom:

Slovo:	A	B	C	D	...	W	X	Y	Z
Numerički ekvivalent:	0	1	2	3	...	22	23	24	25

Važi $d_k(e_k(x))=x$, kao što se zahteva u definiciji kriptosistema. Za k=3 dobija se originalna Cezarova šifra. Za k=13 i n = 26 dobija se ROT13 mehanizam, za koji je karakteristično da se šifrovanje i dešifrovanje obavlja identičnom funkcijom u slučaju engleskog alfabeta: P=ROT13(ROT13(P)).

U dodatku B naveden je jednostavan program u programskom jeziku Java koji demonstrira upotrebu Cezarove šifre (izvorni kod 2.1).

Kriptoanaliza Cezarove šifre je krajnje jednostavna – potrebno je metodom "grube sile" (engl. *brutal force*) ispitati sve moguće ključeve, dok se ne dodje do nekog smislenog teksta. U najgorem mogućem slučaju, potrebno je ispitati onoliko ključeva koliko u datom alfabetu ima karaktera.

Primer 2.1. Dešifrovati šifrat OVMTXSKVEJMNE dobijen Cezarovom šifrom, ukoliko je poznato da originalna poruka i šifrat pripadaju engleskom alfabetu.

d(y, k=1) = NULSWRJUDILMD,

d(y, k=2) = MTKRVQITCHKLC

d(y, k=3) = LSJQUPHSBGJKB,

d(y, k=4) = KRIPTOGRAFIJA

Dakle, ključ je k = 4, a otvoreni tekst je KRIPTOGRAFIJA.

Afina šifra

Osnovni nedostatak Cezarove šifre je mali prostor ključeva, odnosno relativno mali broj ključeva koji je potrebno ispitati prilikom kriptoanalize. Afina šifra donekle rešava ovaj problem (za šifrovanje nad engleskim alfabetom na raspolaganju je 312 ključeva). Definisaćemo operacije šifrovanja i dešifrovanja nad engleskim alfabetom. Neka je $P=C=Z_{26}$ i $K=\{(a,b) \mid (a,b)\in Z_{26}xZ_{26}\}$, pri čemu *a* mora imati multiplikativni inverz a^{-1} u prstenu Z_{26}. Za ključ $k\in K$ važi:

- $e_k(x) = (ax+b) \bmod 26$
- $d_k(y) = a^{-1}(y-b) \bmod 26$

Važi $d_k(e_k(x)) = d_k(ax+b) = a^{-1}((ax+b)-b) = x$, kao što se zahteva u definiciji kriptosistema.

Z_{26} nije polje (jer 26 nije prost), tako da u Z_{26} postoje elementi koji nemaju multiplikativni inverz. Ukupno ima 12 inverzibilnih elemenata. Parovi inverznih elemenata (a, a^{-1}) su dati u sledećoj tabeli:

a	1	3	5	7	11	17	25
a^{-1}	1	9	21	15	19	23	25

Primer 2.2. Šifrovati reč "ZARADA", ako je k=(7,3), a otvoreni tekst pripada engleskom alfabetu.

Primenjujemo $e_k(x) = (ax+b) \bmod 26$ nad numeričkim ekvivalentima svakog slova:

Z: $25 \cdot 7+3=22 \pmod{26}$,

A: $0 \cdot 7+3=3 \pmod{26}$,

R: $17 \cdot 7+3=18 \pmod{26}$,

D: $3 \cdot 7+3=24 \pmod{26}$,

A: $0 \cdot 7+3=3 \pmod{26}$.

Šifrat je: WDSDYD.

Afina šifra ima prostor ključeva od 12*26=312 mogućih ključeva, tako da se na računaru *brutal force* napad može izvesti za relativno kratko vreme. Elegantniji način je analiza učestalosti pojavljivanja slova u govornom jeziku. Broji se pojavljivanje svakog slova u šifratu, a zatim se distribucija slova u šifratu upoređuje sa poznatim podacima o distribuciji slova u jeziku otvorenog teksta. Vrlo je verovatno da slova koja se najčešće pojavljuju u šifratu odgovaraju najfrekventnijim slovima jezika, a verovatnoća raste sa dužinom šifrata. Takođe, podaci o najčešćim bigramima (parovima slova) i trigramima (nizovima od tri slova) u jeziku mogu biti korisni.

Primer 2.3. Dešifrovati šifrat OZWHR YEZCV WFC dobijen afinom šifrom. Otvorena poruka je na srpskom jeziku, a sva slova pripadaju engleskom alfabetu.

Najfrekventnija slova u srpskom jeziku su redom A, I, O, E i N. U našem šifratu uočavamo da su najfrekventnija slova C i W, koja se javljaju po dva puta. Iako je naš šifrat prekratak, možemo očekivati da su ta dva slova šifrati od A, I, O, E ili N. Pretpostavimo da su slova C i W šifrati slova A i I, odnosno da je $e_k(A) = 2$ i $e_k(I) = 22$.

Šifrati slova A i I dobijaju se na sledeći način:

- $e_k(A) = a \cdot 0 + b = b$,

- $e_k(I) = 8 \cdot a + b$.

Dobija se sistem jednačina: b = 2, 8a+b=22 (mod 26), odakle je a=9. Dakle, (a,b) = (9,2) $\rightarrow a^{-1}=3$, pa je odgovarajuća funkcija dešifrovanja:

- $d_{(9,2)}(y) = 3(y-2) \bmod 26$.

Primenom funkcije $d_k(x)$ na šifrat dobija se otvoreni tekst KRIPTOGRAFIJA.

Vigenèreova šifra

Vigenèreova šifra (Blaise de Vigenère) spada u polialfabetske kriptosisteme. Svako slovo otvorenog teksta može se preslikati u jedno od *m* mogućih slova, gde je *m* dužina ključa. Neka je $P=C=K=(Z_{26})^m$, a *m* fiksni prirodan broj. Za ključ $k=(k_1, k_2, \ldots, k_m)$ operacije šifrovanja i dešifrovanja definišu se na sledeći način:

- $e_k(x_1, x_2, \ldots, x_m) = (x_1 + k_1, x_2 + k_2, \ldots, x_m + k_m)$
- $d_k(y_1, y_2, \ldots, y_m) = (y_1 - k_1, y_2 - k_2, \ldots, y_m - k_m)$

Operacije sabiranja i oduzimanja obavljaju se u Z_{26}, tj. po modulu 26.

Primer 2.4. Šifrovati reč VIGENER ako je m = 4, a ključna reč ABYZ.

Numerički ekvivalent ključne reči je ključ k=(0,1,24,25). Šifrovanje se izvodi na sledeći način:

otv. tekst	V (21)	I (8)	G (6)	E (4)	N (13)	E (4)	R (17)	E (4)
ključ	A (0)	B (1)	Y (24)	Z (25)	A (0)	B (1)	Y (24)	Z (25)
šifrat	V (21)	J (9)	E (4)	D (3)	N (13)	F (5)	O (15)	D (3)

Ukoliko se šifrovanje obavlja bez računara, koristiti se Vigenèreov kvadrat (tablica A.1 u dodatku A). Ako slovo I treba šifrovati ključem B, onda se šifrat J nalazi u preseku kolone koja počinje slovom I i reda koji počinje slovom B.

Za razliku od Cezarove i afine šifre, broj mogućih ključeva koje treba ispitati prilikom kriptoanalize Vigenèreove šifre je znatno veći (u opštem slučaju $n+n^1+n^2+...+n^m$, gde je *n* broj slova u alfabetu, a *m* maksimalna dužina ključa). Za engleski alfabet i maksimalnu dužinu ključa od 5 karaktera ispituje se najviše $26+26^2+26^3+26^4+26^5=12.356.630$ ključeva. Broj ključeva približno eksponencijalno raste sa dužinom ključa, tako da je *brutal force* metoda kriptoanalize nepraktična za Vigenèreove šifre. Takođe, jednostavna analiza učestalosti pojedinih slova u šifratu nije od posebne koristi (videti primer 4 - prvo slovo E se preslikalo u D, a drugo u F).

Prilikom kriptoanalize [3] Vigenèreove šifre najpre se određuje dužina ključa pomoću Kasiskijevog testa (Friedrich Kasiski, 1863) ili pomoću metode indeksa koincidencije (William Friedman, 1920).

Kasiskijeva metoda se se zasniva na činjenici da će dva identična segmenta otvorenog teksta biti šifrovana na isti način ukoliko se njihove početne pozicije razlikuju za neki umnožak dužine ključa m. Obrnuto, ako uočimo dva identična segmenta u šifratu, dužine barem 3, tada je vrlo verovatno da oni odgovaraju identičnim segmentima otvorenog teksta. Test se svodi na traženje takvih segmenata u šifratu, i određivanja udaljenosti između njihovih početnih položaja (udaljenost se računa u odnosu na početak prvog segmenta). Pretpostavlja se da dužina ključa deli ove udaljenosti, tj. za dužinu ključa se uzima njihov najveći zajednički delilac.

Indeks koincidencije *Ic(x)* niza $x = x_1x_2 \dots x_n$ od *n* slova definiše se kao verovatnoća da su dva slučajno izabrana elementa niza *x* jednaka. Dva elementa niza *x* možemo odabrati na $^{n(n-1)}/_2$ načina. Neka su $f_0, f_1, \dots, f_{25}$ redom apsolutne frekvencije, tj. broj pojavljivanja slova A, B, C, ... , Z u nizu *x*. Za svako i=0,1,...,25 postoji $f_i(f_i-1)/2$ načina odabiranja dvaput i-tog slova. Indeks koincidencije niza određuje se sledećom formulom:

- $Ic(x) = \Sigma_i \{ f_i(f_i-1)) / n(n-1) \}$.

Neka je *x* neki tekst na engleskom jeziku. Očekivane verovatnoće (relativne frekvencije) pojavljivanja slova A, B, ... , Z označene su redom sa $p_0, p_1, \dots, p_{25}$. U skladu s tablicom frekvencija za engleski jezik za p_i se uzimaju odgovarajuće vrednosti; alternativno, može se uzeti neki malo veci

3 Više informacija o kriptoanalizi Vigenèreove šifre čitaoci mogu naći na web stranici http://www.math.hr/~duje/kript.html (Kriptografija, PMF, Sveučilište u Zagrebu).

tekst na engleskom jeziku i na osnovu njega se mogu odrediti relativne frekvencije. Uzevši u obzir da je verovatnoća su slučajno odabrana dva slova A jednaka p_0^2, slučajno odabrana dva slova B jednaka p_1^2, indeks koincidencije teksta *x* može se odrediti formulom:

- $Ic(x) = \Sigma_i\ p_i^2 \approx 0{,}065$ (tipična vrednost indeksa koincidencije za engleski jezik).

Slično, može se odrediti indeks koincidencije slučajnog teksta:

- $Ic = \Sigma_i\ (1/26)^2 = 26(1/26)^2 \approx 0{,}038$.

Pretpostavimo da kriptoanalitičar raspolaže šifratom teksta $y = y_1y_2 \ldots y_n$ koji je dobiven Vigenèreovom šifrom. Šifrat *y* se rastavlja na m podnizova z_1, z_2, ... , z_m tako što se *y* upisuje po kolonama, u matricu dimenzija m*(n/m). Redovi ove matrice su upravo traženi podnizovi z_1, z_2, ... , z_m. Ako je *m* prava dužina ključne reči, onda su indeksi koincidencije podnizova približno jednaki 0,065. Ako *m* nije dužina ključne reči, indeksi podnizova $Ic(z_i)$ odgovaraju indeksima koincidencije slučajnog teksta.

Nakon određivanja dužine, korišćenjem međusobnog indeksa koincidencije dva niza traži se i ključna reč. Sledeći postupak će suziti prostor pretrage na 26 potencijalnih ključnih reči. Neka su $x = x_1x_2 \ldots x_n$ i $y = y_1y_2 \ldots y_{n'}$ dva niza od *n*, odnosno *n'* slova. Međusobni indeks koincidencije $MIc(x,y)$ nizova *x* i *y*, definiše se kao verovatnoća da je slučajno izabrani element niza *x* jednak slučajno izabranom elementu niza *y*. Ako apsolutne frekvencije pojavljivanja slova A, B, C, ... , Z u nizovima *x* i *y* označimo sa f_0, f_1, ... , f_{15}, i f'_0, f'_1, ... , f'_{15}, onda je:

- $MIc(x,y) = \Sigma_i\ f_i\ f'_i\ /\ nn'$.

Posmatrajmo podnizove z_1, z_2, ... , z_m dobijene prilikom traženja dužine ključa. Pretpostavlja se ključna reč $k = (k_1, k_2, \ldots, k_m)$ i na osnovu nje se vrši procena međusobnog indeksa koincidencije $MIc(z_i,z_j)$ nizova z_i i z_j. Verovatnoća da su proizvoljna slova nizova z_i i z_j jednaki A iznosi $P_{0\text{-}ki}P_{0\text{-}kj}$, da su jednaki B iznosi $P_{1\text{-}ki}P_{1\text{-}kj}$, tj.

- $MIc(z_i, z_j) \approx \Sigma_h\ p_{h\text{-}ki}\ p_{h\text{-}kj} = \Sigma_h\ p_h\ p_{h+ki\text{-}kj}$.

Operacije u indeksima izvode se po modulu 26. Procena zavisi samo od razlike d = k_i - k_j mod 26 (pomeraj između nizova z_i i z_j). Takođe, važi i sledeća jednakost: $\Sigma_h\ p_h\ p_{h-d} = \Sigma_h\ p_h\ p_{h+d}$, što znači su ocene za pomeraj d i pomeraj 26-d jednake. Zato je dovoljno posmatrati pomeraje između 0 i 13. Za d = 0, ocena je 0,065, a za ostale vrednosti pripada intervalu 0,032 i 0,045.

Pretpostavimo da je z_i fiksirani podniz. Posmatra se efekat šifrovanja niza *zj* redom sa A, B, C, ... , pri čemu se dobijaju podnizovi z_j^0, z_j^1, ... , z_j^{25}. Za svako g = 0,1,...,25 računa se međusobni indeks koincidencije $MIc(z_i, z_j^g)$ po formuli: MIc(x, y^g) = $\Sigma_i\ f_i f'_{i-g}$ / nn'. Za g=d, *MIc* bi trebalo da bude približno jednak 0,065, a za g≠d varira između 0.032 i 0.045. Na ovaj način, određuju se pomeraji bilo koja dva podniza z_i i z_j. Nakon toga ostaje 26 mogućih ključnih reči koje se lako mogu ispitati.

Jednostavan XOR algoritam

Binarna operacija ekskluzivno ILI (eXclusive OR) definiše se na sledeći način: a ⊕ b = 0, ako je a = b; a ⊕ b = 1, ako je a ≠ b.

⊕	0	1
0	0	1
1	1	0

Osnovna osobina XOR operacije, koja omogućava njenu upotrebu u kriptografiji je: a ⊕ a = 0, tj. a ⊕ b ⊕ b = a. To znači da se šifrat može dobiti kao rezultat XOR operacije izvršene bit-po-bit nad otvorenim tekstom i ključem, a otvoreni tekst kao rezultat XOR operacije izvršene bit-po-bit nad šifratom i ključem. Jednostavan XOR algoritam je bitski orijentisana varijanta Vigenèreove polialfabetske šifre. Algoritam je simetričan, a ista operacija (odnosno isti kod) koristi se i za šifrovanje (P ⊕ K = C) i dešifrovanje (C ⊕ K = P).

U jednom periodu, XOR algoritam je masovno korišćen na MS-DOS i Macintosh sistemima – svaki "*fast proprietary*" metod šifrovanja značajno brži od DES algoritma u komercijalnom softveru je varijanta jednostavnog

XOR algoritma. Pri tome su proizvođači softvera svoje kupce ubeđivali da je algoritam "siguran gotovo kao i DES !"

Zanimljiva je činjenica da je Američka agencija za bezbednost (*National Security Agency*) dozvolila upotrebu ovog algoritma sa 160-bitnim ključem u mobilnoj telefoniji.

U dodatku B naveden je jednostavan program u programskom jeziku C++ koji demonstrira upotrebu jednostavnog XOR algoritma za šifrovanje datoteka (izvorni kod 2.2).

Slično klasičnoj Vigenèreove šifri, kriptoanaliza XOR algoritma obavlja se u dva koraka: određivanje dužine ključa i pronalaženje samog ključa. Za određivanje dužine ključa koristi se procedura prebrojavanja koincidencije[4]. Za j=1,2,3,...n bajtova računa se vrednost XOR operacije šifrata sa samim sobom pomerenim za j bajtova udesno i prebrojavaju se jednaki bajtovi. Ukoliko je j umnožak veličine ključa, jednako je približno 6% bajtova (pod pretpostavnkom da je ASCII tekst šifrovan slučajno generisanim ključem). U suprotnom, jednako je manje od 0.5% bajtova. Najmanja vrednost j koja ukazuje na umnožak dužine ključa je dužina ključa. Dalje se određuje vrednost XOR operacije šifrata sa šifratom pomerenim za dužinu ključa. Ključ se zbog cikličnosti potire (kao posledica asocijativnosti XOR operacije) i kao rezultat ostaje XOR otvorenog teksta sa otvorenim tekstom pomerenim za dužinu ključa. Dalje se na osnovu osobina otvorenog teksta (količina informacije po bajtu podataka) određuje otvoreni tekst.

Jednokratna beležnica

Jednokratna beležnica (engl. *one-time pad*) može se posmatrati kao varijanta XOR algoritma kod kog je dužina ključa jednaka dužini otvorenog teksta. Ključ i otvoreni tekst se primaocu šalju različitim kanalima. Šifrat je siguran, a poruku je nemoguće ispravno dešifrovati bez pravog ključa (pomoću različitih kjučeva mogu se dobiti različite poruke, od kojih neke imaju smisla, ali se prava poruka dobija samo pomoću pravog ključa). Kriptosistem se koristio u komunikaciji između Moskve i Vašingtona tokom hladnog rata. Sigurnost XOR algoritma se značajno uvećava ukoliko se svaki

4 Detaljno je opisana u knjizi W.F. Friedmana "*The Index of Coincidence and It's Applications in Cryptography*".

put koristi različiti ključ (time se sprečava mogućnost da treće lice dođe do ključa i dešifruje neke poruke).

Playfairova šifra

Playfairova šifra (Charles Wheatstone, ime je dobila po njegovom prijatelju Baronu Playfairu) je polialfabetska bigramska šifra - šifruju se parovi slova i to tako da rezultat zavisi i od jednog i od drugog slova. Ukoliko otvoreni tekst pripada engleskom jeziku, algoritam za šifrovranje koristi matricu veličine 5x5 slova, koja se konstruiše koristeći ključnu reč. Pošto matrica ima 25 elemenata, a engleski jezik 26 slova, dogovor je da se slova I i J poistovete.

Na primer, za ključnu reč PLAYFAIR, matrica će imati sledeći oblik:

P	L	A	Y	F
IJ	R	B	C	D
E	G	H	K	M
N	O	Q	S	T
U	V	W	X	Z

Prilikom šifrovanja, otvoreni tekst se deli na blokove od po dva slova. Ni jedan blok se ne sme sastojati od dva jednaka slova, a dužina teksta mora biti parna. Ukoliko to nije slučaj, u tekst se umeću slova po dogovoru (npr. slova X). Kod šifrovanja bloka od dva slova, mogu nastupiti tri slučaja:

- Slova koja se nalaze u istom redu menjaju se slovima koja se nalaze jedno mesto udesno (ciklički). Na primeru prethodne matrice, EH->GK, ST->TN, FP->PL.

- Slova koja se nalaze u istoj koloni menjaju se slovima koja se nalaze jedno mesto ispod (ciklički). Na primeru prethodne matrice, OV->VL, KY->SC, PI->IE.

- Inače, određuje se pravougaonik koji ta dva slova formiraju. Slova se menjaju slovima koja se nalaze u preostala dva vrha tog

pravougaonika. Redosled je određen tako da najpre dođe ono slovo koje se nalazi u istom redu kao prvo slovo u polaznom bloku. Na primeru prethodne matrice, OC->SR, RK->CG, PD->FI.

Primer 2.5. Šifrovati reč PROTECTION Playfaiovom šifrom ako je ključna reč PLAYFAIR, a otvoreni tekst pripada engleskom jeziku.

E_k (PR) = LI, E_k (OT) = QN , E_k (EC) = JK , E_k (TI) = ND , E_k (ON) = QO.

Šifrat je: LIQNJKNDQO.

Osnovna prednost Playfairove šifre u odnosu na monoalfabetsko supstitucijsko šifrovanje je to što je šifra bigramska, tako da se u šifratu gube jednoslovne reči (npr. "a") koje prilično utiču na frekvencije. Broj bigrama je veći od broja individualnih slova (26 slova - 676 bigrama), a frekvencije najfrekventnijih bigrama znatno su ujednačenije od frekvencija najfrekventnijih slova. Zato je Playfairova šifra dugo vremena smatrana sigurnom i kao takva je korišćena u britanskoj vojsci za vreme I svetskog rata. Ipak, kod dugih tekstova ova šifra postaje nesigurna jer se može iskoristiti analiza frekvencija bigrama.

Ukoliko je šifrat prekratak za dešifrovanje samo analizom frekvencija bigrama, koristi se metoda verovatne reči (pretpostavke da je kriptoanalitičaru poznata jedna reč koja se pojavljuje u otvorenom tekstu). Metoda se sastoji u tome da se napravi lista takvih reči ili fraza i da u šifratu pronađemo segment čija se struktura podudara sa strukturom verovatne riječi. Na osnovu analize frekvencije bigrama i/ili metode verovatne (ili poznate) reči rekonstruiše se matrica šifrovanja i određuje ključ.

Hillova šifra

Hillova šifra (Lester Hill) je poligramska šifra kod koje se m uzastopnih slova otvorenog teksta zamenjuje sa m slova u šifratu. Neka je *m* fiksni prirodan broj, $P=C=(Z_{26})^m$, a K={m×m}, gde je m×m invertibilna matrica nad Z_{26}. Za ključ $k \in K$ operacije šifrovanja i dešifrovanja definišu se na sledeći način:

- $e_k(x) = xk$,

- $d_k(y) = yk^{-1}$, gdje su sve operacije u Z_{26}.

Primer 2.6. Šifrovati otvoreni tekst CIPHER ključem k = $\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 4 \\ 3 & 2 & 1 \end{vmatrix}$.

Otvoreni tekst pripada engleskom jeziku. To znači da se matrični račun izvodi po modulu 26. Numerički ekvivalent otvorenog teksta je 2, 8, 15, 7, 4, 17. Pošto je ključ matrica tipa 3x3, šifrovanje zahteva dve operacije množenja matrica:

CIP: $\begin{vmatrix} 2 & 8 & 15 \end{vmatrix} \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 4 \\ 3 & 2 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 22 & 1 \end{vmatrix}$ =BWB.

HER: $\begin{vmatrix} 7 & 4 & 17 \end{vmatrix} \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 4 \\ 3 & 2 & 1 \end{vmatrix} = \begin{vmatrix} 22 & 16 & 2 \end{vmatrix}$ = WQB.

Šifrat je BWBWQB.

Hillov kriptosistem za ključ 3x3 skriva ne samo sve informacije o frekvencijama slova, već i o frekvencijama bigrama. Već za m=5 Hillov kriptosistem se može smatrati gotovo sigurnim na napad samo šifrat. Međutim, ovu šifru je relativno lako razbiti pomoću napada poznat otvoreni tekst, a pogotovo pomoću napada odabran otvoreni tekst.

Šifrovanje transpozicijom

Ideja transpozicijske šifre je da se ne menjaju elementi otvorenog teksta već njihov raspored. U praksi je najčešće korišćena transpozicija po kolonama. Otvoreni tekst upisuje u se u pravougaonik, a zatim se poruka čita po kolonama, ali s promenjenim poretkom kolona. Ako se poslednji red ne ispuni do kraja, onda se prazna mesta popune proizvoljnim slovima (nulama) koja ne menjaju sadržaj poruke.

Primer 2.7. Šifrovati otvoreni tekst "KRIPTOGRAFIJA I KRIPTOANALIZA"

transpozicijom po kolonama sa ključem 4312576.

4	3	1	2	5	7	6
K	R	I	P	T	O	G
R	A	F	I	J	A	I
K	R	I	P	T	O	A
N	A	L	I	Z	A	X

Šifrat: IFIL PIPI RARA KRKN TJTZ GIAX OAOA.

2.3 Pitanja i zadaci

2.1 Na programskom jeziku C, C++ ili Java sastavite program koji šifruje, odnosno dešifruje tekstualnu datoteku sa diska Afinom šifrom.

Prvi parametar koji korisnik prenosi programu je šta želi da radi: da šifruje ili da dešifruje datoteku. U slučaju da se obavlja šifrovanje, korisnik kao ostale parametre unosi ime datoteke u kojoj se nalazi otvoreni tekst, ime datoteke u kojoj će biti smešten šifrat i ključ. Pretpostaviti da ulazni tekst pripada engleskom alfabetu. Osobine šifrovanog teksta su sledeće: interpunkcijski znaci su zadržani u obliku u kom se nalaze u otvorenom tekstu (ne šifruju se), šifrati velikih slova su velika slova, a malih slova su mala slova.

U slučaju da se obavlja dešifrovanje, korisnik kao parametre unosi ime datoteke u kojoj se nalazi šifrat, ime datoteke u kojoj će biti smešten otvoreni tekst i ključ. Program proverava da li je ključ ispravan, tj. da li se može upotrebiti za šifrovanje, odnosno dešifrovanje (proveriti da li postoji multiplikativni inverz parametra a u ključu).

2.2 Koristeći program iz prethodnog zadatka, šifrujte jedan tekst na engleskom jeziku dužine oko 250 karaktera, koji u sebi sadrži reč “System”.

Pretpostavite da ste vi kriptoanalitičar koji je došao do šifrata i na neki način saznao da se u odgovarajućem otvorenom tekstu nalazi ta reč. Sastavite program koji obavlja kriptoanalizu šifrata *brutal-force*

metodom i određuje i upisuje u datoteku sve moguće potencijalne otvorene tekstove koji u sebi sadrže tu reč i odgovarajuće ključeve.

2.3 Koristeći program iz zadatka 2.1 šifrujte jedan tekst na engleskom jeziku dužine oko 1000 karaktera.

Pretpostavite da ste vi kriptoanalitičar koji je došao do šifrata i da znate da je otvoreni tekst na engleskom jeziku, ali da nemate uzorke tog teksta. Sastavite program koji obavlja kriptoanalizu šifrata metodom frekvencijske analize slova u šifratu.

Obratite pažnju na to da je prilikom određivanja broja pojavljivanja nekog slova potrebno uzeti u obzir broj pojavljivanja kako malih tako i velikih slova (na primer, ako se malo slovo “a” u tekstu pojavilo tri puta, a veliko slovo “A” dva puta, onda se slovo “A” ukupno pojavilo pet puta). Relativne frekvencije najčešćih slova (u promilima) za engleski jezik su: E (127), T (91), A (82), O (75), I (70), N (67), S (63), H (61), R (60).

2.4 Izmenite program iz zadatka 2.1 tako da se šifrovanje obavlja Vigenèreovom šifrom.

2.5. Prethodnim programom šifrujte engleski tekst dužine 1000 karaktera bez interpunkcijskih znakova. Sastavite program koji pomoću Kasiskijevog testa određuje dužinu ključa.

2.6 Sastavite program koji određuje relativne frekvencije karaktera i indekse koincidencije. Program kao ulazni parametar prihvata ime datoteke u kojoj se nalazi tekst, a u istoimenu datoteku sa ekstenzijom *.ind* smešta rezultat u tabelarnom obliku.

Pomoću tog programa odredite indekse koincidencije:

(a) slučajno generisanog teksta,

(b) teksta na engleskom jeziku,

(c) teksta na srpskom jeziku.

2.7 Programom iz zadatka 2.4 šifrujte engleski tekst dužine 1000-2000 karaktera bez interpunkcijskih znakova. Sastavite program koji računanjem indeksa koincidencije određuje dužinu ključa.

2.7. Pokušajte da objasnite logiku postupka određivanja dužine ključa i

nalaženja samog ključa metodom indeksa koincidencije.

2.8 Izmenite program iz zadatka 2.1 tako da se šifrovanje obavlja Hillovom šifrom ukoliko je ključ matrica veličine 3x3.

2.9 Programom iz zadatka 2.8 šifrujte tekst na engleskom jeziku dužine 100-200 karaktera. Pod pretpostavkom da znate prvih 10 karaktera otvorenog teksta, sastavite program koji metodom poznati otvoreni tekst određuje ključ.

3

Simetrični blokovski algoritmi

3.1 DES

Američki nacionalni biro za standarde (*National Bureau of Standards*, NBS) započeo je 1972. godine program zaštite podataka. Jedan od ciljeva programa bio je razvijanje i standardizovanje kriptosistema. Godine 1973. NBS je raspisao javni konkurs za kriptosistem koji je trebalo da zadovolji sledeće uslove:

- visoki nivo sigurnosti koja leži u ključu, a ne u tajnosti algoritma,
- efikasnost,
- potpuna specifikacija i lako razumevanje algoritma,
- ekonomičnost implementacije u elektronskim uređajima,
- dostupnost svim korisnicima i
- mogućnost izvoza (zbog američkih zakona o izvozu - *US export laws*).

Na tom konkursu ni jedan predlog nije zadovljio sve navedene zahteve, tako da je NBS ponovio konkurs sledeće godine. Na ponovljeni konkurs poslat je predlog algoritma koji je razvio IBM-ov tim kriptografa. Algoritam je simetričan, što znači da se isti ključ koristi i za šifrovanje i za dešifrovanje. Algoritam je blokovski orijentisan, što znači da obrađuje jedan po jedan blok bitova otvorenog teksta koristeći ključ. Jedan blok otvorenog teksta se pomoću istog ključa uvek prevodi u isti šifrat. Algoritam je zasnovan na upotrebi Fiestelove mreže[5] (engl. *Feistel network*, nazvana je po tvorcu algoritma Lucifer, Horst Feistelu). Predloženi algoritam je prihvaćen kao standard 1976. godine, nakon nekih izmena u kojima je učestvovala i Američka nacionalna agencija za bezbednost (*National Security Agency* – NSA). Algoritam je dobio ime *Data Encryption Standard* (DES).

5 Blokovski algoritam koji u *i*-toj rundi obavlja operacije: $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$. To znači da se blok podatka deli na dva dela, od kojih se jedan propusti kroz određenu funkciju, a drugi se dalje prenosi neizmenjen. Blokovi zatim zamene mesta, pa se obavi sledeća runda (broj rundi zavisi od algoritma - DES ima 16 takvih rundi).

Šifrovanje i dešifrovanje

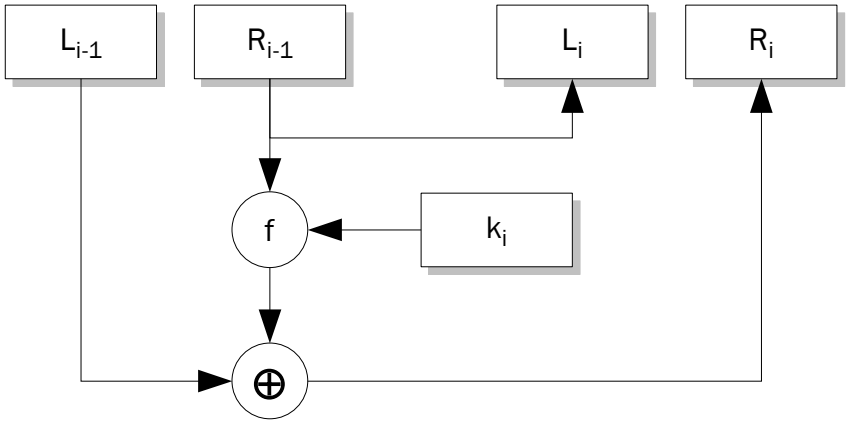
DES je simetričan algoritam koji šifruje tekst u blokovima dužine 64 bita, koristeći ključ *k* dužine 56 bita. Tako se dobija šifrat dužine 64 bita. Tri osnovna koraka u algoritmu su: inicijalna permutacija *IP,* 16 rundi obrade podataka (proširenje, XOR sa ključem, supstitucija) i završna inverzna permutacija IP^{-1}.

Za dati blok otvorenog teksta *x*, pomoću fiksne inicijalne permutacije IP (tablica A.2 u dodatku A) dobija se vrednost $x_0 = IP(x)$ koja se može zapisati u obliku $x_0 = L_0R_0$, gde su L_0 – 32 viša bita u x_0 i R_0 – 32 niža bita u x_0.

Nakon inicijalne permutacije, nad izlaznim podatkom $L_{i-1}R_{i-1}$ ($1 \le i \le 16$) iz prethodne runde se 16 puta obavljaju sledeće transformacije:

- $L_i = R_{i-1}$
- $R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$,

Ulazni podatak za prvu rundu je vrednost dobijena inicijalnom permutacijom bloka otvorenog teksta.



Slika 3.1 *– Jedna runda DES algoritma*

Na kraju se izlazni podatak iz poslednje runde pomoću završne permutacije IP^{-1} (tablica A.3 u dodatku A). transformiše u šifrat dužine 56

bita.

Na slici 3.1 simbolički je prikazana jedna runda DES algoritma. Simbolom $\oplus$ označena je operacija ekskluzivno ILI, a k_1, k_2, ... , k_{16} su potključevi dužine 48 bita dobijeni kao permutacije nekih bitova iz ključa k.

Funkcija f prihvata dva ulazna argumenta: nižih 32 bita izlaza iz prethodne runde (R_{i-1}) i potključ dužine 48 bitova (k_i). Kao rezultat se dobija niz dužine 32 bita. Funkcija se računa u sledeća četiri koraka:

- Niz R_{i-1} se proširuje do niza dužine 48 bita prema fiksnoj funkciji proširenja *EX* (tablica A.4 u dodatku A). Rezultujući Niz $EX(R_{i-1})$ se sastoji od 32 bita niza R_{i-1}, permutovanih na određeni način, s tim da se 16 bitova pojavi dvaput.

- Zatim se računa vrednost $A = EX(R_{i-1}) \oplus k_i$ a rezultat se zapisuje u formi osam 6-bitnih nizova: $A = A_1A_2A_3A_4A_5A_6A_7A_8$.

- Sledeći korak u računanju vrednosti funkcije f je zamena vrednosti *A* pomoću supstitucijskih kutija (engl. *supstitution box*, *S-box*). Svaka supstitucijska kutija S_j (S_1, S_2, ... , S_8) je fiksna matrica dimenzija 4×16, čiji su elementi celi brojevi između 0 i 15 (tablice A.5-A.12 u prilogu A). Za dati niz od 6 bitova $A_j = a_1a_2a_3a_4a_5a_6$, rezultat supstitucije $S_j(A_j)$ računa se na sledeći način:

  - Dva bita a_1a_6 određuju binarni zapis reda ($0 \le red \le 3$), a četiri bita $a_2a_3a_4a_5$ određuju binarni zapis kolone ($0 \le kol \le 15$) u S_j.
  - $B_j = S_j(A_j) = S_j(red,kol)$, zapisano kao binarni broj dužine 4 bita.

  Na ovaj način se određuje $B = B_1B_2...B_8 = S_1(A_1)S_2(A_2)...S_8(A_8)$.

- Niz bitova *B* dužine 32 bita permutuje se pomoću fiksne završne permutacije *P* (tablica A.13 u dodatku A). Tako se dobija P(B), odnosno $f(R_{i-1}, k_i)$.

Objasnićemo kako se formiraju potključevi k_1, k_2, ... , k_{16}. Ključ k dužine 56 bita, koji se koristi prilikom šifrovanja čuva se u obliku K, dužine 64 bita. Bitovi parnosti na pozicijama 8, 16, 24, 32, 40, 48, 56 i 64 definisani su tako da svaki bajt sadrži neparan broj jedinica. Ovi bitovi se ignorišu kod

računanja tablice ključeva.

Za dati 64-bitni ključ *k*, ignorišu se bitovi parnosti, a ostalih 56 bita se permutuje pomoću fiksne permutacije PK_1 (tablica A.14 u dodatku A). Zapisuje se $PK_1(k) = C_0D_0$, gde su C_0 i D_0 viših i nižih 28 bitova u $PK_1(k)$. Za i = 1, 2, ... , 16 računa se:

- $C_i = LS_i(C_{i-1})$,
- $D_i = LSi(D_{i-1})$,
- $k_i = PK_2(C_iD_i)$.

LSi je ciklički pomeraj ulevo za jednu poziciju, ako je i=1, 2, 9 ili 16, a u svim ostalim slučajevima za dve pozicije. PK_2 je fiksna permutacija predstavljena tablicom A.15 u dodatku A.

Za dešifrovanje DES šifrata koristi se isti algoritam kao i za šifrovanje. Polazi se od šifrata *y*, ali se potključevi koriste u obrnutom redosledu: k_{16}, k_{15}, ... , k_1. Kao rezultat se dobija otvoreni tekst *x*.

Ključevi koji se ne koriste

Neki DES ključevi su značajno nesigurniji od ostalih, i kao takvi se ne koriste. U te ključeve spadaju:

- Slabi ključevi, tj. ključevi koji generišu jednake potključeve u svakoj rundi: $k_1 = k_2 = ... = k_{16}$. Postupci šifrovanja i dešifrovanja slabim ključem su jednaki. DES slabi ključevi (zapisani heksadecimalno s bitovima parnosti) su:
  - 0101010101010101
  - 1F1F1F1F0E0E0E0E
  - E0E0E0E0F1F1F1F1
  - FEFEFEFEFEFEFEFE
- Delimično slabi ključevi, tj. ključevi koji generišu samo dva različita potključa, od kojih se svaki koristi u po 8 rundi. Par ključeva (k,k') je

par DES delimično slabih ključeva ako je rezultat kompozicije dva DES algoritma sa ključevima k i k' otvoreni tekst. Jednostavnije rečeno, šifrovanje sa jednim ključem isto je što i dešifrovanje s drugim. Postoji šest pari DES polovično-slabih ključeva:

- 01FE01FE01FE01FE i FE01FE01FE01FE01
- 1FE01FE00EF10EF1 i E0F1E0F1F10EF10E
- 01E001E001F101F1 i E001E001F101F101
- 1FFE1FFE0EFE0EFE i FE1FFE1FFE0EFE0E
- 011F011F010E010E i 1F011F010E010E01
- E0FEE0FEF1FEF1FE i FEE0FEE0FEF1FEF1

- Potencijalno slabi ključevi, tj. ključevi koji generišu samo četiri potključa (48 ključeva).

Ukupno ima 64 ključa koje ne treba koristiti.

Karakteristike i kriterijumi dizajna

Inicijalna i završna permutacija nemaju skoro nikakav uticaj na sigurnost DES algoritma, ali omogućuju lakše učitavanje otvorenog teksta i šifrata bajt po bajt u hardverske implementacije DES algoritma zasnovane na 8-bitnom mikroprocesoru (uzmite u obzir da se DES se pojavio pre 16-bitnih i 32-bitnih mikroprocesora). Zbog toga neki proizvođači softvera preskaču IP i IP^{-1} i otvoreni tekst direktno uvode u prvu rundu - ovako modifikovani algoritmi nisu manje sigurni od DES-a, ali ne prate standard i ne treba ih nazivati “DES algoritmima”.

Kriptoanaliza linearnih kriptosistema napadom “poznati otvoreni tekst” je relativno jednostavna (npr. napad na Hilovu šifru). Sve operacije u DES-u, osim zamene vrednosti u supstitucijskim kutijama, su linearne, što znači da su S-kutije izuzetno značajne za sigurnost DES-a. Supstitucijske kutije i tablica permutacije *P* dizajnirane su tako da povećaju difuziju kriptosistema (na svaki bit šifrata utiče što više bitova otvorenog teksta) i otežaju diferencijalnu kriptoanalizu. S-kutije je dizajnirala NSA prema sledećim kriterijumima:

- Svaki red u svakoj S-kutiji je permutacija brojeva od 0 do 15.
- Ni jedna S-kutija nije linearna ili afina funkcija ulaznih podataka.
- Posledica promene jednog bita ulaznog podatka kod primene S-kutija je promena bar 2 bita u izlaznom podatku.
- Za svaku S-kutiju i svaki ulazni podatak x važi da se vrednosti $S_j(x)$ i $S_j(x \oplus 001100)$ razlikuju u barem 2 bita.
- Za svaku S-kutiju, svaki ulazni podatak x i svako $e,f \in \{0,1\}$ važi sledeće: $S(x) \neq S(x \oplus 11ef00)$.
- Za svaku S-kutiju važi sledeće: ako se fiksira jedan bit u ulaznom podatku i posmatraju vrednosti jednog fiksnog izlaznog podatka, onda će broj ulaznih podataka kod kojih je taj bit jednak 0 biti blizu broja onih kod kojih je taj bit jednak 1. Preciznije, ako fiksiramo prvi ili šesti bit ulaznog podatka, onda su oba broja jednaka 16, dok kod drugih bitova variraju između 13 i 19.

Dizajn permutacije P usklađen je sa sledećim kriterijumima:

- Četiri bita izlaza iz svake S-kutije su u sledećoj rundi ulazni podaci za šest različitih S-kutija, a ni koja dva nisu ulazni podaci za istu S-kutiju.
- Četiri izlazna bita iz svake S-kutije u i-toj rundi su distribuirani tako da dva od njih utiču na središnje bitove u i+1 rundi, a dva na krajnje bitove (dva leva i dva desna od 6 bitova u ulaznom podatku).
- Za dve S-kutije S_j i S_k važi da ako neki izlazni bit iz S_j utiče na neki središnji bit S_k u sledećoj rundi, onda ni jedan izlazni bit iz S_k ne utiče na središnje bitove S_j.
- Za j=k: izlazni bitovi iz S_j ne utiču na središnje bitove S_j.

Poželjno je da promena jednog bita otvorenog teksta ili jednog bita ključa utiče na mnogo bitova šifrata. Ako je promena u šifratu mala, to može značajno smanjiti broj otvorenih tekstova ili ključeva koje treba ispitati prilikom kriptoanalize. Svojstvo da mala promena otvorenog teksta ili ključa dovodi do značajne promene u šifratu karakteristično je za DES algoritam i u literaturi se može naći pod imenom “efekt lavine” (engl. *avalanche*).

Lako se može pokazati da već kod treće runde efekt lavine dolazi do izražaja, a da posle pete runde svaki bit šifrata zavisi od svakog bita

otvorenog teksta i svakog bita ključa. Takođe, može se pokazati i da je posle osme runde šifrat slučajna funkcija bitova otvorenog teksta i ključa. Pretpostavlja se da posle 16 rundi poznati napadi (kao što je, na primer, diferencijalna kriptoanaliza) ne mogu biti efikasniji od napada "grubom silom".

Sigurnost DES-a

Algoritam koji su IBM-ovi kriptografi ponudili Američkom nacionalnom birou za standarde koristio je ključ dužine 112-bitni ključ. U verziji koja je prihvaćena kao standard dužina ključa je pod uticajem NSA smanjena na 56 bita. DES, sa ključem dužine 56 bita, danas ne pruža dovoljnu sigurnost protiv napada "grubom silom".

Diffie i Hellman su 1977. godine ustanovili da tadašnja tehnologija omogućava konstrukciju računara koji bi otkrio ključ za jedan dan (troškovi procenjeni na 20 miliona dolara). Na osnovu toga su zaključili da je takav računar dostupan samo vojnim organizacijama i organizacijam kao što je NSA, ali da će 1990. godine DES postati sasvim nesiguran. Weiner je 1993. godine procenio da se za 100.000 dolara može konstruisati računar koji bi otkrilo ključ za 35 časova, za milion dolara računar koji bi otkrio ključ za 3.5 časa, a za 10 miliona dolara računar koji bi otkrio ključ za 21 minut. Konačno razbijanje DES-a usledilo je 1998. godine: organizacija *Electronic Frontier Foundation* (*EFF*) konstruisala je *DES Cracker* koji košta 250.000 dolara, a otkriva ključ za 56 časova. Sagrađen je od 1536 čipova koji mogu testirati 88 milijardi ključeva u sekundi. [6]

Izraelski kriptografi Eli Biham i Adi Shamir javno su 1990. godine opisali metodu diferencijalne kriptoanalize, koja spada u napade tipa odabran otvoreni tekst. Osnovna ideja diferencijalne kriptoanalize je poređenje rezultata operacije ekskluzivno ILI, izvršene nad dva otvorena teksta sa rezultatom iste operacije izvršene nad dva odgovarajuća šifrata. Jednostavnije rečeno, posmatraju se dva bloka otvorenog teksta sa specifičnim razlikama i analizira se evolucija te razlike pri prolasku kroz algoritam. Na osnovu razlika u dobijenim šifratima, različitim ključevima se dodeljuju verovatnoće. Nakon testiranja većeg broja parova otvorenog

6 Više o tome može se naći na web stranici organizacije EFF http://www.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/.

teksta, određuje se najverovatniji ključ. Metoda je očigledno bila poznata konstruktorima DES-a, što se vidi iz načina na koji su konstruisane supstitucijske kutije i permutacija *P*.

Japanski kriptograf Mitsuru Matsui opisao je 1993. godine metodu linearne kriptoanalize. Ova metoda, najverovatnije nepoznata tvorcima DES-a, zasniva se na činjenici da se neki bitovi bitovi ključa, iako nisu linearne funkcije otvorenog teksta i šifrata, mogu dobro aproksimirati linearnom funkcijom. Pomoću linearne kriptoanalize Matsui je opisao napad tipa poznati otvoreni tekst koji za razbijanje DES-a zahteva prosečno 2^{43} otvorenih tekstova. Ni linearna ni diferencijalna kriptoanaliza nisu dovele do razbijanja DES-a, ali je njihova važnost u tome što su primenljive na svaki simetrični blokovski algoritam.

Sigurnost DES-a može se uvećati kompozicijom operacija šifrovanja sa različitim ključevima. Za fiksni ključ *k*, skup od 2^{56} permutacija dobijenih pomoću DES-a je podskup grupe svih permutacija skupa $\{0,1\}^{64}$, čiji je red 2^{64}. DES (tj. skup svih njegovih permutacija) nije podgrupa ove grupe, što znači da se višestrukom upotrebom DES-a može postići veća sigurnost.

Objasnićemo najpre zašto se dvostruki DES ne koristi. Šifrovanje i dešifrovanje dvostrukim DES algoritmom pomoću ključeva k_1 i k_2 definiše se na sledeći način:

- $y = e_{k2}(e_{k1}(x))$
- $x = d_{k1}(d_{k2}(y))$

Pokazano je da je broj operacija potreban za kriptoanalizu dvostrukog DES-a pomoću napada "susret u sredini" (engl. *meet-in-the-middle*, Diffie, 1997.) reda veličine 2^{56}, što je neznatno više od broja operacija potrebnih za kriptoanalizu običnog DES-a. Radi se o sledećem: pretpostavimo da je poznat jedan par blokova otvoreni tekst – šifrat (x,y). Blok tvorenog teksta *x* redom se šifruje pomoću 2^{56} mogućih ključeva k_1, a rezultati se upisuju u tabelu i sortiraju po vrednostima $e_{k1}(x)$. Zatim se blok šifrata *y* redom dešifruje pomoću 2^{56} mogućih ključeva k_2. Posle svakog dešifrovanja, u tabeli se traži rezultat $d_{k2}(y)$ takav da je $e_{k1}(x)=d_{k2}(y)$. Ukoliko se par ključeva (k_1,k_2) pronađe, testira se na poznatom paru otvoreni tekst – šifrat. Ukoliko ključevi zadovolje taj test, prihvataju se za korektne ključeve. Verovatnoća da je izabran pogrešan par ključeva je $2^{112-64-64} = 2^{-16}$.

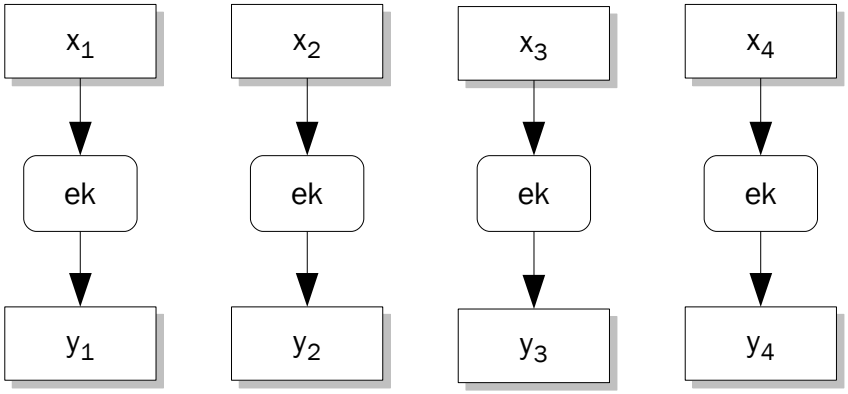
Zbog toga se koristi trostruki DES, koji se definiše na sledeći način:

- $x = d_{k1}(e_{k2}(d_{k3}(y)))$,
- $y = e_{k3}(d_{k2}(e_{k1}(x)))$.

Ukoliko je k2=k3, algoritam se ponaša kao običan DES. Za trostruki DES broj potrebnih operacija prilikom napada "susret u sredini" je reda veličine 2^{112}, što znači da je sigurnost trostrukog šifrovanja onakva kakvu smo očekivali od dvostrukog.

Režimi rada

Do sada je opisano kako DES šifruje jedan bloku dužine 64 bita. U realnim situacijama, poruke su znatno duže. Poznata su četiri režima rada DES-a (engl. *DES modes of operation*) koji pokrivaju sve moguće primene DES-a. Ovi režimi su primjenljivi na bilo koji simetričan blokovski algoritam.

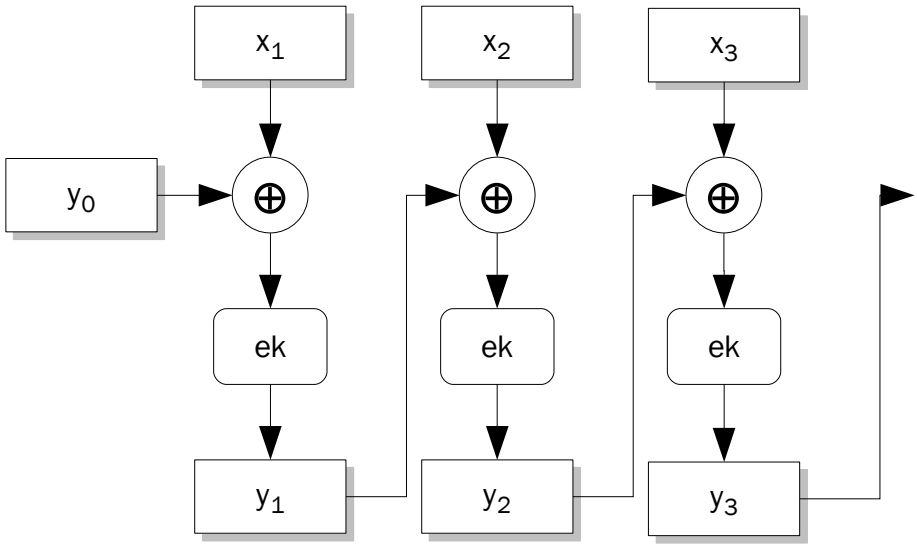


Slika 3.2 *– ECB režim rada*

ECB režim rada (engl. *electronic codebook mode*) je najjednostavniji režim. Poruka se podeli na blokove dužine 64 bita (zadnji blok se dopuni slučajno generisanim nizom ako je potrebno), a šifrovanje se obavlja blok po blok pomoću istog ključa (slika 3.2). Identičnim blokovima otvorenog teksta odgovaraju identični blokovi šifrata.

Prilikom šifrovanja u CBC režimu rada (engl. *cipher block chaining*),

najpre se računa rezultat XOR operacije izvršene nad trenutnim blokom otvorenog teksta i šifratom prethodnog bloka, a zatim se rezultat šifruje ključem *k* (slika 3.3). Povratna sprega postoji, tako da identičnim blokovima otvorenog teksta u opštem slučaju odgovaraju različiti šifrati.



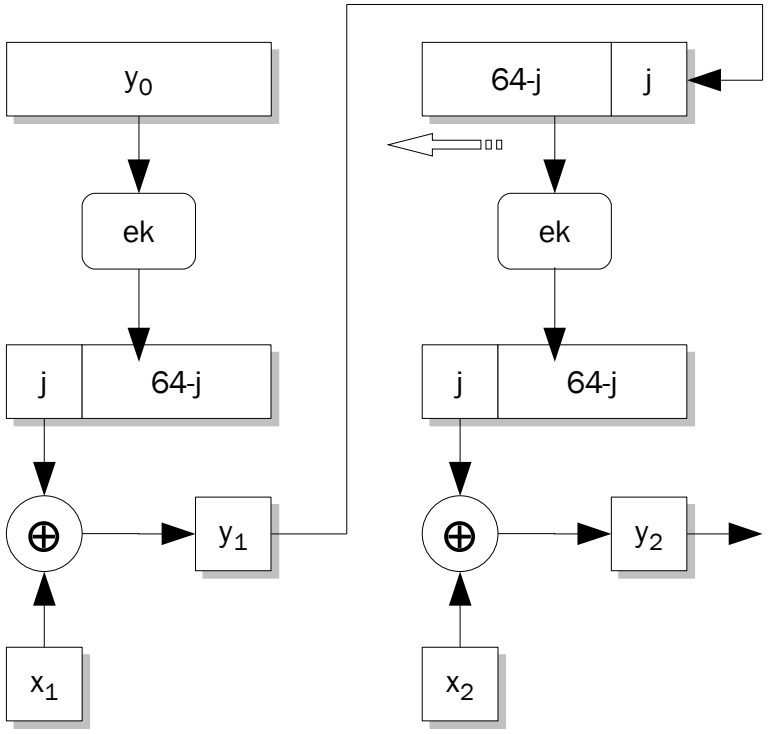
Slika 3.3 *– CBC režim rada*

Vrednost y_0 je inicijalna vrednost (pominje se u literaturi pod imenom inicijalizujući vektor) koja mora biti poznata i primaocu i pošiljaocu (npr. može se poslati ECB režimom). Za šifrovanje i dešifrovanje se koriste sledeće relacije:

- $y_i = e_k(y_{i-1} \oplus x_i)$ za $i \geq 1$.
- $x_i = y_{i-1} \oplus d_k(y_i)$.

U CFB režimu (engl. *cipher feedback*) DES radi kao protočna (engl. *stream*) šifra. Kod protočnih šifri poslednji blok otvorenog teksta se ne dobunjuje slučajno generisanim nizom do dužine 64 bita, što znači da je šifrat iste dužine kao i otvoreni tekst. DES u CFB režimu (slika 3.4) obrađuje odjednom *j* bitova otvorenog teksta ($1 \leq j \leq 64$), s tim da se otvoreni tekst najčešće čita bit po bit (j=1) ili karakter po karakter (j=8, jednom slovu odgovara 8 bitova po ASCII standardu).

Najpre se šifruje 64-bitna inicijalna vrednost y_0. Zatim se y_1 dobija kao rezultat operacije ekskluzivno ILI izvršene nad x_1 i j levih bitova izlaznog podatka. Ulazni podatak za sledeći korak šifrovanja dobija se tako što se prethodni ulazni podatak pomeri za j mesta ulevo, a zatim se sa desne strane doda y_1. Postupak se nastavlja dok se ceo otvoreni tekst ne šifruje.



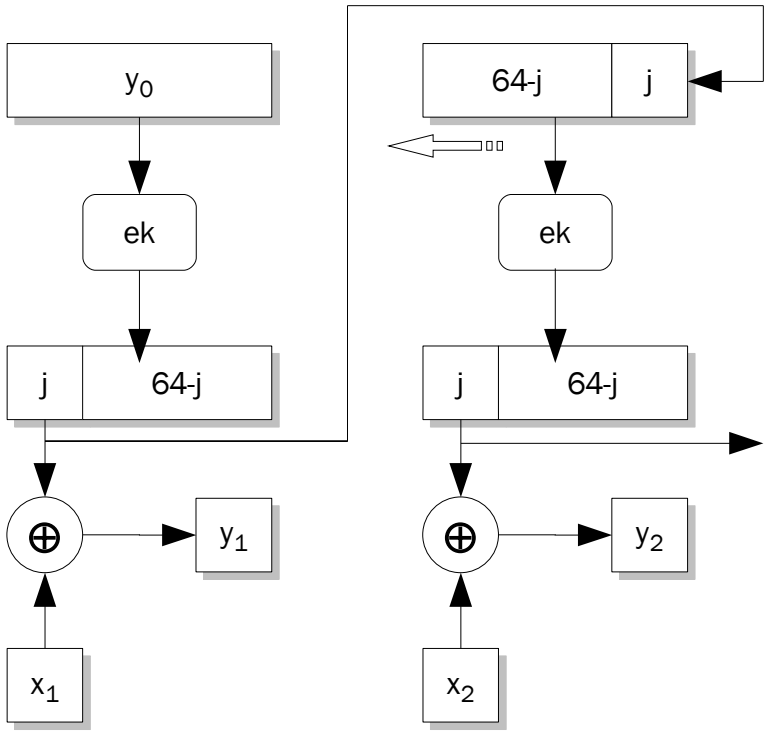
Slika 3.4 *– CFB režim rada*

Dešifrovanje se obavlja na isti način, a otvoreni tekst se dobija pomoću operacije ekskluzivno ILI, izvršene nad odgovarajućim šifratom i izlaznim podatkom funkcije šifrovanja e_K.

OFB režim (engl. *output feedback*) je vrlo sličan CFB, s tim što se ulazni podatak za funkciju e_k u sledećem koraku šalje odmah posle primene e_K u tekućem koraku, a ne posle primene XOR-a.

Prednost OFB režima je u tome što se greške u transmisiji ne propagiraju kroz ostatak šifrata (npr. greška u delu šifrata y_1 prilikom dešifrovanja

proizvešće neispravan deo otvorenog teksta x_1, ali je ostatak otvorenog teksta ispravan).



Slika 3.5 *– OFB režim*

3.2 AES/RIJNDAEL

Trostruki DES je predstavljao adekvatnu privremenu zamenu za DES koja obezbeđuje viši nivo sigurnosti. Međutim, trostruki DES šifruje podatke pomoću ključa dužine 168 bita (3x56) kako bi postigao sigurnost za koju je dovoljan ključ dužine 112 bita. Takođe, trostruki DES obavlja operacije kroz 48 rundi (3x16) kako bi postigao sigurnost za koju je verovatno dovoljno 32 runde. Što se performansi tiče, softverske implementacije trostrukog DES-a su prespore za neke primene (na primer, za šifrovanje video zapisa u realnom vremenu).

Američki institut za standarde i tehnologiju (*National Institute of Standards and Technology*, *NIST*) je 1997. godine raspisao konkurs za kriptosistem koji je trebalo da zameni DES. Uslovi koje je kriptosistem morao da zadovolji su sledeći: algoritam je simetričan blokovski, operacije se obavljaju nad blokovima otvorenog teksta dužine 128 bita pomoću ključa dužine 128, 192 i 256 bita. Od svih prijavljenih kandidata, 15 je zadovoljilo kriterijume. U uži krug izbora ušli su sledeći algoritmi:

- TWOFISH (Counterpane Systems) – algoritam sa Feistelovom mrežom od 16 rundi kod koje se supstitucijske kutije dinamički menjaju u zavisnosti od ključa, što otežava diferencijalnu i linearnu kriptoanalizu.
- RC6 (RSA Security Inc.) – algoritam sa Feistelovom mrežom od 20 rundi koji difuziju postiže pomoću funkcije $f(x) = x(2x+1)$, a otpornost na diferencijalnu i linearnu kriptoanalizu pomoću rotacije koja zavisi od samih podataka.
- MARS (IBM)
- SERPENT
- RIJNDAEL

Godine 2000. objavljeno je da je pobednik konkursa algoritam RIJNDAEL, kog su razvili belgijski kripografi Joan Daemen i Vincent Rijmen. Karakterističan je po tome što prilikom konstrukcije supstitucijskih kutija koristi operacije u konačnom polju $GF(2^8)$. Modifikovani algoritam je dobio ime AES (*Advanced Encryption Standard*). Osnovne karakteristike RIJNDAEL algoritma su sledeće:

- veličina bloka za šifrovanje je promenljiva (128, 192 ili 256 bita),
- dužina ključa je promenljiva (128, 192 ili 256 bita)[7],
- broj rundi je promenljiv i zavisi od dužne ključa i veličine bloka.

Iz prethodno navedenih karakteristika proizilazi da je algoritam zbog veće dužine ključa otporniji na napad "grubom silom" od DES-a.

Mnogo simetričnih blokovskih algoritama je zasnovano na upotrebi Feistelovih mreža - blok podataka se deli na dva dela koji prenošenju u

7 Ključ može biti i kraći i duži od tih vrednosti, ali dužina ključa mora biti deljiva sa 4; u tom slučaju se menja i broj rundi.

sledeću rundu menjaju mesta, s tim da se nad jednim blokom obavi određena funkcija: $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$. RIJNDAEL nije takav algoritam – autori algoritma su stvorili runde koje se razlikuju od Feistelovih. Jedna RIJNDAEL runda sadrži tri sloja:

- linearni difuzioni sloj (engl. *linear mixing layer*), koji obezbeđuje veliku difuziju bitova nakon nekoliko rundi (funkcije *ShiftRow* i *MixColumn*),
- nelinearni sloj (engl. *non-linear layer*), odnosno upotreba supstitucijskih kutiha optimizovanih za najgori slučaj (funkcija *ByteSub*),
- sloj dodavanja ključa (engl. *key addition layer*), u kome se obavlja operacija ekskluzivno ILI nad potključem runde se trenutnim stanjem bloka (funkcija *AddRoundKey*).

Slojevi su dizajnirani pomoću specijalnih metoda dizajniranja algoritama, sa posebnim osvrtom na otpornost protiv diferencijalne i linearne kriptoanalize (*Wide Tail Strategy*, *WTS*).

Matematička osnova [8]

Svi okteti RIJNDAEL algoritma interpretiraju se kao elementi konačnog polja $GF(2^8)$ (engl. *Galois field*). Oktet se može prikazati kao niz bitova $b_7b_6...b_0$ ili kao element konačnog polja pomoću sledeće notacije:

- $b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0x^0 = \sum_i b_i x^i$.

Na primer, oktet 0110 1001 (69h) može se prikazati kao element konačnog polja: $x^6+x^5+x^3+1$. Sabiranje elemenata u konačnom polju izvodi se pomoću operacije ekskluzivno ILI primenjene nad odgovarajućim bitovima polinoma. Na primer,

- $(x^7+x^6+x^5+1) + (x^6+x^5+x+1) = (x^7+x)$

Množenje u konačnom polju $GF(2^8)$ odgovara množenju polinoma po

8 Delom preuzeto iz dokumenta CCERT-PUBDOC-2003-08-37, koji je dostupan sa web stranice http://www.cert.hr

modulu *m(x)*, gde je m(x) nedeljiv polinom [9] osmog stepena. Operacija deljenja po modulu, tj. traženja ostatka pri deljenju sa *m(x)* obezbeđuje da će rezultat biti stepena manjeg od 8, tj. da se rezultat može prikazati kao polinom. Ovako definisano množenje je asocijativno, a multiplikativna konstanta je 1. Množenje i sabiranje su distributivne operacije. RIJNDAEL kao nedeljivi polinom koristi $m(x) = x^8+x^4+x^3+x+1$. Na primer,

- $(x^6+x^4+x^2+x+1) \cdot (x^7+x+1) \bmod (x^8+x^4+x^3+x+1) = x^7+x^6+1$

Takođe, za svaki polinom *b(x)* različit od nula polinoma može se odrediti inverzni polinom $b^{-1}(x)$. Pomoću proširenog Euklidovog algoritma mogu se izračunati polinomi *a(x)* i *c(x)* takvi da je: $b(x) \cdot a(x) + m(x) \cdot c(x) = 1$. Zato je: $a(x) \cdot b(x) \bmod m(x) = 1$, što znači $b^{-1}(x) = a(x) \bmod m(x)$.

RIJNDAEL takođe koristi i operacije nad polinomima sa koeficijentima iz $GF(2^8)$ stepena manjeg od 4. Takvi polinomi se razlikuju od prethodno opisanih polinoma po tome što su koeficijenti a_i elementi konačnog polja, tj. okteti, a ne bitovi. Svaki polinom oblika $a(x) = a_3x^3+a_2x^2+a_1x+a_0$ može se predstaviti kao četvorobajtni vektor tj. kao 32-bitna reč $a_0a_1a_2a_3$. Sabiranje takvih polinoma se svodi na sabiranje (tj. na izvođenje operacije ekskluzivno ILI) koeficijenta članova istog stepena. Da bi se kod množenja kao rezultat dobio polinom stepena manjeg od četiri, množenje se obavlja po modulu polinoma četvrtog stepena (za to je izabran polinom x^4+1). Rezultat množenja polinoma *a(x)* i *b(x)* po modulu x^4+1 je polinom četvrtog stepena *d(x)* definisan na sledeći način:

- $d(x) = d_3x^3+d_2x^2+d_1x+d_0$, gde su:
  - $d_0=(a_0 \cdot b_0) \oplus (a_3 \cdot b_1) \oplus (a_2 \cdot b_2) \oplus (a_1 \cdot b_3)$
  - $d_1=(a_1 \cdot b_0) \oplus (a_0 \cdot b_1) \oplus (a_0 \cdot b_2) \oplus (a_3 \cdot b_3)$
  - $d_2=(a_2 \cdot b_0) \oplus (a_1 \cdot b_1) \oplus (a_0 \cdot b_2) \oplus (a_3 \cdot b_3)$
  - $d_3=(a_3 \cdot b_0) \oplus (a_2 \cdot b_1) \oplus (a_1 \cdot b_2) \oplus (a_0 \cdot b_3)$

Pri tome se koeficijenti a_i i b_i množe kao polinomi po modulu *m(x)*.

9 Polinom f(x) iz $Z_p[x]$ je nesvodljiv ako ne postoje polinomi $f_1(x)$ i $f_2(x)$ iz $Z_p[x]$ takvi da je $f(x) = f_1(x)\, f_2(x)$ gde su $\deg(f_1)>0$ i $\deg(f_2)>0$. Drugim rečima, polinom f(x) je nesvodljiv ukoliko je deljiv samo jedinicom i sa samim sobom. Nesvodljiv polinom je među polinomima isto što i prost broj među brojevima.

Šifrovanje

Stanje bloka (engl. *state*) je trenutno stanje šifrovanog bloka podatka. Stanje se može zamisliti kao matrica bajtova promenljive veličine (blok za šifrovanje je promenljive veličine). Na primer, blok podataka veličine 192 bita može se prikazati na sledeći način:

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$	$s_{0,4}$	$s_{0,5}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$	$s_{1,4}$	$s_{1,5}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$	$s_{2,4}$	$s_{2,5}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$	$s_{3,4}$	$s_{3,5}$

Svaka ćelija predstavlja jedan bajt (oktet). Ulazni okteti se u matricu upisuju redom po kolonama, tj: $s_{0,0}$, $s_{1,0}$, $s_{2,0}$, $s_{3,0}$, $s_{0,1}$, $s_{1,1}$, ... $s_{0,5}$, $s_{1,5}$, $s_{2,5}$, $s_{3,5}$. Broj redova matrice je fiksan (četiri reda), što znači da svaka kolona ima ukupno 32 bita, tj. 4 okteta. Broj kolona matrice (N_w) zavisi od veličine bloka za šifrovanje i dobija se tako što se veličina bloka podeli sa 32. Matrica stanja AES algoritma je uvek veličine 4x4 okteta (AES radi isključivo sa 128-bitnim blokovima), dok originalni RIJNDAEL algoritam dopušta i druge vrednosti; u tom slučaju se koriste i matrice reda 4x6 za 192-bitne blokove, odnosno matrice reda 4x8 za 256-bitne blokove.

Ključ za šifrovanje prikazuje se pomoću tabele sa četiri reda i N_k kolona, slično stanju bloka. Za 192-bitni ključ, tablica ključa će imati sledeći oblik:

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$	$k_{0,4}$	$k_{0,5}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$	$k_{2,4}$	$k_{2,5}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$	$k_{3,4}$	$k_{3,5}$

Stanje i ključ predstavljeni na ovaj način olakšavaju logičku analizu algoritma i koriste se interno u algoritmu.

Broj rundi (*Nr*) je, promenljiv, i zavisi od veličine bloka i dužine ključa. Konkretno. broj rundi je određen onim što je duže – max (N_w, N_k). Ako su i ključ i blok dužine 128 bita ($N_w = N_k = 4$), onda je broj rundi [10] 10. Ako su ili ključ ili blok podatka dužine 192 bita ($N_w = 6$ ili $N_k = 6$), onda je broj rundi 12. Ako su ili ključ ili blok podatka dužine 256 bita ($N_w = 8$ ili $N_k = 8$), onda je broj rundi 14. Što se AES standarda tiče, blok je uvek dužine 128 bita, a ključevi mogu biti dužine 128 bita (AES-128), 192 bita (AES-192) i 256 bita (AES-256).

Sve operacije RIJNDAEL algoritma izvode se na matrici stanja. Šifrovanje počinje tako što se ulazni blok podataka kopira u matricu stanja.

Nakon kopiranja ulaznog podatka, u matricu stanja se inicijalno dodaje potključ (*AddRoundKey*). Zatim se matrica stanja transformiše 10, 12 ili 14 puta, zavisno od dužine ključa, s tim da se poslednja runda transformacija razlikuje od prethodnih (izostavlja se transformacija *MixColumn*). Svaka runda algoritma predstavlja funkciju koja se sastoji od četiri transformacije, koje se izvode nad oktetima:

- zamena okteta na osnovu tabele supstitucije (*ByteSub*)

  Funkcija *ByteSub* je jedina nelinearna transformacija: najpre se svaki element matrice stanja zameni svojim multiplikativnim inverzom u konačnom polju GF(2^8) (element 00h ostaje nepromenjen), a zatim se nad bitima matrice stanja primeni transformacija:

  - $b'_i = b_i \otimes b_{(i+4)mod8} \otimes b_{(i+5)mod8} \otimes b_{(i+6)mod8} \otimes b_{(i+7)mod8} \otimes c_i$

  b_i je i-ti bit okteta, a c_i i-ti bit okteta c=63h=01100011. Funkcija *ByteSub* se može prikazati pomoću supstitucijske kutije (tablica A.16 u prilogu A). Red u supstitucionoj kutiji određen je sa četiri viša, a kolona sa četiri niža bita okteta. Na primer, oktet 69h se pomoću funkcije *ByteSub* transformiše u F9h.

- pomeranje okteta u redovima matrice stanja (*ShiftRow*)

  Funkcija *ShiftRow* obavlja rotaciju okteta u redovima matrice stanja za jedno (drugi red), dva (treći red) i tri mesta ulevo (četvrti red), s tim što

10 Uključuje i poslednju dodatnu rundu.

se prvi red matrice (r=0) ne pomera.

- $s_{0,0}\ s_{0,1}\ s_{0,2}\ s_{0,3} \rightarrow s_{0,0}\ s_{0,1}\ s_{0,2}\ s_{0,3}$
- $s_{1,0}\ s_{1,1}\ s_{1,2}\ s_{1,3} \rightarrow s_{1,1}\ s_{1,2}\ s_{1,3}\ s_{1,0}$
- $s_{2,0}\ s_{2,1}\ s_{2,2}\ s_{2,3} \rightarrow s_{2,2}\ s_{2,3}\ s_{2,0}\ s_{2,1}$
- $s_{3,0}\ s_{3,1}\ s_{3,2}\ s_{3,3} \rightarrow s_{3,3}\ s_{3,0}\ s_{3,1}\ s_{3,2}$

- mešanje podataka u kolonama matrice stanja (*MixColumn*)

  Funkcija *MixColumn* vrši transformacije u svakoj koloni matrice stanja. Kolone se posmatraju kao polinomi četvrtog stepena *s(x)* sa koeficijentima iz GF(2^8) i u tom obliku se množe sa konstantnim polinomom četvrtog stepena $a(x)=03x^3+01x^2+01x+02$ po modulu $m(x)=x^4+1$. Tj. $s'(x)=s(x) \cdot a(x)$. Izmenjena matrica stanja sastoji se od sledećih okteta:

  - $s'[0,k] = (02 \cdot s[0,k]) \oplus (03 \cdot s[1,k]) \oplus s[2,k] \oplus s[3,k]$
  - $s'[1,k] = s[0,k] \oplus (02 \cdot s[1,k]) \oplus (03 \cdot s[2,k]) \oplus s[3,k]$
  - $s'[2,k] = s[0,k] \oplus s[1,k] \oplus (02 \cdot s[2,k]) \oplus (03 \cdot s[3,k])$
  - $s'[3,k] = (03 \cdot s[0,k]) \oplus s[1,k] \oplus s[2,k] \oplus (02 \cdot s[3,k])$

- dodavanje potključa u matricu stanja (*AddRoundKey*)

  Funkcija *AddRoundKey* transformiše matricu stanja dodavanjem potključa pomoću operacije ekskluzivno ILI. Svaki potključ se sastoji se od N_w reči iz linearnog niza dobijenog proširenjem ključa. Te reči se dodaju na sledeći način:

  - $[s'[0,k], s'[1,k], s'[2,k], s'[3,k]] = [s[0,k], s[1,k], s[2,k], s[3,k]] \oplus [w_{runda*Nw+k}]$, za $0 \leq k < N_w$

  - w_i su reči niza dobijenog proširenjem ključa, promenljiva *runda* dobija vrednost tekuće runde algoritma ($0 \leq runda \leq Nr$).

Završna vrednost matrice stanja se kopira u izlazni šifrovani blok podataka.

Proširenje ključa

Ključ *k* se proširuje kako bi se generisali odgovarajući potključevi za operaciju *AddRoundKey*. Proširenjem ključa generiše se $(N_r+1)N_w$ 32-bitnih reči, koje se označavaju sa w_i ($0 \leq i < (N_r+1)N_w$). Potključevi se biraju iz proširenog ključa tako da se prvi međuključ sastoji od prve četiri reči, drugi od sledeće četiri.

Prve četiri reči proširenog ključa predstavlja ključ *k*. Svaka sledeća reč se dobija izvođenjem operacije ekskluzivno nad prethodnom reči i reči koja se nalazi N_k pozicija pre tekuće ($w_i=w_{i-1} \oplus w_{i-Nk}$). Ako je *i* deljivo sa N_k, onda se pre operacije ekskluzivno ILI nad reči w_{i-1} izvršavaju operacije:

- *RotWord*, koja rotira reč w_{i-1} jedno mesto ulevo, tako da se od reči $b_0b_1b_2b_3$ dobije reč $b_1b_2b_3b_0$,
- *SubWord*, koja na reč w_{i-1} primenjuje supstitucionu kutiju (tabela A.16 u dodatku A) i
- Ekskluzivno ILI reči w_{i-1} sa konstantom runde (konstanta *rconst[i]* runde *i* ima vrednost $[02^{i-1},00,00,00]$).

Dešifrovanje

Dešifrovanje se obavlja pomoću istog algoritma, s tim što se u rundama koriste funkcije inverzne funkcijama *ByteSub*, *ShiftRow* i *MixColumn*, i nepromenjena funkcija *AddRoundKey*.

Ukoliko algoritam za šifrovanje predstavimo sledećim pseudo-kodom:

```
stanje=ulaz;
AddRoundKey(stanje,w[0,Nw-1]);
for runda=1 to Nr-1 do {
   ByteSub(stanje);
   ShiftRow(stanje);
   MixColumn(stanje);
   AddRoundKey(stanje,w[runda*Nw,(runda+1)*Nw-1]);
}
ByteSub(stanje);
ShiftRow(stanje);
AddRoundKey(stanje,w[Nr*Nw,(Nr+1)*Nw-1];
```

```
izlaz=stanje;
```

algoritam za dešifrovanje će imati sledeći oblik:

```
stanje=ulaz;
AddRoundKey(stanje,w[Nr*Nw,Nr+1)*Nw-1]);
for runda=Nr-1 to 1 do {
   ShiftRowInv(stanje);
   ByteSubInv(stanje);
   AddRoundKey(stanje,w[runda*Nw,(runda+1)*Nw-1]);
   MixColumnInv(stanje);
}
ShiftRowInv(stanje);
ByteSubInv(stanje);
AddRoundKey(stanje,w[0,Nw-1]);
```

Funkcija *ByteSubInv* je inverzna funkciji *ByteSub*. Na primer, ulazni oktet F9h ova funkcija transformiše u 69h. Oktet 00h ostaje nepromenjen. Funkcija *ShiftRowInv* rotira oktete u redovima matrice stanja za jedno (drugi red), dva (treći red) i tri mesta udesno (četvrti red), s tim što se prvi red matrice (r=0) ne pomera. Funkcija *MixColumnInv* vrši transformacije u svakoj koloni matrice stanja - kolone se predstave kao polinomi četvrtog stepena *s(x)* sa koeficijentima iz GF(2^8) i množe sa konstantnim polinomom četvrtog stepena $a^{-1}(x)=0B9x^3+0Dx^2+09x+0E$ po modulu $m(x)=x^4+1$. Tj. $s'(x)=s(x) \cdot a^{-1}(x)$. Polinom $a^{-1}(x)$ je multiplikativni inverz polinoma *a(x)* kog koristi funkcija *MixColumn*. Izmenjena matrica stanja sastoji se od sledećih okteta:

- $s'[0,k] = (0E \cdot s[0,k]) \oplus (0B \cdot s[1,k]) \oplus (0D \cdot s[2,k]) \oplus (09 \cdot s[3,k])$
- $s'[1,k] = (09 \cdot s[0,k]) \oplus (0E \cdot s[1,k]) \oplus (0B \cdot s[2,k]) \oplus (0D \cdot s[3,k])$
- $s'[2,k] = (0D \cdot s[0,k]) \oplus (09 \cdot s[1,k]) \oplus (0E \cdot s[2,k]) \oplus (0B \cdot s[3,k])$
- $s'[3,k] = (0B \cdot s[0,k]) \oplus (0D \cdot s[1,k]) \oplus (09 \cdot s[2,k]) \oplus (0E \cdot s[3,k])$

Sigurnost AES algoritma

U ovom trenutku dužine ključeva za definisane AES-128, AES-192 i AES-256 standarde zadovoljavaju sigurnosne zahteve u većini primena. AES standard se temelji na RIJNDAEL algoritmu, koji omogućava šifrovanje i pomoću ključeva dužine veće od 256 bita (pod uslovom da je dužina ključa

deljiva sa 4), što omogućuje buduća proširenja standarda, ukoliko to bude potrebno. U algoritmu do sada nisu pronađeni nesigurni ili potencijalno nesigurni ključevi (kao što je na primer pronađeno 64 ključa kod DES-a). Otpornost RIJNDAEL algoritma na linearnu i diferencijalnu kriptoanalizu ukazuje na to da je prilikom razvoja algoritma posebna pažnja posvećena tome da sve moguće strategije napada imaju očekivano trajanje i memorijske zahteve identične ili veće u odnosu na napade na ostale algoritme koji šifruju blokove podataka iste dužine.

3.3 IDEA

IDEA (*International Data Encryption Algorithm*) je algoritam koji su razvili švajcarski kriptografi Xuejia Lai i James Massey. Prva verzija algoritma nazvana PES (*Proposed Encryption Standard*) objavljena je 1990. godine. PES nije bio otporan na diferencijalnu kriptoanalizu, tako da je algoritam prepravljen, a svoj konačni oblik je dobio 1992. godine. Dobra osobina IDEA algoritma je to što prilikom razvoja nije bilo mešanja nekih državnih institucija (kao što je NSA), tako da se ne sumnja u verodostojnost algoritma i postojanje *backdoor*-a. IDEA je patentiran algoritam i za komercijalnu upotrebu je potrebna odgovarajuća licenca. Koristi se, na primer, u PGP (*Pretty Good Privacy*) paketu kao simetrični algoritam.

IDEA koristi ključ dužine 128 bita za šifrovanje blokova otvorenog teksta dužine 64 bita. Prilikom šifrovanja, blok otvorenog teksta p dužine 64 bita najpre se deli na četiri podbloka dužine 16 bita: p_1, p_2, p_3, p_4. Šifrovanje se obavlja pomoću 8 rundi i završne transformacije. U njima se koristi 52 potključa dužine 16-bita (po šest u svakoj rundi i četiri u završnoj transformaciji) generisanih na osnovu polaznog ključa. Nad podblokovima dužine 16 bita obavljaju se sledeće tri operacije:

- ekskluzivno ILI,
- sabiranje po modulu 2^{16},
- množenje po modulu $2^{16}+1$ (može se posmatrati kao supstitucijska kutija).

Ove operacije ne zadovoljavaju zakone asocijativnosti i distributivnosti i

mogu se jednostavno softverski implementirati. Na kraju svake runde, zamenjuju se vrednosti u drugom i trećem podbloku.

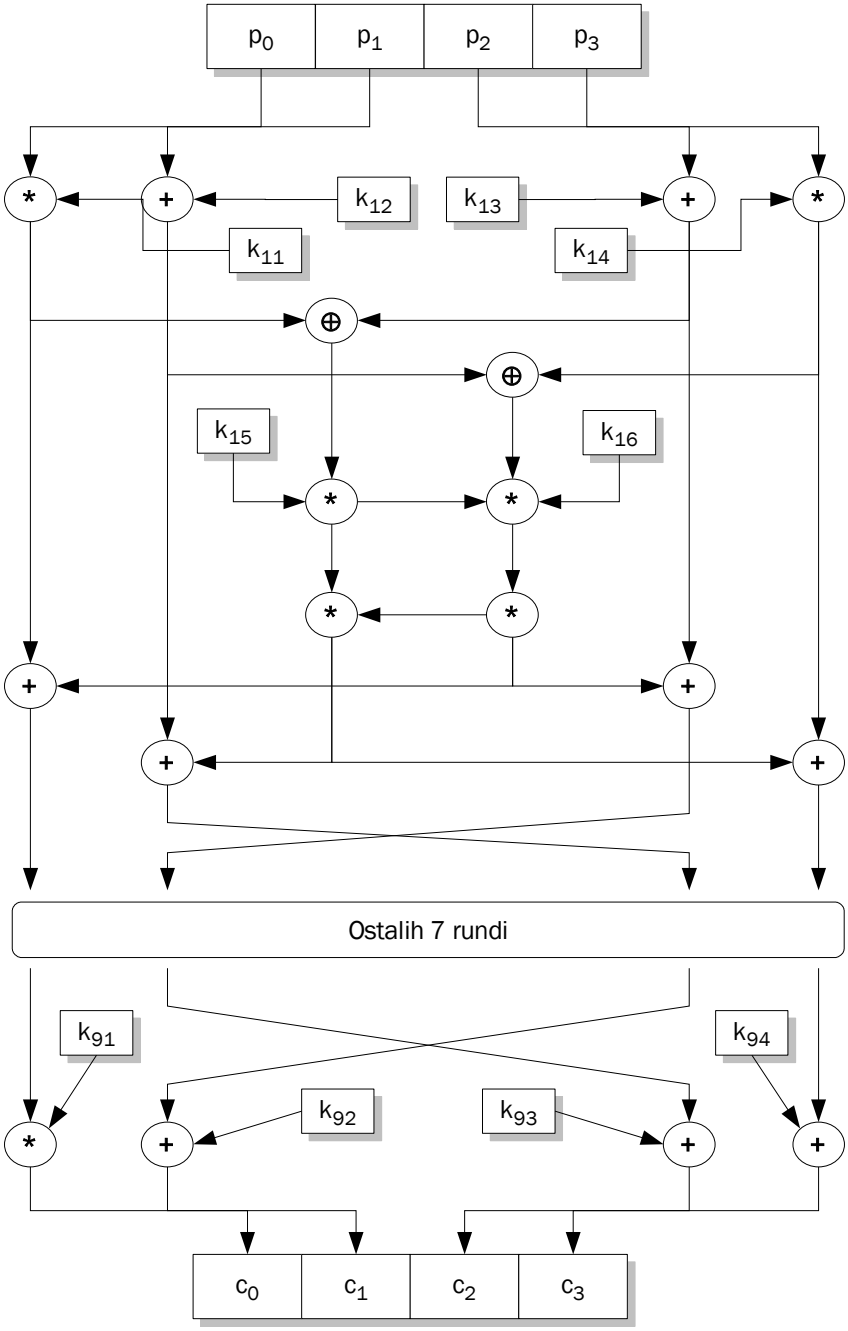
Posle osme runde dobijeni podblokovi prolaze kroz završnu transformaciju; šifrat se dobija konkatenacijom dobijenih podblokova C_1, C_2, C_3 i C_4.

Potključevi se generišu na osnovu polaznog ključa na sledeći način: ključ k dužine 128 bita se deli na osam 16-bitnih potključeva koji se koriste kao potključevi u prvoj rundi (k_{11}, k_{12}, k_{13}, k_{14}, k_{15}, k_{16}) i prva dva ključa druge iteracije (k_{21}, k_{22}). Zatim se bitovi ključa k ciklički pomere za 25 mesta u levo, čime se generiše sledećih osam potključeva koji koriste kao preostala četiri potključa druge iteracije (k_{23}, k_{24}, k_{25}, k_{26}) i četiri potključa treće iteracije (k_{31}, k_{32}, k_{33}, k_{34}). Postupak se nastavlja dok se ne dobiju potključevi potrebni za završnu transformaciju (k_{91}, k_{92}, k_{93}, k_{94}).

Algoritam je prikazan na slici 3.6. Dešifrovanje se obavlja identičnim algoritmom, ali se potključevi generišu na drugačiji način. Ako sa k_{ij} i k_{dij} označimo redom potključeve za šifrovanje i dešifrovanje, onda se postupak generisanja potključeva može opisati na sledeći način:

- Za i = 1 ... 8:
  - $k_{di1} = (k_{(10-i)1})^{-1}$, $k_{di2} = -k_{(10-i)2}$
  - $k_{di3} = -k_{(10-i)3}$, $k_{di3} = -(k_{(10-i)4})^{-1}$
  - $k_{di3} = k_{(9-i)5}$, $k_{di3} = k_{(9-i)5}$
- $k_{d91} = k_{11}^{-1}$, $k_{d92} = -k_{12}$, $k_{d93} = -k_{13}$, $k_{d94} = -k_{14}^{-1}$

Slično kao i kod DES-a, postoji određena klasa ključeva koji su slabi, tj. nesigurni i kao takvi se ne koriste. Sve ključeve oblika 0000 0000 0x000 0000 0000 000x xxxx 0x000, gde je x bilo koji heksadecimalni broj moguće je identifikovati pomoću napada odabran otvoreni tekst.



***Slika 3.6** – IDEA*

Sigurnost IDEA algoritma

IDEA se može koristiti u svim režimima rada (ECB, CBC, CFB, OFB). IDEA algoritam se pokazao prilično sigurnim u odnosu na druge simetrične algoritme. Uzevši u obzir dužinu ključa od 128 bita, napad "grubom silom" skoro da nije moguć, a sam algoritam je dizajniran tako da bude otporan na diferencijalnu i linearnu kriptoanalizu.

Slično kao i kod DES-a, dvostruko šifrovanje sa dva različita ključa ne uvećava sigurnost kriptosistema. Dvostruka IDEA je ranjiva na napad tipa "susret u sredini", ali je sam napad nepraktičan zbog 128-bitnog ključa (potrebno je 2^{128} operacija).

Viši nivo sigurnosti može se postići korišćenjem trostruke IDEA implementacije (slično trostrukom DES-u) sa ključem dužine 384 bita. Dodatno, ukoliko se koriste odgovarajući alati za upravljanje ključevima, moguća je implementacija IDEA algoritma sa nezavisnim potključevima. S obzirom na to da IDEA koristi 52 potključa dužine 16 bita, ukupna dužina ključa bi bila 832 bita.

3.4 Pitanja i zadaci

3.1 Sastavite program koji šifruje, odnosno dešifruje bilo koju datoteku pomoću DES algoritma. Funkcije koje realizuju šifrovanje jednog bloka otvorenog teksta DES algoritmom pronađite na Internetu.

Prvi parametar koji korisnik prenosi programu je šta želi da radi: da šifruje ili da dešifruje datoteku. U slučaju da se obavlja šifrovanje, korisnik kao ostale parametre unosi (1) ime datoteke u kojoj se nalazi otvoreni tekst, (2) ključ u heksadecimalnom obliku bez bitova parnosti i (3) režim rada algoritma (*ECB* ili *CBC*). Program zatim šifruje datoteku (pri čemu poslednji blok teksta dopunjuje slučajnim brojevima do dužine 64bita) i snima je na disk dodajući na ime datoteke ekstenziju *.sec*. Nakon šifrovanja program na disk dodatno snima i istoimenu datoteku sa ekstenzijom *.key* u koju smešta (1) ključ sa bitovima parnosti, (2) informacije o dužini otvorenog teksta, (3) režim rada algoritma koji korišćen pri šifrovanju.

U slučaju da se obavlja dešifrovanje, korisnik kao parametre unosi ime datoteke u kojoj se nalazi šifrat. Program proverava da li na disku postoji odgovarajuća datoteka sa ekstenzijom *.key* i da li je ključ ispravan (provera parnosti). Zatim se na osnovu informacija iz datoteke *.key* obavlja dešifrovanje i odsecaju slučajno dodati biti (dopuna poslednjeg bloka do 64 bita).

3.2 Izmenite program iz zadatka 3.1 tako da se šifrovanje obavlja AES algoritmom. Korisniku obezbedite mogućnost da unese ključ dužine 128, 192 ili 256 bita (shodno tome koristiti AES-128, AES-192 ili AES-256).

3.3 Izmenite program iz zadatka 3.1 tako da se šifrovanje obavlja IDEA algoritmom sa ključem dužine 128 bita.

3.4 Funkciju za šifrovanje iz programa 3.3 izmenite tako da dobijeni algoritam koristi nezavisnih 52 potključa dužine 16 bita (ključ je dužine 832 bita).

3.5 Ispitajte performanse DES, AES i IDEA algoritama. Koristeći programe iz zadatka 3.1 - 3.4 šifrujte datoteku dužine 100MB: (1) DES algoritmom, (2) AES-128, (3) AES-192, (4) AES-256, (5) IDEA algoritmom sa ključem dužine 128 bita, (6) IDEA algoritmom sa nezavisnim potključevima. Odredite vreme potrebno za šifrovanje datoteke za svaki algoritam. Koji je algoritam najbrži ?

3.6 Sastavite program koji računa potključeve za DES algoritam. Ispitajte koji se potključevi dobijaju ukoliko se koriste slabi ili delimično slabi ključevi.

3.7. Program iz primera 2.2 u dodatku B izmenite u funkciju cipher() koja će datoteke šifrovati pomoću XOR algoritma sa ključem dužine 24 bita. Šifrujte datoteku kompozicijom dve funkcije cipher() sa različitim ključevima. Probajte da napišete program koji će obaviti kriptoanalizu: (1) metodom grube sile nad poljem ključeva 2^{48}, (2) metodom susret u sredini. Koliko je operacija primene funkcije cipher() u najgorem slučaju potrebno za metodu susret u sredini ?

4

Generatori pseudoslučajnih sekvenci i protočno šifrovanje

4.1 Pseudoslučajne sekvence i testovi slučajnosti

Generator slučajnih sekvenci je uređaj ili algoritam koji na izlazu generiše sekvencu statistički nezavisnih binarnih brojeva. Sekvenca je slučajna samo ako je pri generisanju korišćen "prirodni" izvor slučajnosti. Prirodni izvori slučajnosti mogu biti:

- Hardverski (nestabilnosti frekvencije oscilatora, razlike u naponskim nivoima prilikom pražnjenja i punjenja kondenzatora, kašnjenje pri pomeraju glava diska za čitanje i pisanje, šum sa mikrofona, nestabilnost napona)

- Softverski (sistemski časovnik, pomeranje miša, sadržaj ulazno-izlaznog bafera, statistički izveštaji operativnog sistema o opterećenju sistema i mreže)

Generator slučajnih sekvenci je pogodan za generisanje ključa za jednokratnu beležnicu (engl. *one-time pad*).

Generator pseudoslučajnih sekvenci (engl. *pseudorandom number generator*, *PRNG*) je deterministički algoritam koji na osnovu slučajne ulazne sekvence[11], tj. ključa *k* generiše izlaznu sekvencu koja na prvi pogled izgleda kao da je slučajna. Generisana pseudoslučajna sekvenca je znatno duža od ključa. Generator se smatra pouzdanim ukoliko zadovoljava test sledećeg bita i neke statističke testove.

Jednostavan primer generatora pseudoslučajnih sekvenci je linearni kongruentni generator proizvodi pseudoslučajnu sekvencu x_1, x_2, x_3, ... na osnovu linearne rekurzije:

- $x_n = (a \cdot x_{n-1} + b) \bmod m, n>1$

Celi brojevi *a*, *b* i *m* su parametri generatora, a x_0 je ključ. Linearni kongruentni generatori zadovoljavaju statističke testove slučajnosti, ali su predvidljivi i samim tim nesigurni. Na osnovu jednog dela sekvence bez

11 Koristi se termin *seed* (seme) zato što cela sekvenca nastaje na osnovu ulazne slučajne sekvence, tj. "raste" iz nje kao biljka iz semena

poznavanja parametara *a*, *b* i *m* može se odrediti ostatak sekvence[12].

Pseudoslučajne sekvence mogu se generisati i pomoću jednosmernih funkcija – postupak je jednostavan ukoliko na raspolaganju imate jednosmernu heš funkciju (kao što su MD5 ili SHA-1) ili simetričan blokovski algoritam[13] sa ključem k_s (na primer, DES). Neka je *f* jednosmerna funkcija, a s_0 ključ. Sekvenca: $f(s_0)$, $f(f(s_0))$, $f(f(f(s_0)))$, ... je pseudoslučajna.

Složeniji generatori pseudoslučajnih sekvenci su ANSI X9.17, FIPS 186, RSA i x^2 mod n.

ANSI X9.17 se koristi za generisanje ključeva za DES algoritam i početnih vrednosti, tj. inicijalizacionih vektora za CBC režim rada. ANSI X9.17 generator u petlji obavlja šifrovanje pomoću trostrukog DES algoritma sa dva ključa (funkcija E^3_k). Početna 64-bitna vrednost *s* generiše se na osnovu trenutnog vremena i datuma, a zatim se obavi šifrovanje: $I=E^3_k(s)$. Dalje se u petlji obavljaju sledeće operacije:

- $x_i = E_k (I \oplus s)$
- $s = E_k (x_i \oplus I)$.

Izlaz iz generatora je *n* pseudoslučajnih 64-bitnih nizova x_1, x_2, ...x_m. Petlja se ponavlja dok se ne dobije sekvenca željene dužine.

FIPS 186 se koristi za generisanje privatnih ključeva za DSA algoritam (*Digital Signature Algorithm*). Navedeni generator se oslanja na FIPS 186 jednosmernu funkciju *G(t,c)* definisanu sledećim algoritmom:

- Ulazni nizovi bitova *t* i *c* dužine 160-bita rastavljaju se na po pet 32-bitnih podnizova t_0, t_1, ... t_4 i c_0, c_1, ... c_4, a zatim se nad njima računa rezultat operacije ekskluzivno ILI, tj. $x_i = t_i \oplus c_i$, za i=0,...,4.

- U petlji se za i=0,...,4 radi sledeće:

12 Videti Jim A. Reeds – "*Cracking Random Number Generators*", Joan Boyar – "*Inferring a Sequence Generated by Linear Congruence*"

13 U slučaju da se kao jednosmerna funkcija koristi simetričan blokovski algoritam, cela pseudoslučajna sekvenca se može lako nastaviti ukoliko se otkrije ključ k_s. Takođe se konkatenacijom operacija dešifrovanja može doći i do polazne vrednosti, tj. do ključa s_0.

- $b_1 = c_{(i+4) \bmod 5}$, $b_2 = c_{(i+3) \bmod 5}$.
- $a_1 = x_i$, $a_2 = x_{(i+1) \bmod 5} \oplus x_{(i+4) \bmod 5}$.
- Određuje se konkatenacija nizova A = a_1 a_2 i B = $b_1^{(LSB24)}$ b_2, gde su $b_1^{(LSB24)}$ 24 najniža bita niza b_1.
- Niz A se šifruje nizom B pomoću DES algoritma: $y_i = E_B(A)$
- Niz y_i se rastavlja u dva 32-bitna podniza: L_i i R_i.

- U petlji se za i=0,...,4 određuje: $z_i = L_i \oplus R_{(i+2) \bmod 5} \oplus L_{(i+3) \bmod 5}$.
- Izlaz iz funkcije G(t, c) je 160-bitni niz koji se dobija kao konkatenacija podnizova z_0 z_1 z_2 z_3 z_4.

Ulazni podatak FIPS 186 generatora pseudoslučajnih sekvenci je prost 160-bitni broj *q*. Za interne potrebe generatora, definiše se parametar *t* koji ima sledeću vrednost:

- t = 67452301 efcdab89 98badcfe 10325476 c3d2e1f0 $_{hex}$

Najpre se pomoću nekog izvora slučajnosti generiše slučajna početna vrednost *s*, a zatim se u petlji i=1,...n (pri čemu je *n* određeno željenom dužinom sekvence), obavljaju sledeće operacije:

- $z_i = (s + y_i) \bmod 2^b$ (parametar y_i služi za prikupljanje slučajnosti – korisnik unosi niz y_i dužine *b* bitova; alternativno se može koristiti i vrednost y_i=0)
- $a_i = G(t, z_i) \bmod q$ (primena FIPS 186 jednosmerne funkcije)
- $s = (1 + s + a_i) \bmod 2^b$.

Izlazna sekvenca je niz od n brojeva a_1, ... a_m na intervalu $0 \le a_i \le q-1$.

RSA generator pseudoslučajnih brojeva radi na sledeći način: najpre se generišu dva prosta broja *p* i *q* i odrede vrednosti n=pq i φ=(p − 1)(q − 1). Zatim se bira slučajni prirodni broj *e* na intervalu 1<e<φ, takav da je najveći zajednički delilac za *e* i *φ* NZD(e,φ)=1. Na intervalu [1, n-1] se bira slučajan broj x_0 (*seed*). U petlji i=1,...l (pri čemu je *l* određeno željenom dužinom sekvence) se obavljajaju sledeće operacije:

- $x_i = x_{i-1}^e \bmod n$
- $z_i = LSB(x_i)$ [14]

Izlazna sekvenca dužine l je niz bitova: $z_1 z_2 z_3 \ldots z_l$.

Za $x^2 \bmod n$ generator (Blum-Blum-Schub) najpre se generišu dva prosta broja *p* i *q*, koji su kongruentni sa 3 po modulu 4 (tj. prilikom deljenja sa 4 daju ostatak 3). Odredi se n=pq i bira broj s (seed) na intervalu [1, n−1] takav da je NZD(s,n)=1. Određuje se i $x_0=s^2 \bmod n$. Zatim se u petlji i=1,...l (pri čemu je *l* određeno željenom dužinom sekvence) obavljaju sledeće operacije:

- $x_i=x_{i-1}^2 \bmod n$
- $z_i = LSB(x_i)$

Izlazna sekvenca dužine l: z1z2z3...zl

Sigurnost RSA i $x^2 \bmod n$ generatora pseudoslučajnih sekvenci zasnovana je na tajnosti prostih brojeva *p* i *q*.

Testovi slučajnosti

Statistički testovi slučajnosti su testovi kojima se meri kvalitet generatora pseudoslučajnih sekvenci. Nemoguće je matematički dokazati da li su generisane sekvence slučajne ili ne – testovima se otkrivaju slabosti koje generator ima.

Uzorak pseudoslučajne sekvence propušta se kroz seriju statističkih testova u kojima se traže atributi koji nisu karakteristični za slučajne sekvence. Na primer, proverava se da li binarna sekvenca ima znatno veći broj nula od broja jedinica. Generator koji ne zadovolji jedan od testova deklariše se kao ne-slučajan i testiranje se prekida. U suprotnom, testiranje se nastavlja. Ukoliko generator zadovolji sve testove prihvata se kao slučajni generator (preciznije: ne odbacuje se).

Opisaćemo dve metode za testiranje slučajnosti: Golombove postulate i

14 z_i je najniži bit podniza x_i

FIPS 140-1 statistički test. Pre toga navodimo nekoliko neophodnih definicija.

Monotona sekvenca, tj. monoton podniz (engl. *run*) je podniz uzastopnih nula ili jedinica u nizu. Podniz uzastopnih jedninica naziva se blok, a podniz uzastopnih nula šupljina (engl. *gap*).

Sekvenca $s = s_0\, s_1\, s_2 \ldots$ je N-periodična ako je $s_i = s_{i+N}$ za svako $i>0$. Sekvenca je periodična ako je N-periodična za neko pozitivno N.

Autokorelaciona funkcija: $c(t) = N^{-1} \sum_{i=0,N-1} (2s_i - 1)(2s_{i+t} - 1)$ služi za merenje sličnosti između sekvence *s* i sekvence *s* pomerene za *t* mesta. Ako je sekvenca *s* N-periodična slučajna sekvenca onda je vrednost autokorelacione funkcije NC(t) mala za sve vrednosti t na intervalu $0<t<N$.

PN sekvenca (engl. *pseudo-noise*) je binarna sekvenca koja zadovoljava sledeća tri Golombova postulata slučajnosti:

- broj nula i jedinica u sekvenci se razlikuje najviše za 1,
- najmanje 50% monotonih sekvenci ima dužinu 1, najmanje 25% monotonih sekvenci ima dužinu 2, namanje 12.5% monotonih sekvenci ima dužinu 3 itd,
- autokorelaciona funkcija *c(t)* ima dve vrednosti: $c(t)=N$, ako je $t=0$, i $c(t)=K$, ako je $0<t<N$.

Primer 4.1. Odredite da li je s_{15} = 011001000111101 PN sekvenca.

Broj nula je 7, broj jedinica je 8, pa je prvi postulat zadovoljen. U sekvenci s_{15} ima 8 monotonih podnizova: četiri dužine 1 (dve šupljine i dva bloka), dva dužine dva (jedna šupljina i jedan blok), jedan dužine tri (šupljina) i jedan dužine četiri (blok). To znači da je i drugi postulat zadovoljen. Vrednosti korelacione funkcije su $c(0)=1$ i $c(t)=-1/15$ $(1\leq t\leq 14)$ pa je i treći postulat zadovoljen, što znači da je sekvenca s_{15} PN sekvenca.

Prema FIPS 140-1, generator se može prihvatiti kao generator slučajnih sekvenci ukoliko generisana pseudoslučajna sekvenca s dužine n=20000 bitova zadovoljava sledeće testove:

- Monobitni test (odnos broja "0" i "1"). Broj jedinica u sekvenci *s* mora pripadati intervalu $9654<n_1<10346$.

- Poker test (učestanost pojavljivanja različitih sekvenci dužine *m*) za sekvence dužine m=4. Sekvenca *s* se deli na 5000 delova dužine 4. Definišu se n_1, n_2, ..., n_{16} kao brojevi pojavljivanja određenog tipa sekvence dužine 4 (ukupno imamo 24=16 mogućih sekvenci dužine 4). Određuje se: $X = {}^{16}/_{5000}(\sum_{i=1,\ 16} ni^2) - 4$, a sekvenca prolazi poker test ukoliko dobijena vrednost pripada intervalu $1.03<X<57.4$.

- Određuju se broj blokova B_i i broj šupljina G_i dužine i=1,2,...,6, pri čemu se monotonim sekvencama dužim od 6 formalno pripisuje dužina 6. Sekvenca *s* prolazi test ako se vrednosti B_i i G_i nalaze u sledećim intervalima: B_1,G_1 (2267−2733), B_2,G_2 (1079−1421), B_3,G_3 (502−748), B_4,G_4 (223−402), B_5,G_5 (90−223), B_6,G_6 (90−223).

- Test dugačkih sekvenci (engl. *long run test*). Ne sme biti monotonih sekvenci dužine veće od 34.

4.2 Protočno šifrovanje

Pomoću blokovskih algoritama otvoreni tekst se šifruje u blokovima fiksne dužine. Ukoliko dužina otvorenog teksta nije umnožak dužine bloka, zadnji blok se dopunjuje pseudoslučajnom sekvencom bitova. Na primer, ukoliko se otvoreni tekst dužine 620 bita šifruje DES algoritmom (veličina bloka 64 bita), zadnji blok se mora dopuniti pseudoslučajnom sekvencom dužine 20 bita. Pri tome se mora sačuvati i informacija o dužini poruke ili dužini sekvence kojom je poruka dopunjena.

Protočno šifrovanje rešava ovaj problem – poruka se šifruje bit po bit (ili karakter po karakter) koristeći transformaciju koja se obično menja u vremenu. Nema dopunjavanja poruke generisanim pseudoslučajnim sekvencama – šifrat poruke duge 753 bita je dužine 753 bita. Hardverski realizovano protočno šifrovanje je brže i u odnosu na blokovsko šifrovanje zahteva hardver manje složenosti. Protočno šifrovanje se koristi u situacijama u kojima blokovsko šifrovanje ne može da se koristi. Primer za to su telekomunikacije: ukoliko se karakteri obrađuju pojedinačno a

ulazno/izlazni bafer primopredajnog uređaja je ograničen, blokovsko šifrovanje se ne može primeniti. Protočni kriptosistemi imaju malu propagaciju greške duž šifrata.

Jednostavan primer protočnog šifrovanja je DES algoritam koji radi u CFB ili OFB režimu rada. Još jedan jednostavan protočni kriptosistem može se formirati pomoću generatora pseudoslučajnih sekvenci: šifrat se dobija kao rezultat operacije ekskluzivno ILI primenjene nad otvorenim tekstom i pseudoslučajnom sekvencom generisanom pomoću ključa *k* (tzv. ključna sekvenca). Ukoliko se ista operacija primeni nad šifratom i sekvencom, dobija se otvoreni tekst. Ista pseudoslučajna sekvenca može se generisati dva puta bez ikakvih problema jer su pseudoslučajni generatori deterministički, tj. na osnovu jednog ključa uvek generišu istu sekvencu.

Protočni kriptosistemi se dele na sinhrone i asinhrone. U slučaju sinhronih, ključna sekvenca (engl. *keystream*) se generiše nezavisno od otvorenog teksta i šifrata. Za sinhroni kriptosistem važi:

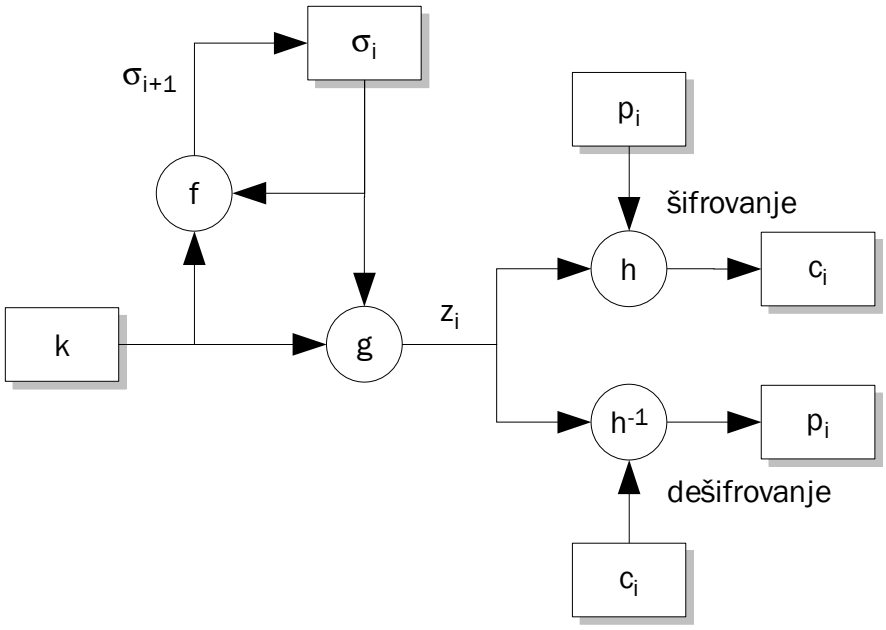
- $\sigma_{i+1} = f(\sigma_i, k)$,
- $z_i = g(\sigma_i, k)$,
- $c = h(z_i, m_i)$.

Funkcija *f* prevodi sistem iz stanja σ_i u sledeće stanje σ_{i+1}, pri čemu je σ_0 inicijalno stanje koje se izvodi iz ključa *k*. Funkcija *g* generiše ključnu sekvencu na osnovu trenutnog stanja i ključa, a izlazna funkcija *h* generiše šifrat na osnovu otvorenog teksta i ključne sekvence.

Dešifrovanje se obavlja slično, s tim što se prilikom generisanja otvorenog teksta na osnovu šifrata koristi inverzna funkcija h^{-1}:

- $\sigma_{i+1} = f(\sigma_i, k)$,
- $z_i = g(\sigma_i, k)$,
- $c = h(z_i, m_i)$.

Proces šifrovanja i dešifrovanja prikazan je na slici 4.1.

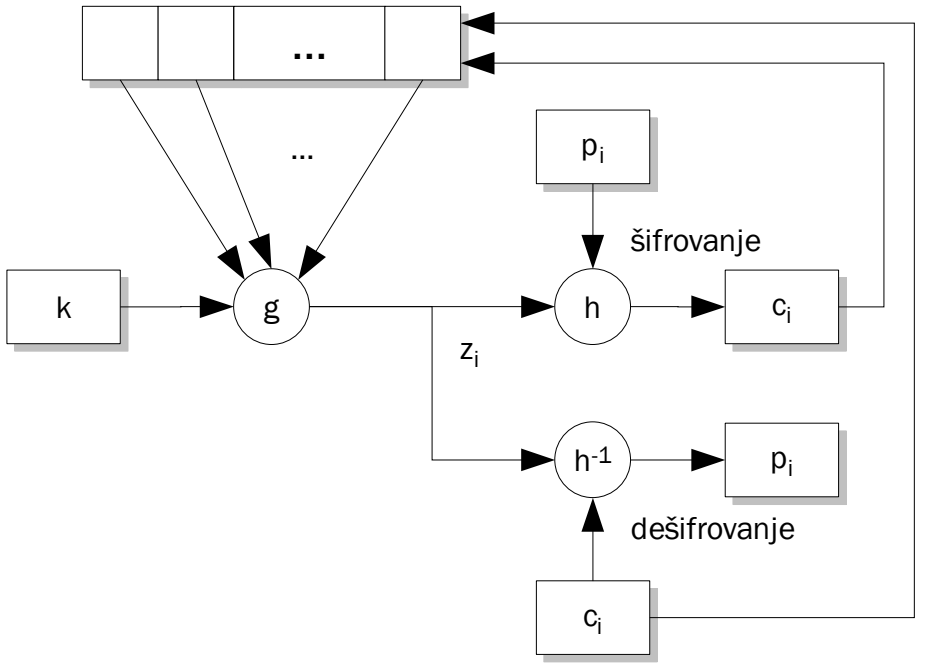


Slika 4.1 *– Sinhrono protočno šifrovanje*

Za sinhrono protočno šifrovanje karakterističan je problem sinhronizacije – pošiljalac i primalac moraju biti sinhronizovani po pitanju ključa i stanja. U slučaju gubitka sinhronizacije dešifrovanje je nemoguće. To znači da je slaba tačka sinhronih protočnih kriptosistema mogućnost izvođenja aktivnog napada umetanjem besmislenog teksta u poruku, čime se sprečava komunikacija. Da bi se dešifrovanje moglo obaviti u slučaju gubitka sinhronizacije, u kriptosistem se ugrađuju posebni re-sinhronizacioni mehanizmi.

Još jedna karakteristika sinhronih protočnih kriptosistema je nepostojanje propagacije greške duž šifrata. Ukoliko je izlazna funkcija *h* relativno prosta (npr. ekskluzivno ILI), napadač može namerno modifikovati šifrat tako da primaoc dobije modifikovanu poruku. To znači da je potrebno implementirati autentifikacioni mehanizam i na neki način osigurati integritet podataka.

U slučaju asinhronih (samo-sinhronišućih), kriptosistema (slika 4.2), ključna sekvenca se generiše na osnovu ključa i fiksnog broja prethodnih bitova šifrata.



Slika 4.2 *– Asinhrono protočno šifrovanje*

- $\sigma_i = f(\sigma_{i-t}, \sigma_{i-t+1}, \sigma_{i-t+2}, \ldots \sigma_{i-1})$
- $z_i = g(\sigma_i, k)$
- $c_i = h(z_i, m_i)$

Za funkcionisanje asinhronog protočnog šifrovanja problem gubitka sinhronizacije nije velikog značaja. Kako se ključna sekvenca generiše na osnovu ključa i fiksnog broja prethodnih bitova šifrata, kriptosistem je sposoban da sam ponovo uspostavi sinhronizaciju u slučaju da napadač umetne ili obriše bitove šifrata.

Za razliku od sinhronog protočnog šifrovanja, propagacija greške duž šifrata postoji, zato što *n* bitova šifrata utiče na generisanje ključne sekvence. Ukoliko napadač modifikuje jedan bit šifrata, *n* sledećih bitova će se neispravno dešifrovati.

Linearni pomerački registar sa povratnom spregom

Pomerački registar sa povratnom spregom (engl. *feedback shift register, FSR*) je registar kod koga se prelazak u sledeće stanje ostvaruje pomoću sledeće dve operacije:

- kružni pomeraj, odnosno rotacija bitova registra za 1 bit udesno,
- generisanje najznačajnijeg bita (engl. *most significant bit*) na osnovu funkcije povratne sprege čiji su argumenti ostali bitovi registra.

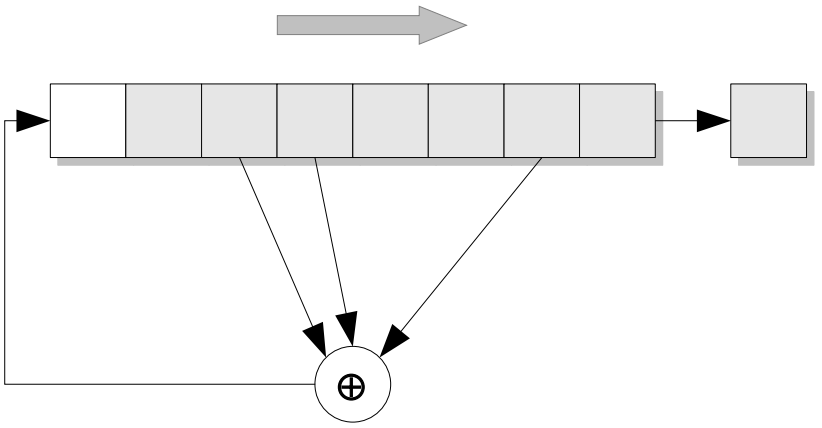
Najmanje značajan bit (engl. *least significant bit*) je izlaz iz registra – na taj način se uzastopnim prelaskom u sledeće stanje može generisati sekvenca. Perioda FSR je broj bitova nakon kojih sekvenca počinje da se ponavlja. Portočni kriptosistemi se mogu lako implementirati u hardveru ukoliko se koriste ovi registri.

Linearni pomerački registar sa povratnom spregom (engl. *linear feedback shift register, LFSR*) je pomerački registar kod kog se povratna sprega realizuje pomoću operacije ekskluzivno ILI nad određenim bitovima (bitovi poznati pod imenom *tap sequence*). LFSR je prikazan na slici 4.3.

Primer 4.2. Odrediti izlaznu sekvencu i periodu 4-bitnog LFSR ukoliko *tap* sekvencu čine prvi i četvrti bit.

Pretpostavićemo da se u LFSR inicijalno nalazi 1111 (može se uzeti i bilo koja druga vrednost). LFSR prolazi kroz sledeća stanja: 1111, 0111, 1011, 0101, 1010, 1101, 0110, 0011, 1001, 0100, 0010, 0001, 1000, 1100, 1110, 1111. Nakon toga sledi druga perioda, jer je LFSR doveden u inicijalno stanje. Izlazna sekvenca LFSR je: 111101011001000, a perioda registra 15.

Idealni n-bitni LFSR može se naći u 2^n-1 internih stanja. Teoretski, LFSR može da generiše pseudoslučajnu sekvencu dužine 2^n-1 bitova; nakon toga se sekvenca ponavlja. Pseudoslučajnu sekvencu dužine 2^n nemoguće je generisati – ukoliko se LFSR napuni nulama na početku, na izlazu se dobija beskonačan niz nula. Realni LFSR generiše m-sekvencu, odnosno sekvencu dužine *m* bitova, pri čemu je $m<2^n-1$.



Slika 4.3 *– Linearni pomerački registar sa povratnom spregom*

LFSR je u *Galois* konfiguraciji ukoliko se nakon rotacije nad svakim bitom *tap* sekvence i izlazom registra izvrši operacija ekskluzivno ILI i rezultat upiše na mesto tog bita. se Najmanje značajan bit postaje najznačajniji. Ovakav LFSR nije kriptografski bolji, ali se lako realizuje u softveru i daje dobru periodu.

RC4

RC4[15] je simetrični protočni algoritam sa ključem promenljive veličine. Iako je "javno dostupan" (anonimno poslat na Cypherpunks mailing listu i proširen na Usenet newsgrupu sci.crypt) počev od 1994. godine – što znači da ga možete naći u svakoj knjizi koja se iole ozbiljnije bavi kriptografijom – algoritam je patentiran. Ukoliko želite da ga koristite, moraćete da kupite odgovarajuću licencu od RSA Data Security, Inc (ili da nakon izvesnog vremena platite kaznu za zloupotrebu intelektualne svojine i, naravno, sudske troškove). Algoritam nije komplikovan i može se jednostavno opisati.

RC4 radi u OFB režimu rada. Ključna sekvenca se generiše nezavisno od otvorenog teksta. RC4 sadrži 8*8 supstitucijskih kutija (S_0,S_1,...,S_{255}) koje u svakom trenutku predstavljaju permutaciju brojeva 0,1,...,255. Inicijalno se popunjavaju vrednostima S_i=i, tj. S_0=0, S_1=1,..., S_{255}=255. Zatim se niz od 256 bajtova K_0, K_1,..., K_{255} puni ciklički ključem (ključ se ponavlja kako bi

15 RC je najverovatnije skraćenica od "Rivest Cipher"

se popunio ceo niz). Promenljivoj *j* se dodeli vrednost 0, a zatim se u petlji i=0,255 radi sledeće:

- $j = (j + S_i + K_i) \bmod 256$,
- S_i and S_j zamenjuju vrednosti.

Slučajni bajt se generiše na sledeći način:

- $i = (i + 1) \bmod 256$,
- $j = (j + S_i) \bmod 256$,
- S_i i S_j zamenjuju vrednosti,
- $t = (S_i + S_j) \bmod 256$,
- $K = S_t$.

Jedan bajt šifrata se dobija kao rezultat ekskluzivno ILI operacije primenjene nad bajtom ključne sekvence K i bajtom otvorenog teksta. Slično se dobija i otvoreni tekst – operacija ekskluzivno ILI se primeni nad šifratom i ključnom sekvencom. Šifrovanje RC4 algoritmom je oko 10 puta brže od šifrovanja DES algoritmom.

Perioda generisane sekvence relativno velika, a sam algoritam je većim delom nelinearan. RSA Data Security tvrdi da je algoritam otporan na linearnu i diferencijalnu kriptoanalizu. Prethodno opisana varijanta je poznata kao 8-bitni RC4; 16-bitna varijacija algoritma zahteva 65536 operacija za inicijalizaciju supstitucijskih kutija – ali je sam algoritam generisanja ključne sekvence brži.

4.3 Pitanja i zadaci

4.1 Sastavite program koji generiše pseudoslučajne sekvence koristeći linearnu rekurziju. Dužinu sekvence (broj iteracija), parametre generatora *a*, *b* i *m* i početnu vrednost x_0 unosi korisnik.

4.2 Sastavite program koji generiše RSA pseudoslučajne sekvence. Slučajnu ulaznu sekvencu (seed) generišite na osnovu tekućeg

vremena. Dužinu sekvence (broj generisanih bitova) unosi korisnik.

4.3 Izmenite program iz prethodnog zadatka tako da se pseudoslučajne sekvence generišu po algoritmu x^2 mod n.

4.4 Koristeći gotovu MD5 heš funkciju (izvorni kod možete naći na Internetu) realizujte generator pseudoslučajnih sekvenci. Početnu vrednost s_0 i dužinu sekvence[16] zadaje korisnik, a sekvenca se generiše kao konkatenacija nizova:

- md5(s0), md5(md5(s0), md5(md5(md5(s0)))...

4.5 Sastavite program koji ispituje pseudoslučajne sekvence pomoću FIPS 140-1 statističkog testa slučajnosti. Pomoću generatora iz zadataka 4.1, 4.2, 4.3 i 4.4 generišite pseudoslučajne sekvence dužine 20000 karaktera i proverite da li sekvence zadovoljavaju test.

4.6 Sastavite program za protočno šifrovanje otvorenog teksta koji ključnu sekvencu generiše pomoću kompozicije MD5 heš funkcija (zadatak 4.3), a kao funkciju *h* koja na osnovu ključne sekvence i otvorenog teksta generiše šifrat koristi ekskluzivno ILI. Korisnik unosi sledeće parametre: šta želi da radi (da šifruje ili da dešifruje datoteku), imena datoteka u kojima se nalaze otvoreni tekst i šifrat i ključ.

4.7 Sastavite program koji simulira rad 16-bitnog linearnog registra sa povratnom spregom. Korisnik kao parametre zadaje početnu vrednost generatora i određuje bitove koji čine *tap* sekvencu. Program određuje periodu LFSR i na ekranu ispisuje jednu periodu generisane sekvence.

4.8 Koristeći program iz zadatka 4.7 odredite *tap* sekvence 16-bitnog LFSR za koje generator ima najveću periodu.

16 MD5 proizvodi heš dužine 128 bita, što znači da generator za n iteracija proizvodi pseudoslučajnu sekvencu dužine n*128 bitova.

5

Jednosmerne heš funkcije

5.1 Šta je heš, a šta jednosmerna funkcija ?

Jednosmerna funkcija (engl. *one-way*) je funkcija oblika y=f(x) takva da važi:

- za dato *x*, *f(x)* se određuje relativno lako i efikasno i
- za dato y=f(x), $x=f^{-1}(y)$ se određuje relativno teško.

Da ne shvatite pogrešno: to što je rečeno da se "$f^{-1}(y)$ teško određuje" ne znači da je nemoguće odrediti *x* na osnovu poznatog *y*, već da je za to potrebno nekoliko miliona godina ukoliko se koristi procesorska snaga svih računara u svetu.

Jednosmernost se može jednostavno objasniti na primeru koji nije vezan za kriptografiju. Ako slomite tanjir (promenljiva *x*), dobićete parčiće keramike - *f(x)*. Ovo se lako radi, u šta se možete i sami uveriti. Ako iz *f(x)* pokušate da dobijete *x* (rekonstrukcija tanjira), potrebno je malo više vremena (i lepka).

Ne može se matematički dokazati da jednosmerne funkcije postoje. Ukoliko se funkcija efikasno izračunava, a vrednost inverzne funkcije $f^{-1}(y)$ relativno teško nalazi, funkcija se uzima u obzir za dalje razmatranje. Na primer, vrednost funkcije $f(x)=x^2$ u konačnom polju se relativno lako određuje. Međutim, $f^{-1}(y)=x^{1/2}$ se nalazi teško.

Jednosmerna funkcija sa zamkom, tj. privatna jednosmerna funkcija (engl. *trapdoor one-way*) je funkcija za koju važi:

- za dato *x*, *f(x)* se određuje relativno lako i efikasno,
- za dato y=f(x), $x=f^{-1}(y)$ se određuje relativno teško,
- za dato y=f(x) i tajnu informaciju *z* (zamka), $x=g(f^{-1}(y),z)$ se određuje relativno lako i efikasno.

Na primer, posmatrajte ručni sat (y). Rasklapanje časovnika u delove y=f(x) je jednostavan posao. Sklapanje časovnika iz delova, tj. određivanje $x=f^{-1}(y)$ je vremenski zahtevno i komplikovano, ali se može relativno brzo

obaviti ako na raspolaganju imate uputstvo za sklapanje *z*.

Heš funkcija (engl. *hash*) pretvara ulazni podatak promenljive dužine (engl. *pre-image*) u izlazni podatak fiksne dužine – heš. Jednostavan primer je računanje vrednosti operacije ekskluzivno ILI nad svim bajtovima poruke. Bez obzira na dužinu poruke, kao rezultat se dobija jedan bajt. Heš funkcije se koriste za utvrđivanje integriteta poruke. Posmatrajte heš kao otisak prsta (engl. *fingerprint*) – prilikom slanja poruke, pošiljaoc šalje i heš poruke. Na osnovu otiska primaoc može odrediti da li je poruka koju je primio stigla u originalnom ili izmenjenom obliku. Heš funkcija je preslikavanje tipa više u jedan – beskonačan skup poruka preslikava se u konačan skup heš vrednosti (kardinalnost skupa je 2^n, pri čemu je *n* broj bitova u hešu). Međutim, iako primaoc ne može biti 100% u verodostojnost poruke, verovatnoća da napadač izmeni poruku tako da izmenjena i originalna poruka generišu isti heš je vrlo mala.

Heš funkcije se dele na jednoparametarske (ulazni argument je samo poruka) i dvoparametarske (ulazni argumenti su poruka i tajni ključ). U praksi se pojavila i drugačija podela heš funkcija, zasnovana na specifičnoj primeni pojedinih funkcija. Prema funkcionalnoj podeli, heš funkcije se dele na:

- Mehanizme za uočavanje promena (engl. *modification detection codes*, *MDC*). Osnovna uloga funkcije je da obezbedi sažetak poruke kojim se poruka može jednoznačno identifikovati u daljoj obradi. Obično se ovaj metod koristi zajedno sa dodatnim metodima za obezbedjivanje integriteta podataka.

- Mehanizme za autentifikaciju poruka (engl. *message authenticaton codes, MAC*). MAC su heš funkcije koje na osnovu dva funkcionalno nezavisna ulaza (poruka i tajni ključ) proizvode heš. Sistem je projektovan tako da bude skoro nemoguće dobiti originalni heš bez poznavanja tajnog ključa. Koriste se prilikom utvrdjivanja porekla poruke. Ova klasa funkcija je podklasa dvoparametarskih heš funkcija.

Više o jednosmernim heš funkcijama

Jednosmerna heš funkcija[17] h = H(m) je preslikavanje za koje važi sledeće:

- na osnovu ulaznog podatka *m* ma koje dužine, heš *h* fiksne dužine *n* se lako i efikasno određuje
- na osnovu heš vrednosti *h*, odgovarajući ulazni podaci m_1,m_2,... se ne mogu odrediti ili se određuju teško i neefikasno.

Heš funkcije su zasnovane na ideji kompresije. Kompresijom se dobija blok manji od ulaznog podatka. Ulaz za funkciju su sledeći blok poruke i vrednost funkcije primenjene na prethodnom bloku. Tj, heš vrednost bloka m_i je $h_i=f(m_i,h_{i-1})$, što znači da na vrednost h_i utiču svi blokovi do bloka m_i. Heš poslednjeg bloka je heš poruke.

Heš funkcije su preslikavanja više-na-jedan i kao takve nisu imune na pojavu kolizija ili "sudara". Kolizija (engl. *collision*) je pojava kada dva (ili više) različitih ulaza rezultuju istim izlazom. Drugim rečima, postoji šansa da dve različite ulazne poruke rezultuju identičnim izlazom. Ovo predstavlja veliki problem ukoliko se heš funkcije koriste u okviru mehanizma autentifikacije. Međutim, verovatnoća da dva slučajno odabrana ulaza proizvedu isti heš dužine *n* je 2^{-n}. Dobra heš funkcija je "oslobođena kolizije", tj. teško se generišu dve poruke na osnovu kojih se proizvodi isti heš H(m)=H(m'). Kod dobrih heš funkcija, promena jednog bita u ulaznom podatku rezultuje promenom najmanje polovine bitova izlaza. U opštem slučaju, algoritam koji opisuje heš funkciju se ne skriva, a sigurnost funkcije zavisi od njene jednosmernosti.

Heš funkcije se pominju pod različitim imenima: funkcije sažimanja (engl. *message digest*), funkcija kreiranja otiska prsta, kriptografska ček-suma. Veoma su značajne za kriptografiju i primenjuju se:

- u kriptografskim protokolima,
- za digitalno potpisivanje i proveru integriteta poruke,
- za autentifikaciju.

17 U daljem tekstu se podrazumeva jednosmernost i koristi termin heš funkcija.

Dužina heša

Jedan od parametara koji utiču na odabir heš funkcije je dužina proizvedene heš vrednosti. 64-bitni heš je prekratak, što se može ilustrovati jednostavnim primerom rođendanskog napada:

- Ana priprema dve verzije ugovora. Jedan je ispravan (*goodfile*), drugi dovodi firmu čiji je vlasnik Boban u bankrot (*badfile*), ali njoj uvećava sumu na računu u banci.
- Ana pravi par kozmetičkih izmena u oba dokumenta (dodaje ili briše blanko karaktere, prazne linije, poneku tačku ili zarez) generišući ukupno 2^{32} verzije oba dokumenta.
- Ana upoređuje heš vrednosti dokumenata i nalazi par dokumenta (ispravan – "malo manje" ispravan) sa istim hešom.
- Ana i Boban potpisuju dokument koristeći protokol za digitalno potpisivanje u kome Boban potpisuje heš dokumenta. Pri tome, Boban potpisuje ispravan dokument (*goodfile*).
- Nakon par dana, Ana može na sudu da prikaže dokazni materijal u vidu neispravnog ugovora (*badfile*) i dokaže da je Boban potpisao heš vrednost tog ugovora.

Ukoliko sumnjate da je generisani heš prekratak, a imate poverenja u heš funkciju, dužinu proizvedenog heša možete uvećati pomoću sledećeg algoritma:

Heš dužine 128 bita je prihvatljiv – napadač koji primenjuje napad zasnovan na rođendanskom paradoksu mora da računa heš 2^{64} različitih dokumenata (što je vremenski mnogo zahtevnije od računanja 2^{32} heša) kako bi našao dva sa istim heš vrednostima. MD2, MD4 i MD5 proizvode heš dužine 128 bita. SHA i RIPEMD-160 proizvode heš dužine 160 bita, što odgovara dužini koju je NIST propisao za SHS (*secure hash standard*).

- $h_1 = H(m)$
- $h_2 = h_1 | H(h_1 | m)$
- $h_3 = h_2 | H(h_2 | m)$

Postupak se nastavlja dok se ne dobije hash željene dužine. Na primer. ako funkcija H proizvodi heš dužine 128 bita, koristeći ovaj algoritam sa 8 iteracija dobićete heš dužine 1024 bita.

5.2 Značajnije heš funkcije

MD2, MD4 i MD5 algoritme razvio je Ronald Rivest za RSA Data Security, Inc. Sva tri agoritma proizvide 128-bitni heš, s tim što je MD2 prilagođen za 8-bitnim mikroprocesorima, dok su MD4 i MD5 prilagođeni 32-bitnim računarima. Ovi algoritmi se mogu besplatno koristiti – za njihovo korišćenje nije potrebna nikakva licenca.

Rogier i Chavaud su opisali kako se može doći do kolizije u MD2, što je jedini poznati kriptoanalitički napad na MD2. Rivest je 1990. godine dizajnirao algoritam MD4 koji poruku obrađuje pomoću Damgard-Merkle iterativnih struktura u tri runde. Den Boer i Bosselaers su opisali napad na MD4 sa nedostatkom prve ili poslednje runde, a zatim je Dobbertin pokazao kako kolizija za kompletan MD4 može da se odredi za manje od minut vremena na prosečnom personalnom računaru. Dobbertin je takođe pokazao da redukovana verzija MD4 algoritma (izostavljena treća runda) nije jednosmerna.

Ronald Rivest je 1991. godine razvio MD5 heš algoritam. MD5 se uslovno može prihvatiti kao ojačani MD4 – algoritam se sastoji od četiri različite runde (koje su donekle slične rundama MD4 algoritma). Veličina heša i mehanizam dopunjavanja poruke do određene dužine ostali su nepromenjeni. Za sada jedini opisani napad kojim bi se moglo doći do kolizije u MD5 je metoda grube sile.

MD2

Neka je *M* poruka, a S_0–S_{255} permutacije. Poruka se najpre nastavlja sa *p* bajtova vrednosti *p* tako da rezultujuća poruka bude dužine n×16 bajtova. Na poruku se dodaje 16-bitna ček-suma. Zatim se inicijalizuje blok od 48 bajtova X_0, X_1, ..., X_{47} na sledeći način:

- bajtovi X_0–X_{15} se postave na 0,
- bajtovima X_{16}–X_{31} se dodeljuju vrednosti bajtova 0-15 poruke,
- bajtovima X_{32}–X_{47} se dodeljuju se rezultati operacije eksklizivno ILI izvršene nad bajtovima X_0–X_{15} i X_{16}–X_{31}.

Algoritam za kompresiju je dat sledećim pseudo-kodom:

```
t = 0
FOR j=0 TO 17 {
   FOR k=0 TO 47 {
        t = Xk XOR St;
        Xk = t;
        t = (t + j) MOD 256;
   }
}
```

Nakon kompresije ažurira se vrednost bloka bajtova:

- bajtovima X_{16}–X_{31} dodeljuju se vrednosti bajtova 16-31 poruke,
- bajtovima X_{32}–X_{47} dodeljuju se rezultati operacije eksklizivno ILI izvršene nad bajtovima X_0–X_{15} i X_{16}–X_{31}.

Kompresija i ažuriranje bloka bajtova X_0, X_1, ..., X_{47} ponavlja se za svakih 16 bajtova poruke. Izlaz algoritma, tj. heš vrednost poruke su prvih 16 bajtova u bloku X.

MD5

MD5 obrađuje tekst u 512-bitnim blokovima, koji su podeljeni u 16 32-bitnih blokova. Izlaz iz algoritma je 128 bitni heš, tj. četiri 16-bitna bloka.

Inicijalna obrada obuhvata nastavljanje poruke (engl. *padding*): poruka se nastavlja nizom 1000...0000 do dužine (n×512)-64 bita. Na kraj se dodaju 64 bita koji predstavljaju dužinu poruke. Ukupna dužina dopunjene poruke je n×512 bita.

Zatim se inicijalizuju četiri 32-bitne promenljive *A*, *B*, *C* i *D* (*chaining variables*), čija se vrednost dodatno upisuje u četiri promenljive *a*, *b*, *c* i *d*

nad kojima će se u rundama glavne petlje izvršavati nelinearne operacije:

- A = 0x01234567, a = A
- B = 0x89abcdef, b = B
- C = 0xfedcba98, c = C
- D = 0x76543210, d = D

Glavna petlja algoritma se ponavlja onoliko puta koliko dopunjena poruka ima 512-bitnih blokova. Svaka iteracija sastoji se iz četiri runde. U svakoj rundi se izvršava 16 nelinearnih operacija (u svakoj rundi različita operacija) nad tri od četiri promenljive *a*, *b*, *c* i *d*. Rezulat se ažurira konstantom, četvrtom promenljivom i blokom poruke i upisuje u jednu od promenljivih – *a*, *b*, *c* ili *d*).

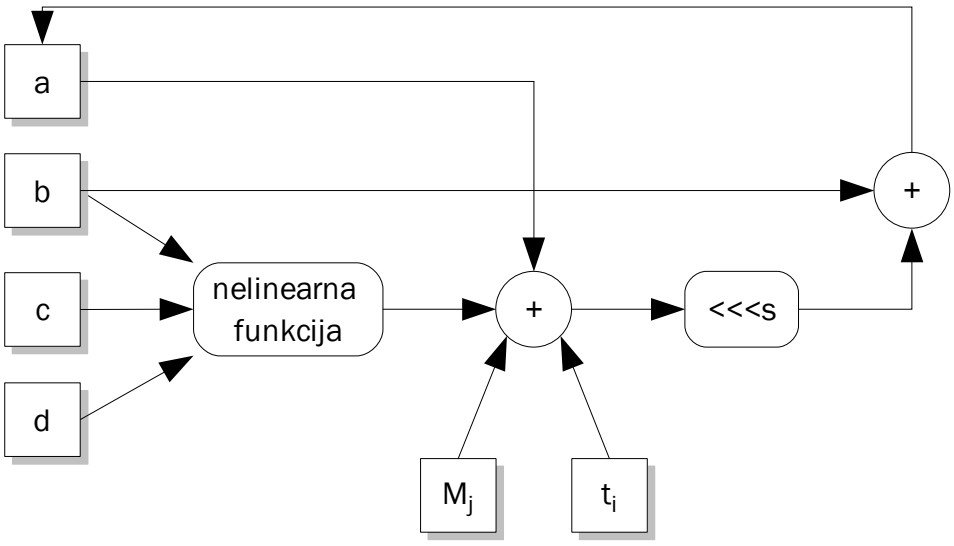
Definisaćemo najpre četiri funkcije koje se izvršavaju bit-po-bit:

- $F(X,Y,Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$ – ako X onda Y, inače Z
- $G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge (\neg Z))$
- $H(X,Y,Z) = X \oplus Y \oplus Z$ – računanje parnosti
- $I(X,Y,Z) = Y \oplus (X \vee (\neg Z))$

Definisaćemo i nelinearne operacije (slika 5.1) koje se izvršavaju u rundama:

- $FF(a,b,c,d,M_j,s,t_i)$: $a=b+((a+F(b,c,d)+M_j+t_i)<<<s)$
- $GG(a,b,c,d,M_j,s,ti)$: $a= b+((a+G(b,c,d)+M_j+t_i)<<<s)$
- $HH(a,b,c,d,M_j,s,ti)$: $a=b+((a+H(b,c,d)+M_j+t_i)<<<s)$
- $II(a,b,c,d,M_j,s,ti)$: $a=b+((a+I(b,c,d)+M_j+t_i)<<<s)$

M_j je *j*-ti 32-bitni podblok 512-bitnog bloka poruke. Konstanta t_i ima vrednost $t_i = int(232 \cdot abs(sin(i)))$, pri čemu je *i* izraženo u radijanima. Operacija <<<s je kružno pomeranje ulevo za s bitova.



Slika 5.1 *– Nelinearne MD5 operacije*

Operacije se po rundama u jednoj iteraciji algoritma izvode prema sledećem rasporedu i dole navedenim argumentima (konstante t_i zapisane su heksadecimalno):

- Prva runda
  - FF(a,b,c,d,M0,7,d76aa478), FF(d,a,b,c,M1,12,e8c7b756)
  - FF(c,d,a,b,M2,17,242070db), FF(b,c,d,a,M3,22,c1bdceee)
  - FF(a,b,c,d,M4,7,f57c0faf), FF(d,a,b,c,M5,12,4787c62a)
  - FF(c,d,a,b,M6,17,a8304613), FF(b,c,d,a,M7,22,fd469501)
  - FF(a,b,c,d,M8,7,698098d8), FF(d,a,b,c,M9,12,8b44f7af)
  - FF(c,d,a,b,M10,17,ffff5bb1), FF(b,c,d,a,M11,22,895cd7be)
  - FF(a,b,c,d,M12,7,6b901122), FF(d,a,b,c,M13,12,fd987193)
  - FF(c,d,a,b,M14,17,a679438e), FF(b,c,d,a,M15,22,49b40821)
- Druga runda:

- GG(a,b,c,d,M1,5,f61e2562), GG(d,a,b,c,M6,9,c040b340)
- GG(c,d,a,b,M11,14,265e5a51), GG(b,c,d,a,M0,20,e9b6c7aa)
- GG(a,b,c,d,M5,5,d62f105d), GG(d,a,b,c,M10,9,02441453)
- GG(c,d,a,b,M15,14,d8a1e681), GG(b,c,d,a,M4,20,e7d3fbc8)
- GG(a,b,c,d,M9,5,21e1cde6), GG(d,a,b,c,M14,9,c33707d6)
- GG(c,d,a,b,M3,14,f4d50d87), GG(b,c,d,a,M8,20,455a14ed)
- GG(a,b,c,d,M13,5,a9e3e905), GG(d,a,b,c,M2,9,fcefa3f8)
- GG(c,d,a,b,M7,14,676f02d9), GG(b,c,d,a,M12,20,8d2a4c8a)

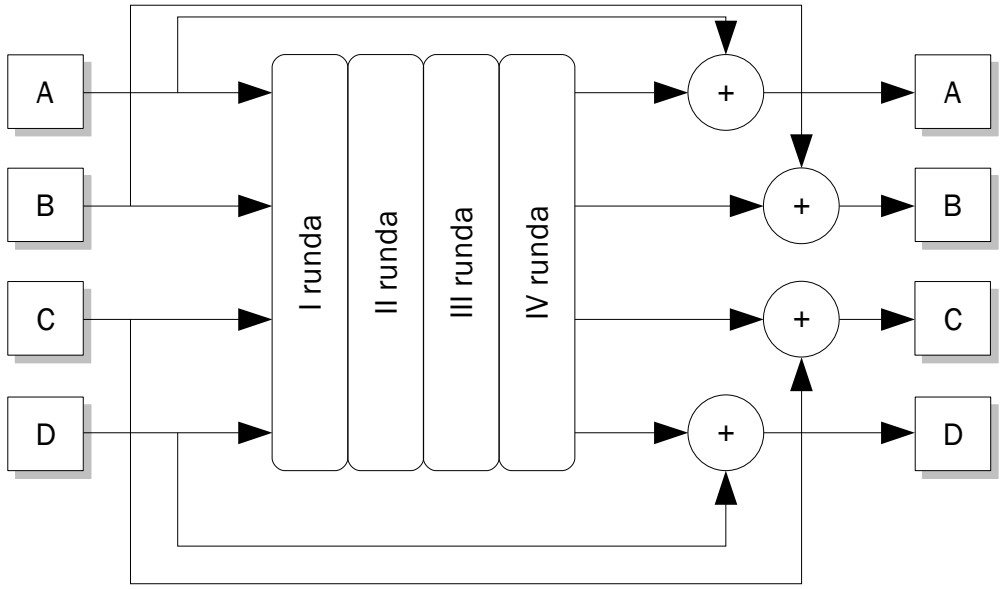
- Treća runda:

  - HH(a,b,c,d,M5,4,fffa3942), HH(d,a,b,c,M8,11,8771f681)
  - HH(c,d,a,b,M11,16,6d9d6122), HH(b,c,d,a,M14,23,fde5380c)
  - HH(a,b,c,d,M1,4,a4beea44), HH(d,a,b,c,M4,11,4bdecfa9)
  - HH(c,d,a,b,M7,16,f6bb4b60), HH(b,c,d,a,M10,23,bebfbc70)
  - HH(a,b,c,d,M13,4,289b7ec6), HH(d,a,b,c,M0,11,eaa127fa)
  - HH(c,d,a,b,M3,16,d4ef3085), HH(b,c,d,a,M6,23,04881d05)
  - HH(a,b,c,d,M9,4,d9d4d039), HH(d,a,b,c,M12,11,e6db99e5)
  - HH(c,d,a,b,M15,16,1fa27cf8), HH(b,c,d,a,M2,23,c4ac5665)

- Četvrta runda:

  - II(a,b,c,d,M0,6,f4292244), II(d,a,b,c,M7,10,432aff97)
  - II(c,d,a,b,M14,15,ab9423a7), II(b,c,d,a,M5,21,fc93a039)
  - II(a,b,c,d,M12,6,655b59c3), II(d,a,b,c,M3,10,8f0ccc92)
  - II(c,d,a,b,M10,15,ffeff47d), II(b,c,d,a,M1,21,85845dd1)
  - II(a,b,c,d,M8,6,6fa87e4f), II(d,a,b,c,M15,10,fe2ce6e0)
  - II(c,d,a,b,M6,15,a3014314), II(b,c,d,a,M13,21,4e0811a1)
  - II(a,b,c,d,M4,6,f7537e82), II(d,a,b,c,M11,10,bd3af235)
  - II(c,d,a,b,M2,15,2ad7d2bb), II(b,c,d,a,M9,21,eb86d391)

Nakon jedne iteracije (odrađene četiri runde nad blokom od 512 bita), promenljive *a*, *b*, *c* i *d* se dodaju na *A*, *B*, *C* i *D*, respektivno, i algoritam nastavlja rad sa sledećim blokom podataka. Izlaz algoritma je konkatenacija promenljivih *A*, *B*, *C* i *D* (otuda naziv *chaining variables* – promenljive koje se ulančavaju).



Slika 5.2 *– MD5 algoritam*

Poboljšanja MD5 u odnosu na MD4 algoritam su sledeća:

- MD5 ima jednu rundu više po iteraciji,
- u svakom koraku se koristi jedinstvena aditivna konstanta t_i,
- umesto funkcije G(X,Y,Z) = ((X $\wedge$ Y) $\vee$ (X $\wedge$ Z) $\vee$ (Y $\wedge$ Z)) u drugoj rundi se koristi G(X,Y,Z) = (X $\wedge$ Z) $\vee$ (Y $\wedge$ ($\neg$ Z)) radi smanjenja simetričnosti,
- efekat lavine je ubrzan korišćenjem rezultata iz prethodnih koraka i optimizacijom kružnog pomeranja ulevo,
- redosled pristupanja pod-blokovima u drugoj i trećoj rundi algoritma je izmenjen kako bi se izbegle sličnosti.

SHA

Američki institut za standarde i tehnologiju (NIST) i nacionalna agencija za bezbednost (NSA) sastavili su heš algoritam SHA (*Secure Hash Algorithm*) za upotrebu u standardu za digitalno potpisivanje (*Digital Signature Standard*). SHA generiše 160-bitni heš. Zasnovan je na ideji na kojoj je zasnovan i MD4. Osnovni principi dizajna SHA algoritma su sledeći:

- iz heša se teško može izvući poruka (jednosmernost),
- dve poruke sa istim hešom se teško nalaze (do heš kolizije se teško dolazi).

Inicijalna obrada poruke identična je kao i kod MD5 algoritma: poruka se nastavlja nizom 1000...0000 do dužine (n×512)-64 bita. Na kraj se dodaju 64 bita koji predstavljaju dužinu poruke. Ukupna dužina dopunjene poruke je n×512 bita.

Zatim se inicijalizuju pet 32-bitnih promenljivih *A*, *B*, *C*, *D* i *E* (jedna promenljiva više u odnosu na MD5, jer SHA proizvodi 160-bitni heš). Iste vrednosti se dodatno upisuju u promenljive *a*, *b*, *c*, *d* i *e*.

- A = 0x67452301, a = A
- B = 0xefcdab89, b = B
- C = 0x98badcfe, c = C
- D = 0x10325476, d = D
- E = 0xc3d2e1f0, e = E

Definisaćemo najpre SHA skup nelinearnih funkcija:

- $f_t(X,Y,Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$, za $t \in [0, 19]$
- $f_t(X,Y,Z) = X \oplus Y \oplus Z$, za $t \in [20, 39]$
- $f_t(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z)$ or $(Y \wedge Z)$, za $t \in [40, 59]$
- $f_t(X,Y,Z) = X \oplus Y \oplus Z$, za $t \in [60, 79]$

Vrednost *t* označava redni broj operacije u rundi. Definisaćemo i

vrednosti konstante K_t:

- $Kt = 2^{1/2} \cdot 2^{32} / 4 = 0x5a827999$, za $t \in [0, 19]$
- $Kt = 3^{1/2} \cdot 2^{32} / 4 = 0x6ed9eba1$, za $t \in [20, 39]$
- $Kt = 5^{1/2} \cdot 2^{32} / 4 = 0x8f1bbcdc$, za za $t \in [40, 59]$
- $Kt = 10^{1/2} \cdot 2^{32} / 4 = 0xca62c1d6$, za za $t \in [60, 79]$

Glavna petlja se ponavlja onoliko puta koliko poruka ima 512 bitnih blokova. Glavna petlja ima četiri runde. U svakoj rundi se obavlja 20 operacija nad tri od pet promenljivih *a*, *b*, *c*, *d* ili *e*, a zatim se radi pomeranje i sabiranje slično kao i kod MD5.

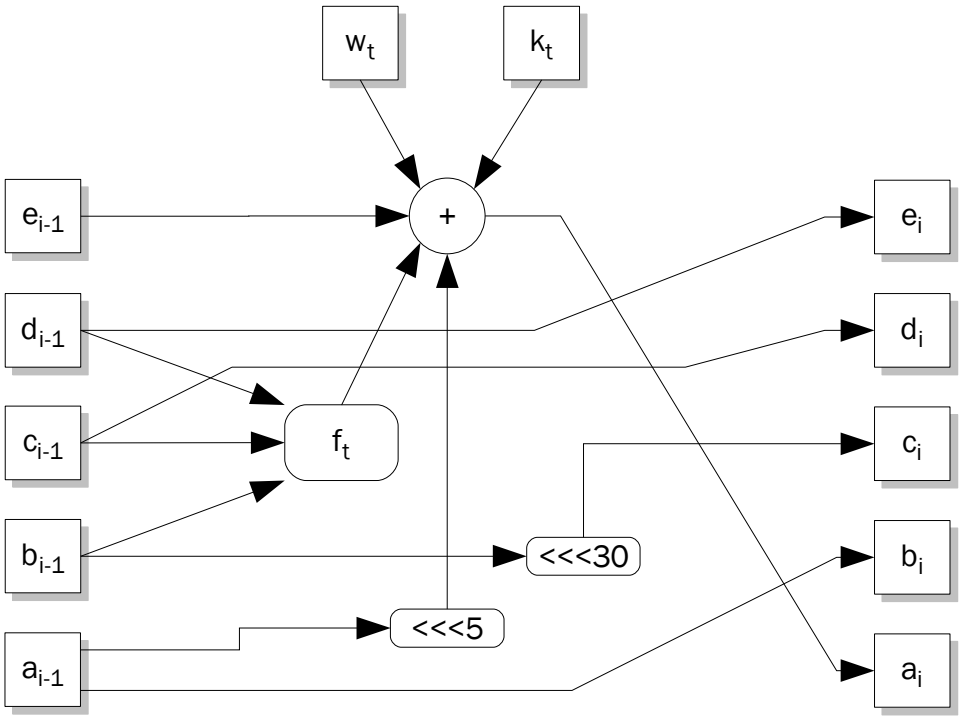
Blok poruke se proširuje iz 16 reči dužine 32 bita (M_0 do M_{15}) u 80 32-bitnih reči (W_0 do W_{79}):

- $W_t = M_t$, za $t \in [0,15]$
- $W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})<<<1$, za $t \in [16,79]$

Operacija <<<s je kružno pomeranje ulevo za s mesta. U glavnoj petlji (slika 5.3) obavljaju se operacije prikazane sledećim pseudo-kodom:

```
FOR t = 0 TO 79 {
   TEMP = (a <<< 5) + ft (b,c,d) + e + Wt + Kt
   e = d
   d = c
   c = b <<< 30
   b = a
   a = TEMP
}
```

Nakon jedne iteracije (obrađen blok od 512 bita), promenljive *a*, *b*, *c*, *d* i *e* se dodaju na *A*, *B*, *C*, *D* i *E* respektivno, i algoritam nastavlja rad sa sledećim blokom podataka. Izlaz algoritma je 160-bitna konkatenacija promenljivih *A*, *B*, *C*, *D* i *E*.



Slika 5.3 *– Jedna iteracija SHA petlje*

5.3 Primena heš funkcija

Jedan od glavih problema sigurnosti je autentičnost korisnika - operativni sistem na neki način mora da zna da li se radi o ovlašćenom korisniku ili ne. Zbog toga, svaki korsinik pre korišćenja sistema mora da se identifikuje, odnosno da se predstavi. Nakon autentifikacije, sistem korisniku daje prava da koristi samo one resurse za čije je korišćenje ovlašćen. Generalno, autentifikacija se može obaviti na tri načina:

- navođenjem poverljivih informacija, kao što je lozinka,
- specijalnim hardverom, kao što su specijalni ključevi i identifikacione kartice (engl. *smartcard*) koje korisnici umeću u čitač kartica,
- biološkim atributima korisnika (engl. *biometrics*), kao što su otisak prsta, snimak mrežnače oka (engl. *retina scan*) i potpis.

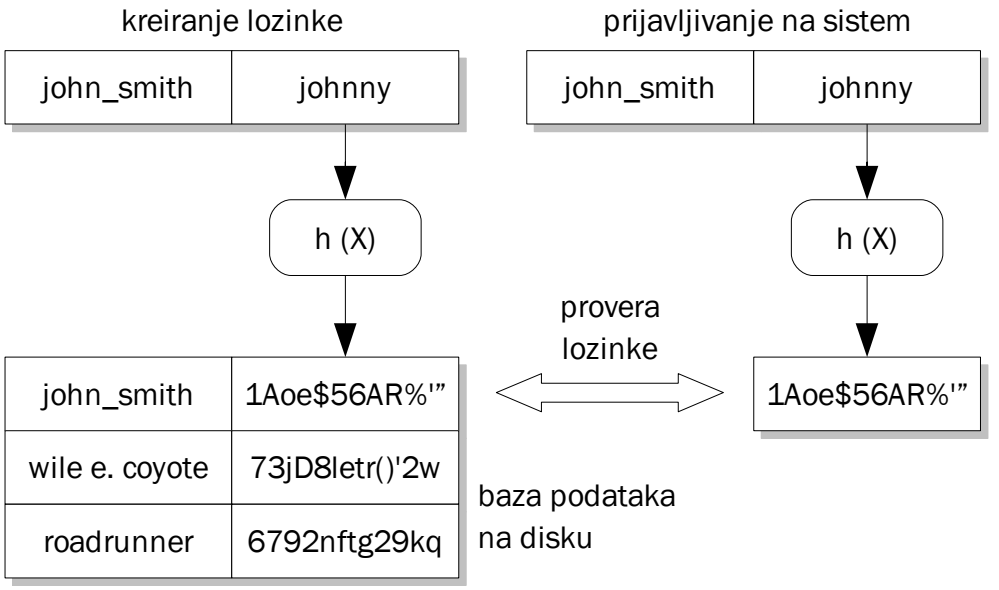
Identifikacija korisnika pomoću poverljivih informacija je najčešće korišćen metod autentifikacije jer osim tastature ne zahteva neki specijalni hardver. Korisnik se, najpre, predstavi sistemu, odnosno identifikuje svojim korisničkim imenom (engl. *username*), a sistem zatim traži potvrdu, odnosno zahteva da korisnik navede odgovarajuću lozinku. Ako uneta vrednost lozinke odgovara vrednosti koja se nalazi na sistemu, operativni sistem smatra da je korisnik prošao autentifikaciju. Osim lozinke koja se koristi za prijavljivanje na sistem, lozinka se može pridružiti svakom objektu sistema, kao što je datoteka.

Lozinke su ranjivo mesto. Preciznije, lozinke su omiljena meta napadača. Ukoliko su kratke ili jednostavne, lako se mogu pogoditi ili pribaviti metodom grube sile. Na primer, ukoliko je korisničko ime *ppetar*, lozinka *pera* se lako pogađa. Zato je potrebno da lozinke budu složene, tj. da se sastoje od većeg broja alfanumeričkih karaktera i specijalnih znakova. Takođe, poželjno je da se često menjaju, i da se pri promeni lozinke ne navode one koje su već korišćene. Lozinke se mogu slučajno otkriti nepoznatom licu (neko vas posmatra dok unosite lozinku) ili namerno, odnosno ilegalno proslediti neovlašćenim korisnicima u cilju ostvarivanja nekakve dobiti.

Heš funkcije i čuvanje lozinki na disku sistema

Poseban problem predstavlja čuvanje informacije o lozinkama na disku računarskog sistema. Ukoliko problem kontrole pristupa na sistemu nije dobro rešen, uljezi lako mogu doći do tih informacija. U tom slučaju, uljez raspolaže lozinkama svih korisnika, uključujući i lozinke povlašćenih korisnika, kao što su sistem administratori. Zbog toga se informacije o lozinkama obrađuju jednosmernim heš funkcijama. Prilikom prvog prijavljivanja na sistem, korisnik smišlja lozinku. Smeštanje lozinke na disk obavlja se na sledeći način (slika 5.4):

- korisnik unosi novu lozinku ($lozinka_A$),
- operativni sistem računa heš unete lozinke: $h_A = h(lozinka_A)$,
- dobijena vrednost se smešta na disk u odgovarajuću tabelu koju čine uređeni parovi (*korisničko ime*, *heš*).



Slika 5.4 *– Heš funkcije i autentifikacija korisnika*

Svaki sledeći put kada želi da se prijavi, korisnik navodi korisničko ime i lozinku. Sistem računa heš unete lozinke h_B = h (*lozinka*$_B$), u tabeli traži heš koji odgovara navedenom korisničkom imenu i upoređuje ga sa dobijenom vrednošću. Ako je h_A=h_B, korisnik je autentifikovan.

Napadač može doći do tabele u kojoj su opisani parovi (*korisničko ime*, *heš*), ali na osnovu tih vrednosti ne može rekonstruisati lozinke, jer su heš funkcije jednosmerne. Ovako zaštićeni sistemi najčešće se napadaju:

- metodama grube sile. Ova metoda napada može se jednostavno sprečiti. Na primer, može se zabraniti prijavljivanje korisnika na sistem posle nekoliko neuspešnih pokušaja – na primer, sistem se blokira nakon tri. Takođe se može uvesti i period mirovanja u trajanju od nekoliko sekundi posle svakog neuspešnog prijavljivanja (čime se smanjuje učestalost pokušaja i višestruko povećava vreme potrebno za napad),

- nasilnom izmenom heša lozinke korisnika sa administrativnim privilegijama u tabeli. Ovaj napad najčešće zahteva da administrator sistema dozvoli podizanje operativnog sistema sa diskete ili CD-

uređaja. Napadač podiže svoj operativni sistem na kom ima sva prava, pristupa particiji na kojoj se nalazi žrtveni operativni sistem i menja tabelu (npr datoteke /etc/passwd i /etc/shadow na Linux sistemu).

Uspešan napad u oba slučaja garantuje sticanje privilegija povlašćenih korisnika. Jedna od metoda koja se u praksi pokazala vrlo uspešnom je ucenjivanje i iznuđivanje informacija od korisnika koji imaju administratorske privilegije.

Heš funkcije i CHAP autentifikacija

CHAP (*Challenge Handshake Authentication Protocol*) je protokol koji se koristi za autentifikaciju pri uspostavljanju veze i povremenu proveru identiteta udaljenog rutera koristeći *three-way handshake* proceduru. Nakon uspostavljanja PPP veze (engl. *link establishment*) lokalni ruter šalje slučajno generisanu poruku udaljenom čvoru – izazov (engl. *challenge*). Udaljeni ruter odgovara na poruku tako što generiše MD5 heš na osnovu primljene poruke i lozinke i izračunatu vrednost šalje lokalnom ruteru. Lokalni ruter upoređuje tu vrednost sa hešom koji sam generiše. Ukoliko se vrednosti poklapaju, autentifikacija se potvrđuje; u suprotnom se veza prekida.

CHAP sprečava potencijalne napade grubom silom tako što lokalni ruter, tj. ruter koji odgovara na zahtev za autentifikacijom:

- slučajno generiše poruke izazova, što znači da će odgovor, tj. heš takođe biti slučajna vrednost i jedinstven za svaki pokušaj autentifikacije (bez obzira na to što se lozinka ne menja) i
- određuje učestalost slanja challenge poruka.

Heš funkcije i digitalno potpisivanje

Digitalno potpisivanje se uslovno može posmatrati kao šifrovanje podataka privatnim ključem. Ukoliko poruku dužine 2 MB šifrujete nekim asimetričnim algoritmom, dobićete šifrat dužine 2 MB. To znači da ćete nekom poslati duplo veću količinu podataka. Da bi se potpis sveo na razumnu dužinu a da pri tom ne izgubi svoj integritet, pošiljalac računa heš

poruke, potpisuje ga svojim privatnim ključem i šalje originalnu poruku i potpis primaocu. Primalac računa heš primljene poruke i proverava primljeni potpis javnim ključem pošiljaoca. Kao rezultat provere potpisa, dobija se dobija heš koji je izračunao pošiljaoc – upoređivanjem sa izračunatim hešom proverava se identitiet pošiljaoca.

5.3 Pitanja i zadaci

5.1 Sastavite program koji simulira prijavljivanje korisnika na operativni sistem koristeći mehanizam za autentifikaciju opisan u 5.3.

Program inicijalno zahteva da se unese administrativna lozinka (korisnik *admin*), a zatim u tabelu na disku smešta uređeni par (admin, SHA heš unete lozinke). Dalje se nudi mogućnost da se na sistem prijavi bilo koji korisnik.

Korisniku se, ukoliko se na sistem prijavi kao *admin*, nudi mogućnost da promeni svoju lozinku ili kreira druge korisnike i dodeli im inicijalne lozinke. Ostali korisnici mogu samo da promene svoje lozinke.

5.2 Sastavite program koji omogućava korisniku da potvrdi integritet šifrovane datoteke (kriptografska ček-suma). Program nakon šifrovanja računa heš datoteke i smešta ga na kraj šifrata u izlaznu datoteku. Po okončanom dešifrovanju, program računa heš dobijenog otvorenog teksta i upoređuje ga sa hešom iz datoteke u kojoj je smešten šifrat. Za šifrovanje iskoristite program iz zadatka 2.1, 2.4, 2.8, 3.1, 3.2, 3.3 ili 3.4.

6

Kriptografija sa javnim ključevima

6.1 Uvod u kriptografiju sa javnim ključevima

Svi prethodno opisani kriptosistemi su simetrični – pošiljalac i primalac tajno biraju ključ *k* i na osnovu njega iz kriptosistema dobijaju funkcije za šifrovanje i dešifriranje čiji su argumenti otvoreni tekst, odnosno šifrat. Pritom je funkcija za dešifrovanje d_k identična funkciji šifrovanja e_k ili se na osnovu nje lako dobija Na primer, funkcija dešifrovanja DES algoritmom se dobija izmenom redosleda potključeva u funkciji šifrovanja. Sigurnost simetričnih kriptosistema zavisi od tajnosti ključa, što je istovremeno i njihov veliki nedostatak, jer pre šifrirovanja pošiljalac i primalac moraju na neki način razmeniti ključ preko nekog sigurnog komunikacionog kanala. Pošto šifrovanje većeg broja poruka istim ključem znatno smanjuje sigurnost, pošiljalac i primalac moraju često menjati ključ.

Diffie-Hellmanov protokol za razmenu ključeva

Godine 1976. Whitfield Diffie i Martin Hellman su ponudili rešenje problema razmene ključeva, zasnovano na diskretnom logaritamskom problemu, odnosno na težini računanja diskretnih logaritama u konačnom polju. Pretpostavimo da se dve osobe (Ana i Bane) moraju dogovoriti o ključu za šifrovanje preko nekog nesigurnog komunikacionog kanala. Takođe, pretpostavimo su se te dve osobe izabrale veliki prost broj *n* i broj *g*, takav da je G={0, 1, ..., n-1} ciklična multiplikativna grupa a *g* njen generator. Brojevi *g* i *n* nisu tajna, što znači da ih može koristiti veći broj osoba koje međusobno komuniciraju. Diffie-Hellmanov protokol za razmenu ključeva obuhvata sledeće korake:

- Ana bira slučajan veliki broj *x* i šalje Banetu $X = g^x \bmod n$,
- Bane bira slučajan veliki broj *y* i šalje Ani $Y = g^y \bmod n$,
- Ana određuje $k = Y^x \bmod n$,
- Bane određuje $k' = X^y \bmod n$.

Vrednosti *k* i *k'* su jednake $g^{xy} \bmod n$ i ne može ih izračunati neko ko prisluškuje kanal. To znači da napadač može doći do vrednosti *n*, *g*, *X*, i *Y*, ali da bi dobio vrednost *k* mora izračunati diskretni logaritam. Dakle, *k* je

tajni ključ koji Ana i Bane nezavisno računaju.

Izbor vrednosti g i n značajno utiče na sigurnost protokola. Najvažnije je da n bude veliki broj, jer je sigurnost protokola zasnovana na problemu određivanja diskretnog logaritma, odnosno faktorizacije brojeva reda veličine *n*. Takođe, broj (n-1)/2 treba biti prost. Generator g ne mora ni prost ni veliki broj – za generator se može izabrati i jednocifreni broj, ali taj broj mora generisati grupu G.

Kriptosistemi sa javnim ključem

Diffie i Hellman se smatraju začetnicima kriptografije javnog ključa. Ideja javnog ključa se sastoji u konstrukciji kriptosistema takvih da je na osnovu javno poznate funkcije šifrovanja nemoguće u nekom razumnom vremenu odrediti tajnu funkciju dešifrovanja. Kriptosistem sa javnim ključem se sastoji od dve familije funkcija koje se ne kriju (za šifrovanje i dešifrovanje), a konkretne funkcije se izvode na osnovu privatnog i javnog ključa. Funkcija šifrovanja algoritmom sa javnim ključem na osnovu javnog ključa i ulaznih podataka proizvodi šifrat. Funkcija dešifrovanja na osnovu privatnog ključa i šifrata proizvodi originalnu poruku. Javni ključ je poznat onim osobama sa kojima korisnik želi da komunicira, dok je tajni ključ poznat samo korisniku koji je ovlašćen da dešifruje poruke. Privatni i javni ključ su matematički povezani, ali se privatni ključ ne može odrediti na osnovu javnog ključa.

Ključnu ulogu u kriptografiji sa javnim ključevima imaju jednosmerne funkcije sa zamkom, tj. lične jednosmerne funkcije (videti 5.1). Jednosmerna funkcija je funkcija oblika y=f(x) takva da se *f(x)* određuje relativno lako i efikasno za svako zadato *x*, ali se $x=f^{-1}(y)$ određuje relativno teško za dato *y*. Ukoliko se inverz $x=g(f^{-1}(y),z)$ određuje relativno lako i efikasno za dato *y* i tajnu informaciju *z*, onda se funkcija f(x) naziva privatna jednosmerna funkcija (engl. *trapdoor one-way*).

Dva korisnika (Ana i Bane) komuniciraju na sledeći način:

- Ana šalje Banetu svoj javni ključ k_A^{public},
- Bane šalje Ani svoj javni ključ k_B^{public},
- Ana šalje Banetu poruku šifrovanu Banetovim javnim ključem:

$c_{A-B} = Ek_B^{public}(p_{A-B})$,

- Bane dešifruje poruku svojim privatnim ključem $p_{A-B} = Dk_B^{private}(c_{A-B})$,
- Bane odgovara, tj. šalje Ani poruku šifrovanu njenim javnim ključem: $c_{B-A} = Ek_A^{public}(p_{B-A})$,
- Ana dešifruje poruku svojim privatnim ključem $p_{B-A} = Dk_A^{private}(c_{B-A})$.

Ukoliko grupa korisnika želi da komunicirati na ovaj način, situacija je još jednostavnija. Svi korisnici svoje javne ključeve smeštaju u neku javnu, svima dostupnu datoteku ili ne server ključeva (engl. *keyserver*). Tada učesnici u komunikaciji ne moraju slati svoje javne ključeve jedni drugima, jer su isti javno dostupni na nekom serveru.

Osnovno svojstvo kriptografije sa javnim ključem je poverljivost (engl. *confidentiality*) – poruku koju Ana šalje Banetu ne može pročitati niko drugi, jer nema Banetov privatni ključ

Ovde se može postaviti pitanje kako Bane može biti siguran da mu je Ana poslala poruku? Ukoliko se ključevi čuvaju na serveru, svako ima pristup Banetovom javnom ključu, a samim tim i funkciji za šifrovanje $Ek_B^{public}(x)$, pa se može lažno predstaviti kao Ana. Dakle, poslavlja se pitanje verodostojnosti, tj. autentičnosti poruke. Neki kriptosistemi korisnicima nude mogućnost da digitalno potpišu svoju poruku. Digitalno potpisivanje se uslovno može posmatrati kao šifrovanje podataka privatnim ključem. Pri tom se ne šifruje sama poruka, već njen heš. Pretpostavite da je Ana poslala Banetu potpisanu poruku. Iako ona kasnije može reći da to nije učinila, Bane uvek može na osnovu potpisa dokazati da je poruku koju je primio poslala Ana, zato što je ona vlasnik privatnog ključa kojim je poruka potpisana. Digitalno potpisivanje nameće sledeće karakteristike kriptografiji sa javnim ključem:

- verodostojnost (engl. *autenticity*) – Bane zna da je samo Ana mogla poslati poruku koju je on primio na osnovu njenog digitalnog potpisa,
- integritet (engl. *integrity*) – Bane zna da poruka koju mu je Ana poslala nije promenjena prilikom slanja, zato što Ana potpisuje heš poruke,
- nepobitnost, tj. neporicanje (engl. *non-repudiation*) – Ana ne može kasnije reći da nije poslala poruku koju je digitalno potpisala.

Kriptosistemi sa javnim ključem imaju nekoliko prednosti u odnosu na simetrične kriptosisteme. Kriptosistem sa javnim ključem ne zahteva sigurnan komunikacioni kanal za razmenu ključeva. Za komunikaciju grupe koju čini *n* osoba potrebno je 2n ključeva (ukoliko bi se koristio simetrični kriptosistem bilo bi potrebno n(n-1)/2 ključeva). Takođe, korisnici svoje poruke mogu potpisati, što u slučaju simetričnog kriptosistema nije moguće.

Ipak, kriptografija sa javnim ključem ne predstavlja zamenu za simetrične kriptosisteme i najčešće se ne koristi za šifrovanje poruka, već za šifrovanje ključeva (tzv. hibridni kriptosistem). Ana i Bane komuniciraju pomoću simetričnog kriptosistema koristeći ključ koji su razmenili pomoću kriptosistema sa javnim ključem. Osnovni razlog zašto se javni ključ ne koristi za šifrovanje poruka je to što su algoritmi s javnim ključem znatno sporiji (oko 1000 puta) od modernih simetričnih algoritama. Drugi nedostatak kriptosistema sa javnim ključem je njihova osetljivost na napad odabrani otvoreni tekst ukoliko je domen funkcije šifrovanja mali.

6.2 RSA

RSA je verovatno najpopularniji asimetrični kriptosistem. Objavljen je 1978. godine, a ime je dobio po svojim tvorcima Ronaldu Rivestu, Adi Shamiru i Leonardu Adlemanu (*RSA Data Security*). Postojanje slabih tačaka pravilno implementiranog kriptosistema, sa ključevima generisanim na osnovu određenih preporuka, do sada nije ni dokazano ni opovrgnuto.

Sigurnost RSA zasniva se na složenosti faktorizacije velikih brojeva. Javni i tajni ključ određeni su parom velikih prostih brojeva (200 dekadnih cifara i više). Smatra se da je težina određivanja otvorenog teksta na osnovu šifrata bez adekvatnog privatnog ključa jednaka težini faktorizaciji proizvoda dva velika prosta broja. Sam algoritam i njegova sigurnost su, dakle, zasnovani na sledećim činjenicama da je:

- lako odrediti odrediti da li je veliki broj prost i pomnožiti dva velika prosta broja,
- teško faktorisati veliki broj koji je proizvod dva velika prosta broja (odnosno dobiti njegove početne proste faktore).

Matematička podloga RSA algoritma

Ukoliko se rezultat neke računske operacije uvek nalazi na intervalu [0, m-1], onda *m* nazivamo modulom. Na primer. sati se posmatraju po modulu 12 ili 24 (zavisno od toga da li ste Amerikanac ili Srbin), dani u nedelji po modulu 7, a godine po modulu 365. Izračunati *n mod m* znači pomnožiti *m* konačno mnogo puta sve dok razlika *n* i tog proizvoda ne bude izmedju 0 i *m*. Na primer:

- 17 mod 5 = 2, 7 mod 11 = 7, 20 mod 3 = 2,
- 11 mod 11 = 0, 25 mod 5 = 0.

Za brojeve *r* i s kažemo da su kongruentni po modulu *m* ako važi:

- r mod m = s mod m

U tom slučaju pišemo $r \equiv s \ mod \ m$. Sledi da je r-s=km za neku vrednost celog broja k. Na primer, $4 \equiv 9 \equiv 14 \equiv 19 \ \text{mod} \ 5$.

Proizvod dva broja $(a \cdot b) \ \text{mod} \ m$ u modularnoj aritmetici određuje se na sledeći način:

- $(a \cdot b) \ \text{mod} \ m = ((a \ \text{mod} \ m) \cdot (b \ \text{mod} \ m)) \ \text{mod} \ m$

Na primer:

- $39 \cdot 15 \ \text{mod} \ 11 = ((39 \ \text{mod} \ 11) \cdot (15 \ \text{mod} \ 11)) \ \text{mod} \ 11 =$
  $= 6 \cdot 4 \ \text{mod} \ 11 = 24 \ \text{mod} \ 11 = 2$

Za efikasnost RSA kriptosistema, važna je činjenica da se modularno stepenovanje može izvesti vrlo efikasno. Vrednost $x^e \ \text{mod} \ m$ računa se metodom "kvadriraj i množi". Najpre se e prikaže kao:

- $e = e_0 + 2 \cdot e_1 + \ldots + 2^{s-1} \cdot e_{s-1} = \Sigma_{i=0,s-1} \ 2^i \ e^i$

a zatim se primenjuje sledeći algoritam:

- postavi se y = 1

- Za i = s−1 do 0 radi se sledeće
  - $y = y^2 \mod m$
  - ako je $e_i = 1$ onda: $y = y \cdot x \mod m$

Deljenje po modulu se u modularnoj aritmetici realizuje kao množenje sa multiplikativnim inverzom delioca. Na primer, odredićemo 5÷3 mod 11. Multiplikativni inverz od 3 mod 11 je 4 jer je $3 \cdot 4 = 1 \mod 11$. Dobija se:

- $5 \div 3 \mod 11 = 5 \cdot 4 \mod 11 = 9 \mod 11$

Navodimo malu Fermaovu teoremu. Ako je *p* prost broj (pazite, teorema ne važi ako *p* nije prost broj) i *a* bilo koji broj izmedju 1 i p, onda važi:

- $a^{p-1} \mod p = 1$.

RSA problem

Neka su dati prost broj *n* i broj *e* sa intervala [1, n-1]. Za neki broj *m* (otvoreni tekst) sa intervala [1, n] može se odrediti šifrat:

- $c = m^e \mod n$.

Postavlja se pitanje: ako je dat šifrat *c* i ključ (*e*, *n*), kako se može naći otvoreni tekst *m*? Najpre je potrebno naći broj *d* takav da važi:

- $e \cdot d = 1 \mod n-1$

i zatim izračunati $m = c^d \mod p$.

Zašto se sistem ponaša na taj način? Ukoliko pronađemo *d* koje zadovoljava jednakost $e \cdot d = 1 \mod n-1$, dobićemo $e \cdot d - 1 = k(n-1)$ za neko *k*, tj. $e \cdot d = k(n-1) + 1$. Odatle se vidi da je *d* multiplikativni inverz vrednosti *e* po modulu n-1.

- $c^d = (m^e)^d \mod n$
- $= m^{ed} \mod n$

- $= m^{k(n-1)+1} \bmod n$
- $= m^{k(n-1)} \cdot m \bmod n$ (ključni korak)
- $= 1 \cdot m \bmod n$

  $= m \bmod n$

Obrazloženje: prethodni problem se mogao rešiti zbog Male Fermaove teoreme. Prema teoremi je $a^{p-1} = 1 \bmod p$ za svako *a* između 1 i *p*, a u ovom slučaju imamo $c^{k(p-1)} = 1 \bmod n$ (*k* kao stepen ne utiče na rezultat).

Napomena: ako modul n nije prost broj metod ne funkcioniše ! Uopšteno, $a^{n-1} \neq 1 \bmod n$ ako *n* nije prost. Međutim, mogli bismo rešiti problem nalaženja *m* ako bismo našli broj *r* takav da je $a^r = 1 \bmod n$. Ako su *a* i *n* uzajamno prosti postojaće takav broj *r*, kao i način da se dođe do njega. U nameri da nađemo *r* tako da važi $a^r = 1 \bmod n$, neophodno je da faktorišemo broj *n* tj. da pronađemo njegove proste faktore. Ako je $n = p \cdot q$ gde su *p* i *q* prosti brojevi, tada je:

- $r = (p-1) \cdot (q-1)$

RSA algoritam se implementira shodno sledećim tvrđenjima:

- lako je odrediti da li je veliki broj prost ili složen. Lako je izračunati proizvod dva velika prosta broja $n = p \cdot q$.
- postavljanjem $r = (p-1) \cdot (q-1)$ dobijamo $m^r = 1 \bmod n$ za svako *m* uzajamno prosto sa *n*.
- za dat broj *e*, uzajamno prost sa *r*, lako je odrediti broj *d* takav da važi: $e \cdot d = 1 \bmod r$.

Operacije šifrovanja i dešifrovanja definišu se na sledeći način:

- $e_{(e, n)}(m) = m^e \bmod n$,
- $d_{(d, n)}(c) = c^d \bmod n$.

Otvoreni tekst $m = c^d \bmod n$ se lako računa ako je *d* poznato. Vrednost *d* se može naći samo ako nadjemo *r*, a *r* možemo naći samo ako faktorišemo *n*. Faktorizacija velikog broja *n* je teška – zahteva veliku procesorsku snagu i puno vremena, a ne postoji ni jedan efikasan algoritam koji bi to vreme

redukovao. Odatle sledi da je algoritam siguran.

Osnovu sigurnosti RSA algoritma čini čuvanje i skrivanje prostih faktora *p* i *q*. Bez njih je nemoguće naći *r* a potom i *d*, tj. $m = c^d \bmod n$ se ne može odrediti. Osoba koja ne zna *p* i *q* ne može otkriti poruku *m*. Dakle, otkrivanje poruke *m* ekvivalentno je faktorizaciji *n*.

Generisanje ključeva, šifrovanje i dešifrovanje

Generisanje para ključeva obavlja se na sledeći način:

- najpre se generišu dva prosta broja *p* i *q* (oba preko 100 decimalnih cifara) i izračunavaju vrednosti $n = p \cdot q$ i $r = (p-1) \cdot (q-1)$,
- bira se slučajan broj *e* na intervalu [1, r-1] koji je uzajamno prost sa *r* (tj. jedini zajednički faktor za *e* i *r* je 1),
- izračunava se *d* tako da važi: $e \cdot d = 1 \bmod r$.

Vrednosti *p*, *q* i *r* se čuvaju ili brišu. Privatni ključ (*d*, *n*) se čuva u tajnosti, dok je javni ključ (*e*, *n*) dostupan svima sa kojim vlasnik privatnog ključa želi sigurno da komunicira. Relacija koja povezuje ključeve je $e \cdot d = 1 \bmod r$.

Ukoliko Ana želi da pošalje Banetu poruku m, ona preuzima Banetov javni ključ (*e*, *n*) sa servera, izračunava $c = m^e \bmod n$ i šalje Banetu šifrat *c*. Bane prima šifrat i dešifruje ga svim privatnim ključem $m = c^d \bmod n$.

Poruka *m* mora biti manja od *n*. Zato pošiljalac deli svoju poruku na blokove čija je vrednost manja od *n* i parcijalno ih šifruje.

Primer 6.1. Neka je $p = 47$ i $q = 71$. Generišite par RSA ključeva i šifrujte poruku $m = 6882326879666683$.

Najpre se određuje $n = p \cdot q = 3337$ i $r = (p-1) \cdot (q-1) = 46 \cdot 70 = 3220$.

Bira se e sa intervala [1, 3219]. Može se uzeti $e = 79$, što je uzajamno prosto sa 3220. Za određivanje multiplikativnog inverza broja 79 po modulu 3220 primenjujemo Euklidov algoritam na brojeve 3220 i 79:

- $3220 = 40 \cdot 79 + 60$
- $79 = 1 \cdot 60 + 19$
- $60 = 3 \cdot 19 + 3$
- $19 = 6 \cdot 3 + 1$
- $3 = 3 \cdot 1$

Primenom gornjih jednakosti po obrnutom redu dobija se izraz za jedinicu kao linearnu kombinaciju brojeva e = 79 i r = 3220.

- $1 = 19 - 6 \cdot 3 = 19 - 6 \cdot (60 - 3 \cdot 19)$
  $= -6 \cdot 60 + 19 \cdot 19 = -6 \cdot 60 + 19 \cdot (79 - 60)$
  $= 19 \cdot 79 - 25 \cdot 60$
  $= 19 \cdot 79 - 25 \cdot (3220 - 40 \cdot 79)$
  $= 1019 \cdot 79 - 25 \cdot 3220$

Dobijena jednačina po modulu 3220 postaje $1019 \cdot 79 = 1 \bmod 3220$ što znači da je 1019 inverz za 79 tj. d = 1019.

Znači, privatni ključ je (79, 3337), a javni (1019, 3337).

Poruku m = 6882326879666683 razbićemo na manje blokove tako da svaki bude manji od n=3337. U ovom slučaju, to su sledeći blokovi: 688, 232, 687, 966, 668, 3.

Šifrovanjem prvog bloka dobija se: $688^{79} \bmod 337 = 1570$. Nastavljajući na isti način dobijamo šifrat: c = 1550 2756 2714 2276 2423 0158.

Dešifrovanjem se dobija: $1550^{1019} \bmod 3337 = 688$. Nastavljajući na isti način, dobićemo originalnu poruku *m*.

Digitalno potpisivanje poruka

Ukoliko je potrebno dokazati verodostojnost neke poruke, RSA kriptosistem se može primeniti za digitalno potpisivanje. Pretpostavite da Ana, čiji je privatni ključ (*d*, *n*), a javni ključ (*e*, *n*), šalje Banetu poruku m, ali

da Bane zahteva od Ane da na neki način dokaže da je baš ona poslala tu poruku. U tom slučaju, komunikacija se odvija po sledećem protokolu:

- Ana računa potpis *s* pomoću svog privatnog ključa: $s = m^d \bmod n$,
- Ana šalje Banetu poruku i potpis, odnosno uređeni par (m,s),
- Bane dešifruje potpis *s* koristeći Anin javni ključ: $m_1 = s^e \bmod n$,
- ako je $m_1=m$, Bane prihvata poruku zato što jedino Ana zna svoj privatni ključ kojim je poruka potpisana.

Ukoliko se za komunikaciju primeni prethodno opisani protokol, Ana mora da pošalje dva puta veću poruku (RSA šifrat je iste dužine kao i otvoreni tekst). U slučaju da je poruka dužine 10 MB, potpis će takođe imati dužinu 10 MB, pa se primaocu šalje 20 MB (probajte da pošaljete nekom poruku dužine 20 MB preko dial-up konekcije). Ovo premašenje se može smanjiti ukoliko se pre slanja ne potpisuje sama poruka, već njen heš. U tom slučaju premašenje je znatno manje – najčešće 128 ili 160 bita, tj. određeno je dužinom heša koji se generiše. Izbor heš funkcije je deo komunikacionog protokola između pošiljaoca i primaoca, ali se najčešće koriste MD5 ili SHA. Komunikacija između pošiljaoca i primaoca se odvija prema sledećem protokolu:

- Ana računa heš $h = H(m)$ poruke *m*, a zatim digitalni potpis pomoću svog privatnog ključa: $s = h^d \bmod n$,
- Ana šalje Banetu poruku i potpis, odnosno uređeni par (m,s),
- Bane dešifruje potpis *s* koristeći Anin javni ključ: $h = s^e \bmod n$,
- Bane računa heš primljene poruke: $h_1 = H(m)$,
- ako je $h_1=h$, Bane prihvata poruku, zato što jedino Ana zna svoj privatni ključ kojim je heš poruke potpisan.

Kriptoanaliza RSA

Pretpostavimo da je broj *n* koji želimo da faktorišemo neparan. Najjednostavniji način faktorizacije je "probno deljenje" koje se sastoji iz deljenja broja n sa svakim neparnim brojem manjim od $\sqrt{n}$. Probno deljenje je jedan oblik napada "grubom silom"; ukoliko je $n>10^{12}$, koristite

se suptilnije metode, kao što je Pollardov p-1 algoritam:

- a = 2
- Za j=2 do B izračunaj a = a^j mod n
- d = NZD(a-1, n)
- Ako je 1<d<n vrati faktor d

Pretpostavite da postoji izvesna granica *B*. Što je granica veća, to je Pollardov p-1 algoritam uspešniji. Ipak, ako je granica preterano velika (npr. blizu $\sqrt{n}$) algoritam sigurno daje rešenje, ali nije ništa brži od "probnog deljenja".

Primer 6.2. Faktorišite n=15770708441 koristeći Pollardov p-1 algoritam ako je granica B=180.

Primenom algoritma nalazimo da je a=11620221425 i d=135979, odakle dobijamo 15770708441 = 135979 · 115979. Važno je napomenuti da je faktorizacija uspela jer p-1 = 135978 sadrži faktore koji su manji od B=180: 135978 = 2 · 3 · 131 · 173. To je posledica činjenice da u algoritmu p-1 deli B!.

Na sledećoj ideji zasnovani su mnogi algoritmi faktorizacije: pretpostavimo da možemo naći *x* i *y* koji zadovoljavaju sledeće osobine:

- $x \neq \pm y \bmod n$
- $x^2=y^2 \bmod n$.

Tada *n* deli (x+y)(x–y) ali ne (x+y) ili (x–y) pojedinačno. To znači da su NZD(x+y,n) i NZD (x–y,n) netrivijalni faktori broja *n*.

Na primer, za jednačinu $10^2 = 32^2 \bmod 77$ dobijamo:

- NZD(10+32,77) = 7
- NZD(32–10,77) = 11.

Ako je n=pq i ako znamo Φ(n)=(p-1)(q-1) tada zamenom q=n/p dobijamo:

- $p^2 - (n-\Phi(n)+1) \cdot p + n = 0$

Iz dobijene kvadratne jednačine lako izračunavamo *p* pa potom i *q*. Međutim, saznati *Φ(n)* podjednako je teško kao i faktorisati *n*. Još je interesantnije da izračunavanje dekriptujućeg eksponenta *a* takođe nije ništa lakše od faktorizacije broja *n*. Ukoliko bismo ipak uspeli da saznamo eksponent *a*, primenom sledećeg algoritma može se doći da faktora broja *n*.

Neka su *a*, *b* i *x* privatni ključ (dekriptujući eksponent), javni ključ i traženi faktor broja n, respektivno.

- $a \cdot b - 1 = 2^s \cdot r$, gde je r neparan broj,
- bira se slučajan broj *w* sa intervala [1, n-1],
- neka je x = NZD(w,n),
- ako je 1<x<n vrati *x* (*x* je faktor broja *n*),
- neka je $v = w^r \bmod n$,
- ako je $v = 1 \bmod n$, algoritam nema odgovor,
- dok je $v \neq 1 \bmod n$, radi $\{v_0 = v, v_i = v_{i-1}^2 \bmod n\}$,
- ako je $v_0 = -1 \bmod n$, algoritam nema odgovor,
- u suprotnom vrati $x = NZD(v_0+1,n)$.

Na primer, neka je n=89855713, b=34986517, a=82330933 i w=5. Tada je $ab-1 = 2^3 \cdot 360059073378795$ i $w^r \bmod n = 85877701$. Kako je $85877701^2 = 1 \bmod n$, algoritam vraća vrednost: x = NZD(85877702,n) = 9103. Drugi faktor je n/9103 = 9871.

Prethodno opisani algoritam je tipa "Las Vegas" – ako ima odgovor, on je uvek tačan. Jednostavno rečeno, ako smo dovoljno srećni da na početku izaberemo *w* takvo da sadrži *p* ili *q*, odmah dobijamo faktor *x*. Ako je *w* uzajamno prosto sa *n*, tada će, u najgorem slučaju posle s iteracija za $k=2^s \cdot r$ važiti $w^k = 1 \bmod n$ (jer je $k = ab-1 = 0 \bmod \Phi(n)$). Na kraju je još potrebno ispitati da li je v_0 trivijalni kvadratni koren od 1 tj. da li je $v_0 \neq \pm 1$.

Prethodni rezultat je veoma važan jer pokazuje da ako je tajni ključ otkriven tada nije dovoljno promeniti samo par ključeva već je potrebno promeniti i moduo *n*. U suprotnom, znajući par (a,b) mogu se čitati naredne

poruke.

Polazeći od činjenice da Φ(n)|(ab–1) potrebno je da za novi javni ključ *b'* odredimo *a'* tako da važi Φ(n)|(a'b'–1) (Φ(n) je nepoznato). Najpre se nalazi:

- g = (ab–1)/NZD(ab-1,b')

S obzirom na to da Φ(n)|(ab–1) i da Φ(n) ne deli NZD(ab-1,b') (jer NZD(Φ(n), b')=1) sledi Φ(n)|g. Nađimo a' tako da je a'b' = 1 mod g. Kako Φ(n)|g i g|(a'b'–1), onda se Φ(n)|(a'b'-1) tj. imamo novi par ključeva.

6.3 El-Gamal

El-Gamalov kriptosistem (Taher El-Gamal, 1985) zasnovan je na težini određivanja diskretnog logaritma u konačnim poljima. Diskretan logaritamski problem svodi se na sledeće: za dati prost broj *p* i vrednosti *g* i *y*, potrebno je naći *x*, tako da važi $y = g^x \bmod p$. Za male vrednosti modula, diskretni logaritam se može odrediti metodom grube sile, tj. prostim isprobavanjem različitih vrednosti. Na primer, za dato p=11, g=2 i y=9, možemo probati različite vrednosti *x* sve dok ne dobijemo $2^x \bmod 11 = 9$. Međutim, za velike vrednosti modula (broj p ima 100 decimalnih cifara i više) zvanično nije moguće rešiti diskretan logaritamski problem pomoću današnje tehnologije.

Generisanje ključeva, šifrovanje i dešifrovanje

Javni i privatni ključ za El-Gamalov kriptosistem određuju se na sledeći način:

- generiše se veliki prost broj *p*,
- određuje se generator *g* grupe {0, 1, …., p-1}, odnosno broj *g* takav da $g^x \bmod p$ daje različit rezultat za svako *x*; na osnovu male Fermaove teoreme važi $g^{p-1} \bmod p = 1$,
- bira se slučajni broj a sa intervala [1, p-1],

- izračunava se $y = g^a \bmod p$.

Uređena trojka (*p*, *g*, *y*) je javni ključ, a broj *a* privatni.

Prilikom slanja poruke, pošiljalac najpre uzima privatni ključ primaoca (*p*, *g*, *y*) i deli poruku na blokove tako da svaki blok bude manji od *p*. Svaki blok *m* poruke pošiljaoc šifruje na sledeći način:

- generiše slučajan broj *k* na intervalu [1, p-1],
- izračunava: $r = g^k \bmod p$,
- izračunava: $x = y^k \bmod p$,
- izračunava: $c = (m \cdot x) \bmod p$.

Šifrat jednog bloka je uređen pad (*r*, *c*). Primalac dešifruje svaki blok šifrata (*r*, *c*) koristeći svoj privatni ključ *a* na sledeći način:

- Izračunava $r^a = (g^k)^a = (g^a)^k = y^k = x$,
- Iz jednačine $c = (m \cdot x) \bmod p$ određuje vrednost m.

Šifrovana poruka se može poslati preko nesigurnog komunikacionog kanala - napadač može preuzeti poruku sa mreže, ali je ne može dešifrovati jer nema odgovarajući privatni ključ (i najverovatnije ne ume da reši diskretan logaritamski problem).

Poređenje RSA i El-Gamal kriptosistema

Osnovne karakteristike RSA kriptosistema su:

- sigurnost je zasnovana na težini faktorizacije velikih brojeva,
- šifrat je iste dužine kao i otvoreni tekst,
- isti algoritam (stepenovanje po modulu) se koristi i za šifrovanje i za dešifrovanje,
- algoritam je patentiran.

Osnovne karakteristike El-Gamal kriptosistema su:

- sigurnost je zasnovana na diskretnom logaritamskom problemu,
- šifrat je uređeni par brojeva i dva puta je duži od poruke,
- za šifrovanje i dešifrovanje se koriste različiti algoritmi,
- algoritam nije patentiran (što je očigledna prednost u odnosu na RSA).

6.4 Sertifikati

Kao što je već izloženo, kriptografija sa javnim ključevima rešava problem sigurnog kanala za razmenu ključeva i broja ključeva potrebnih za sigurnu komunikaciju većeg broja osoba. Ukoliko mali broj korisnika želi međusobno da komunicira koristeći kriptosistem sa javnim ključevima, razmena ključeva se može obaviti preko elektronske pošte, pomoću disketa ili fleš diskova. Ukoliko je ta grupa korisnika veća, ovaj način razmene ključeva je nepraktičan. Takođe, napadač može podmetnuti svoj javni ključ i na taj način čitati šifrovane poruke namenjene drugim entitetima. Zamislimo da dve osobe (Ana i Bane) žele da komuniciraju preko elektronske pošte koristeći kriptosistem sa javnim ključem. U tom slučaju, Ana mora da poseduje Banetov javni ključ ukoliko želi da mu pošalje poruku. S druge strane, Bane mora da poseduje Anin javni ključ, ukoliko želi da joj pošalje poruku. Osnovni problem koji se ovde može postaviti je pitanje integriteta njihovih javnih ključeva, osnosno kako se može garantovati da je Banetov javni ključ stvarno Banetov, a ne ključ napadača koji želi da čita Anine poruke? Ovaj problem se rešava pomoću sertifikata i infrastrukture javnih ključeva. Uvešćemo pojam digitalnog sertifikata.

Digitalni sertifikat (engl. *certificate*) čine:

- javni ključ,
- informacije o identitetu (ime, identifikator korisnika – UID, ...),
- informacije koje se tiču autorizacije korisnika, npr. dozvole za pristup resursima (opciono),
- jedan ili više digitalnih potpisa.

Digitalni potpis je overa sertifikata. Sertifikate potpisuju strane kojima se veruje. Potpisom se ne “overava” sertifikat u celini, već samo veza između

identiteta korisnika i javnog ključa. Sertifikat je, dakle, javni ključ sa opisom identiteta korisnika (jednim ili više) i potpisom koji je izdala strana kojoj se veruje, kojim je overena veza između identiteta i ključa. Zbog toga se ovakvi sertifikati nazivaju identifikacionim sertifikatima (engl. *identity certificate*) i koriste se za identifikaciju pojedinaca, servera ili neke kompanije. Za razliku od identifikacionih sertifikata, sertifikati akreditiva (engl. *credential certificate*) opisuje dozvole ili akreditive (npr. pristup određenom računaru).

Digitalni sertifikati obezbeđuju podršku za:

- Autentifikaciju identiteta. Digitalni sertifikati koje izdaje PKI omogućavaju pojedinačnim korisnicima i organizacijama da provere identitet učesnika u komunikaciji/transakciji. U mrežnom segmentu, autentifikacija predstavlja identifikaciju entiteta, a sertifikati su jedan od oblika podrške autentifikaciji. Primer autentifikacije na mreži je autentifikacija klijenta serveru i servera klijentu. Sledeći primer je digitalni potpis elektronske pošte u kombinaciji sa sertifikatom koji identifikuje pošiljaoca, obezbeđuje snažne dokaze da je osoba identifikovana sertifikatom zaista poslala tu poruku.

- Proveru integriteta. Digitalni sertifikat obezbeđuje integritet poruka, odnosno onemogućava da korisnik primi izmenjenu ili oštećenu poruku.

- Autorizaciju pristupa. Digitalni sertifikati zamenjuju često zaboravljene korisnička imena i lozinke na Internetu.

- Neporicanje. Digitalni sertifikati potvrđuju korisnički identitet, čineći ga skoro nemogućim za kasnije odbacivanje digitalno „označenih“ transakcija, kao na primer kupovina preko web sajta. Takođe, sertifikat onemogućava potpisnika da kasnije ne prizna slanje digitalno potpisane elektronske pošte.

6.5 Infrastruktura javnih ključeva

Server sertifikata (engl. *certificate server*) je baza podataka na mrežnom serveru koji obezbeđuje sekundarnu memoriju za skladištenje sertifikata i

mehanizme za razmenu. Serveri sertifikata ne obezbeđuju mehanizme za izdavanje ili poništavanje sertifikata, već samo za njihovo skladištenje i distribuciju - zato se ponekad nazivaju i skladišta sertifikata (engl. *certificate repositories*).

Za razliku od servera sertifikata, infrastruktura javnih ključeva (engl. *public key infrastructure*, PKI) je struktuirani sistem koji, osim skladištenja, obezbeđuje dodatne funkcije (servise i protokole) za izdavanje i poništavanje sertifikata, kao i funkcije za uspostavljanje relacija poverenja. PKI predstavlja kombinaciju kriptografskih tehnika, softvera, i mrežnih servisa koja integriše digitalne sertifikate, asimetrično šifrovanje i sertifikacione centre u kompletnu, široko rasprostranjenu sigurnosnu arhitekturu. Tipičan primer je PKI sistem preduzeća koji uključuje izdavanje digitalnih sertifikata individualnim korisnicima ili serverima, korisnički softver, integraciju sa direktorijumima aktivnih sertifikata, alatima za upravljanje, obnavljanje i poništavanje sertifikata i odgovarajuće servise i podršku.

Tri osnovne komponente infrastrukture javnih ključeva su:

- Sertifikacioni centar (engl. *certificate authority*, *CA*). CA je centralna komponenta PKI koja generiše, izdaje i poništava sertifikate i potpisuje izdate sertifikate svojim privatnim ključem CA. CA je odgovoran za generisanje sertifikata i njihov integritet, slično kao što je MUP odgovoran za lične karte i vozačke dozvole. Korišćenjem javnog ključa CA svako može proveriti potpis CA na sertifikatu i samim tim integritet sertifikata. CA se štiti metodama samouništenja u slučaju napada[18] (engl. *tamper-proof* metode) – u slučaju napada koji ugrožava integritet PKI, CA uništava sve ključeve. CA se može realizovati zatvoreno, implementacijom gotovih PKI rešenja ili pomoću javnih CA servisa. Osnovni zadatak ustanove koja pruža uslugu izdavanja digitalnih sertifikata jeste da bude poverljiva treća strana kojoj veruju učesnici u komunikaciji. Navodimo neke značajnije sertifikacione centre:
  - EuroSign, UK (http://www.eurosign.com),

18 *Tamper-proof* metode zaštite često se sreću kod hardverskih modula ("crnih kutija") za šifrovanje. Modul se štiti eksplozivnom materijom nanešenom na čip ili štampanu ploču – ukoliko napadač pokuša da na neki način otvori modul, aktivira se mehanizam koji dovodi do slabe eksplozije koja uništava internu elektroniku modula.

  - Cybertrust GTE, USA (http://www.cybertrust.com),
  - VeriSign, USA (http://www.verisign.com),
  - Thawte, South Africa (http://www.thawte.com).
- Registracioni centar (engl. *registration authority*, *RA*). RA je komponenta PKI koja osigurava proces registracije korisnika, prihvata i obrađuje zahteve za izdavanjem sertifikata, i iste prosleđuje CA radi izdavanja sertifikata. RA se odnosi na ljude, procese i alate za registraciju i administraciju korisnika PKI. RA/CA podseća na službu za izdavanje pasoša: određena grupa ljudi (RA) proverava identitet čoveka koji želi da mu se izda pasoš i da li sme da mu se izda pasoš, a zatim CA kreira pasoš i prosleđuje ga korisniku. Identifikacija korisnika prilikom registracije je ključni korak u izdavanju sertifikata. Proces registracije predstavlja prvu i najvažniju kariku u realizaciji neporecivosti.
- Skladište sertifikata. U skladištu sertifikata se prave javni ključevi i sertifikati korisnika, kao i tzv. liste poništenih sertifikata (engl. *certificate revocation list*, *CRL*). Spremište se najčešće realizuje pomoću LDAP kompatibilnog direktorijumskog servera.

Sertifikat koji CA izdaje sadrži ime entiteta (ime osobe, naziv organizacije ili naziv servera) i javni ključ. CA potpisuje sertifikat svojim javnim ključem na osnovu javnog ključa entiteta i njemu odgovarajućeg privatnog ključa i na taj način uspostavlja vezu između entiteta i para ključeva. Kao dodatak, sertifikat uključuje datum isticanja sertifikata, naziv CA koji je izdao sertifikat, serijski broj i druge informacije. Sertifikovan korisnik je entitet koji se oslanja na informacije zastupljene u sertifikatu. Sertifikovani korisnici veruju izdavaču po pitanju verodostojnosti izdatih sertifikata tj. sertifikatima koji u potpunosti identifikuju entitet.

Funkcije PKI

PKI je osnova za druge sigurnosne servise. PKI obezbeđuje distribuciju javnih ključeva i sertifikata uz visok nivo sigurnosti i integriteta. Sistemi koji često zahtevaju upotrebu sigurnosnih mehanizama baziranih na PKI su elektronska pošta, razmena podataka putem elektronske trgovine, kućno bankarstvo i elektronski poštanski sistemi. PKI obezbeđuje osnovne sigurnosne servise za sisteme kao što su:

- SSL (Secure Socket Layer), IPsec (Internet Protocol Security) i HTTPS protokoli za sigurnu komunikaciju i transakcije,
- S/MIME[19] (Secure Multipurpose Internet Mail Extension) i OpenPGP[20],
- SET (Secure Electronic Transaction) za razmenu vrednosti.

Osnovne funkcije PKI su izdavanje, osvežavanje, potvrda i oduzimanje sertifikata.

Sertifikati se najčešće izdaju na određeno vreme. Pošto sertifikati prestaju da važe posle određenog vremena, potrebno ih je "osvežavati" ako ništa značajno nije izmenjeno u okruženju. U nekim slučajevima krajnji entiteti će komunicirati direktno sa CA, bez obzira na prisustvo RA. Na primer, komunikacija se obavlja direktno sa CA kad se sertifikat obnavlja.

Sledeća važna funkcija PKI je potvrda sertifikata. Podaci u sertifikatu su vremenom podložni izmenama. Sertifikovani korisnik želi biti siguran u tačnost podataka, što zahteva potvrdu sertifikata. Postoje dva načina za izvršenje potvrde, a PKI može koristiti oba:

- *on-line* potvrda (korisnik zahtevati potvrdu sertifikata direktno od CA svaki put kad mu je potreban) i
- *off-line* potvrda (CA izdraje izdati vreme važenja sertifikata, odnosno par datuma koji definišu period unutar kojeg se informacija sadržana u sertifikatu može smatrati validnom).

Ukrštena sertifikacija (engl. *cross-certification*) je proces razmene javnih ključeve dva CA. Svaki CA će kreirati sertifikat koji čini vezu između jedinstvenog naziva drugog CA i njegovog javnog ključa i potpisati sertifikat sopstvenim privatnim ključem. Na taj način, Ana, kao korisnik čiji je sertifikat izdao CA_1, može da potvrdi svoj identitet Banetu čiji je sertifikat potpisao CA_2.

Pojam koji je blisko vezan za potvrdu je oduzimanje, tj. poništenje (engl. *revocation*) sertifikata. Poništenje je proces objavljivanja u javnosti da je informacija u sertifikatu postala netačna. Do poništenja može doći:

19 Standard koji upotrebom poznatih mehanizama simetričnog i asimetričnog šifrovanja, digitalnih potpisa i sertifikata, omogućuje tajnost i sigurnost elektronske pošte.

20 Hibridni kriptosistem i skup protokola o kome će više biti rečeno u sledećem poglavlju

- u slučaju da je privatni ključ entiteta kompromitovan
- usled izmena podataka (npr. izmenjen broj telefona korisnika), što je češći slučaj

Ako se sertifikat potvrđuje *on-line*, mogućnost opoziva gubi na značaju jer CA može jednostavno dati do znanja da sertifikat nije više važeći. Ipak, ukoliko je potvrda sertifikata *off-line*, metod poništenja postaje kritičan (naročito u slučaju kompromitovanja privatnog ključa).

U nedostatku *on-line* pristupa, najčešće se koristi lista poništenih sertifikata (CRL), na kojoj se nalaze poništeni sertifikati koje je potpisao i izdao CA. Prilikom potvrde sertifikata, veoma je važno da korisnik proveri poslednju verziju liste, kako bi bio siguran da li je sertifikat koji namerava da koristi aktivan ili opozvan.

Jedan od glavnih problema vezanih za liste poništenih sertifikata je problem čekanja na novu CRL (engl. *time-granularity*), odnosno period između trenutka u kom CA dobije informaciju o tome da je određeni sertifikat nevažeći i trenutka izdavanja nove liste. Sve dok se opozvani sertifikat ne pojavi na listi, svaki korisnik koji proverava listu neće znati da je došlo do opoziva i prihvatiće sertifikat za validan. Problem se može rešiti uvođenjem takozvanih inkrementalnih, tj. delta-listi, koje predstavljaju listu izmena nastalih između dva izdavanja potpune CRL. Delta-liste omogućavaju češća obaveštenja o opozivima i time smanjuju mogućnost zloupotrebe nevažećeg sertifikata.

Drugi problem je veličina CRL. Od CA se može očekivati da sertifikuje desetine ili stotine hiljada entiteta, tako da je lako predvidivo da će CRL biti velika. Jedan od načina za rešenje problema veličine CRL je izdavanje različitih lista prema razlozima opoziva ili prema različitim subjektima – rešenje se svodi na deljenje velikih lista na veći broj manjih koje je lakše obraditi. Na primer, CA može izdati jednu listu za standardne opozive (npr. izmena podataka o entitetu) i drugu za slučajeve narušavanja sigurnosti. Tada korisnik može pažnju usmeriti samo na one certifikate koji su oduzeti zbog narušavanja sigurnosti. Slično tome, CA može izdati jednu listu za krajnje korisnike, a drugu za one CA koje mora da sertifikuje.

Primer upotrebe PKI u hibridnom kriptosistemu

U ovom primeru, Ana i Bane dele istu tačku poverenja, odnosno oboje koriste serifikate koje je potpisao isti CA. Primer ilustruje hibridni kriptosistem, pogodan za efikasno šifrovanje i slanje većih poruka u sprezi sa PKI. Najpre opisujemo postupak kreiranja ključeva i sertifikacije.

- Ana i Bane generišu po jedan par ključeva,
- Ana i Bane prosleđuju svoje javne ključeve, nazive i opisne informacije RA,
- RA proverava njihove akreditive i prosleđuje zahtev za izdavanjem sertifikata ka CA,
- CA generiše sertifikate i potpisuje ih svojim privatnim ključem CA,
- Bane i Ana razmenjuju javne ključeve i proveravaju ih na osnovu sertifikata koje preuzimaju sa PKI servera.

Pretpostavimo da Ana želi da pošalje Banetu poruku m. Ana radi sledeće:

- računa heš $h_1 = H(m)$ poruke *m* i potpisuje heš svojim privatnim ključem: $s = Sk_A^{priv}(h_1)$,
- generiše tajni simetrični ključ *k*,
- šifruje poruku *m* i potpisani heš *s* tajnim simetričnim ključem koji se koristi za komunikaciju: $c = Ek(m|s) = Ek(m \mid Sk_A^{priv}(H(m)))$,
- formira digitalni omot, tj. šifruje tajni simetrični ključ Banetovim javnim ključem: $w = Ek_B^{public}(k)$,
- šalje Banetu šifrovanu poruku (sa potpisom) i digitalni omot c|w.

Bane prima šifrovanu poruku i digitalni omot i radi sledeće:

- dešifruje omot svojim privatnim ključem i na taj način dobija tajni simetrični ključ: $k = Dk_B^{private}(w)$,
- dešifruje poruku i potpisani heš koristeći tajni simetrični ključ koji je izračunao u prethodnom koraku: $Dk(c) = (m|s)$,
- određuje heš poruke: $h_2 = H(m)$,

- proverava Anin potpis pomoću njenog javnog ključa, čime dobija heš koji je Ana potpisala: $h_1 = Vk_A^{public}(s)$,
- upoređuje heš koji je on izračunao sa hešom koji je dobio proverom potpisa. Ukoliko se heš vrednosti poklapaju, Bane je siguran u integritet poruke.

Delegiranje odgovornosti

U velikim organizacijama moguće je delegirati odgovornost za izdavanje sertifikata većem broju različitih CA. Do delegacije ovlašćenja za izdavanje sertifikata dolazi ukoliko:

- broj zahtevanih sertifikata predstavlha veliko opterećenje za jedan CA,
- različite organizacione jedinice imaju različite zahteve,
- ukoliko je važno da ovlašćeni CA bude na istom geografskom području na kom se nalazi i entitet kom je potreban sertifikat.

Odgovornost izdavanja sertifikata može se delegirati na podcentre. X.509 standard sadrži model za uspostavljanje hijerarhije CA. Prema ovom modelu, osnovni CA je na vrhu hijerarhije i on sam sebi izdaje samo-potpisani sertifikat (engl. *self-signed certificate*). Vrhovni CA potpisuje sertifikate svim centrima koji se nalaze na sledećem nivou hijerarhije. Sve sertifikate CA na nižem nivou potpisuje CA sa višeg nivoa. Organizacije imaju veliku fleksibilnost po pitanju uspostavljanja hijerarhije CA hijerarhije. Hijerarhija sertifikacionih centara može se prikazati pomoću lančane strukture, odnosno pomoću niza sertifikata koje su izdali podcentri, počev od nižih hijerarhijskih nivoa zaključno sa vrhovnim CA. Lančana struktura prati proceduru sertifikovanja od ogranaka ka korenu hijerarhije. Na primer:

- sertifikat vrhovnog CA (samopotpisan) → sertifikat CA_1 (potpisan od CA) → sertifikat CA_1^A (potpisan od CA_1) → sertifikat entiteta (potpisan od CA_1^A)

Sertifikati i LDAP direktorijum

Protokol za pristup direktorijumskim servisima (*lightweight directory*

access protocol, LDAP) omogučava veliku fleksibilnost u upravljanju sertifikatima unutar jedne organizacije. Na primer, CA može iskoristiti informacije o imenu i drugim podacima novog zaposlenog iz direktorijuma pre dodele sertifikata. CA može iskoristiti informacije iz direktorijuma na druge načine kako bi izdao sertifikate pojedinačno ili masovno upotrebom većeg broja identifikacionih tehnika, zavisno od sigurnosnih polisa date organizacije. Ostali vidovi upravljanja kao što su upravljanje ključevima, obnavljanje i opoziv, mogu biti delimično ili potpuno automatizovani pomoću direktorijuma. Informacije sadržane u direktorijumu takođe mogu biti korišćene sa sertifikatima u kontroli pristupa različitim mrežnim resursima od strane različitih grupa ili korisnika. Izdavanje sertifikata i drugi poslovi upravljanja sertifikatima mogu biti osnovni deo upravljanja korisnicima i grupama. Pre nego što se sertifikat može izdati, javni ključ koji sadrži i odgovarajući privatni ključ moraju biti generisani. Ključ može generisati klijentski softver ili CA; u slučaju da ključeve generiše CA, distribucija se obavlja putem LDAP direktorijuma. Ponekad se jednoj osobi izdaju dva sertifikata: jedan sa parom ključeva za potpisivanje, a drugi za šifrovanje. Podela sertifikata na taj način daje mogućnost zadržavanja privatnog ključa za potpisivanje samo na lokalnom računaru, čime se obezbeđuje najviši nivo neporicanja.

6.6 X.509 sertifikati

X.509 je autentifikaciona platforma, tj. radni okvir (engl. *framework*) dizajniran radi podrške X.500 direktorijumskih servisa. X.509 i X.500 su deo X serije internacionalnih standarda koje su predložili ISO i ITU1. X.500 standard je dizajniran s namerom da obezbedi direktorijumske servise velikih računarskih mreža, dok X.509 obezbeđuje PKI platformu za autentifikaciju X.500 servisa. Prva verzija X.509 standarda pojavila se 1988. godine i predstavlja prvi PKI projekat. Postoji veliki broj kompanija širom sveta čiji su proizvodi zasnovani na X.509 standardu. Na primer, VISA i MasterCard primenjuju X.509 kao osnovu za njihov standard sigurnih elektronskih transakcija. Trenutno je aktuelna izrada X.509 baziranih PKI koji će podržavati globalnu mrežu kao što je Internet. Pored PGP sertifikata, X.509 je jedini PKI sistem koji se praktično upotrebljava.

X.500

Da bismo objasnili X.509 PKI standard, potrebno je najpre dati neke osnovne smernice o X.500 direktorijumu za koji je X.509 dizajniran. X.500 opisuje standarde i mrežne protokole vezane za javne otvorene imenike (engl. *directory*), čije su karakteristike:

- striktna hijerarhijska organizacija i
- dobijanje podataka na osnovu "jedistvenih prepoznatljivih imena" (engl. *distinguished names*, *DN*).

X.500 direktorijum je veoma sličan telefonskom imeniku. X.500 može sadržati imena atributa, kao što su naziv organizacije u kojoj osoba radi, radno mesto, adresu elektronske pošte i sl. Objekat X.500 direktorijuma može predstavljati bilo koji realni entitet – osobu, računar, štampač, organizacionu jedinicu ili kompaniju. Objekat, takođe, može sadržati i javni ključ entiteta.

X.500 direktorijum je hijerarhijski organizovan kao stablo. Svaki čvor, osim vrhovnog, ima jednog roditelja i veći broj dece i označen je relativnim imenom koje je jedinstveno na tom hijerarhijskom nivou. Na primer, ispod vrhovnog čvora postoje zapisi o svakoj državi (oznaka komponente C) i označeni su relativnim imenom koje predstavlja jedinstvenu dvokarakternu oznaku države koju dodeljuje ISO. Ispod svakog čvora države nalaze se čvorovi koji predstavljaju sve oblasti (npr. pokrajine ili republike, oznaka komponente P), a ispod tih čvorova nalaze se čvorovi koji predstavljaju gradove (oznaka komponente L). Slede čvorovi koji predstavljaju organizacije i organizacione jedinice (oznake komponenti O i OU, respektivno). Konačno, svaka organizacija sadrži zapise o svim entitetima koje bi mogla da kontroliše. Svaki od njih je, takođe, označen relativnim imenom (oznaka komponente CN – engl. *common name*).

Jedinstveno prepoznatljivo ime objekta formira se na osnovu relativnih imena, koja su komponente DN, počev od vrha hijerarhije. Na primer, relativno ime osobe *Petar Petrović*, zaposlene u odseku za razvoj kompanije *Pera & Sons,* koja se nalazi u Boru, Republika Srbija, država SCG:

- C = YU, P = "Srbija", L = "Bor", O = "Pera & Sons", OU = "razvoj", CN = "Petar Petrović"

X.509 sertifikati – verzije 2 i 3

X.509 je kreiran kao podrška autentifikaciji zapisa X.500 direktorijuma. X.509 sertifikat u verziji 2 sadrži sledeća polja:

- verzija X.509 sertifikata,
- jedinstveni serijski broj koji je sertifikatu dodelio CA,
- identifikator algoritma kojim je CA potpisao sertifikat,
- X.500 naziv izdavača sertifikata,
- opseg važenja (par datuma između kojih se sertifikat smatra važećim),
- X.500 naziv entiteta koji sadrži privatni ključ koji odgovara javnom ključu sadržanom u sertifikatu,
- jedinstveni identifikator izdavača,
- jedinstveni identifikator entiteta,
- javni ključ i identifikator algoritma za koji se ključ korist,
- digitalni potpis sertifikata privatnim ključem CA.

Kako je X.509 blizak X.500, njegov CA je, uglavnom, slične hijerarhije kao X.500 stablo. X.509 i X.500 su originalno dizajnirani sredinom 1980-tih za *off-line* okruženje međusobno povezanih računara. Prve dve verzije X.509 serifikata koriste slične liste poništenih sertifikata. CRL za X.509 je tabela koja sadrži sledeća polja:

- verzija CRL,
- identifikator algoritma kojim je CA potpisao sertifikat,
- X.500 naziv izdavača sertifikata,
- datum i vreme trenutnog ažuriranja,
- datum i vreme sledećeg ažuriranja,
- tabelu parova serijski broj sertifikata – datum opoziva.

CRL se potpisuje privatnim ključem CA. Za prve dve verzije X.509 sertifikata problemi veličine liste i vremena čekanja na izdavanje nove liste nisu razrešeni. U trećoj verziji sertifikata načinjen je mali pomak u rešenju

tih problema.

Najznačajnija izmena uneta u trećoj verziji sertifikata je proširivost formata sertifikata i CRL. Organizacija koja implementira standard može definisati sadržaj sertifikata shodno svojim potrebama. Na primer, treća verzija X.509 sertifikata daje mogućnost CA da doda sertifikatu liste polisa koje su sledile kreiranju sertifikata. Ove polise imaju ulogu da pomognu korisniku u odluci da li je sertifikat podesan za određenu upotrebu. Na primer, polisa može ukazivati na to da se sertifikovani ključ može koristiti za potpisivanje elektronske pošte ali ne i finansijskih transakcija. Polise predstavljaju vid sigurnosne politike CA. Treća verzija sertifikata može sadržati jedno ili više alternativnih imena entiteta ili izdavača, što omogućava upotrebu X.509 bez pozivanja na podatke iz X.500 direktorijuma. Primeri alternativnih imena uključuju adrese elektronske pošte i web adrese. Alternativna imena se mogu koristiti i za identifikaciju izdavača liste poništenih sertifikata. Proširenja u trećoj verziji omogućavaju da vrednosti atributa zapisa X.500 direktorijuma budu uključene u sertifikat, tako da sertifikat može sadržati dodatne identifikacione podatke, osim naziva subjekta.

CRL takođe može sadržati neka proširenja. Na primer, svaka izdata CRL može biti označena brojem u rastućem redosledu, što omogućava korisnicima da provere da li eventualno neka lista nedostaje. Takođe, za svaki sertifikat u CRL može se navesti razlog opoziva. CRL distribucione tačke omogućavaju redukciju veličine CRL. Da se korisnici ne bi primoravali na prihvatanje kompletnih listi, CA može na određeni način izdeliti listu i izdati svaki deo sa različite distribucione liste. Na primer, korporacijski CA može izdati različit CA za svako odeljenje kompanije. Tada, kad korisnik želi da proveri sertifikat za nekog iz određenog odeljenja, on treba da proveri samo listu tog odeljenja, a ne celu listu. Drugi način podele liste je prema razlozima opoziva sertifikata. Na primer, opozvani sertifikati iz razloga izmene ličnih podataka se postavljaju na jednu listu, dok se oni koji su na primer opozvani iz razloga narušavanja sigurnosti, postavljaju na drugu. Na taj način se liste mogu mnogo brže i efikasnije pretraživati, bez potrebe za procesiranjem lista opozvanih sertifikata iz svih razlika.

Delta-liste su dodatak koji omogućava redukovanje liste CRL na taj način da CA izdaje sažetu listu samo sa izmenama nastalim od poslednjeg izdavanja kompletne liste. Korisnici koji izrađuju sopstvenu CRL bazu mogu koristiti delta-liste kako bi samo dopunjavali svoju listu sa poslednjim

izmenama, bez potrebe da svaki put uzimaju kompletnu listu štedeći time i svoje vreme i smanjujući mrežni saobraćaj. Indirektne liste omogućavaju da CRL budu izdate od strane entiteta, a ne isključivo od strane CA koji je izdao sertifikat. Ovo takođe dozvoljava prikupljanje listi sa više CA i obezbeđivanje jedinstvene distribucione tačke za sve njih. Svi ovi dodaci ne rešavaju problema perioda čekanja na izdavanje nove CRL (*time-granularity*). Čak i ako se koriste izdeljene ili delta-liste, postoji realna verovatnoća zloupotrebe nevažećih sertifikata. X.509 platforma može se koristiti za *on-line* proveru sertifikata, potpuno izbegavajući potrebu za listama poništenih sertifikata.

Registracija objekata

Svaki put kad X.509 ima potrebu da identifikuje neki objekat, kao što je algoritam potpisa, sertifikaciona polisa, korisnički definisan alternativni naziv ili korisnički definisan dodatak, on koristi internacionalno definisan mehanizam identifikacije pomoću identifikatora objekata (engl. *object identifier*, OID). OID je numerička vrednost, sačinjena od niza celih brojeva i jedinstven je u odnosu na druge OID.

OID je označen u skladu sa hijerarhijskom strukturom vrednosno-označenih centara. U osnovi, svaka kompanija može biti centar za dodelu vrednosti. Kompanija sadrži vrednost koju je sebi dodelila i koja predstavlja prefiks za sve kasnije vrednosti koje ga definišu. Posmatrajte, na primer, CA koji funkcioniše u SAD čiji je OID 2-16-840-1-45356. Ovi brojevi imaju značenje u okviru hijerarhije: na primer, prefiks 2 ukazuje na ogranak hijerarhije koji zajednički administriraju ISO i ITU, 840 je kod države SAD. Američki nacionalni registracioni centar (ANSI) dodeljuje prefiks 1 svim organizacijama koje on registruje. Broj 45356 ANSI dodeljuje CA. Oid sertifikacionog centra će biti prefiks identifikatora svih objekata koje CA namerava da registruje. Na primer, ukoliko CA želi da sertifikuje u hijerarhijskom stablu (npr. ogranak broj 3) određenu sertifikacionu polisu kojoj je dodelila neku vrednost (recimo 15), polisa će biti identifikovana kao objekat pod brojem 2-16-840-1-45356-3-15.

6.7 Pitanja i zadaci

6.1 Sastavite program koji generiše proste brojeve na intervalu koji zadaje korisnik.

6.2 Sastavite program za modularno stepenovanje. Korisnik unosi vrednosti *n*, *g* i *a*, a program vraća rezultat $a = g^b \bmod n$.

6.3 Sastavite program koji proverava da li je vrednost *g* generator multiplikativne ciklične grupe {0, 1, ..., n-1}. Parametre g i n unosi korisnik.

6.4 Koristeći programe iz zadataka 6.1-6.3, sastavite program koji simulira Diffie-Hellmanov protokol za razmenu ključeva. Program treba da generiše prvi prost broj *n* veći od neke vrednosti koje zadaje korisnik i odredi generator *g* multiplikativne ciklične grupe {0, 1, ..., n-1}. Program zatim nudi korisniku da unese dva broja *x* i *y* manja od *n* i na osnovu njih generiše vrednosti $X = g^x \bmod n$ i $Y = g^y \bmod n$ koje učesnici u protokolu razmenjuju kao i ključ $k = Y^x \bmod n = X^y \bmod n$.

6.5 Sastavite program koji generiše RSA par ključeva. Kao faktore broja n koristite prve proste brojeve *p* i *q* veće od dve vrednosti koje zadaje korisnik (vrednosti ne smeju biti manje od 1000). Javni ključ *e* program treba da odabere slučajno sa odgovarajućeg intervala i na osnovu njega da generiše privatni ključ. Javni ključ se smešta u datoteku *public.RSA.key*, a privatni u datoteku *private.RSA.key*.

6.6 Sastavite program koji šifruje, odnosno dešifruje binarne ili tekstualne datoteke sa diska RSA algoritmom.

Prvi parametar koji korisnik prenosi programu je šta želi da radi: da šifruje ili da dešifruje datoteku. U slučaju da se obavlja šifrovanje, korisnik kao ostale parametre unosi ime datoteke u kojoj se nalazi otvoreni tekst, ime datoteke u kojoj će biti smešten šifrat i ime datoteke u kojoj se nalazi javni ključ. U slučaju da se obavlja dešifrovanje, korisnik kao ostale parametre unosi ime datoteke u kojoj se nalazi šifrat, ime datoteke u koju će biti smešten otvoreni tekst i ime datoteke u kojoj se nalazi privatni ključ.

Šifrovanje i dešifrovanje datoteke možete obaviti konverzijom otvorenog teksta ili šifrata u odgovarajuće numeričke vrednosti i obratno, na nivou grupe bitova, karaktera ili grupe karaktera – obratite samo pažnju na ograničenje da prilikom šifrovanja *c* i *m* moraju biti manji od *n*. Konverzija u numeričku vrednost obavlja se nakon čitanja jedne grupe bitova, karaktera ili grupe karaktera ulazne datoteke (pre šifrovanja ili dešifrovanja jedne vrednosti). Konverzija u suprotnom smeru se obavlja pre upisa u datoteku.

6.7 Izmenite program iz prethodnog zadatka tako da obavlja potpisivanje datoteke i proveru potpisa RSA algoritmom. Program bi trebalo da generiše datoteku sa potpisom istog imena, ali sa ekstenzijom *.sig*.

6.8 Sastavite program koji obavlja faktorizaciju velikih brojeva metodom grube sile. Pri faktorizaciji prirodnog broja n pretražite skup prirodnih brojeva na intervalu [2, $\sqrt{n}$].

6.9 Sastavite program koji obavlja faktorizaciju velikih brojeva Pollardovim p-1 algoritmom za faktorizaciju. Od korisnika se očekuje da zada veliki broj i granicu *B*.

6.9 Sastavite program koji obavlja faktorizaciju RSA velikih brojeva ako je poznat privatni i javni ključ.

6.10 Sastavite program koji generiše par ključeva za El-Gamal algoritam. Javni ključ se smešta u datoteku *public.ElG.key*, a privatni u datoteku *private.ElG.key*. Program treba da generiše prvi prost broj *p* veći od neke vrednosti koje zadaje korisnik i odredi generator *g* multiplikativne ciklične grupe {0, 1, ..., p-1}. Broj *a* sa intervala [1, p-1] se bira slučajno.

6.11 Izmenite program iz zadatka 6.6 tako da se šifrovanje datoteke obavlja El-Gamal algoritmom. Koristite ključeve generisane programom iz zadatka 6.10.

7

Kriptografski softver

7.1 Šifrovanje diskova i komunikacionih kanala

U praksi se najčešće nailazi na šifrovanje elektronske pošte i mrežnog saobraćaja, kao i pojedinačnih datoteka na masovnim memorijskim medijumima i kompletnih sistema datoteka – ove kriptografske metode zaštite ćemo kao najčešće korišćene ovde i opisati. Međutim, svaki podatak na računaru može se zaštititi šifrovanjem – postoje programi koji šifruju kompletne diskove, prenosive medijume, čak i kod programa instaliranog na računaru.

Šifrovanje komunikacionih kanala

Teorijski posmatrano, šifrovanje se može obaviti na bilo kojem sloju OSI modela. U praksi, šifrovanje se obavlja:

- link po link (engl. *link-by-link encryption*) – obavlja se na nižim slojevima OSI referentnog modela (prvi i drugi sloj). Šifruje se sve što prođe kroz odgovarajući link; šifrovanje i dešifrovanje se obavlja u svakom čvoru.

- s kraja na kraj (engl. *end-to-end encryption*) – obavlja se na višim slojevima referentnog modela (kao što je aplikacioni sloj). Šifruje se samo poruka; šifrovanje obavlja pošiljalac a dešifrovanje primalac.

Prednosti šifrovanja link-po-link su transparentnost za korisnika i jednostavnost rukovanja ključevima (potreban je samo jedan ključ po linku). Ovakav način šifrovanja obezbeđuje sigurnost u smislu skrivanja informacija o rutiranju i učesnicima u komunikaciji (jer se sve šifruje). Osnovni nedostatak je izloženost podataka u međučvorovima – pogrešno rutiranje može izložiti podatke u čvorovima na kojima rade zlonamerni korisnici. Šifrovanje link-po-link je transparentno za korisnika i ne nudi mogućnost slanja otvorenog teksta, jer se svi podaci šifruju. Šifrovanje link-po-link se implementira u hardveru, što je najčešće skupo.

Prednosti šifrovanja s kraja na kraj su viši nivo sigurnosti, mogućnost odabira podataka koji će se prenositi u šifrovanom obliku (ostali podaci se

prenose kao otvoren tekst) i mogućnost softverske implementacije (što je znatno jeftinije). Dakle, šifrovanje nije transparentno za korisnika – korisnik bira algoritam i određuje podatke koji će biti šifrovani, a zatim primenjuje operaciju šifrovanja. Šifrovanje s kraja na kraj zahteva složeniji mehanizam za rad sa ključevima – u slučaju da je algoritam simetričan, potreban je jedan ključ po paru korisnika, što za veći broj korisnika implicira korišćenje kriptosistema sa javnim ključem ili hibridnih kriptosistema. Analiza saobraćaja je moguća jer se informacije o rutiranju i učesnicima u komunikaciji ne šifruju.

Kombinovanjem prethodne dve tehnike dobija se sistem koji obezbeđuje veći nivo sigurnosti: šifrovanjem fizičkog linka skrivaju se informacije o rutiranju, dok se šifrovanjem s kraja na kraj sprečava izlaganje otvorenog teksta u čvorovima. Upravljanje ključevima za ove dve tehnike se razdvaja: mrežni administrator je zadužen za šifrovanje linka, dok su korisnici zaduženi za šifrovanje s kraja na kraj.

Šifrovanje podataka na diskovima

Ako Bane ne primi poruku ili ne može da je dešifruje, Ana je može poslati ponovo. Ako Ana ne može da dešifrovati podatke sa diska (npr. izgubila je ključ), podaci su zauvek izgubljeni! Pri radu sa ključevima i šifrovanim podacima treba uzeti u obzir da podaci (i ključevi) mogu biti smešteni na disku dosta dugo pre nego što budu dešifrovani, kao i da ključ za dešifrovanje podataka ima istu vrednost kao i sami podaci! Ključeve treba čuvati na sigurnom mestu!

Šifrovanje podataka na diskovima može se obaviti na nivou datoteka i na nivou drajvera. Za šifrovanje na nivou datoteka koriste se klasični sistemi datoteka i posebni programi za šifrovanje datoteka i direktorijuma (kao što su PGP, GPG, BestCrypt Archive i slični). Jednostavno se implementira i koristi. Takođe, korisnici mogu jednostavno pomerati podatke na drugi računar ili napraviti rezervnu kopiju. Degradacija performansi je zanemarljiva. Za različite datoteke mogu se koristiti različite lozinke.

Najveći nedostatak ovog načina šiftovanja je mogućnost ostavljanja otvorenog teksta u raznim direktorijumima za smeštaj privremenih datoteka. Ukoliko kod kuće imate instaliran Microsoft Windows, primetićete

da prilikom rada sa 99,99% aplikativnih programa, program formira privremene datoteke u nekom direktorijumu[21]. Krajnji korisnici najčešće toga nisu ni svesni. Vi možete da šifrujete svoj dokument kao datoteku – on će u tom obliku za napadača predstavljati nešto što neće moći da pročita. Međutim, isti takav dokument u obliku otvorenog teksta napadač će naći u direktorijumu u kom taj program čuva privremene datoteke. Da ne shvatite pogrešno – ovde niste samo vi krivi, kriv je i proizvođač softvera koji u svoj program nije ugradio rutine za uklanjanje "smeća" sa računara. Dakle, podaci, osim na željenim mestima, najčešće postoje u obliku otvorenog teksta u raznim privremenim direktorijumima (temp, recent, recycle-bin), u programskom kešu, na drugim računarima (ukoliko radite na projektu u mreži) ili na izmenljivim medijumima (arhiva). Takođe, uzmite u obzir da se prilikom brisanja datoteke ažuriraju samo meta-podaci sistema datoteka (na primer, indeksni čvorovi ili FAT tabele), dok se pravi podaci sa diska ne brišu! Ovde paranoja koja prati problematiku zaštite dobija svoje opravdanje. Postojanje otovrenog teksta u najboljem slučaju implicira mogućnost napada tipa "poznat otvoreni tekst", dok su u najgorem slučaju vaši podaci "javno dostupni".

Ukoliko šifrujete celu particiju ili disk, napadač neće moći da ostvari pristup vašim podacima bez ključa, zato što je ključ potreban za pristup bilo kom direktorijumu. Šifrovanje cele particije ili diska obavlja se na nivou drajvera – sistem datoteka ili ceo disk se najpre šifruje, a zatim poseban drajver upravlja rutinama za šifrovanje i obezbeđuje virtuelni sistem datoteka prema operativnom sistemu. Šifrovanje na nivou drajvera obezbeđuju programi, kao što su TrueCrypt, PGPDisk, EFS (uslovno) i razni kriptografski sistemi datoteka pod Linuxom.

Osnovna prednost šifrovanja na nivou drajvera je smeštaj privremenih datoteka na šifrovanom sistemu datoteka. Šifrovanje je transparentno za korisnika – korisnik ne može da zaboravi da šifruje nešto jer se to od njega i ne očekuje. Upravljanje ključevima je znatno komplikovanije zbog velikog broja korisnika koji na različit način pristupaju različitim delovima datoteka, a zahteva se i postojanje sigurnog smeštaja za ključeve. Što se tiče kriptografskih napada, svi se upućuju na glavni ključ ili lozinku koja štiti glavni ključ – otkrivanje ključa ili lozinke omogućava napadaču pristup svim podacima. Osnovni nedostatak ovog načina šifrovanja su problemi koji se često javljaju kod drajvera za virtuelne sisteme datoteka[22], a i pad

21 Na primer, direktorijum "Local Settings\Temp" u vašem profilu

22 Vidite Murphyjev zakon, primenljiv je na 99,99% zaštitnih mera koje ćete primeniti

performansi je primetan. Pošto su brzina šifrovanja i dešifrovanja veoma značajan faktor[23], za šifrovanje se obično koristi specijalizovani hardver[24] ili namenski dizajnirani algoritmi, optimizovani po pitanju brzine.

7.2 Pretty Good Privacy

U predgovoru knjige *Applied Cryptography - Protocols, Algorithms and Source Code in C*, Bruce Schneier kaže da postoje dve vrste kriptografije: kriptografija koja će sprečiti vašu mlađu sestru da čita vaše datoteke i kriptografija koja će sprečiti državu da čita vaše datoteke. Ova druga vrsta kriptografije je trn u oku svake države, bez obzira na prava i slobode koje ta država javno zagovara. Argument koji vrlo često upotrebljavaju protivnici dostupnosti jake kriptografije jeste da kriptografija nije potrebna poštenim građanima koji ne rade ništa protivzakonito. Međutim, policiji je potreban nalog za pretres ukoliko žele pretražiti nečiji stan? Setite se poznate analogije: nezaštićeni e-mail je razglednica koju svako može pročitati, a šifrovani e-mail je pismo u koverti koje je skriveno od očiju javnosti. Da li to znači da su ljudi koji šalju pisma sumljivi i da li treba po inerciji pretpostaviti da nešto kriju? Ili jednostavno žele da sačuvaju svoju privatnost?

U svakom slučaju, javna dostupnost jakog kriptografskog softvera je nešto što vojno-bezbedonosne organizacije žele onemogućiti na svaki način. Tako, na primer, u Sjedinjenim Američkim Državama zakon o izvozu oružja ITAR[25] (*International Traffic in Arms Regulations*) strogo zabranjuje izvoz jakog kriptografskog softvera, svrstavajući ga u istu kategoriju sa delovima borbenih aviona, kao i sa klasičnim, hemijskim i biološkim naoružanjem. Slični zakoni postoje u drugim državama.

Godine 1991. u američkom Senatu predstavljen je predlog novog zakona (Senate Bill 266), prema kom bi svi proizvođači opreme za sigurnu

23 Performanse računarskih sistema u velikoj meri zavise od brzine diskova, koji su relativno spori uređaji u odnosu na ostatak računarskog sistema. Ukoliko brzinu diska, dodatno smanjite neefikasnim algoritmom za šifrovanje, ništa korisno niste uradili, jer će se korisnici sistema pre ili kasnije žaliti i od vas zahtevati da performanse povećate, čak i ukoliko to znači da se odustaje od šifrovanja.

24 "*tamper-proof*" crna kutija na disk kontroleru, na samom disku ili kao posebna kartica

25 http://www.pmdtc.org/reference.htm, *International Traffic in Arms Regulations*, Directorate of Defense Trade Controls, U.S. Department of State

komunikaciju bili prisiljeni ugrađivati u svoje proizvode mogućnost da država prema potrebi dešifruje zaštićene komunikacione kanale. Navodimo kraći isečak ovog zakona na engleskom jeziku:

> "*It is the sense of Congress that providers of electronic communications services and manufacturers of electronic communications service equipment shall insure that communications systems permit the Government to obtain the plain text contents of voice, data, and other communications when appropriately authorized by law.*"

Navedeni predlog nikada nije zaživeo kao važeći zakon, ali je inspirisao Philipa R. Zimmermanna da te iste godine napiše program PGP (Pretty Good Privacy), inicijalno namenjen zaštiti sadržaja elektronske pošte. Prvu verziju svog programa Zimmermann je dao prijateljima; PGP se ubrzo našao na Internetu, a samim tim i van Sjedinjenih Američkih Država. Zimmermann je tako prekršio ITAR i postao meta višegodišnjeg sudskog progona američkih vlasti tokom kojegg je dobar deo vremena proveo u zatvoru. Sudski proces protiv Zimmermanna obustavljen je 1996. godine, pa su odmah počele kružiti glasine kako se Zimmermann nagodio sa državom i ugradio u svoj proizvod mehanizam pomoću kog bi država imala uvid u podatke zaštićene PGP-om. Ova glasina pokazala se neistinitom, a najbolji dokaz za to je dostupnost izvornog programskog koda čija bi analiza već odavno otkrila eventualnu zamku (engl. *trapdoor*). Zimmermann je, takođe, imao i problema sa kompanijom RSA Data Security čiji je algoritam RSA upotrebio bez plaćanja licence. Da bi izbegao sudsku tužbu, Zimmermann je potpisao sporazum kojim se obvezao da neće više distribuirati PGP. Razvoj i distribuciju nastavili su drugi, a PGP je postao veoma popularan.

Godine 1997. pojavila se prva legalna verzija PGP-a izvan Sjedinjenih Država. Strogi zakoni izričito su zabranjivali izvoz kriptografskog softvera, pa je PGP izvezen u obliku knjige koja je sadržavala njegov programski kod. Knjiga je skenirana i pomoću softvera za prepoznavanje karaktera (OCR) PGP je vraćen u elektronski oblik. Ovom bizarnom operacijom uklonjena je i zakonska prepreka distribuciji PGP-a.

Navodimo neke od značajnijih nagrada koje su Philipu Zimmermannu dodeljene za PGP:

- Information Week Most Important Products of 1994,
- Chrysler Award for Innovation in Design (1995),
- Pioneer Award from the Electronic Frontier Foundation (1995),
- Time Magazine Net 50, one of the 50 most influential people on the internet in 1995,
- Norbert Wiener Award from Computer Professionals for Social Responsibility (1996),
- PC Week IT Excellence Award (1996),
- Network Computing Well-Connected Award for Best Security Product (1996),
- Lifetime Achievement Award from Secure Computing Magazine (1998),
- Louis Brandeis Award from Privacy International (1999).

Šta je PGP i koliko je siguran?

Najkraće rečeno, PGP je hibridni kriptosistem. Pridev hibridni odnosi se na činjenicu da PGP prilikom zaštite podataka koristi kombinaciju najboljih karakteristika simetričnih algoritama i kriptografije sa javnim ključem. Osnovna namena PGP-a je zaštita elektronske pošte (šifrovanje i digitalno potpisivanje), ali su u novijim verzijama dodate funkcije za šifrovanje i digitalno potpisivanje datoteka, formiranje šifrovanih sistema datoteka i šifrovanje celih diskova (engl. *whole disk encryption*). Najjednostavnije rečeno, šifrovanje poruka PGP kriptosistemom obavlja se na sledeći način:

- otvoreni tekst se komprimuje i šifruje simetričnim ključem,
- simetrični ključ se šifruje javnim ključem pošiljaoca,
- pošiljalac šalje šifrate otvorenog teksta i simetričnog ključa primaocu.

Ukoliko se ne naznači drugačije, PGP će u prvom koraku izvršiti kompresiju podataka ZIP algoritmom. Kompresijom se smanjuje količina podataka koja se prenosi preko mreže. Kompresijom se postiže i viši nivo sigurnosti, jer se očekivani uzorci u otvorenom tekstu uklanjaju, što otežava kriptoanalitičke napade. PGP zatim generiše 128-bitni simetrični ključ za

jednokratnu upotrebu[26] (engl. *session key*), deli poruku na blokove dužine 64 bita koje šifruje IDEA algoritmom[27]. Simetrični ključ PGP šifruje RSA algoritmom koristeći javni ključ primaoca. Primaocu se šalje šifrat komprimovanog teksta i šifrat simetričnog ključa. Dešifrovanje se obavlja u suprotnom smeru: primalac dešifruje simetrični ključ svojim privatnim ključem, dešifruje zaštićene podatke simetričnim ključem i obavlja dekompresiju.

Osim šifrovanja, PGP nudi korisnicima mogućnost da digitalno potpišu poruke koje šalju. Digitalni potpis ostvaruje se pomoću MD5 heš funkcije i RSA algoritma na sledeći način:

- PGP računa MD5 heš otvorenog teksta,
- heš se potpisuje privatnim ključem pošiljaoca,
- pošiljaoc šalje otvoreni teksta i potpisani heš primaocu.

Primalac proverava potpis tako što upoređuje vrednost dobijenu dešifrovanjem potpisa javnim ključem pošiljaoca i heš primljene poruke koji je sam izračunao.

PGP se sastoji od četiri kriptografske komponente: simetričnog algoritma, asimetričnog algoritma, heš funkcije i generatora pseudo-slučajnih brojeva. Kao što već pretpostavljate, njegova sigurnost zavisi od sigurnosti najslabije komponente.

- Ukoliko ne koristite DES, simetrični algoritam ne predstavlja slabu tačku, jer je napad grubom silom za sada jedini opisani napad koji može dovesti do otkrivanja ključa AES, IDEA i Blowfish algoritama (naravno, pod uslovom da neko može da ispita 2^{128} kombinacija ključeva u dovoljno kratkom vremenskom intervalu).

- RSA je zaštićen problemom faktorizacije velikih brojeva, što predstavlja računski veoma komplikovan, gotovo nerešiv zadatak. Takozvanim GNFS algoritmom 1999. godine faktorizovan je broj sa 155 cifara. Nakon četvoromesečnih priprema na CRAY

26 PGP generiše novi simetrični ključ za svaku poruku.

27 IDEA je podrazumevani simetrični algoritam, a umesto njega se mogu koristiti i Triple-DES, CAST-128, Blowfosh, AES i drugi algoritmi. U tom slučaju PGP generiše simetrične ključeve odgovarajuće dužine, shodno simetričnom algoritmu koji se koristi.

superračunarima, postupak distribuirane faktorizacije na 292 računara trajao je nešto više od 5 meseci. Utrošeno je 8000 MIPS godina (1 MIPS godina je ekvivalent $3{,}15 \cdot 10^{13}$ operacija). Ukoliko se uzme u obzir da minimalna dužina RSA ključa (1024 bita) odgovara broju sa 309 cifara, dok dužina ključa od 2048 bita odgovara broju sa 617 cifara, zaključuje se da je RSA veoma siguran algoritam.

- Heš funkcija MD5 koju PGP koristi za potpisivanje je tako konstruisana da za dva ulaza koja se razlikuju samo u jednom bitu daje dva potpuno različita izlaza. Teoretski, postoji mogućnost da za dva različita ulaza heš funkcija ponudi identičan izlaz, ali verovatnoća tog događaja ekstremno je mala. Pretpostavite da imate poruku koja je već digitalno potpisana i da želite pronaći drugu poruku čiji je MD5 heš identičan. Ukoliko računar upoređuje heš vrednosti milijardu poruka u sekundi, za pronalaženje željene poruke bilo bi potrebno oko 10^{22} godina. Postoji još jedan pristup koji se može iskoristiti za pronalaženje dva različita ulaza za koje heš funkcija daje jednake izlaze. Reč je o tzv. rođendanskom napadu (engl. *birthday attack*), kojim se prostor pretraživanja za MD5 smanjuje na 264 slučajeva. Opisanom računaru koji poredi milijardu poruka u sekundi trebalo bi u proseku 585 godina da pronađe par poruka sa identičnim hešom.

- Generisanje slučajnih brojeva predstavlja nepremostiv problem, jer je računar deterministički uređaj. Statistička analiza izlaza generatora pseudo-slučajnih brojeva pokazala se vrlo uspešnom u predviđanju nizova koji će biti generisani, pod uslovom da je početno stanje generatora poznato.

PGP Desktop 8.x/9.x

Počev od prve verzije, pa zaključno sa verzijom 7.0.3 (odnosno verzijom PGPDisk 6.5), PGP je distribuiran kao besplatan softver (engl. *freeware*)[28]. Nakon toga, distribuciju preuzima PGP Coorporation koja PGP distribuira kao komercijalan proizvod, dostupan u nekoliko varijanti za računare koji rade pod Windows i Mac OS X operativnim sistemom. PGP Coorporation distribuira proizvode za kućnu i kancelarijsku upotrebu (PGP Desktop Home i Professional, PGP Whole Disk Encrypyion for Professionals) i proizvode

28 Besplatne verzije možete preuzeti sa web stranice www.pgpi.com/download/.

namenjene srednjim i većim kompanijama, sposobne da opslužuju do 10000 korisnika (PGP Whole Disk Encryption for Enterprises, PGP Universal Series 100, 200, 500). Navodimo značajnije funkcije integrisane u PGP Desktop programski paket:

- šifrovanje datoteka (uključujući mogućnost kreiranja izvršnih SDA[29] datoteka),
- sigurno brisanje datoteka (engl. *shredding*)[30],
- generisanje PGP volumena, tj. šifrovanih sistema datoteka u datoteci (engl. *volume disk encryption*), koji se aktiviraju preko logičkih diskova,
- šifrovanje celog diska (engl. *whole disk encryption*), pod uslovom da je disk neparticionisan i da nije u pitanju RAID volumen (na primer, *stripe* ili *mirror* set),
- podrška za rad sa identifikacionim karticama,
- šifrovanje i potpisivanje elektronske pošte (podrška za Outlook i Outlook Express).

Opisaćemo najpre postupak rada sa programskim paketom PGP Desktop 8.1. Pošto je paket prilično jednostavan za upotrebu[31], postupci se opisuju samo ukratko. Više uputstva možete naći u pratećoj dokumentaciji programskog paketa i na Internetu.

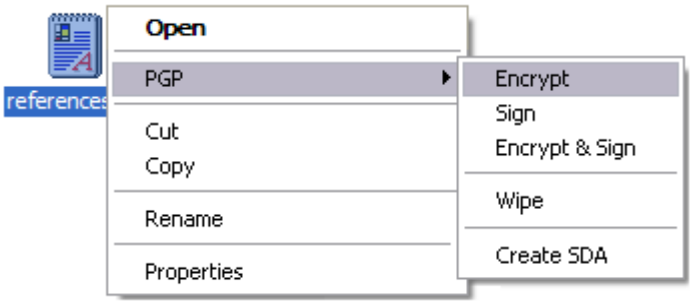
Ukoliko želite da šifrujete datoteku simetričnim algoritmom pomoću

29 Arhiva šifrovanih datoteka koja u sebi sadrži izvršni kod koji će prilikom pokretanja zatražiti od korisnika da unese lozinku i na osnovu lozinke dešifrovati i dearhivirati sadržaj datoteke

30 U blokove podataka koje datoteka zauzima nekoliko se puta upisuju različiti podaci (shodno algoritmu za sigurno brisanje koji se koristi), a zatim nule. Na kraju se na sličan način čiste i informacije o datoteci u delu za meta podatke. Nakon toga datoteku ne možete povratiti programom za povratak obrisanih datoteka. Ukoliko koristite sistem datoteka sa dnevnikom transakcija (engl. *journal*), sigurno brisanje NIJE garancija da se delovi datoteke (ili datoteka u celosti) ne mogu povratiti (što zavisi od sadržaja dnevnika).

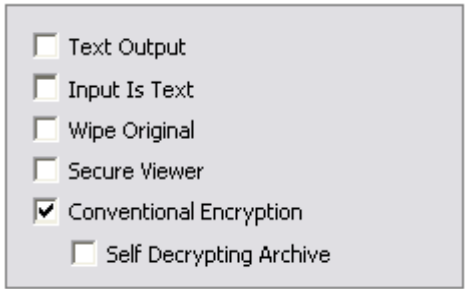
31 Jednostavnost za upotrebu i naklonjenost ka korisniku se i očekuje od komercijalnog paketa koji ima svoju potpuno funkcionalnu besplatnu alternativu. Takođe, možete očekivati (iako se to teško dokazuje i obično sama pomisao na to može dovesti do sudske tužbe) da komercijalan kriptografski softver koji NIJE meta napada države ili bezbednonosnih organizacija (čitaj: vojska, CIA i njima slični), poseduje i odgovarajući *trapdoor* (videti zakon ITAR).

lozinke (a ne pomoću ključa), dovoljno je da u kontekstnom meniju datoteke odaberete opciju PGP → Encrypt (slika 7.1).



Slika 7.1 *– Šifrovanje datoteke simetričnim algoritmom*

Potrebno je da, zatim, u dijalogu koji sledi štiklirate Conventional Encryption (slika 7.2). Ukoliko želite da uz to i sigurno obrišete originalnu datoteku, štiklirajte i opciju Wipe Original.

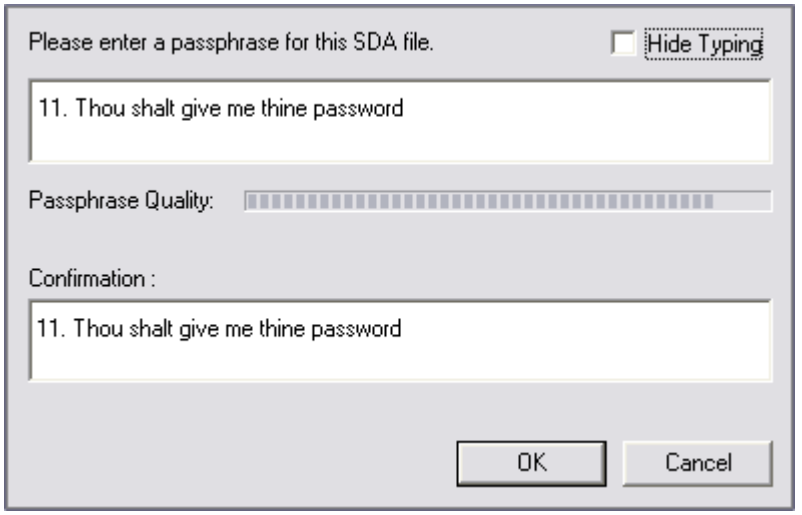


Slika 7.2 *– Izbor opcije za šifrovanje lozinkom umesto ključem*

Sledeći dijalog će od Vas tražiti da unesete lozinku (slika 7.3). Lozinku morate uneti dva puta, čime se sprečavaju slučajne greške. PGP će vas, pritom, obavestiti o složenosti vaše lozinke (što složenija, to bolja). Ukoliko želite da lozinku sakrijete od tuđih očiju, štiklirajte Hide Typing .

Ukoliko želite da šifrujete datoteku nečijim javnim ključem, koristi se hibridni kriptosistem. Potrebno je da javne ključeve primaoca (kojima je šifrat namenjen) iz skupa javnih ključeva ubacite u listu Recipients – tada

će ključ kojim je datoteka šifrovana biti šifrovan javnim ključevima svih primaoca. Šifrovane datoteke imaju ekstenziju *.pgp* (na primer, šifrat datoteke MyDoc.txt dobija ime MyDoc.txt.pgp) i mogu se dešifrovati pozivanjem opcije PGP→Decrypt & Verify iz kontekstnog menija datoteke.

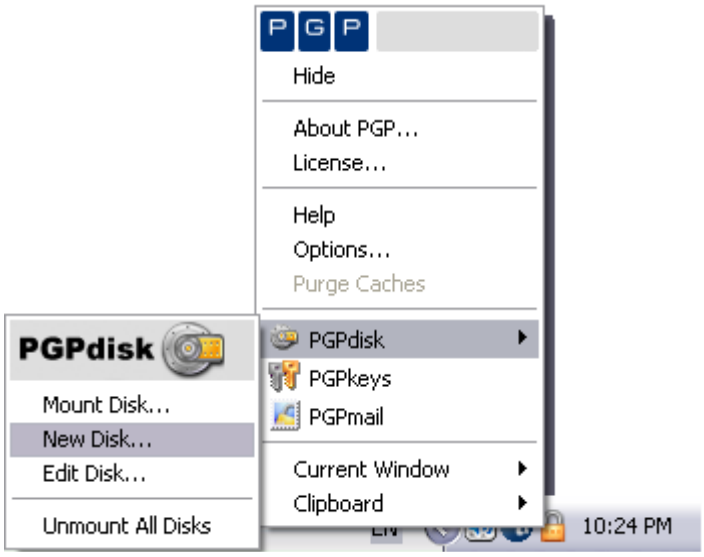


Slika 7.3 *– Unos lozinke*

Opcija Wipe u kontekstnom meniju otvoriće dijalog za sigurno brisanje datoteke. Datoteku možete obrisati ukoliko za to imate adekvatne NTFS dozvole – PGP ne dozvoljava da obrišete datoteku ukoliko nemate pravo da to uradite.

Izvršnu SDA datoteku možete kreirati pomoću opcije Create SDA iz menija ili štikliranjem dugmeta Self Decrypting Archive u dijalogu sa slike 7.2. SDA se štite isključivo lozinkom – ne možete generisati SDA i zaštititi je ključem. Prilikom dešifrovanja, odnosno pokretanja SDA arhive, korisnik unosi lozinku; nakon toga, arhiva se dešifruje i iz nje se obavlja ekstrakcija datoteka. Prednost SDA u odnosu na klasične šifrate je u tome što na računaru na kome želite dešifrovati arhivu ne morate imati instaliran PGP.

Opisaćemo ukratko i postupak generisanja šifrovanih sistema datoteka u datoteci. U meniju System Tray ikonice PGP-a (žuti katanac) odaberite opciju PGP Disk → New Disk (slika 7.4). Otvoriće se čarobnjak za kreiranje PGP volumena (PGPdisk Creation Wizard) koji će Vas provesti kroz ceo postupak.



Slika 7.4 *– Generisanje PGP volumena*

Od Vas se, najpre, zahteva da unesete:

- ime i veličinu volumena,
- način na koji želite da aktivirate volumen – mapiranjem na logički disk ili montiranjem (engl. *mount*) na neki drugi NTFS direktorijum,
- algoritam za šifrovanje volumena,
- sistem datoteka na volumenu.

PGP nudi dva načina zaštite volumena – pomoću javnog ključa (hibridni kriptosistem) ili lozinke (simetrični kriptosistem).

PGP, takođe, obezbeđuje i mehanizam za potpisivanje i šifrovanje elektronske pošte u Outlook i Outlook Express klijentima (PGPmail) i alat za rad sa ključevima (PGPkeys).

OpenPGP

Godine 1997. organizacija IETF (Internet Engineering Task Force)

formirala je OpenPGP radnu grupu u nameri da na osnovu zaštićenog proizvoda (PGP) definiše standard koji se može implementirati u bilo kom softverskom proizvodu bez plaćanja licence. Kao rezultat rada radne grupe nastao je OpenPGP (RFC 2440[32]) – protokol za šifrovanje elektronske pošte pomoću kriptografije sa javnim ključevima, zasnovan na originalnoj PGP distribuciji Phillipa Zimmermanna. OpenPGP protokol definiše standardne formate šifrovanih poruka, potpisa i sertifikata za razmenu javnih ključeva. Infrastruktura javnih ključeva definisana OpenPGP standardom primenljiva je, osim šifrovanja elektronske pošte, i na druge aplikacije. Trenutno je OpenPGP trenutno vodeći standard u kriptografiji sa javnim ključevima.

7.3 GNU Privacy Guard

Prema zvaničnoj dokumentaciji, GNU Privaci Guard (GnuPG) je alat koji obezbeđuje mehanizme za sigurnu komunikaciju i skladištenje podataka. GnuPG se može upotrebiti za šifrovanje i potpisivanje podataka i obezbeđuje mehanizme za relativno jednostavno upravljanje ključevima. Paket je prilagođen OpenPGP standardu, opisanom u RFC2440. Licenciran je pod GNU GPL[33] licencom, što znači da je besplatan za korišćenje za privatne i poslovne namene, kao i da je izvorni kod programa javno dostupan. GnuPG tokom rada koristi sledeće algoritme:

- BLOWFISH, TWOFISH, RIJNDAEL (sa ključem dužine 128, 192 i 256 bita), 3DES i CAST5 za simetrično šifrovanje
- RSA i El-Gammal za šifrovanje javnim ključem
- DSA za digitalno potpisivanje
- MD4, SHA1 i RIPEMD160 heš funkcije

GNU Privacy Guard[34] je dostupan za Linux i Windows operativne sisteme. Većina Linux disribucija isporučuje se sa GnuPG paketom i pratećim

32 Više o tome možete naći na http://www.ietf.org/rfc/rfc2440.txt

33 Ukoliko niste sigurni koja su vaša prava u odnosu na softver licenciran pod GNU opštom javnom licencom (GPL), pogledajte stranicu www.gnu.org/copyleft/gpl.html (obratite pažnju na ime direktorijuma copyleft !) Videćete da je licenca krajnje naklonjena korisnicima.

34 Zvanična web stranica je www.gnupg.org. Sa te stranice besplatno možete preuzeti izvršni i izvorni kod programa, prateću dokumentaciju, proširenja i dodatke.

softverom (kao što je Kgpg front-end). Veliki broj programa na Linux sistemima (kao što je, na primer, Kmail klijent za elektronsku poštu) uključuje podršku za OpenPGP standard (samim tim i za GnuPG). Ukoliko u distribuciju Linux sistema koju koristite nije uključen GnuPG, tar.gz paket možete preuzeti sa zvanične GnuPG web stranice[35]. Win32 distribucija se isporučuje sa programom koji vas vodi kroz jednostavan postupak instalacije.

Rad sa GnuPG paketom

Gnu Privacy Guard je aplikacija koja radi u tekstualnom režimu rada. Ukoliko na računaru instalirate samo GnuPG, biće vam omogućeno da radite sa ključevima, šifrujete i potpisujete datoteke – iz komandne linije. Rad u komandnoj liniji ponekad može da bude odbojan (ukoliko niste sistem administrator ili "*geek*"). Ilustrovaćemo to primerom kreiranja para ključeva, što se od svakog korisnika i očekuje na početku rada sa paketom. Kako bi ceo postupak izgledao još složenije, pretpostavićemo da korisnik radi na Linux sistemu.

Pretpostavićemo da administrator sistema želi da kreira par ključeva za root nalog. Pod root nalogom, on zadaje sledeću komandu:

```
# gpg --gen-key
gpg: /root/.gnupg: directory created
gpg: /root/.gnupg/options: new options file created
gpg: you have to start GnuPG again, so it can read the new
options file
```

Ukoliko je GnuPG pokrenut prvi put, u ličnom direktorijumu korisnika (u ovom slučaju, to je direktorijum /root) biće kreiran skriveni poddirektorijum .gnupg. za smeštaj ključeva i konfiguracionih datoteka. Administrator ponovo zadaje identičnu komandu:

```
# gpg --gen-key
gpg: /root/.gnupg/secring.gpg: keyring created
gpg: /root/.gnupg/pubring.gpg: keyring created
```

35 Detaljan postupak instalacije .tar.gz paketa opisan je u raznoj literaturi (na primer, *Securing and Optimizing Linux – The Ultimate Solution*, Gerharda Mourannija). Postupak instalacije je znatno jednostavniji ukoliko GnuPG pronađete u obliku RPM ili DEB paketa.

GnuPG dalje postavlja pitanja. Prvo pitanje se odnosi na tip ključeva. Zavisno od tipa ključeva koji odaberete, imaćete mogućnost da šifrujete i potpisujete (El-Gammal ključevi), odnosno samo da potpisujete (DSA ključevi).

```
Please select what kind of key you want:
(1) DSA and ElGamal (default)
(2) DSA (sign only)
(4) ElGamal (sign and encrypt)
Your selection? 1
DSA keypair will have 1024 bits.
About to generate a new ELG-E keypair.
```

Nakon toga, program zahteva od korisnika da odabere veličinu ključeva (podrazumevana vrednost je 1024 bita):

```
minimum keysize is 768 bits
default keysize is 1024 bits
highest suggested keysize is 2048 bits
What keysize do you want? (1024) 1024
Requested keysize is 1024 bits
```

Korisnik, zatim, određuje koliko će dugo ključevi biti validni. Podrazumeva se da ključevi nemaju "rok trajanja", odnosno period posle koga se više ne mogu koristiti:

```
Please specify how long the key should be valid.
0 = key does not expire
<n> = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct (y/n)? y
```

Potrebno je, zatim, da programu date neke osnovne informacije o vlasniku ključeva:

```
You need a User-ID to identify your key; the software
constructs the user id from Real Name, Comment and Email
Address in this form:
"Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"
```

```
Real name: Pera Kojot
Email address: pera@coyote.org
Comment: Suvi genije
You selected this USER-ID:
"Pera Kojot (Suvi genije) <pera@coyote.org>"
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
```

Nakon toga se programu daje lozinka kojom se štiti par ključeva. Prilikom generisanja ključeva, generator pseudoslučajnih brojeva oslanja se na razne akcije koje se dešavaju u pozadini sistema (procesorski šum, pomeranje miša, aktivnosti koje nastaju kao posledica keširanja diskova). Poželjno je radi postizanja slučajnije vrednosti da korisnik izvršava neku aktivnost (na primer, da kopira neku datoteku ili pomera miša).

```
You need a Passphrase to protect your secret key.
Enter passphrase: iamagenius
Repeat passphrase: iamagenius
```

Novi par ključeva se nakon toga generiše, potpisuje i smešta u lični direktorijum root korisnika. Root korisnik može izvesti javni ključ i distribuirati ga korisnicima sa kojima je potrebno ostvariti šifrovani komunikacioni kanal. GnuPG obezbeđuje, na primer, mehanizam za snimanje javnog ključa u ASCII zaštićenom formatu:

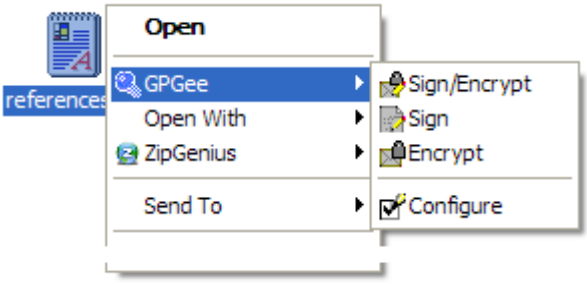
```
# gpg --export –ao Pera Kojot
```

Kao što je već rečeno, rad u komandnoj liniji može biti odbojan i ukoliko ste komformista, GnuPG nećete koristiti. Međutim, GnuPG morate prihvatiti kao kriptografski podsistem, odnosno paket koji obezbeđuje kriptografsku funkcionalnost neophodnu za rad drugih programskih paketa i namenskih GnuPG grafičkih interfejsa. Tome ćemo posvetiti više pažnje.

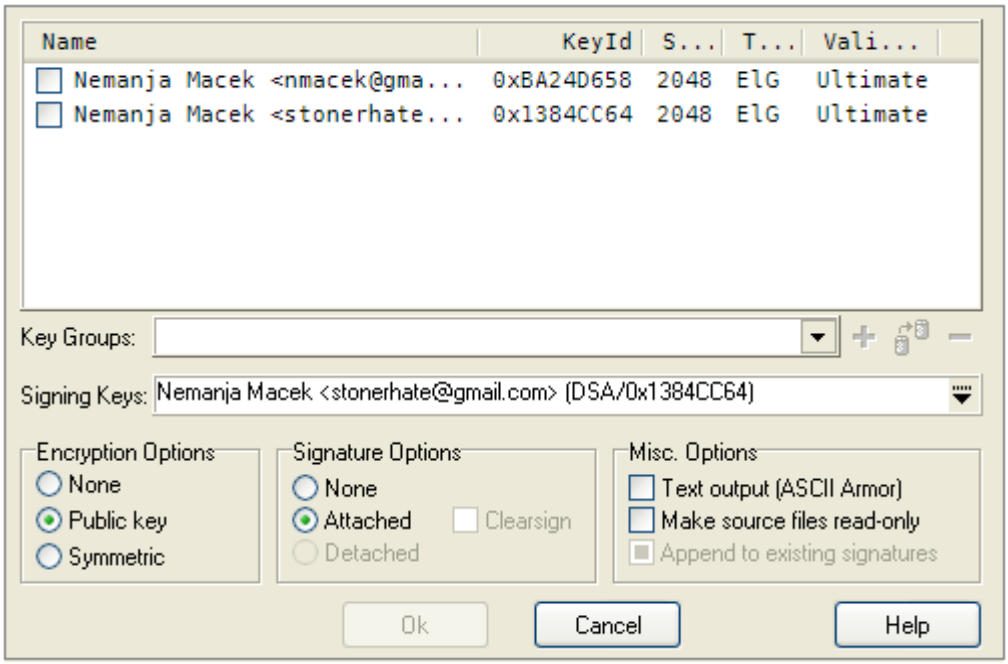
GnuPG Explorer Extension

GPGee je dodatak koji omogućava integraciju GnuPG paketa u Windows Explorer. Licenciran je pod GNU GPL licencom i može se preuzeti sa zvanične GnuPG web stranice. GPGee zahteva da u konfiguracionom dijalogu navedete putanju GnuPG izvršne i konfiguracione datoteke, kao i putanju i imena datoteka u kojima se nalaze privatni i javni ključevi (engl. *keyring*). GPGee ne obezbeđuje mehanizam za rad sa ključevima.

Nakon instaliranja paketa, u konteksnom meniju datoteke pojaviće se novi GPGee podmeni sa opcijama koje otvaraju odgovarajuće dijaloge (slika 7.5). Ukoliko želite da potpišete ili šifrujete neku datoteku (simetrično ili pomoću javnog ključa), GPGee će otvoriti grafički orijentisani dijalog (slika 7.6) u kome možete zadati sve potrebne parametre.



Slika 7.5 – *GPGee kontekstni podmeni*



Slika 7.6 *– GPGee dijalog za šifrovanje i potpisivanje*

Win Privacy Tray (WinPT)

WinPT[36] je, najjednostavnije rečeno, System Tray ikonica koja vam omogućava pristup dijalozima za šifrovanje i potpisivanje sadržaja tekućeg prozora, Clipboard-a i datoteka. WinPT ne obezbeđuje funkcionalnost šifrovanja putem kontekstnih menija Windows Explorer-a, ali, za razliku od GPGee, obezbeđuje mehanizme za rad sa ključevima.

WinPT zahteva da se na sistem prethodno instalira GnuPG; međutim, sa zvanične web stranice može se preuzeti i instalacioni paket koji, osim WinPT, uključuje i poslednju verziju GnuPG paketa i GPG Relay. Licenciran je pod GNU GPL licencom.

GPG Shell

GPG je još jedan GnuPG interfejs prema korisniku koji objedinjuje funkcionalnost WinPT i GPGee programa. To znači da ćete, nakon instalacije, dobiti jednu System Tray ikonicu (zeleni katanac) i GPG Shell podmeni kontekstnog menija datoteke u Windows Explorer-u. GPG Shell obezbeđuje solidan mehanizam za upravljanje ključevima (GPGkeys).

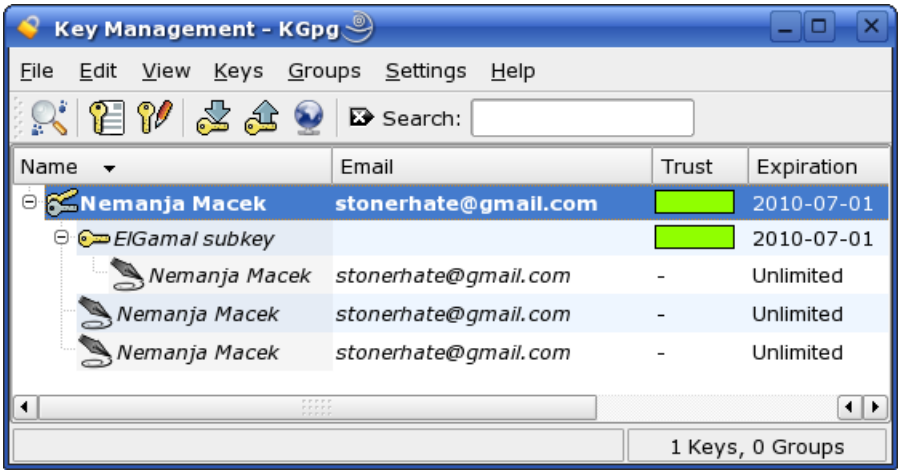
Za razliku od GPGee i WinPT, GPG Shell nije licenciran pod GNU GPL licencom, ali je besplatan i može se preuzeti sa web stranice proizvođača[37].

KGpg

GPGee, WinPT i GPG Shell su interfejsi namenjeni Windows okruženju. Ukoliko preferirate Linux (za što postoji mnogo razloga), na raspolaganju vam je KGpg interfejs, licenciran pod GNU GPL licencom.

KGpg je, u suštini, ikonica (verovatno već pogađate – žuti katanac) koja otvara meni sa opcijama za šifrovanje i potpisivanje tekućeg prozora i clipboard-a, odnosno alat za upravljanje sa kjučevima – KGpg Key Management.

36 WinPT možete preuzeti sa web stranice www.winpt.org.
37 http://www.jumaros.de/rsoft/index.html



Slika 7.7 *– KGpg Key Management*

Slika 7.8 – KGpg dijalog za šifrovanje datoteke

KGpg Key Management (prikazan na slici 7.7) obezbeđuje interfejs za generisanje para ključeva, objavljivanje javnih ključeva na server i preuzimanje tuđih javnih ključeva sa servera, kao i za potpisivanje ključeva i uspostavljanje nivoa poverenja.

KGpg se integriše u KDE grafičko okruženje Linux sistema, čiji je sastavni deo. U kontekstnom meniju datoteke možete, na primer, odabrati opciju za šifrovanje datoteke. KGpg će Vam, u tom slučaju, ponuditi dijalog (slika 7.8) u kome možete odabrati da li datoteku želite da šifrujete pomoću javnog ključa ili simetrično, pomoću lozinke. Takođe, ponuđena vam je i opcija da sigurno obrišete originalnu datoteku, tj. otvoreni tekst (*Shred source file*).

Izloženo predstavlja samo jedan mali deo mogućnosti KGpg paketa.

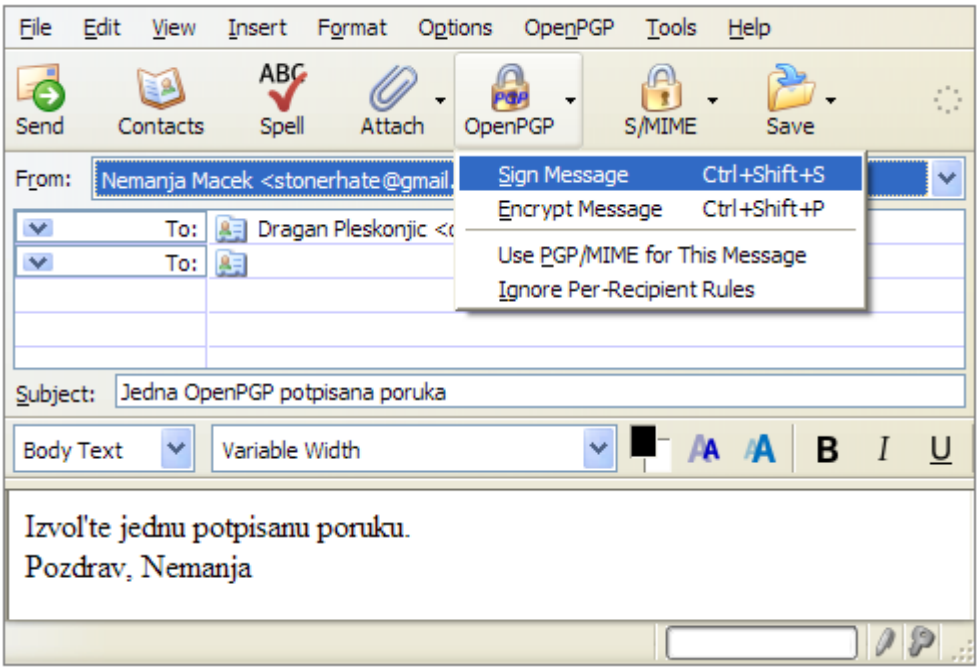
Enigmail

Enigmail je besplatni Mozilla Thunderbird dodatak koji omogućava potpisivanje i šifrovanje elektronske pošte[38]. Instalacijom ovog dodatka dobićete novu stavku u padajećem meniju klijenta za elektronsku poštu, tj. OpenPGP podmeni i nekoliko ikonica koje vam omogućuju da poštu potpišete i šifrujete.

U Enigmail je uključen pristupačan sistem za generisanje ključeva i rukovanje ključevima. Svoje javne ključeve možete izvesti na neki server javnih ključeva[39]; takođe, sa tih servera možete preuzeti i javne ključeve drugih osoba sa kojima želite da komunicirate. Pretpostavićemo da ste generisali svoje parove ključeva i sa servera preuzeli sve javne ključeve koji su vam potrebni. Dakle, postupak potpisivanja pošte obavlja se krajnje jednostavno: iz podmenija (ili ikonice) OpenPGP potrebno je da izaberete opciju Sign Message. Ukoliko je vaš privatni ključ zaštićen lozinkom, OpenPGP će zatražiti da unesete lozinku i ponuditi vam mogućnost da tu lozinku privremeno sačuva u svom programskom kešu (to znači da je narednih pet minuta nećete ponovo unositi).

38 Enigmail možete preuzeti sa web stranice http://enigmail.mozdev.org.

39 Navodimo neke servere javnih ključeva: random.sks.keyserver.penguin.de, pgp.dtype.org, keyserver.kjsl.com, ldap://certserver.pgp.com



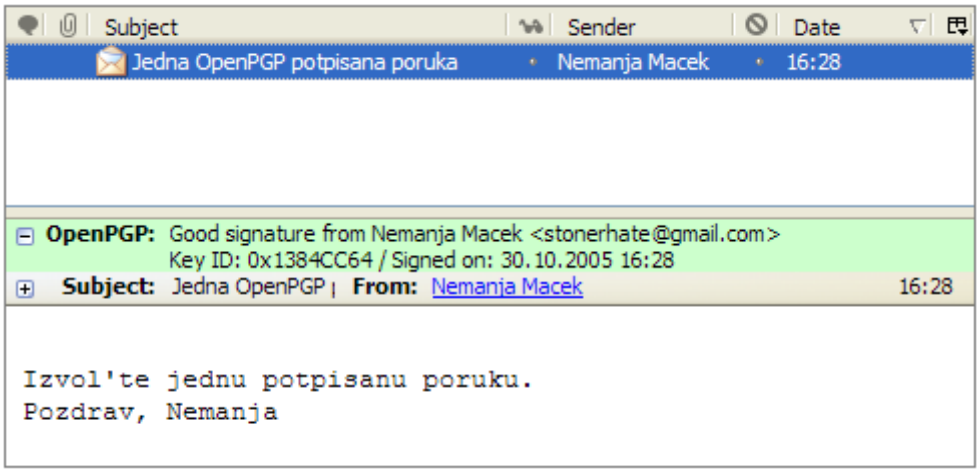
Slika 7.7 *– Mozilla Thunderbird i Enigmail: potpisivanje pošte*

Pre slanja, Enigmail će pozvati GnuPG da potpiše poruku. Pretpostavite da tekst poruke odgovara slici 7.7. Primalac čiji klijent za elektronsku poštu nema podršku za OpenPGP dobiće sledeći tekst:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Izvol'te jednu potpisanu poruku.
Pozdrav, Nemanja
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.2 (MingW32)
Comment: Using GnuPG with Thunderbird
 - http://enigmail.mozdev.org

iD8DBQFDZOFTZL/TTBOEzGQRAmlXAJ9tax24aTUhHeCseEFZJhV
QRiCHLgCeNp0H0IIoSKQyvK0FgHKAWaoQd0U=
=kW2y
-----END PGP SIGNATURE-----
```

Međutim, ukoliko e-mail klijent primaoca poruke ima podršku za

OpenPGP, primalac ne vidi ovakvu strukturu poruke. Pretpostavićemo da primalac takođe koristi Enigmail i da ima Vaš javni ključ. U tom slučaju on dobija sledeću poruku:

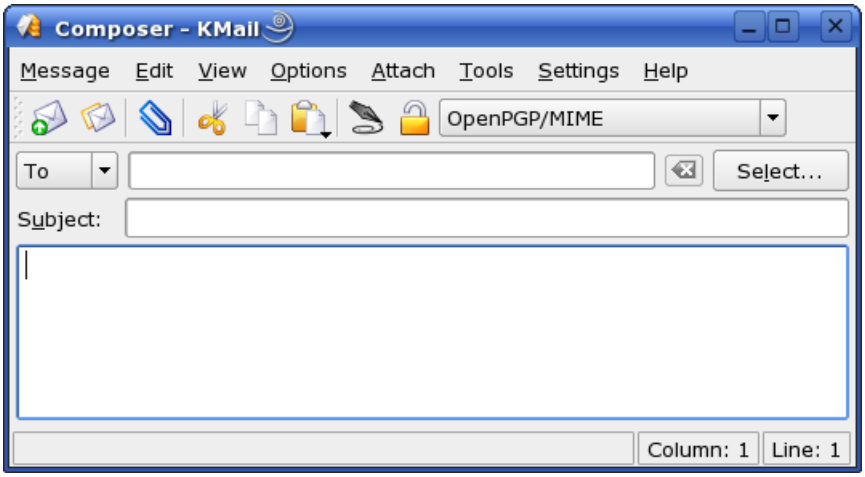


Slika 7.8 *– Šta vidi primalac ...*

Obratite pažnju na sledeće: ukoliko nekome želite da pošaljete šifrovanu poruku, morate imati njegov javni ključ. Nemojte poruku šifrovati svojim javnim ključem ukoliko je želite nekome poslati (naravno, ukoliko želite da je primalac pročita).

KMail

Kao i KGpg, KMail je sastavni deo KDE grafičkog okruženja na Linux sistemima. Ukoliko na nekoj Linux distribuciji instalirate KDE okruženje, KMail će najverovatnije biti podrazumevano instaliran. KMail je klijent za elektronsku poštu u koji je integrisan kriptografski interfejs koji kod Mozilla Thunderbird e-mail klijenta obezbeđuje Enigmail. Korisnik može elektronsku poštu da potpiše ili da šifruje – potpisivanje i šifrovanje se odvija po OpenPGP standardu. U KMail nisu integrisani nikakvi interfejsi za upravljanje ključevima; ovaj paket se delimično oslanja na funkcionalnost koju obezbeđuje KGpg Key Management (što je i logično, s obzirom na to da su oba paketa sastavni deo KDE okruženja). Na slici 7.9 prikazan je KMail Composer.



Slika 7.9 *– KMail Composer*

7.4 TrueCrypt

TrueCrypt[40] je besplatan program otvorenog koda koji korisnicima nudi funkcionalnost čuvanja podataka na šifrovanim sistemima datoteka. Korisnicima su na raspolaganju sledeći simetrični algoritmi: Blowfish, Twofish, AES, CAST5, Serpent i Triple DES, kao i konkatenacije funkcija šifrovanja (npr. AES-Twofish, AES-Blowfish-Serpent i njima slične). TrueCrypt vam nudi mogućnost da ispitate koji algoritam na Vašem sistemu postiže najbolje performanse, tj. koji algoritam najbrže radi. Što se tiče sigurnosti algoritma, poželjno je da o njima nešto znate i da jednostavno odlučite da li više verujete organizacijama kao što su NSA i NIST ili Bruce Schneieru. Šifrovani sistemi datoteka mogu se generisati kao volumeni smešteni u nekoj datoteci na postojećem sistemu datoteka ili kao zasebne particije na disku. U oba slučaja, kripto sistem datoteka se aktivira preko logičkih diskova. TrueCrypt nudi i funkcionalnost kreiranja skrivenih volumena, o kojima će kasnije biti više reči. Podaci se štite lozinkom, ključ-datotekom (engl. *keyfile*) ili kombinacijom lozinke sa ključ-datotekom.

Za sada nije dokazano da je u TrueCrypt ugrađena klopka; takođe, program ne priznaje nikakve autoritete koji mogu nasilno dešifrovati vaše

40 TrueCrypt (trenutno u verziji 4.0) možete preuzeti sa adrese www.truecrypt.org.

podatke niti ostavlja mogućnost administratoru sistema da to uradi – ključ-datoteke se ne distribuiraju u profil korisnika niti na bilo koje drugo mesto, a program je prilično oprezan po pitanju čuvanja lozinki i ključ-datoteka u operativnoj memoriji. Korisnik može definisati kombinaciju tastera koja će izazvati deaktiviranje svih šifrovanih sistema datoteka (po potrebi i nasilno) i brisanje memorijskog keša za skladištenje ključeva i lozinki.

Ispitivanje performansi algoritama

Pre nego što generišete neki kripto sistem datoteka, poželjno je da ispitate performanse algoritama. Odaberite Tools → Benchmark iz glavnog menija programa i odaberite veličinu bafera (na primer, 50 ili 100 MB). Test se odvija u operativnoj memoriji računara, i u zavisnosti od veličine bafera, traje od nekoliko sekundi do nekoliko minuta. Na test računaru, algoritmi Blowfish i Twofish pokazali su se kao najbrži.

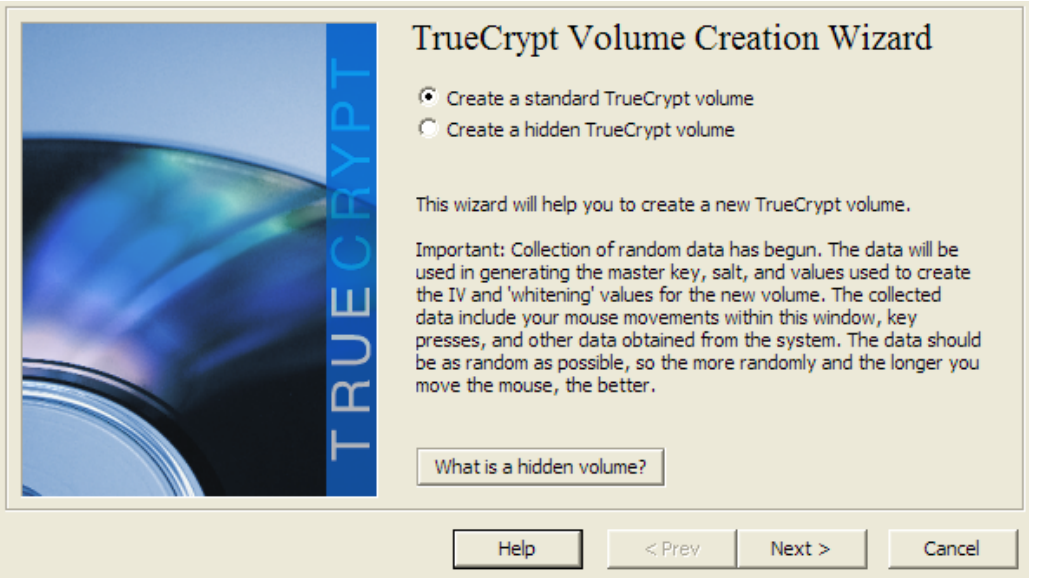
Ovaj test ne sme vam biti jedini kriterijum za izbor algoritma (iako na osnovu njega definitivno možete zaključiti da nije preporučljivo izabrati Triple DES). Međutim, ukoliko se dvoumite između nekoliko algoritma kojima verujete, odaberite brži. U daljim primerima koristićemo Blowfish.

Kreiranje standardnog volumena

Ilustrovaćemo primerom postupak kreiranja standardnog volumena u datoteci koji će biti zaštićen lozinkom. Odaberite Volumes → Create New Volume iz glavnog menija programa. Otvoriće se čarobnjak (engl. *wizard*), prikazan na slici 7.11, koji će Vas provesti kroz jednostavan postupak kreiranja volumena.

Štiklirajte “Create a standard TrueCrypt volume”, a zatim na sledećem dijalogu unesite ime datoteke u koju će program smestiti šifrovani volumen (npr. h:\secret.tc). Datoteke ne moraju imati ekstenziju *.tc*, osim ako želite da se volumen asocira sa programom na nivou Windows Explorera. Na sledećem dijalogu odaberite tip algoritma (npr. Blowfish), a zatim unesite veličinu volumena (npr. 10 MB). Sledeći dijalog zahteva od Vas da unesete lozinku dva puta. Ukoliko želite viši nivo sigurnosti, preporučuje se volumen dodatno zaštitite ključ-datotekom. Nakon toga, odaberite sistem datoteka

koji želite (FAT ili NTFS) i veličinu klastera. TrueCrypt će na osnovu podataka koje ste uneli kreirati prazan šifrovani volumen koji kasnije možete aktivirati preko logičkih diskova.

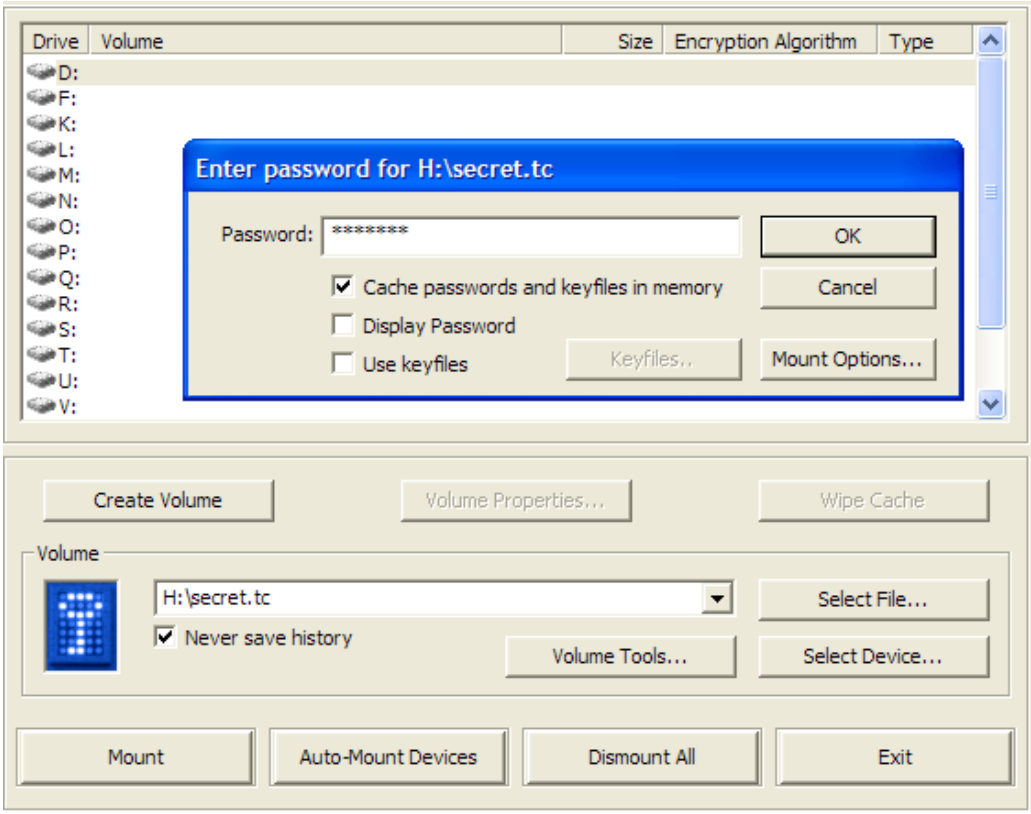


Slika 7.10 *– Dijalog za kreiranje volumena*

Štiklirajte "Create a standard TrueCrypt volume", a zatim na sledećem dijalogu unesite ime datoteke u koju će program smestiti šifrovani volumen (npr. h:\secret.tc). Datoteke ne moraju imati ekstenziju *.tc*, osim ako želite da se volumen asocira sa programom na nivou Windows Explorera. Na sledećem dijalogu odaberite tip algoritma (npr. Blowfish), a zatim unesite veličinu volumena (npr. 10 MB). Sledeći dijalog zahteva od Vas da unesete lozinku dva puta. Ukoliko umesto lozinke želite da zaštitite volumen ključ-datotekom, odaberite neku datoteku sa diska (i sklonite je na sigurno mesto). Nakon toga, odaberite sistem datoteka koji želite (FAT ili NTFS) i veličinu klastera. TrueCrypt će na osnovu podataka koje ste uneli kreirati prazan šifrovani volumen koji kasnije možete aktivirati preko logičkih diskova.

Aktiviranje i deaktiviranje volumena

Pokretanjem datoteke sa ekstenzijom .tc u Windows Exploreru otvara se TrueCrypt dijalog za aktiviranje volumena. Šifrovani sistem datoteka aktivira se preko logičkog slova koje Vi birate, kao što je prikazano na slici 7.12. Od Vas se zahteva da unesete lozinku kojom je sistem datoteka zaštićen. Alternativno, volumen možete aktivirati iz TrueCrypt glavnog prozora pozivanjem opcije Volumes → Mount Volume i unošenjem imena datoteka u kojoj se nalazi volumen.



Slika 7.11 *– Dijalog za aktiviranje volumena*

Nakon toga, volumenu pristupate kao lokalnom disku. Deaktiviranje se može obaviti iz TrueCrypt glavnog prozora pozivanjem opcije Volumes → Dismount Volume ili Volumes → Dismount All Mounted Volumes (ukoliko

želite da deaktivirate sve šifrovane sisteme datoteka). Deaktiviranje se može obaviti i odgovarajućom kombinacijom tastera, ukoliko je aktivna System Tray ikonica TrueCrypt koja omogućava izvršenje TrueCrypt procesa u pozadini. Ova opcija može biti veoma korisna, pa se zato korisnicima preporučuje da ne isključuju ovaj proces. Kombinacije tastera za deaktiviranje i nasilno deaktiviranje definiše sam korisnik.

Ključ-datoteke

Ključ-datoteka (engl. *keyfile*) je datoteka čiji se sadržaj pomoću specijalnog algoritma "kombinuje" sa lozinkom[41] i na taj način obezbeđuje viši nivo zaštite volumena. Korisnik ne može aktivirati šifrovani volumen zaštićen ključ-datotekom ukoliko tu datoteku ne poseduje. Program ne insistira na korišćenju ovakve zaštite, ali u zvaničnom uputstvu upozorava korisnike na prednosti koje ona donosi[42]:

- obezbeđuje zaštitu protiv programa za krađu lozinki (engl. *keyloggers*) koji skriveno prate šta ste Vi otkucali na tastaturi i to beleže u neku datoteku ili šalju preko Interneta napadaču. Napadač koji otkrije Vašu lozinku, neće moći da aktivira volumen jer nema ključ-datoteku;

- uvećava kvalitet lozinke, tj. dužinu i složenost, a samim tim i otpornost na napade grubom slilom.

Volumen se može zaštititi jednom datotekom ili većim brojem datoteka. Ukoliko se volumen štiti većim brojem ključ-datoteka, one se mogu redistribuirati većem broju korisnika – u tom slučaju se volumen može aktivirati samo ako svi korisnici istovremeno prilože svoje ključ-datoteke. Prednost ovog načina zaštite je smanjenje uticaja ljudskog faktora na redukciju sigurnosti (npr. eliminiše se mogućnost podmićivanja jedne osobe

41 Više informacija o tome kako se sadržaj datoteke "kombinuje" sa lozinkom možete naći u zvaničnom TrueCrypt uputstvu: *TrueCrypt User's Guide*, poglavlje *Technical Details*, *Keyfiles*.

42 Program Vas, takođe, neće naterati ni da koristite TrueCrypt rezidentni deo (ali će Vas upozoriti na funkcionalnost koju gubite – npr, nećete moći da deaktivirate volumene pomoću neke kombinacije tastera). Program će vam čak dozvoliti da volumen zaštitite slabom lozinkom, ali će Vas upozoriti da je ona osetljiva na napad grubom-silom. Ukoliko se odlučite da ovakva upozorenja ignorišete (što se vrlo često dešava), sami ste krivi za eventualne posledice – ne krivite program i njegove autore.

radi ostvarivanja pristupa).

Bilo koji tip datoteke može se koristiti kao ključ. U zvaničnom uputstvu se preporučuje korišćenje komprimovanih datoteka ili datoteka sa čitljivim, tj. razumljivim sadržajem (.mp3, .jpg, .avi, .zip i njima slične). TrueCrypt ne modifikuje sadržaje ovakvih datoteka, što znači da korisnik u nekom direktorijumu sa oko 500 mp3 datoteka može čuvati i nekoliko ključeva. Analizom ovih datoteka ne može se utvrditi da li se one koriste kao ključevi ili ne; dodatno, napadač koji po inerciji pretražuje skrivena mesta (npr. profil, USB fleš disk i slično) najverovatnije neće pretpostaviti da je ključ mp3 datoteka.

Navodimo i dve važne napomene vezane za ključ datoteke:

- Ukoliko se volumen štiti ključ-datotekom, program ne insistira na korišćenju lozinke. Međutim, korišćenje lozinke sa ključ-datotekom obezbeđuje vrlo visok nivo zaštite volumena.

- Poželjno je da ukupna veličina svih datoteka kojima se štiti volumen bude veća od 20 bajtova kako bi se sprečila mogućnost napada grubom silom. Ovo je naročito značajno ukoliko se volumen štiti samo ključ-datotekom bez lozinke.

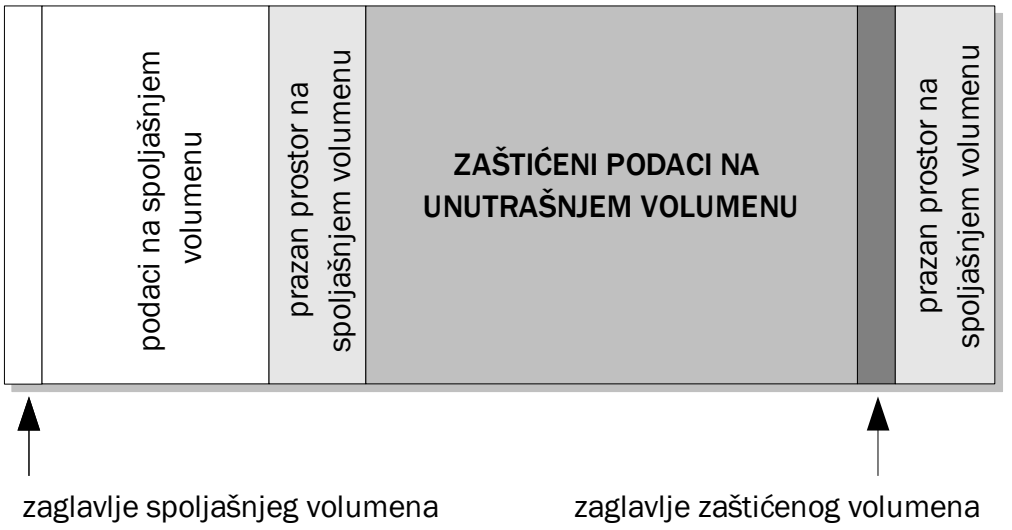
- Pazite da ne izgubite ključ-datoteku. Ukoliko je izgubite, podaci na šifrovanom volumenu koji je njome zaštićen biće nedostupni.

Skriveni volumeni

Kao što je opisano u poglavlju 2, postoji nekoliko tipova kriptoanalitičkih napada. Jedan od efikasnijih napada je napad potkupljivanjem, ucenom, krađom i sličnim aktivnostima. Postoje situacije u kojima ne možete odbiti da napadaču predate ključ ili lozinku (npr. ukoliko napadač primenjuje fizičku silu). Korišćenje skrivenih volumena je jedan od načina kojim se možete odbraniti od ovakvih napada.

U okviru praznog prostora spoljašnjeg volumena, koji se prilikom kreiranja popunjava pseudoslučajnim podacima, TrueCrypt generiše skriveni, unutrašnji volumen (engl. *hidden volume*). Deo praznog prostora u

kom je kreiran skriveni volumen ne razlikuje se ni po čemu od ostatka praznog prostora. Čak i ako aktivirate spoljašnji volumen, nemoguće je dokazati postojanje unutrašnjeg volumena (slika 7.12).



Slika 7.12 *– Skriveni TrueCrypt volumen*

Preporučuje se da na spoljašnji volumen iskopirate neke podatke koji će napadaču stvoriti iluziju pravog sadržaja čije je otkrivanje cilj njegovog napada. U većini slučajeva dovoljno je da u spoljašnji volumen iskopirate datoteku sa lažnim lozinkama i nekoliko bezvrednih dokumenata ili slika koje, navodno, želite da sakrijete, ali čije otkrivanje ne sme imati nikakve posledice. Ovi dokumenti će biti dostupni svakome ko Vas primora da mu predate lozinku i/ili ključ-datoteku.

Datoteke koje su poverljive čuvaju se na skrivenom volumenu, čijeg postojanja napadač nije svestan. Lozinka skrivenog volumena mora biti različita od lozinke spoljašnjeg volumena koji služi kao maska za prevaru napadača. Skriveni volumen se aktivira slično spoljašnjem – pri tome, program aktivira volumen za koji navedete lozinku. TrueCrypt će najpre pokušati da dešifruje zaglavlje spoljašnjeg volumena. Ukoliko u tome uspe, tj. ukoliko ste Vi naveli lozinku kojom je zaštićen spoljašnji volumen, biće vam dostupni podaci koji nisu osetljivi. Ukoliko je dešifrovanje zaglavlja spoljašnjeg volumena neuspešno, tj. ako je uneta pogrešna lozinka,

TrueCrypt će pokušati da dešifruje zaglavlje skrivenog volumena (obično se nalazi u trećem bloku s kraja spoljašnjeg volumena). Ukoliko u tome uspe, tj. ukoliko ste Vi naveli lozinku kojom je zaštićen skriveni volumen, biće vam dostupni pravi podaci.

7.5 EFS

EFS (Encryption File System) obezbeđuje mehanizme zaštite podataka na lokalnim i udaljenim NTFS volumenima šifrovanjem pomoću javnih ključeva. EFS je sastavni deo Microsoft Windows operativnog sistema i izvršava se u pozadini kao integrisani sistem usluga transparentan za korisnika. Prilikom izvršavanja, nalazi se u jezgru operativnog sistema i za smeštanje ključeva za šifrovanje koristi memoriju koja se ne straniči, čime osigurava da ključevi nikad ne dospeju u datoteku za straničenje (engl. *page file*).

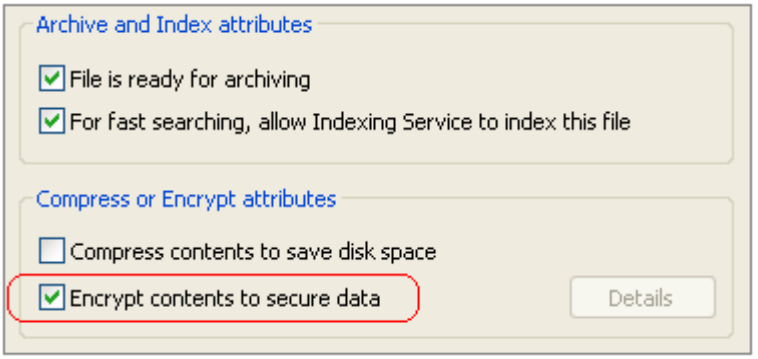
Svaka datoteka ili folder se šifruju relativno brzim simetričnim algoritmom DESX. Ključ za šifrovanje datoteka (*File Encryption Key* – FEK), dužine 128 bita, nezavistan je od javnog ili tajnog ključa korisnika i dobija se pomoću generatora slučajnih brojeva. Proces šifrovanja i dešifrovanja se odvija brzo i transparentno tokom normalnih ulazno/izlaznih operacija i ne degradira performanse sistema. FEK se šifruje javnim ključem korisnika pomoću RSA algoritma i u šifrovanom obliku se smešta uz datoteku u posebnom EFS atributu DDF (*Data Decryption Field*). Privatni i javni ključ korisnika se čuvaju u korisničkom profilu[43]. Preporučuje se da korisnik izveze privatni ključ i čuva ga na nekom sigurnom mestu.

Ključeve za šifrovanje generiše EFS i sertifikuje ih kod nadležnog sertifikacionog centra (CA) koji izdaje EFS sertifikate. Ako nadležni CA ne postoji, EFS će sam potpisati ključeve. Izdavanjem digitalnog sertifikata uspostavlja se vremenski ograničena relacija između javnih ključeva i identiteta korisnika.

43 EFS koristite samo ako verujete administratoru sistema. Ukoliko ne verujete administratoru sistema, nemojte da koristite EFS. Efektivne NTFS dozvole administratora su podešene tako da on može da čita podatke iz svih profila, uključujući i Vaš. Ukoliko to ne može da uradi, preuzeće vlasništvo i prilagoditi NTFS dozvole, što znači da veoma lako može doći do vašeg para ključeva. Obratite pažnju na činjenicu da Vaš privatni ključ nije zaštićen lozinkom! Dodatno, administrator je najčešće i agent za oporavak. Smatrajte da ste upozoreni.

Kako se šifruje?

Šifrovanje se obavlja iz Windows Explorer-a (u kartici General dijaloga Properties datoteke ili direktorijuma pritisnuti dugme Advanced i štiklirati Encrypt contents to secure data – slika 7.13) ili iz komandne linije (komandom cipher).



Slika 7.13 *– EFS: šifrovanje iz Windows Explorer-a*

Moguće je šifrovati foldere (direktorijume) ilii pojedinačne datoteke. Ako je folder šifrovan, sve datoteke koje se kreiraju u njemu automatski će biti šifrovane, a svaka datoteka će imati sopstveni FEK ključ. Datoteka se ne mora eksplicitno dešifrovati pre upotrebe – za dešifrovanje i ponovno šifrovanje nakon upotrebe se brine sam EFS.

Oporavak podataka – klopka ili "za nedaj Bože" ?

EFS takođe omogućava kvalifikovanim agentima za oporavak (engl. *recovery agent*) da podatke dešifruju ukoliko za to postoji potreba[44]. Ukoliko postoji bar jedan kvalifikovani agent za oporavak, za svaku datoteku ili folder se generiše i poseban ključ koji agent za oporavak upotrebljava ukoliko je korisnik izgubio svoj privatni ključ ili je napustio radno mesto. FEK ključ se RSA algoritmom šifruje i pomoću javnog dela ključa za oporavak i

44 Ovo se može smatrati kao potencijalna klopka (*trapdoor*). Ukoliko ne verujete agentima za oporavak, koristite True Crypt. True Crypt ne podržava (za sada) nikakve agente za oporavak niti bilo kakve druge agente.

tako šifrovan se smešta u poseban EFS atribut DRF (*Data Recovery Field*).

Ukoliko se EFS primenjuje u manjim kancelarijama, u kojima je mreža organizovana u modelu radne grupe, centralna baza sa ključevima ne postoji – za svaki računar korisnik ima drugi ključ koji se čuva u lokalnom profilu tog računara. Lokalni administrator, tj. administrator samostalnog servera ili radne stanice u tom slučaju je i agent za oporavak. Ako administrator obriše sve sertifikate sa lokalnog računara, dešifrovanje podataka će biti nemoguće. To znači da bi korisnici trebalo da čuvaju svoje tajne ključeve na nekom drugom medijumu, a ne na lokalnom računaru.

Ukoliko je mreža organizovana u Windows 2000/2003 domena, definiše se polisa oporavka podataka (*Data Recovery Policy*). Ova polisa je, kao i sve druge polise, deo objekta grupne polise (*Group Policy Object*). Odnosi se na javne ključeve i brine se o tome da li u sistemu postoje agenti za oporavak. Polisa je vezana za računar, a ne za korisnika i može biti primenjena na nivou domena, organizacione jedinice ili lokalno.

7.6 Pitanja i zadaci

7.1 Objasnite prednosti i mane šifrovanja link-po-link u odnosu na šifrovanje s kraja na kraj.

7.2 Koja metoda šifrovanja podataka (na nivou datoteke ili drajvera) obezbeđuje transparentnost za korisnike? Koja je metoda pogodnija i sigurnija za upotrebu ukoliko se često generiše rezervna kopija podataka na trakama ili DVD medijumima.

7.3 Instalirajte GnuPG paket na Vašem računaru i generišite dva para ključeva (pretpostavite da će paket koristiti dve osobe). Probajte da šifrujete i potpišete datoteku iz komandne linije; probajte da šifrovanu datoteku dešifrujete i proverite potpis.

7.4 Isprobajte GPGee, WinPT i GPG Shell programske pakete.

7.5 Isprobajte funkcionalnost KGpg programskog paketa na Linux sistemima. Isprobajte procedure za šifrovanje i potpisivanje

elektronske pošte u Kmail paketu.

7.6 Instalirajte Mozilla Thunderbird klijent za elektronsku poštu sa Enigmail dodatkom. Generišite dva para ključeva koji odgovaraju dvema različitim e-mail adresama (iskoristite neki besplatni web-mail server sa POP3 i SMTP servisima). Probajte da se "dopisujete" šifrovanim i potpisanim porukama.

7.7 Koristeći TrueCrypt paket napravite:

- šifrovani volumen u datoteci zaštićen lozinkom i
- skriveni volumen u okvirima šifrovanog volumena zaštićen kombinacijom ključ-datoteke i lozinke.

Koristite algoritam koji na vašem sistemu daje najbolje performanse. Pokušajte da aktivirate ove volumene. Prilikom aktiviranja spoljašnjeg volumena onemogućite mogućnost oštećenja, tj. brisanja podataka na skrivenom volumenu usled popunjenja praznog prostora na spoljašnjem volumenu.

7.8 Prijavite se na Windows sistem kao korisnik sa limitiranim ovlašćenjima (neko ko ne pripada grupi lokalnih administratora). Na disku vašeg računara napravite direktorijum C:\Secret i u njega iskopirajte nekoliko datoteka. Šifrujte ceo direktorijum i izvezite svoj privatni ključ, a zatim ga izbrišite iz svog profila. Probajte sada da pročitate neku datoteku sa šifrovanog direktorijuma. Prijavite se na sistem kao lokalni administrator i probajte da pročitate neku datoteku.

8

Kontrola pristupa i zaštita operativnog sistema Windows 2000/XP/2003

8.1 Opšte o Windows 2000/XP/2003 sigurnosti

Godine 1988. Microsoft je odlučio da razvije prenosivi operativni sistem koji podržava i OS/2 i POSIX API-je. Izvorno, Windows NT (*new technology*) je trebao da koristi OS/2 API kao svoje prirodno okruženje, ali je tokom razvoja izmenjen tako da koristi Win32 API, što je odražavalo popularnost Windowsa 3.0. Prve verzije NT operativnog sistema, Windows NT 3.1 i Windows NT 3.1 Advanced Server, prihvatile su korisnički interfejs tada aktuelne verzije 3.1 16-bitnog Windows-a. Posle toga su objavljene verzije NT Workstation 4.0 i NT Server 4.0, zasnovane na korisničkom interfejsu Windows 95. Prema statistici, 1996. godine je prodato više NT server licenci nego Unix licenci. Takođe, NT 4.0 je poslednja verzija koja je prenešena i na druge arhitekture, osim Intelove. Kao posledica stanja na tržištu, u verzijama koje slede (Windows 2000/XP/2003) ukinuta je podrška za sve procesore koji nisu Intel kompatibilni.

Windows NT je slojeviti sistem koji se sastoji od modula. Sloj apstrakcije hardvera (HAL), jezgro i izvršni deo, tj. egzekutiva (*executive*) rade u zaštićenom režimu rada (engl. *protected mode*), a kolekcija podsistema u korisničkom režimu rada (engl. *user mode*). Podsistemi okruženja (MS-DOS podsistem, 16-bitni i 32-bitni Windows podsistem i POSIX podsistem) oponašaju različite operativne sisteme, tj. obezbeđuju platformu za izvršavanje aplikacija namenjenih tim operativnim sistemima. Zaštitni podsistemi obezbeđuju sigurnosne funkcije.

Jedan od ključnih ciljeva razvoja Windows NT operativnog sistema je sigurnost – zaštitu lokalnih i mrežnih resursa od neovlašćenog pristupa Windows NT obezbeđuje pomoću *Security Reference Monitora*, žetona za pristup (engl. *Access Token*), identifikacije korisnika i grupa i listi za kontrolu pristupa. *Security Reference Monitor* (SRM) je deo egzekutive koji obavlja kontrolu pristupa objektima, odnosno dozvoljava pristup objektima samo ovlašćenim korisnicima i obezbeđuje funkcionalnost praćenja pristupa objektu (engl. *auditing*). Upravljač objekta (*Object Manager*) koristi usluge koje pruža SRM – pre izvršenja neke akcije, upravljač objekta konsultuje SRM i tako određuje da li proces ima pravo da izvrši akciju koju želi nad objektom.

SRM implementira C2 nivo sigurnosti:

- vlasnik objekta određuje ko može pristupiti objektu,
- identifikacija korisnika obavlja se pomoću jedinstvenog korisničkog imena i lozinke,
- obezbeđuje mogućnost praćenja uspešnih i neuspešnih pokušaja pristupa objektima,
- štiti sadržaj memorije i datoteka od neovlašćenog pristupa.

Sigurnosni podsistem čine sledeće komponente:

- *Local Security Authority*. LSA upravlja sigurnosnom polisom računara. Generiše pristupne žetone, odnosno informacije koje se dodaju procesima i koje određuju čemu sve korisnik može da pristupi. LSA upravlja polisom praćenja pristupa resursima. Komunicira sa *Security Reference Monitorom* i vodi evidenciju o porukama vezanim za praćenje objekata koje šalje jezgro.

- *Logon* proces. Prihvata zahteve za prijavljivanje na sistem; komunicira sa LSA kako bi odredio da li korisnik, koji želi da se prijavi na sistem, ima na to pravo ili ne.

- *Security Account Manager* (SAM). Upravlja bazom podataka u kojoj su opisani korisnici i grupe i obezbeđuje pristup toj bazi.

Jednostavno rečeno: Windows 2000/XP/2003 zahteva od vas da se autentifikujete. Lozinke korisnika se na lokalnim diskovima i na domen kontrolerima ne čuvaju u obliku otvorenog teksta, nego u obliku heš vrednosti korisničkih lozinki (vidite 5.3). Nakon autentifikacije Vi možete pristupiti samo onim objektima u odnosu na koje koje imate potrebna ovlašćenja. Ovlašćenja određuju šta Vi sa objektom možete, a šta ne možete uraditi. Svi neuspešni i uspešni pokušaji pristupanja, odnosno čitanja ili izmene objekata se po potrebi mogu pratiti.

Koliko je sve to zaista sigurno?

Verovatno ste čuli da postoji nekoliko alata pomoću kojih možete izmeniti

lozinku administrativnog naloga na Windows NT/2000/XP/2003 sistemima. Ilustrovaćemo na primeru kako možete da odete korak dalje, tj. otkrijete lozinku administratora ukoliko je ona slaba. Ovaj napad možete isprobati na bilo kom računaru kome možete fizički pristupiti i na kom se sistem može podići sa CD ili DVD uređaja[45].

Za napad vam je potrebna "živa" (engl. *live*) distribucija Linux sistema koja sadrži sve potrebne "open-source" programe koji će vam pomoći u ovom postupku: Samdump2, Bkhive i John the Ripper. Primer takve distribucije je Auditor CD. Napad obuhvata sledeće korake:

- podizanje operativnog sistema sa CD medijuma,
- montiranje particije sa Windows sistemom na neki direktorijum,
- izvlačenje sistemskog ključa[46] iz Registry baze pomoću programa Bkhive (ili Bkreg za starije sisteme, zaključno sa Windows NT 4),
- izvlačenje heša lozinke u datoteku u PWDump formatu[47] pomoću sistemskog ključa i programa Samdump2,
- odabiranje rečnika, tj. liste reči (engl. *wordlist*) koji želite da koristite pri napadu; rečnike možete naći na Internetu ili u direktorijumu /opt/auditor/full/share/wordlists/ Auditor distribucije,
- napad na heševe pomoću programa John the Ripper. John redom generiše heševe svih reči iz rečnika i obaveštava napadača kada pronađe heš jednak hešu lozinke koju napadač želi da otkrije.

Ukoliko prethodno opisani napad vrati lozinku koja se nalazi u engleskom rečniku, može se zaključiti da je slaba tačka sistema ljudski faktor (tj. administrator koji nije svestan problema sigurnosti). Do ovakvih rezultata se dolazi vrlo brzo.

45 Jednostavnije rešenje: ukoliko se operativni sistem ne može podići sa CD ili DVD uređaja, izvadite disk i montirajte ga u računar za koji znate BIOS lozinku i nastavite dalje sa "radom". Komplikovanije rešenje: pročitajte dodatak C ove knjige.

46 Sistemskim ključem (*SysKey*) uveden je dodatni nivo kriptografske zaštite nad heševima u SAM bazi. Uveden je u Windows NT 4 sistemu sa Service Pack 3 i od tada se podrazumevano koristi.

47 Preporučuje se da ekstrakt heša i rečnik smestite na RAM disk koji je mnogo puta brži od bilo kog magnetnog medijuma, pa je i napad na heš brži. Dodatno, ukoliko koristite samo RAM disk, na hard disku sistema neće ostati nikakvi tragovi.

Ukoliko napadi rečnikom ne vrate željeni rezultat, napadač može isprobati i napad grubom silom.[48] Kao što je već rečeno, napadi grubom silom su, zbog nedovoljne procesorske snage, neuspešni ukoliko je lozinka dovoljno dugačka. Napad rečnikom takođe neće biti uspešan ukoliko lozinka nije jednostavna, što znači da gore opisani postupak neće pomoći napadaču da otkrije lozinku.

Propusti ovakvog tipa nisu posledica nepostojanja ili loše implementirane zaštite samog operativnog sistema, već ljudskog faktora. Ovo ne važi samo za Windows – do datoteka /etc/passwd i /etc/shadow na Linux sistemima se takođe lako dolazi, pa će John odraditi posao za napadača ukoliko je lozinka kratka ili se nalazi u nekom rečniku. Svaki ozbiljniji operativni sistem administratorima nudi mehanizme zaštite, ali će sistem ostati nesiguran ukoliko administrator te mehanizme ne ume da iskoristi. Jednostavno rečeno, nemojte kritikovati operativni sistem ukoliko ne umete da se zaštitite! Kritikujte ga ukoliko ne obezbeđuje mehanizme zaštite.

U ovom poglavlju ćemo ukratko opisati neke mehanizame zaštite implementirane na Windows 2000/XP/2003 sistemima. Najpre navodimo spisak zaštitnih mera koje je neophodno da primenite ukoliko želite da Vaš Windows sistem bude sigurniji:

- instalirajte sve sigurnosne zakrpe koje proizvođač operativnog sistema objavi,
- regulišite prava korisnika tako što ćete korisnike učlaniti u grupe kojima ćete dodeliti ovlašćenja pomoću grupnih polisa,
- postavite NTFS dozvole kako biste ograničili lokalni i udaljeni pristup datotekama i direktorijumima,
- postavite dozvole za deljene foldere kako bi ste ograničili pristup datotekama preko mreže,
- šifrujte datoteke od velikog značaja ili ih smestite u šifrovani sistem datoteka (jako kriptografija je najbolja zaštitna mera); ukoliko nemate poverenja u EFS, koristite drugi softver,
- omogućite praćenje značajnih događaja (kao što je pristup osetljivim

48 Napadi grubom silom limtirani su na lozinke dužine osam karaktera. Ukoliko želite da povećate dužinu lozinke koje će John uzeti u obzir, morate pre toga da izmenite i prevedete izvorni kod programa. Obratite pažnju na to da je otkrivanje lozinke čija je dužina veća od 10 karaktera vremenski izuzetno zahtevan posao!

datotekama),

- aktivirajte mrežnu barijeru (više o mrežnim barijerama možete naći u poglavlju 10),
- instalirajte i aktivirajte rezidentni antivirus i program za zaštitu od špijunskih programa (više o tome možete naći u poglavlju 14),
- isključite Windows servise kojima ne verujete[49].

Dodatno, ukoliko je računar koji obezbeđujete server, potrebno je da regulišete i kontrolu pristupa servisima i obezbedite viši nivo fizičke sigurnosti. Ukoliko obezbeđujete domen kontroler, potrebno je da obavite i kontrolu pristupa na nivou aktivnog direktorijuma (na primer, da pažljivo proberete osobe kojima ćete dodeliti ulogu domenskog administratora).

8.2 Korisnički nalozi, grupe i prava korisnika

Operativni sistem Windows 2000/XP/2003 je višekorisnički, što znači da na jednom računaru može raditi više korisnika. Ukoliko se radi o serveru sa instaliranim Terminal services uslugama, više korisnika može raditi istovremeno. Za svakog korisnika koji želi da koristi Windows potrebno je napraviti korisnički nalog. Korisnički nalog je skup podataka o korisniku koji, između ostalog, sadrži i korisničko ime (engl. *username*) i odgovarajuću lozinku kojom je taj nalog zaštićen. Korisnik navodi lozinku prilikom prijavljivanja na računar ili na domen. Postoje dve vrste korisničkih naloga: lokalni i domenski. Lokalni nalozi postoje na svakom Windows 2000/XP/2003 računaru i služe za autentifikaciju korisnika na tom računaru i kontrolu pristupa resursima tog računara. Administriraju se pomoću *Computer Management* alata.

Osim korisničkih, postoje i takozvani grupni nalozi, tj. korisničke grupe. Administratori sistema ili domena organizuju, tj. učlanjuju korisnike u grupe kako bi jednostavnije obavili neke administrativne zadatke, kao što je

49 Neki korisnici, na primer, isključuju Messenger, Error i Time Synhronization servise, podršku za Remote Desktop i Remote Assistance, onemogućuju praćenje korisnika preko Media Playera, kreiranje liste skoro korišćenih dokumenata (Recent). Ukoliko spadate u grupu takvih korisnika, programi XPAntiSpy (www.xp-antispy.org) i SafeXP (www.theorica.net/safexp.htm) biće vam od koristi.

dodela prava za pristup objektima. Postoje dve vrste korisničkih grupa: lokalne (postoje na svakom računaru) i domenske (postoje u Windows 2000/2003 domenu i čuvaju se u aktivnom direktorijumu).

Na Windows 2000/XP/2003 sistemima postoje i takozvane sistemske grupe. Sistemske grupe nemaju prave članove, ali se često koriste za obavljanje raznih administrativnih zadataka. Uzmite za primer grupu *Everyone*. U ovu grupu administrator nikada ne učlanjuje korisnike. Međutim, pravo dato grupi *Everyone* dato je svim korisnicima tog sistema. Kao drugi primer navešćemo grupu *Creator Owner* – ako grupi *Creator Owner* date neko pravo nad grupom nekih objekata, to pravo se prenosi na vlasnike tih objekata.

Korisnički nalozi (Windows XP)

Na Windows XP računarima postoje dva tipa korisničkih naloga:

- Administrator računara (engl. *Computer Administrator*). Nalozi ovog tipa su članovi lokalne grupe Administrators i imaju potpunu kontrolu nad računarom. Instaliranje softvera, preuzimanje vlasništva nad datotekama ili kreiranje drugih korisničkih naloga može se obaviti samo sa nalogom ovakvog tipa.
- Ograničeni nalog (engl. *Limited*). Nalozi ovog tipa namenjeni su krajnjim korisnicima i ne obezbeđuju prava potrebna za instalaciju softvera i administraciju drugih naloga.
- Gost (engl. *Guest*). Ugrađeni nalog koji omogućava korisnicima koji nemaju svoj nalog da se prijave na računar. U podrazumevanom stanju gost nije zaštićen lozinkom, ali je isključen i preporučuje se da tako i ostane.

Osim korisničkih i grupnih naloga koje kreira administrator, na Windows sistemima postoje i ugrađene (engl. *built-in*) grupe i korisnički nalozi. Primer ovakvog naloga je Administrator, koji se ne može obrisati niti ukloniti iz ugrađene grupe Administrators.

Od značajnijih ugrađenih grupa navešćemo sledeće:

- *Aministrators*. Korisnička grupa u koju su učlanjeni nalozi tipa Computer Administrator.
- *Network Configuration Operators*. Članovi ove grupe mogu obavljati administrativne zadatke koji se tiču konfigurisanja mreže (kao što je modifikacija LAN i dial-up konekcija).
- *Power Users*. Članovi ove grupe mogu deliti štampače i datoteke na mreži, izmeniti sistemsko vreme, promeniti prioritete procesa ili obaviti udaljeno gašenje računara. Članovi ove grupe ne mogu da kreiraju nove ili modifikuju tuđe korisničke naloge, preuzmu vlasništvo nad nekom datotekom, izvrše backup ili učitaju drajvere.
- *Remote Desktop Users*. Članovi ove grupe mogu se prijaviti na sistem koristeći *Remote Desktop* servis.

Prava dodeljena korisnicima

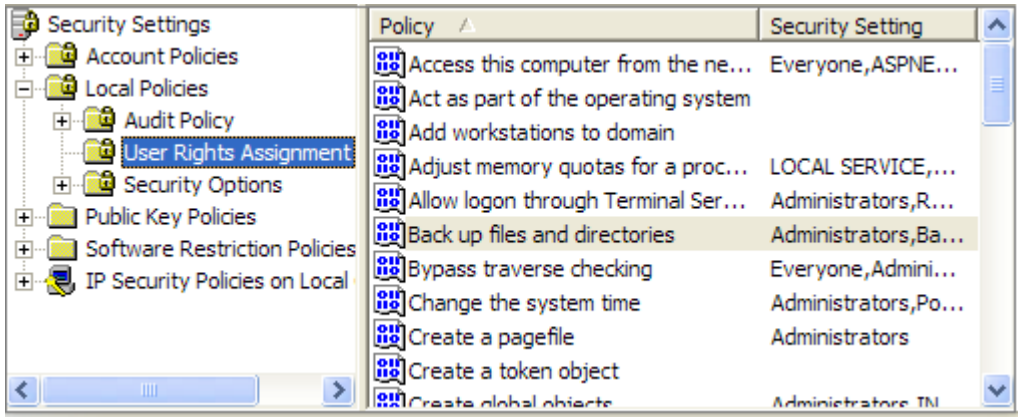
Korisnička prava su jedan od najznačajnijih aspekata sigurnosti Windows XP sistema. U Windows XP terminologiji, korisničko pravo se definiše kao sposobnost korisnika da izvrši neku akciju. Korisnička prava se dodeljuju pomoću *Local Security Settings* alata (slika 8.1), dostupnog iz *Control Panela* (*Administrative Tools → Local Security Settings*).

Potrebno je najpre u hijerarhijskom stablu *Security Settings* odabrati *Local Policies → User Rights Assigment*. Alat prikazuje listu korisničkih prava – većina ovih prava je prilično jasna, a mi navodimo samo neka od njih:

- *Change the System Time*. Korisnik kome je ovo pravo dodeljeno može da izmeni sistemsko vreme. Sistemsko vreme je značajno za funkcionisanje mreže i zato ne treba dozvoliti svakom da ga menja.
- *Force Shutdown from a Remote System*. Korisnik kome je ovo pravo dodeljeno može da nasilno ugasi sistem sa udaljenog računara (jedan od alata koji obavlja udaljeno gašenje isporučuje se uz *Resource Kit*).
- *Load and Unload Device Drivers*. Pod drajverima se u Windows terminologiji, osim drajvera za hardver (npr. za grafičku karticu), podrazumevaju i drajveri koji omogućuju funkcionalnost neke aplikacije (npr. drajver za virtuelne CD/DVD uređaje koji omogućava

simulaciju nekog CD uređaja sa ubačenim medijumom pomoću ISO datoteke na hard disku). Korisnik kome je ovo pravo dodeljeno može da učita ili ukloni drajver iz memorije.

- *Log on Locally*. Korisnik kome je ovo pravo dodeljeno može da se prijavi na sistem.
- *Take Ownership of Files or Other Objects*. Korisnik kome je ovo pravo dodeljeno može da se preuzme vlasništvo nad nekim objektom (npr. datotekom).



Slika 8.1 *– Local Security Settings alat*

Domenski korisnički nalozi i grupe

Domenski nalozi se čuvaju u aktivnom direktorijumu i služe za autentifikaciju korisnika na računarima koji su članovi domena i kontrolu pristupa resursima u domenu. Domenske korisničke grupe mogu biti sigurnosne, koje služe za organizovanje korisnika radi dodele pristupnih prava, i distribucione, koje se u te svrhe ne mogu iskoristiti. Na osnovu opsega važenja, domenske korisničke grupe se mogu podeliti na:

- univerzalne (članovi grupa mogu biti objekti iz celog aktivnog direktorijuma; grupi se mogu dodeliti prava nad objektima iz celog aktivnog direktorijuma);
- globalne (u globalnu grupu se mogu učlaniti korisnici iz istog domena;

grupi se mogu dodeliti prava u bilo kom domenu);

- domenske lokalne (u domensku lokalnu grupu se mogu učlaniti korisnici iz celog aktivnog direktorijuma; grupi se mogu dodeliti prava isključivo nad resursima iz istog domena).

Navodimo neke značajnije ugrađene domenske naloge:

- *Administrator*. Korisnički nalog koji ima potpunu kontrolu nad domenom. Ne može se obrisati niti ukloniti iz grupe *Administrators*.
- *Guest*. Ugrađeni nalog koji omogućava korisnicima koji nemaju svoj nalog da se prijave na domen. Podrazumevano je isključen i preporučuje se da tako i ostane.
- *Administrators*. Domenska lokalna grupa čiji su članovi korisnik *Administrator* i globalna grupa Domain *Admins*.
- *Account operators*. Članovi ove domenske lokalne grupe mogu administrirati domenske korisničke i grupne naloge.
- *Server operators*. Članovi ove domenske lokalne grupe mogu administrirati kontrolere domena.
- *Domain Admins*. Globalna grupa u koju treba učlaniti sve korisnike koji su administratori domena.

Grupne polise i dodela prava korisnicima domena

Grupna polisa je skup različitih podešavanja kojima se može uticati na konfiguraciju računara. Grupne polise omogućavaju da se neka potrebna konfiguracija koja obuhvata podešavanja operativnog sistema, aplikacija, sigurnosti kao i radnog okruženja, definiše kao načelo koje se primenjuje na grupu korisnika. Grupna polisa ima dva osnovna dela:

- deo koji određuje konfiguraciju računara i primenjuje se kada se računar pokrene (*Computer Configuration*),
- deo koji određuje okruženje korisnika i primenjuje se kada se korisnik prijavi na mrežu ili lokalni računar (*User Configuration*).

Neki delovi polise se nakon podizanja operativnog sistema i prijavljivanja

korisnika primenjuju u određenim intervalima, a osvežavanje, tj. ponovna primena grupne polise može se obaviti i ručno, naredbom gpupdate.

Osim lokalnih polisa, kojima se formira konfiguracija za lokalne korisničke naloge, polise se definišu na nivou domena i organizacionih jedinica. Grupna polisa je objekat aktivnog direktorijuma koji se kreira i vezuje za domen ili organizacionu jedinicu. Redosled primene polisa je bitan jer se njime određuje i njihov prioritet polisa: najpre se primeni lokalna polisa, zatim grupna polisa vezana za domen i na kraju grupne polise vezane za organizacione jedinice (hijerarhijski, počev od polise organizacione jedinice na vrhu hijerarhije). Shodno tome da li se podiže operativni sistem ili se korisnik prijavljuje na domen, primenjuju se one polise koje odgovaraju organizacionim jedinicama u kojima se nalazi računar, odnosno korisnik. Polisa sa najvećim prioritetom je polisa vezana za organizacionu jedinicu u kojoj se nalazi računar odnosno korisnik.

U delu koji se tiče konfiguracije računara (*Computer Settings*) nalazi se skup podešavanja koja se tiču sigurnosti i upotrebe korisničkih naloga (Security Settings). U ova podešavanja spadaju:

- podešavanja bezbednosti korisničkih lozinki (*Account Policy → Password Policy*),
- zaključavanje naloga u slučaju pokušaja neautorizovanog pristupa (*Account Policy → Account Lockout Policy*),
- podešavanja koja se tiču praćenja pristupa resursima (*Local Policy → Audit Policy*),
- posebnih prava koji korisnici imaju na sistemu (*Local Policy → User Rights Assigments*),
- dodatna sigurnosna podešavanja (*Local Policy → Security Options*).

8.3 NTFS objekti i deljeni mrežni resursi

Počev od MS-DOS operativnog sistema, pa zaključno sa Windows 2003 Server familijom, Microsoft je u svoje operativne sisteme ugrađivao podršku za FAT, FAT32 i NTFS sisteme datoteka. NTFS je vidljiv isključivo iz NT

zasnovanih Windows operativnih sistema, kao što su Microsoft Windows NT, 2000, XP i 2003 Server. Navodimo ukratko neke prednosti NTFS sistema datoteka (koje se tiču sigurnosti) u odnosu na FAT i FAT32:

- mogućnost šifrovanja datoteka s ciljem povećanja sigurnosti,
- kontrola pristupa pomoću određenih skupova pristupnih prava dodeljenih ovlašćenim korisnicima i grupama,
- praćenje aktivnosti na sistemu datoteka, koje omogućava brz oporavak od logičkih oštećenja.

Datoteke i direktorijumi su objekti NTFS sistema datoteka i slične su strukture (NTFS tretira direktorijum kao specijalnu datoteku). Svaki objekat ima svoje ime i sadržaj. Sadržaj datoteke su konkretni podaci. Sadržaj direktorijuma čine indeksi relativnih lokacija, veličina direktorijuma i bit-mapirana reprezentacija strukture direktorijuma. NTFS podržava Unicode imena datoteka, dužine do 255 karaktera. Svaki objekat u sistemu datoteka osim imena opisuju i:

- Standardni atributi. Ove atribute NTFS nasleđuje sa FAT sistema datoteka. Standardni atributi su H (*hidden*), S (*system*), R (*read-only*) i A (*archive*).
- Prošireni atributi. U navedene atribute spadaju razni atributi koje dodaju programi pomoću kojih su datoteke kreirane. Na primer, Microsoft Word dokumentu pridružuje ime autora, naslov i predmet.
- Sigurnosni opis objekta (*Security Descriptor*)

Sigurnost NTFS volumena je izvedena iz Windows objektnog modela, karakterističnog za Windows 2000/XP/2003. Svaki NTFS objekat ima svoj sigurnosni opis koji je smešten u MFT. Sigurnosni opis dodeljuje se svakom objektu, odnosno datoteci i direktorijumu na NTFS sistemu datoteka prilikom kreiranja. Sigurnosni opis sadrži informacije o vlasniku resursa, tj. žeton za pristup vlasnika (engl. *access token*) i listu kontrole pristupa objektu koja utvrđuje privilegije koje su dodeljene svakom korisniku koji ima pristup datoteci.

NTFS dozvole se dodeljuju ili oduzimaju korisnicima i grupama. Lista za kontrolu pristupa (engl. *Access Control List*, *ACL*), realizovana je u vidu tabele u kojoj svaku zapis (engl. *Access Control Entry*, *ACE*) predstavlja

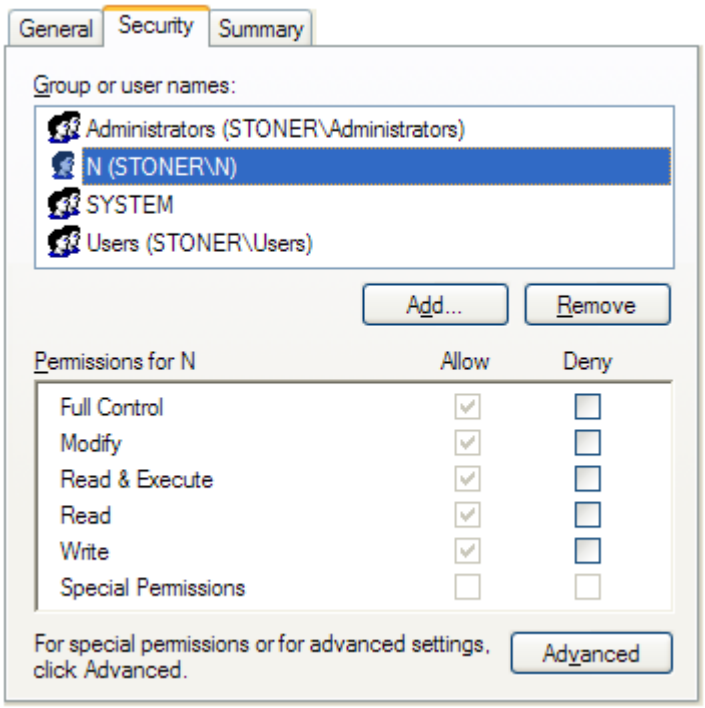
jednog korisnika ili grupu. Korisnicima i grupama se na taj način eksplicitno dodeljuju ili oduzimaju prava u odnosu na objekat. Grupe i korisnici su u ACL predstavljeni numeričkim vrednostima SID (*Security Identifiers*) koje ih jednoznačno identifikuju. Kontrola pristupa objektima moguća je isključivo na NTFS volumenima. Dozvole važe, bez obzira da li korisnik pristupa resursu preko mreže ili lokalno. Prava pristupa data grupi prenose se na grupe i korisnike koji su u tu grupu učlanjeni.

Atomarne dozvole omogućavaju korisniku da nad objektima fajl sistema izvrši određenu akciju. Standardne NTFS dozvole su sistemski predefinisani skupovi atomarnih dozvola i kao takve omogućavaju korisniku da izvrši određeni skup akcija nad objektima sistema datoteka. Slično, specijalne NTFS dozvole su korisnički definisan skup atomarnih dozvola koji se može dodeliti nekom objektu.

U standardne NTFS dozvole spadaju:

- *Read* – minimalna dozvola koja omogućava korisniku kom je data da pregleda sadržaj objekta, tj. da pročita sadržaj datoteke;
- *Write* – dozvola za upis novog sadržaja u objekat; izmena sadržaja objekta je moguća ukoliko je korisniku pored ove dozvole data i dozvola *Read*, koja omogućava korisniku pregledanje sadržaja;
- *Read and Execute* – standardna dozvola *Read* proširena specijalnom dozvolom T*raverse Folder/Execute File* (omogućava zaobilaženje restrikcija na višim hijerarhijskim nivoima u stablu direktorijuma i izvršavanje datoteka);
- *Modify* – unija standardnih dozvola *Read and Execute* i *Write* proširena atomarnom dozvolom za brisanje objekta;
- *Full Control* – uključuje sve atomarne dozvole; korisnik kome je data ova dozvola ima sva prava nad objektom, uključujući i mogućnost dodele i oduzimanja NTFS dozvola i preuzimanja vlasništva;
- *List Folder Contents* – dozvola za pregledanje sadržaja direktorijuma; logično, može se dati samo nad direktorijumom.

Standardne dozvole za pristup objektu dodeljuju se pomoću kartice *Security* dijaloga *Properties* tog objekta (slika 8.2).



Slika 8.2 *- Postavljanje NTFS dozvola*

NTFS uključuje mogućnost nasleđivanja dozvola sa direktorijuma koji se u hijerarhiji stabla nalazi na višem nivou, odnosno propagacije dozvola na poddirektorijume i datoteke. U tom smislu, dozvole se mogu svrstati u eksplicitno dodeljene (dozvole koje su direktno priključene samom objektu) i nasleđene.

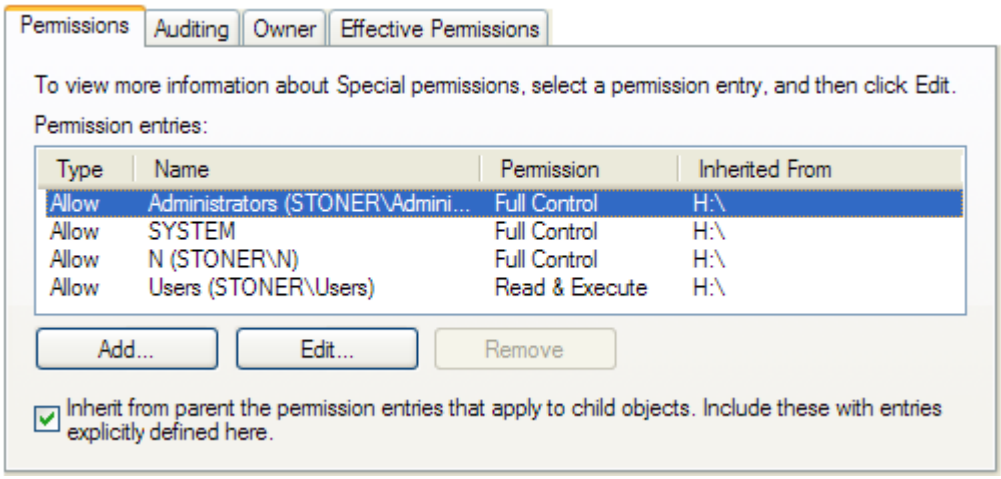
Svaki korisnik može biti član više grupa. Dozvole nad jednim objektom sistema datoteka mogu biti dodeljene većem broju grupa čiji je dati korisnik član. Efektivne dozvole korisnika formiraju se na sledeći način:

- različite dozvole dodeljene grupama kojima korisnik pripada se sabiraju,
- različite nasleđene i eksplicitno dodeljene dozvole se sabiraju i
- zabrana dozvole nadjačava dodelu.

Koristeći kartice dijaloga *Advanced Security Settings* (slika 8.3) korisnici

mogu:

- postaviti specijalne dozvole,
- omogućiti ili onemogućiti nasleđivanje dozvola,
- omogućiti praćenje pristupa NTFS objektima,
- obaviti prenos vlasništva,
- proračunati efektivne dozvole koje će neki korisnik imati u odnosu na taj objekat.



Slika 8.3 – *Advanced Security Settings*

Deljeni mrežni resursi

Direktorijume, tj. foldere (prema Windows terminologiji) na mreži mogu deliti samo oni korisnici kojima je dodeljeno pravo *Create Permanent Shared Objects*. Ove privilegije su date članovima grupa *Administrators* ili Power Users na radnim stanicama, odnosno *Administrators* ili *Server Operators* na serveru.

Dozvole za pristup deljenim resursima se primenjuju isključivo na korisnike koji podacima u deljenom direktorijumu pristupaju preko mreže – na primer, mapiranjem deljenih foldera na logičke diskove, ili iz *Windows*

Explorera (\\fileserver\sharedfolder). Dozvole se primenjuju na direktorijum koji je deljen, kao i na njegov celokupan sadržaj (sve datoteke i poddirektorijumi koji se u njemu nalaze). Deljeni direktorijum može biti na FAT ili NTFS sistemu datoteka. Efektivne dozvole se određuju kao presek dozvola, tj. unija restrikcija dozvola za pristup deljenim direktorijumu i NTFS dozvola konkretnog objekta u tom direktorijumu. Preporučuje se da dozvole za pristup deljenom resursu budu što manje restriktivne (na primer, možete dati grupi *Authenticated Users* dozvolu *Full Control*), a da se kontrola pristupa u pravom smislu te reči obavi NTFS dozvolama.

Postoje tri tipa dozvola za pristup deljenim direktorijumima:

- *Read*. Dozvola koja omogućava pregledanje imena poddirektorijuma i datoteka i njihovih atributa, otvaranje poddirektorijuma, čitanje sadržaja datoteka i pokretanje programa.
- *Change.* Proširenje dozvole Read mogućnostima kreiranja, izmene sadržaja i brisanja datoteka i poddirektorijuma.
- *Full Control*. Proširenje dozvole Change mogućnostima izmene NTFS dozvola i preuzimanjem vlasništva nad poddirektorijumima i datotekama (pod uslovom da se direktorijum nalazi na NTFS sistemu datoteka).

Dozvole za pristup preko mreže se dodeljuju u kartici *Sharing* dijaloga *Properties* deljenog direktorijuma.

8.4 Praćenje događaja i pristupa resursima

Praćenje sigurnosnih događaja (engl. *auditing*) i pristupa resursima je jedna od važnijih zaštitnih mera. Sigurnosni događaji su sve akcije usmerene na resurse koji su zaštićeni nekom sigurnosnom merom, kao što je kontrola pristupa. Na primer, sigurnosni događaj je promena sadržaja ili pristupnih prava direktorijuma, prijavljivanje na domen, kreiranje ili izmena naloga i izmena grupne polise. Praćenje događaja se najčešće primenjuje na domen kontrolerima i serverima, ali i na radnim stanicama koje su deo domena ukoliko se na njima nalaze značajniji resursi. Cilj primene ove zaštitne mere je formiranje dnevnika događaja (engl. *log*) na osnovu koga se

mogu otkriti mogući propusti u primeni nekih sigurnosnih mera.

Postoji nekoliko kategorija događaja koje se mogu pratiti:

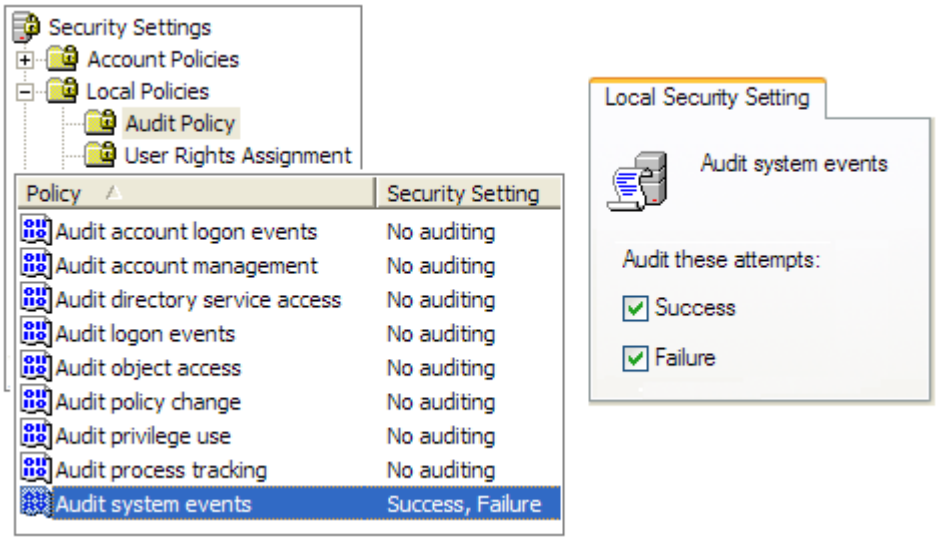
- prijavljivanje korisnika na domen (*Account Logon Events*), prijavljivanje korisnika na računar i autentifikacija korisnika radi pristupa deljenim resursima (*Logon Events*),
- kreiranje, izmena svojstava i brisanje korisničkih naloga i grupa (*Account Management*), npr. promena lozinke ili članstva u grupama,
- pristup objektima aktivnog direktorijuma (*Directory Service Access*),
- pristup resursima (*Object Access*),
- korišćenje privilegija, tj. prava dodeljenih korisnicima (*Privilege Use*),
- promena polise (*Policy Change*), kao što je dodela ili oduzimanje prava korisnicima,
- pokretanje, gašenje i promena prioriteta procesa (*Process Tracking*),
- sistemski događaji (*System Events*), kao što je promena sistemskog vremena.

Zavisno od kategorije, moguće je pratiti uspešne, neuspešne ili obe vrste događaja. Logično je da se događaji poput prijavljivanja korisnika na domen i pristup objekima aktivnog direktorijuma, tj. korišćenje direktorijumskih servisa, prate na domen kontrolerima. Ostali događaji se prate na onim mestima koja su vam sumnjiva po pitanju postojanja sigurnosnih propusta. Događaji se beleže u dnevnik koji se nalazi na računaru na kome je omogućeno praćenje tih događaja.

Događaji zabeleženi u dnevniku mogu se pregledati pomoću alata Event Viewer. Ovaj alat omogućava pretraživanje događaja po određenom kriterijumu, podešavanje osobina dnevnika (npr. šta će se desiti sa starim događajima kada se dnevnik, čija je veličina ograničena, napuni), snimanje dnevnika u drugu datoteku i brisanje događaja iz dnevnika.

Praćenje događaja je deo grupne polise. Praćenje nekih događaja, kao što su prijavljivanje korisnika na sistem, pokretanje procesa ili sistemski događaji, omogućava se jednostavnom izmenom odgovarajuće grupne polise (slika 8.4), tj. omogućavanjem praćenja odgovarajuće kategorije

događaja.



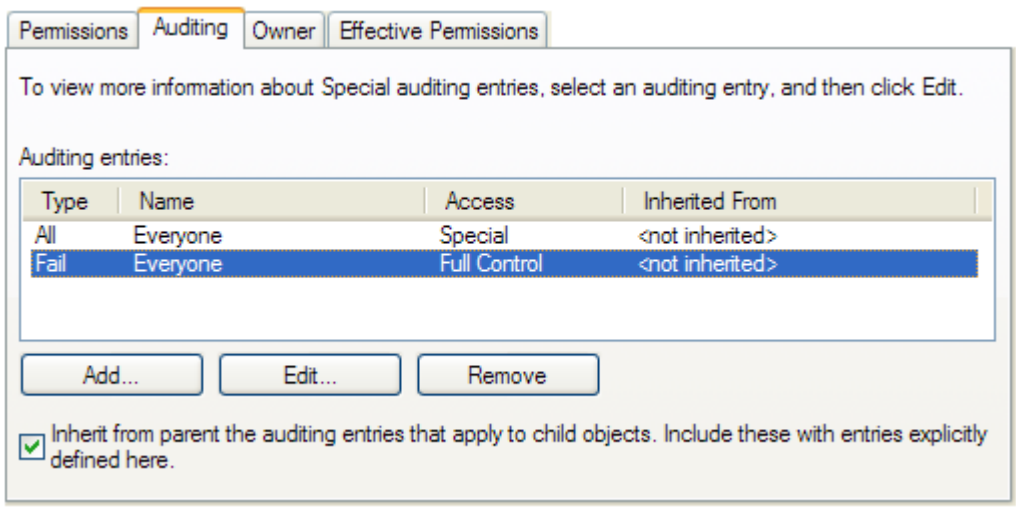
Slika 8.4 *– Praćenje sistemskih događaja*

Praćenje pristupa resursima

Ukoliko želite da pratite pristup resursima nekog računara (na primer, direktorijumu sa podacima i štampaču), potrebno je da:

- omogućite praćenje kategorije događaja – pristup resursima (*Object Access*)
- na samom resursu specificirate šta želite da pratite (kartica *Auditing* dijaloga *Advanced Security Settings* resursa), odnosno koje korisnike, koje aktivnosti i da li želite da pratite samo uspešne ili neuspešne događaje, ili obe vrste događaja (slika 8.5).

Praćenjem pristupa resursima možete, na primer, otkriti ko štampa velike količine dokumenata na štampaču ili ko pokušava da pristupi datotekama kojima mu je pristup zabranjen.



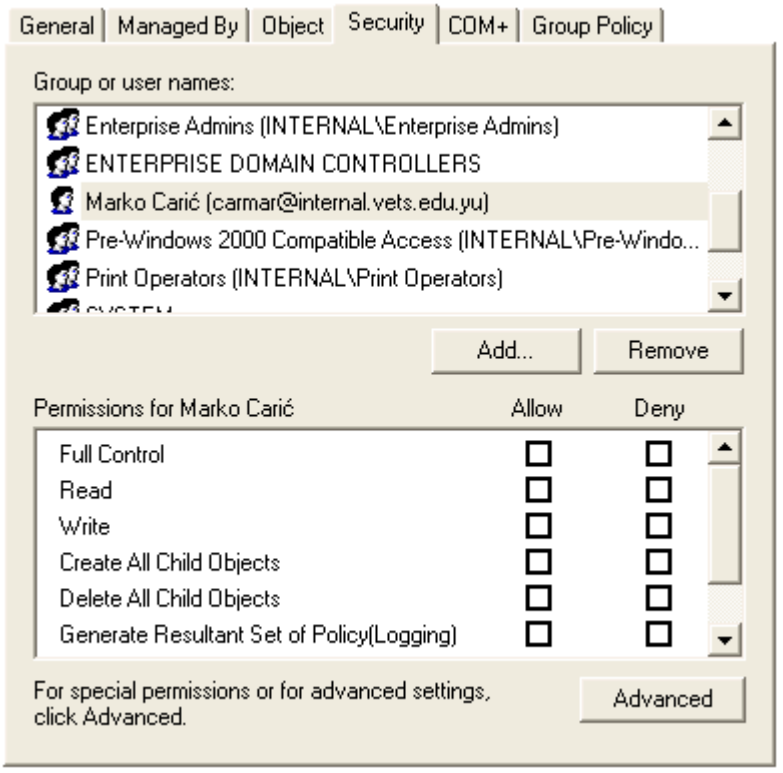
Slika 8.5 *– Dijalog Advanced Security Setting, kartica Auditing*

8.5 Delegacija ovlašćenja u aktivnom direktorijumu

Direktorijum je baza podataka o mrežnim resursima koja klijentima omogućava da pomoću direktorijumskog servisa pronađu odgovarajuće informacije o mrežnim servisima, računarima i korisnicima. Na primer, direktorijumski servisi se mogu koristiti za autentifikaciju korisnika na mrežu i kontrolu pristupa; u tom slučaju, direktorijum administratoru olakšava upravljanje mrežnim resursima. Aktivni direktorijum (Active Directory) je Microsoftova implementacija LDAP protokola (omogućava pristup X.500 direktorijumskim servisima, koji su opisani u 6.6) za mreže sa Windows operativnim sistemom. Aktivni direktorijum je distribuirana baza podataka sa strukturom stabla u kome su mrežni resursi predstavljeni objektima. Objekti su opisani skupom atributa koji je određem tipom objekta. Objekti se organizuju pomoću organizacionih jedinica (engl. *organizational unit*) koje se nalaze unutar domena i mogu po potrebi oslikavati stvarnu raspodelu resursa (na primer, korisnici organizovani po odeljenjima u kojima rade).

Ovakav način organizacije mrežnih resursa omogućava da se poslovi administriranja direktorijuma hijerarhijski raspodele. Na primer,

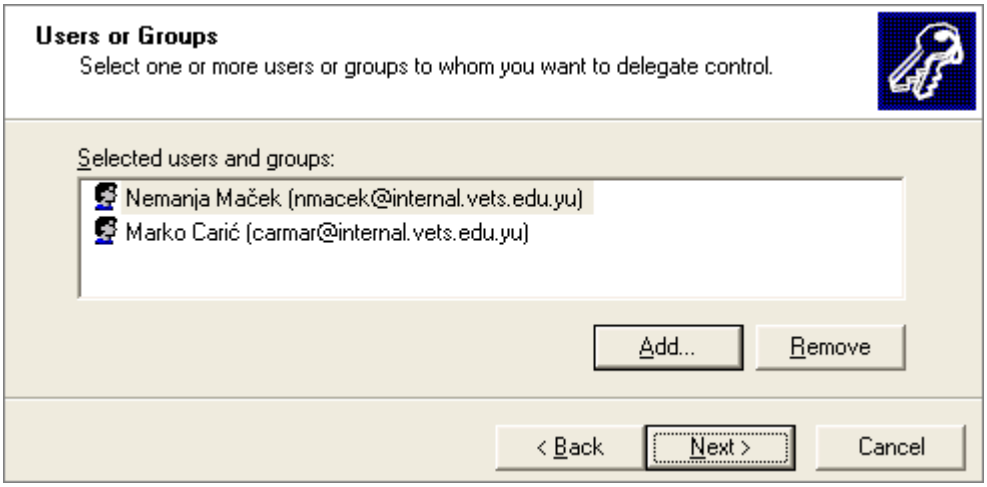
administrator aktivnog direktorijuma može u svakom domenu proglasiti po jednog korisnika domenskim administratorom (tako što će ga učlaniti u domensku grupu Domain Admins), a domenski administratori mogu poslove administriranja domena delegirati korisnicima koji pripadaju odgovarajućim organizacionim jedinicama. Delegiranje administrativnih zadataka svodi se na delegiranje ovlašćenja nad objektima aktivnog direktorijuma. Za svaki objekat postoji specijalna lista za kontrolu pristupa koja određuje skup akcija koje korisnici mogu da obave nad tim objektom (slika 8.6).



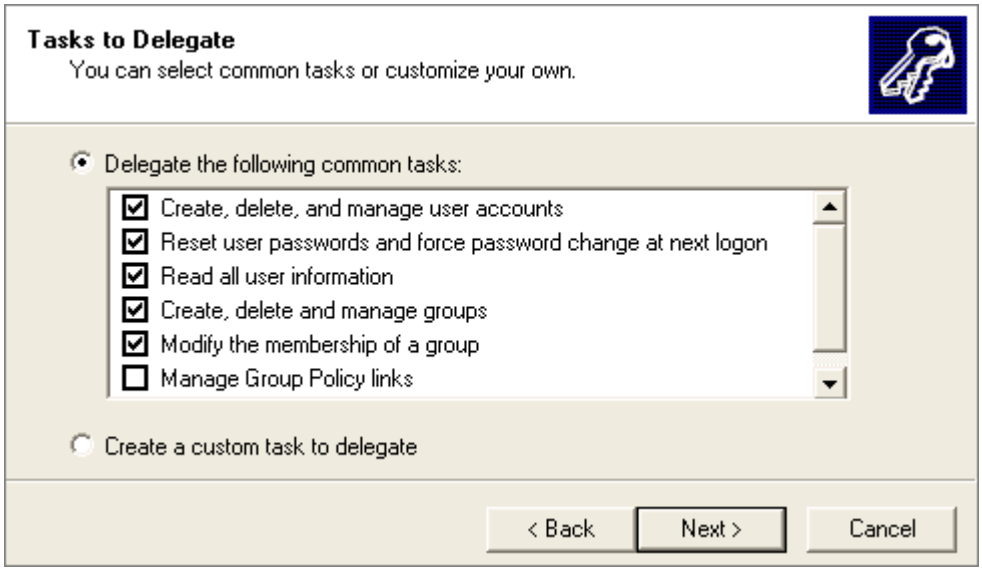
Slika 8.6 *– Ovlašćenja u aktivnom direktorijumu*

U akcije koje korisnici mogu izvršiti nad nekim objektom spadaju, na primer, kreiranje, brisanje i promena atributa korisničkih naloga i grupa, regulisanje članstva u grupama, zamena lozinki drugih korisnika, čitanje i izmena vrednosti atributa nekog objekta, kreiranje ili brisanje podobjekata (engl. *child objects*). Ovlašćenja se dodeljuju pomoću čarobnjaka Delegation of Control, koji zahteva da se navedu korisnici i/ili grupe kojima se dodeljuju

ovlaščenja (slika 8.7) i skup ovlašćenja koja se dodeljuju (slika 8.8).



Slika 8.7 *– Odabir korisnika kojima se daju ovlašćenja*



Slika 8.8 *– Odabir ovlašćenja koja se delegiraju*

8.6 Pitanja i zadaci

8.1 Kreirajte dva lokalna korisnička naloga Korisnik1 i Korisnik2 tipa Limited Account. Prijavite se na sistem kao Korisnik1 i kreirajte neku datoteku. Dodelite kao Administrator korisniku Korisnik2 pravo preuzimanja vlasništva nad tuđim datotekama. Prijavite se kao Korisnik2 i preuzmite vlasništvo nad prethodno kreiranom datotekom. Dodelite sebi odgovarajuće pravo pristupa i pročitajte njen sadržaj.

8.2 Probajte pod korisničkim nalogom Korisnik1 tipa Limited Account da kreirate lokalni korisnički nalog Korisnik3. Da li ste u tome uspeli? Zašto? Dodelite kao administrator korisniku Korisnik1 pravo da kreira i modifikuje druge korisničke naloge i grupe. Probajte sada da kao Korisnik1 kreirate neki korisnički nalog.

8.3 Kreirajte lokalne korisničke grupe Test i Admin i direktorijum Test na nekom NTFS volumenu. Postavite prava pristupa nad direktorijumom Test tako da:

- članovi grupe Admin imaju sva prava nad direktorijumom Test i njegovim sadržajem,
- članovi grupe Test mogu da čitaju sadržaj svih objekata iz direktorijuma Test, kreiraju objekte u direktorijumu Test i da nad objektima koje u njemu kreiraju imaju sva prava.

8.4 Izmenite prava grupe Test iz prethodnog zadatka tako da njeni članovi ne mogu menjati sadržaj datoteka koje su kreirali.

8.5 Instalirajte štampač na vašem računaru i ograničite pristup štampaču tako da na njemu mogu štampati samo članovi grupe Test.

8.6 Obezbedite praćenje sledećih događaja na Vašem računaru:

- pokušaj neovlašćenog pristupa direktorijumu Test iz zadatka 8.3,
- štampanje na štampaču koji ste instalirali u zadatku 8.5,
- prijavljivanje na sistem (uspešno i neuspešno).

9

Kontrola pristupa i zaštita operativnog sistema Linux

9.1 Opšte o sigurnosti Linux sistema

Linus Torvalds je 1991. započeo rad na novom operativnom sistemu koji objedinjuje System V R4 i BSD UNIX standarde. Svoj rad je objavio na Internetu i podsticao druge programere širom sveta da se priključe njegovom daljem razvoju. Linux je ubrzo postao veoma popularan među računarskim entuzijastima koji su tražili alternativno rešenje za postojeće operativne sisteme za PC računare (DOS, Windows). Iako je prvobitno namenjen 32-bitnim Intel x86 mikroprocesorima (počevši od 80386), na kojima može funkcionisati kao radna stanica (workstation) ili kao server, jezgro Linux sistema je modifikovano i prilagođeno procesorima koji ne pripadaju Intel x86 klasi.

Linux je višekorisnički, višeprocesni operativni sistem sa potpunim skupom UNIX kompatibilnih alata, projektovan tako da poštuje relevantne POSIX standarde. Linux sistemi podržavaju tradicionalnu UNIX semantiku i potpuno implementiraju standardni UNIX mrežni model. Linux je raspoloživ kao besplatan operativni sistem licenciran pod GNU GPL licencom (GNU General Public Licence). Izvorni kod Linux sistema javno je raspoloživ i može biti modifikovan tako da odgovara specifičnim potrebama. Linux se može slobodno distribuirati među korisnicima. Brojne profitne i neprofitne organizacije čine Linux raspoloživim u formi distribucija koje sadrže kolekciju CD ili DVD medijuma na kojima se nalaze operativni sistem, izvorni kod, dokumentacija, kao i štampana uputstva za instalaciju i upotrebu sistema. Cene ovakvih distribucija su u većini slučajeva simbolične, osim ako se u distribuciji nalazi komercijalni softver ili je distribucija specifične namene.

Verovatno ste do sada čuli i pročitali da se Linux sistemi veoma često koriste kao mrežni serveri. U ovoj knjizi nećemo navoditi prednosti Linuxa u odnosu na Windows operativni sistem; ukoliko ste navikli na Windows i možete da odvojite dodatnu svotu novca za softver i klijentske licence, slobodno ga koristite kao serversku platformu. U poglavlju 8 ukratko je objašnjeno kako se može otkriti lozinka Windows administrativnog naloga – postupak je veoma sličan ukoliko želite da otkrijete lozinku root naloga na Linux sistemima. Ponavljamo, operativni sistem će ostati nesiguran ukoliko ne iskoristite mehanizme zaštite koje taj sistem obezbeđuje. Jednostavno

rečeno, Linux sistem je siguran onoliko koliko ga administrator učini sigurnim.

Počnite sa osnovnim merama zaštite

Nakon eliminisanja potencijalnog rizika sigurnosti uklanjanjem neželjenih mrežnih servisa, potrebno je preuzeti određene mere u cilju povećanja sigurnosti preostalih servisa i softvera na računaru (serveru). Ukoliko se primene sledeći postupci, administrator sistema može za relativno kratak vremenski period sprečiti razne vrste napada na sistem.

- Postavljanje lozinke za pristup BIOS konfiguraciji i zabrana podizanja operativnog sistema sa flopi diska i CD-ROM uređaja

  Time se sprečava mogućnost podizanja operativnog sistema sa izmenljivog medijuma. Korisnik koji podigne "live" distribuciju Linux sistema sa CD-ROM medijuma prijavljuje se kao root, a nakon toga lako može aktivirati sve lokalne sisteme datoteka i preuzeti i/ili uništiti sve podatke na njima.

- Programi za podizanje operativnog sistema (LILO i GRUB)

  LILO i GRUB su najčešće korišćeni programi koji pune memoriju jezgrom operativnog sistema. U konfiguracionim datotekama ovih programa moguće je navesti lozinku kojom se slika jezgra štiti lozinkom ili se određuje vremenski period čekanja na izbor nakon kog se prihvata podrazumevana opcija. Na primer, ispravna konfiguracija boot menadžera na računaru na kome se nalazi samo jedan operativni sistem omogućava korisnicima da u vremenskom periodu od pet sekundi navedu parametre kernelu. U tom slučaju se od njih zahteva lozinka. Nakon tog perioda, operativni sistem se podiže bez dodatnih parametara i za to se ne zahteva lozinka. Ukoliko kao boot menadžer koristite LILO, ovakvu konfiguraciju ćete postići navođenjem direktiva default, timeout, restricted i password u datoteci /etc/lilo.conf. Dodatno, potrebno je zaštititi datoteku /etc/lilo.conf adekvatnim pravima pristupa (na primer, 600) pošto se lozinka čuva kao otvoreni tekst, i zabraniti izmenu njenog sadržaja (postavljanjem imutability flega).

- Privremeno isključivanje servera sa mreže

  Sigurnosne mere ne treba primenjivati dok je server "na mreži". Preporučuje se da root deaktivira mrežne interfejse pre primene sigurnosnih mera. Mrežni interfejs se može deaktivirati sledećom komandom:

```
# ifdown eth0
```

  Nakon primene sigurnosnih mera mrežni interfejs se može aktivirati komandom:

```
# ifup eth0
```

  Ukoliko na sistemu postoji više mrežnih uređaja, svi se mogu isključiti komandom:

```
# /etc/rc.d/init.d/network stop
Shutting down interface eth0 [OK]
Disabling Ipv4 packet forwarding [OK]
```

Nakon primene sigurnosnih mera svi mrežni interfejsi mogu se aktivirati komandom:

```
# /etc/rc.d/init.d/network start
Setting network parameters [OK]
Bringing up interface lo [OK]
Bringing up interface eth0 [OK]
```

Prevođenje monolitnog jezgra sa odgovarajućim parametrima

Najveći broj korisnika Linux sistema koristi jezgro koje se isporučuje kao podrazumevano za konkretnu distribuciju. Međutim, ukoliko Linux koristite kao serversku platformu, poželjno je da se upoznate sa postupcima preuzimanja izvornog koda jezgra sa Interneta, izmenom i prilagođavanjem konfiguracije i prevođenjem jezgra. Izgradnja prilagođenog jezgra ima svoje prednosti – na taj način možete specificirati sigurnosne opcije koje želite i u jezgro uključiti samo neophodnu drajversku podršku. Takođe, prevođenjem jezgra postajete svesni šta se sve nalazi u vašem jezgru i šta predstavlja potencijalan sigurnosni propust i šta se mora hitno “zakrpiti” (rešenja

sigurnosnih problema ponekad se objavljuju kao zakrpe koje zahtevaju ponovno prevođenje jezgra).

U slučaju mrežnih servera, hardver se ne dodaje često (izuzetak su diskovi i mrežne kartice). Zato se preporučuje da root prevede monolitni kernel u koji će biti uključena podrška za sve što je potrebno kako bi server normalno funkcionisao. Nepostojanje modula smanjuje šansu da napadač uključi modul u rezidentni modularni kernel i time omogući neke funkcije koje za taj server nisu predviđene, a koje bi on mogao da iskoristi za dalje eksploatisanje sistema.

U svakom slučaju, postoje neke preporuke vezane za prevođenje jezgra. Na primer, poželjno je da se sledeće opcije uključe (prilikom izmene konfiguracije komandom make menuconfig):

- CONFIG_INET – TCP/IP skup protokola
- CONFIG_FIREWALL – podrška za mrežnu barijeru
- CONFIG_IP_FIREWALL – podrška za filtriranje IP paketa
- CONFIG_SYN_COOKIES – zaštita od DoS i syn-flood napada
- CONFIG_NETLINK – dvosmerna komunikacija između jezgra i korisničkih procesa
- CONFIG_PACKET – protokol koji omogućava komunikaciju između aplikacije i mrežnog uređaja bez posredovanja protokola u jezgru (*intermediate kernel protocol*)

Takođe, poželjno je da se sledeće opcije isključe (osim u slučaju da sistem koristite kao ruter):

- CONFIG_NET_IPIP – enkapsulacija IP paketa u IP paketu
- CONFIG_IP_ROUTER – rutiranje IP paketa
- CONFIG_IP_FORWARD – rutiranje IP paketa (engl. *forwarding*)
- CONFIG_IP_MULTICAST – podrška za slanje IP paketa na više odredišta

Čišćenje sistema od nepotrebnog softvera

Nakon instaliranja svih potrebnih softverskih paketa i prevođenja prilagođenog jezgra, paket menadžeri (poput programa RPM) nisu potrebni i mogu se skloniti u neki direktorijum (na primer /root) čiji sadržaj niko osim root korisnika ne može pročitati. Dodatno, samim izvršnim datotekama treba promeniti prava pristupa u 700, čime se jedino root korisniku dozvoljava korišćenje tog programa. Alternativna metoda je kopiranje tih programa na neki drugi medijum i uklanjanje sa diska1. Time se onemogućava dalja instalacija programskih paketa, što predstavlja dodatni korak u povećanju sigurnosti.

U ostali softver koji se može ukloniti sa mrežnog servera ubrajaju se i programski prevodioci (na primer, prevodilac za jezik C i asembler), izvorni kod jezgra i programskih paketa, kao i svi mrežni servisi i programi koji se koriste za administraciju mreže a na tom serveru nisu neophodni. Na primer, ukoliko server ima funkciju Web servera, servisi ftpd, telnetd, routed i njima slični nisu potrebni i mogu se ukloniti.

Sigurnost skriptova u /etc/init.d direktorijumu

U /etc/init.d direktorijumu nalaze se skriptovi koji se prilikom podizanja operativnog sistema koriste za zaustavljanje i pokretanje procesa. Pristupna prava ovih skriptova nisu dovoljno restriktivna, tako da ih root mora promeniti, odnosno dozvoliti samo sebi pristup skriptovima u /etc/init.d direktorijumu:

```
# chmod -R 700 /etc/init.d/*
```

9.2 Korisnički nalozi i lozinke

Da Vas podsetimo: na Linux sistemima postoje dve vrste korisničkih naloga – sistemski i regularni. Sistemski korisnički nalozi nastaju prilikom instalacije sistema i služe za specijalne namene. Jedini sistemski korisnik koji se može prijaviti na sistem je root (root ima sve privilegije i služi za administraciju). Nakon instalacije, root kreira regularne korisnikčke naloge

koji služe za prijavljivanje korisnika na sistem. Datoteka /etc/passwd (“*password file*”) je baza podataka koja opisuje sve korisničke naloge jednog Linux sistema. Svaki korisnik je opisan jednom linijom datoteke koju čini sedam polja razdvojenih dvotačkom: korisničko ime, lozinka, UID, GID, opis, lični direktorijum i komandni interpreter. Datoteka /etc/shadow (“*shadow file”*) štiti lozinke korisnika jakom kriptografskom zaštitom. Datoteka sadrži heševe lozinki svih korisnika sistema i informacije o vremenskim ograničenjima.

Svaki korisnik UNIX sistema mora pripadati najmanje jednoj grupi. Identifikator primarne grupe korisnika (GID) naveden je u liniji kojom je korisnik opisan u datoteci /etc/passwd. Primarnoj grupi korisnika se dodeljuju objekti sistema datoteka koje korisnik kreira. Korisnik može pripadati većem broju grupa i u svima je ravnopravan član. Grupe mogu biti sistemske (nastaju prilikom instalacije) i regularne (koje root kreira i upotrebljava u svrhe lakše administracije). Svaka grupa je opisana jednom linijom datoteke /etc/group.

Lozinke korisnika i /etc/shadow

Povećanje opšte sigurnosti Linux sistema obično počinje formulisanjem ograničenja nad lozinkama korisnika. Veliki broj korisnika čuva vredne informacije na računaru, a jedino što njihove podatke štiti od neovlašćenog pristupa je niz od osam karaktera - lozinka. Nasuprot opštem mišljenju, neprobojna lozinka ne postoji. Postoje samo slabe i jake lozinke, od kojih svaka može biti otkrivena korišćenjem odgovarajućeg softvera ili alternativnim metodama, koje se najčešće i koriste. Ukoliko se sledeća pravila ispoštuju lozinke će dobiti svoj zaštitni smisao:

- svaka lozinka mora biti duga bar šest (još bolje, osam) karaktera, formirana od velikih i malih slova, uključujući najmanje jedan numerički karakter,
- lozinke ne smeju biti trivijalne, odnosno formirane na osnovu korisničkog imena, imena devojke, firme i slično,
- za pojedine ili sve korisnike definisan je rok važenja lozinki, odnosno period nakon koga se lozinka mora zameniti novom.

Svaki korisnik može videti sadržaj datoteke /etc/passwd i na taj način

se, na primer, može informisati o drugim korisnicima sistema. To znači da je i polje u kom se nalazi lozinka korisnika javno dostupno - zato se lozinke svih korisnika čuvaju u obliku heša, čime se postiže viši nivo sigurnosti. U datoteci /etc/shadow, kojoj može da pristupi samo root, nalaze se heševi lozinki svih korisnika sistema i informacije o vremenskim ograničenjima. Datoteka /etc/shadow se može kreirati pomoću komande pwconv, koja na osnovu postojeće /etc/passwd kreira novu shadow datoteku. Obrnuti proces, odnosno uklanjanje shadow datoteke, obavlja se komandom pwunconv.

Svaka linija datoteke sadrži ime jednog korisnika iz /etc/passwd, heš lozinke i opisne podatke (npr. datum poslednje izmene lozinke, broj neaktivnih dana pre zaključavanja lozinke, datum prestanka važenja lozinke, itd). Vremenska ograničenja lozinki se mogu pregledati i promeniti komandom chage (Change Age):

```
$ chage -m 10 -M 40 -W 20 -I 7 jsmith
$ chage -l jsmith
Minimum:        10
Maximum:        40
Warning:        20
Inactive:       7
Last Change:            avg 23, 2005
Password Expires:       sep 02, 2005
Password Inactive:      sep 09, 2005
Account Expires:        Never
```

Root korisnički nalog

Korisnički nalog root je deo sistema i kao takav nema sigurnosne restrikcije, odnosno ima najveće privilegije. To znači da sistem podrazumeva da administrator koji koristi taj nalog zna šta radi i da neće ispitivati njegove postupke. Root korisnik može, ukoliko pogreši u pisanju, da obriše kritične sistemske datoteke, tako da je jako bitno da svi koji taj nalog koriste budu veoma pažljivi. Navešćemo neke značajnije zaštitne mere vezane za root korisnički nalog

- prijavite se na sistem kao root samo ako za to postoji izričita potreba (na primer, kreiranje većeg broja korisničkih naloga ili administracija mrežnih servisa),

- root nalog ne sme biti dostupan regularnim korisnicima – na primer, ne smete se prijaviti na računar kao root i napustiti prostoriju[50],
- ograničite sistemsku putanju root korisnika na direktorijume nad kojima obični korisnici nemaju pravo upisa – na taj način sprečavate mogućnost pokretanja trojanaca koje napadači mogu postaviti u neki direktorijum koji se može naći u sistemskoj putanji,
- koristite opisni root prompt, tj. navedite svoje korisničko ime i tekući direktorijum u odzivniku – to je najjednostavniji način da se spreče slučajne greške (kao što je rm -rf *). Na primer:

```
$ PS1="\u@\h, \W $ "
root@abstrakt, tmp $
```

- koristite sistemske naloge za specijalne svrhe ili program sudo – operatoru kome je potrebna privilegija da ugasi sistem ne treba dati root lozinku,
- ograničite prijavljivanje root korisnika na sistem isključivo na sistemske konzole. Datoteka /etc/securetty (“*secure getty*”) određuje terminale i sistemske konzole sa kojih se root može prijaviti na sistem. Sadržaj ove datoteke čita proces login i na osnovu nje utvrđuje da li se root može prijaviti na sistem sa tog terminala ili ne. Prijavljivanje root korisnika samo sa konzole ograničava se upisom linija tty1, tty2, … tty8 u ovu datoteku (ostale linije iskomentarišite ili obrišite); root se na terminal može najpre prijaviti kao regularan korisnik, a zatim iskoristiti komandu su da se prijavi kao root,
- smanjite broj komandi koje se čuvaju u datoteci ~/.bash_history dodavanjem linije HISTSIZE=n u datoteku /etc/profile. Preporučuje se da taj broj bude reda veličine 10 komandi. Obezbedute da se se sadržaj datoteke .bash_history obriše svaki put kada se korisnik odjavi sa sistema (upišite liniju HISTFILESIZE=0 u datoteku /etc/profile).

50 Ovaj problem možete rešiti definisanjem vremena neaktivnosti nakon kog će sistem automatski odjaviti korisnike. Postavite vrednost promenljive TMOUT na broj sekundi neaktivnosti (na primer, TMOUT=300) u datoteci /etc/profile ili bashrc datoteci root korisnika.

Sistemski korisnički nalozi

U cilju povećanja opšte sigurnosti sistema potrebno je ukloniti sve nepotrebne sistemske korisničke naloge. Ovi nalozi služe za specijalne namene, a isporučuju se kao deo Linux distribucije. Većina njih postoji na sistemu čak i ako servisi, za koje su ti nalozi namenjeni, nisu instalirani. Nakon svake nadogradnje ili instaliranja novog softverskog paketa potrebno je proveriti datoteku /etc/passwd i u njoj otkriti nove sistemske korisnike. Sistemski korisnici koji su vezani za servise koji nisu instalirani mogu se ukloniti komandom userdel. U takve korisnike spadaju adm, lp, shutdown, halt, news, mail, uucp, operator, games, gopher, ftp i njima slični. Nakon uklanjanja nepotrebnih sistemskih korisnika, potrebno je ukloniti i prateće sistemske grupe, poput adm, lp, news, mail, uucp, i games. Sintakse komandi za uklanjanje sistemskih korisnika i grupa su:

```
# userdel ime_korisnika
# groupdel ime_grupe
```

Napomena: korisnički nalog mail i grupa mail se ne smeju obrisati ukoliko se Sendmail koristi kao mail server. U slučaju da se kao alternativni mail server koristi qmail, ovi nalozi se mogu obrisati. Nakon brisanja specijalnih naloga, root može kreirati regularne korisničke naloge i grupe i podesiti članstvo korisnika u grupama. U slučaju da se radi o mrežnom serveru, root će kreirati jednog ili dva korisnika (na primer, admin i operator). Kao dodatna mera zaštite mrežnog servera, preporučuje se postavljanje *immutability* flega (sprečava izmene datoteka) za datoteke /etc/passwd, /etc/shadow i /etc/group:

```
# chattr +i /etc/passwd
# chattr +i /etc/shadow
# chattr +i /etc/group
```

Ovakva zaštitna mera ne predstavlja opterećenje u daljem radu, jer se na serverima korisnici ne kreiraju često, tako da se ove datoteke smatraju relativno nepromenljivim. Ukoliko postoji potreba za dodavanjem novog korisnika, potrebno je, najpre, ukinuti immutable fleg sa ovih datoteka.

sudo

Najgora stvar koju iole ozbiljan administrator Linux sistema može da uradi jeste da otkrije root lozinku korisnicima koji ponekad žele da pokrenu neku komandu sa root privilegijama. Zbog toga Linux obezbeđuje sudo alat koji dozvoljava određenom skupu regularnih korisnika da izvršavaju komande sa privilegijama root korisnika. Preporučujemo vam da instalirate program sudo na svim Linux sistemima koje administrirate.

Kao i većina drugih Linux alata, sudo se isporučuje sa pratećom konfiguracionom datotekom (/etc/sudoers). Izmenom sadržaja ove datoteke administrator određuje skup korisnika kojima je dozvoljeno da pokrenu komandu sudo i skup komandi koje sudo korisnici mogu izvršiti sa tuđim privilegijama. Linije u datoteci /etc/sudoers imaju sledeći format:

```
korisnik računar = komanda(e)[pokreni_kao_korisnik]
```

Na primer, ukoliko želite da korisniku johndoe dodelite pravo da pokrene komandu shutdown, korisniku webmaster da zaustavi i ponovo pokrene web server, a korisniku admin da pokrene bilo koju komandu sa root privilegijama, unećete sledeću liniju u ovu datoteku:

```
johndoe ALL =  /sbin/shutdown
webmaster ALL = /etc/init.d/httpd restart
admin ALL = ALL
```

Korisnici kojima su ove privilegije date pokreću komande sa tuđim privilegijama koristeći komandu sudo:

```
$ sudo [-u korisnik] komanda
```

Ukoliko se ne navede parametar -u korisnik, podrazumevaju se root privilegije. Komanda sudo -l korisnik će izlistati spisak dozvoljenih sudo komandi za navedenog korisnika. Navodimo dva primera izvršavanja komande sudo: u prvom, korisnik zadaje komandu za prikazivanje sadržaja shadow datoteke (zahteva root privilegije), a u drugom, pokreće editor jed sa privilegijama korisnika wcoyote:

```
$ sudo cat /etc/shadow
$ sudo -u wcoyote jed /home/wcoyote/.bash_profile
```

Ukoliko koristite sudo da nekim korisnicima date root pristup određenom skupu komandi, obratite pažnju na to da se iz tih komandi ne može "pobeći" u komandni interpreter. Ukoliko korisnik koji pokrene sudo komandu uspe da "izađe" u shell, imaće pristup svim komandama sa root privilegijama.

U podrazumevanom stanju, sudo prati aktivnosti izvršavanja komandi sa tuđim privilegijama i događaje beleži u datoteku /var/log/sudo.log. Svaki događaj predstavljen je jednom linijom u ovoj datoteci. Za svaku pokrenutu sudo komandu beleži se datum pokretanja, ime korisnika koji je pokrenuo komandu, računar i kontrolišući terminal sa koga je komanda pokrenuta i tekući direktorijum u trenutku pokretanja komande.

9.3 Sistemi datoteka i kontrola pristupa

Jedna od najznačajnijih komponenti svake ozbiljne zaštitne polise je kontrola pristupa na nivou sistema datoteka. Kontrolom pristupa na nivou sistema datoteka određuje se skup korisnika koji mogu pristupiti objektima, tj. datotekama i direktorijumima, i nivo pristupa, tj. skup akcija koje autorizovani korisnici mogu izvršiti nad tim objektima. Kontrola pristupa se zasniva na vlasničkim odnosima i pravima pristupa. Prava pristupa se dodeljuju svakoj datoteci i direktorijumu i na osnovu tih prava se određuje nivo pristupa tim objektima.

Prava pristupa za datoteke i direktorijume najlakše se mogu odrediti pomoću komande ls -l koja prikazuje listing sadržaja direktorijuma u dugom formatu.

```
$ ls -ld
drwxrwxr-x  2  jsmith  nm      4096  dec 20  19:52  dir1
drwxrwxr-x  2  nm      nm      4096  dec 20  19:52  dir2
-rw-r--r--  1  root    staff    957  dec 20  19:51  file1
-rw-r--r--  1  root    root   13597  dec 20  19:52  file2
```

Prvi karakter u polju od 10 karaktera ukazuje na tip datoteke: "–" označava običnu, tj. regularnu datoteku (na primer, pdf dokument, sliku ili izvršnu datoteku), "d" označava direktorijum, a "l" simbolički link. Sledećih devet znakova predstavljaju prava pristupa objektu za tri vlasničke

kategorije. Prva tri karaktera definišu prava pristupa vlasnika, druga tri prava pristupa grupe kojoj datoteka pripada i poslednja tri karaktera prava pristupa za ostatak sveta.

Vlasnik (engl. *owner*) je korisnik koji je kreirao objekat, odnosno korisnik kome je root naknadno dodelio vlasništvo. Grupa (engl. *group*) je korisnička grupa kojoj je datoteka formalno priključena. Za razliku od korisnika koji mogu pripadati većem broju grupa, objekti sistema datoteka mogu pripadati samo jednoj grupi. Najčešće je to primarna grupa korisnika koji je objekat kreirao, a root ili vlasnik objekta naknadno mogu promeniti pripadnost objekta grupi. Ostali korisnici (engl. *others*) su svi korisnici koji nisu ni vlasnik objekta, niti pripadaju grupi kojoj objekat pripada. Prava pristupa nekog korisnika određuju se prema vlasničkoj kategoriji kojoj korisnik pripada. Ukoliko vlasnik pripada grupi kojoj je objekat dodeljen, efektivno je vlasničko pravo.

Potpuni skup prava za svaku vlasničku kategoriju čine tri prava: pravo čitanja ("r", *read*), pravo upisa ("w", *write*) i pravo izvršavanja ("x", *execute*). U svim vlasničkim kategorijama prva pozicija je pravo čitanja, druga pravo upisa i treća pravo izvršavanja – "rwx". Ukoliko se na nekoj poziciji umesto slova r, w ili x nalazi crtica ("-"), pravo je ukinuto. Pristupna prava korisnicima omogućavaju da nad datotekom izvrše sledeće akcije:

- "r" – čitanje sadržaja datoteke (prikazivanje na ekranu, štampanje),
- "w" – izmena sadržaja datoteke (ali ne i brisanje datoteke),
- "x" – izvršavanje shell programa ili binarne izvršne datoteke.

Pristupna prava korisnicima omogućavaju da nad direktorijumom izvrše sledeće akcije:

- "r" – čitanje sadržaja direktorijuma, tj. file-info struktura koje se nalaze u direktorijumskom bloku (korisnik može da izvrši komandu ls),
- "w" – izmena sadržaja direktorijuma, tj. dodavanje novih i brisanje postojećih objekata u njemu,
- "x" – pozicioniranje na direktorijum, prikazivanje dugačkog listinga sadržaja i pretraživanje direktorijuma.

Prava pristupa se, takođe, mogu prikazati i sa tri oktalne cifre, pri čemu

svaka cifra odgovara jednoj vlasničkog kategoriji. Pravo čitanja ima vrednost 4, pravo upisa 2, a pravo izvršavanja 1. Na primer, `-rwxr-x---` se može predstaviti kao 750, a `-r-wr-wr--` kao 664.

Na javno dostupnim direktorijumima tipična prava pristupa su 777, što znači da svi korisnici mogu da kreiraju nove i brišu postojeće datoteke, bez obzira na to kome te datoteke pripadaju. Postavljanjem takozvanog *sticky* bita za direktorijum uvodi se dodatno ograničenje: bez obzira na pravo upisa koje korisnik ima nad tim direktorijumom, on u njemu ne može obrisati tuđe datoteke (odnosno datoteke kojima on nije vlasnik).

Podrazumevani vlasnički odnosi i prava pristupa

Prilikom kreiranja Linux dodeljuje objektu vlasnika i grupu i postavlja takozvana podrazumevana prava pristupa. Korisnik koji kreira objekat postaje njegov vlasnik, a objekat se dodeljuje primarnoj grupi tog korisnika. Inicijalna prava pristupa dodeljuju se na osnovu vrednosti promenljive umask. Svaki korisnik ima svoju promenljivu umask, koja se postavlja prilikom prijavljivanja na sistem, a može se videti i promeniti pomoću istoimene komande. Inicijalna prava pristupa za direktorijum određuju se tako što se od potpunog skupa prava (777) ukinu prava definisana promenljivom umask. Inicijalna prava za nove datoteke određuju se tako što se od potpunog skupa prava za datoteku (666)[51] ukinu prava definisana promenljivom umask. Na primer:

```
$ umask 027
$ mkdir my_dir
$ touch my_file
$ ls -l
drwxr-x---  2  johndoe  users  4096  dec  23 14:33  my_dir
-rw-r-----  1  johndoe  users  0     dec  23 14:33  my_file
```

Promena vlasničkih odnosa i prava pristupa

Vlasnik, grupa i prava pristupa se dodeljuju svakom objektu prilikom kreiranja, a kasnije se mogu promeniti. Vlasnik objekta ili root korisnik mogu

51 Nad datotekama se inicijalno ne dodeljuje pravo izvršavanja, pa je potpuni skup prava rw- za sve tri vlasničke kategorije, tj. 666

promeniti prava pristupa, ali samo root može promeniti vlasništvo, tj. "pokloniti" datoteku nekom drugom korisniku. Na taj način se sprečava mogućnost da trenutni vlasnik pokloni veliku datoteku nekom drugom korisniku i na taj način prepuni njegovu kvotu.

Vlasnički odnosi se menjaju komandom chown (Change Owner) i chgrp (Change Group). Na primer, sledećim komanda root može dodeliti datoteku korisniku pperic i grupi admin.

```
# ls -l my_file
-rw-r--r--  1  johndoe  users  0  Apr 28 12:07  my_file
# chown pperic my_file
# chgrp admin my_file
# ls -l myfile
-rw-r--r-- 1   pperic   admin   0  Apr 28 12:07  my_file
```

Pristupna prava se menjaju komandom chmod (Change Mode). Komanda nudi korisnicima dva režima rada – simbolički i oktalni. U oktalnom režimu korisnik navodi sve tri oktalne cifre, odnosno pristupna prava za sve tri vlasničke kategorije, u tačnom redosledu. Na primer:

```
$ ls -l my_file
-rw-rw-rw- 1  johndoe  users  0  dec 23 15:25  my_file
$ chmod 755 my_file
$ ls -l my_file
-rwxr-xr-x 1  johndoe  users  0  dec 23 15:25  my_file
```

U simboličkom režimu vlasnik može dodeliti ili ukinuti prava koja želi, bez poznavanja trenutnih prava pristupa datoteke, što je pogodno za dodelu ili oduzimanje prava većem broju datoteka sa različitim trenutnim skupom prava. Sintaksa komande je sledeća:

```
$ chmod kategorija operator pravo ime
```

gde su:

- vlasničke kategorije: "u" – vlasnik, "g" – grupa, "o" – ostali korisnici, "a" – sve vlasničke kategorije,
- operatori: "+" – dodela prava, "–" – ukidanje prava, "=" – dodela tačno određenih prava "=" nekoj vlasničkoj kategoriji,

- prava: “r” - čitanje, “w” - izmena , “x” - izvršavanje.

Sledećim primerom ilustrovano je postavljanje tačnog skupa prava (r-x) vlasniku i oduzimanje prava upisa grupi i ostalim korisnicima.

```
$ ls -l myfile
-rw-rw-rw-  1  johndoe  users  0  dec 23 15:25  my_file
$ chmod u=rx,go-w my_file
$ ls -l my_file
-r-xr--r--  1  johndoe  users  0  dec 23 15:25  my_file
```

Veoma koristan parametar karakterističan za komande chown, chgrp i chmod je parametar -R, koji omogućava promenu vlasništva, odnosno pristupnih prava kompletnog sadržaja direktorijuma, tj. svih datoteka i poddirektorijuma. Na primer, svim regularnim korisnicima koji ne pripadaju grupi root mogu se sledećom komandom ukinuti sva prava nad direktorijumom /root direktorijumom i njegovim sadržajem:

```
# chmod -R o-rwx /root
```

Sticky bit se postavlja i ukida komandom chmod tako što se u simboličkom režimu svim vlasničkim kategorijama dodeli pravo “t”, a ukida oduzimanjem prava “t”. Sledećim primerom ilustrovano je postavljanje i identifikacija *sticky* bita:

```
$ chmod +t my_dir
$ ls -l my_dir
-rwxrwxrwt  1  root  root  4096  dec 23 15:25  my_dir
```

Dodatno, *sticky* bit se može postaviti komandom chmod u oktalnom režimu, navođenjem cifre 1 pre pristupnih prava.

```
$ chmod 1777 my_dir
```

Datoteke bez vlasnika

Datoteke koje nemaju vlasnika ili nisu dodeljene ni jednoj grupi obično ukazuju na jednu od sledećih činjenica:

- root je ukinuo neki korisnički nalog (datoteke bez vlasnika na

direktorijumima /home ili /tmp),

- neki programski paket koji zahteva postojanje specijalnih sistemskih korisnika je deinstaliran; ti sistemski korisnici su, takođe, uklonjeni prilikom deinstalacije paketa,
- obrisana je neka od sistemskih grupa (karakteristično za specijalne datoteke u direktorijumu /dev),
- napadač je uspeo da uđe u sistem.

Ukoliko datoteka nema vlasnika ili grupu, u izlazu komande ls -l umesto imena vlasnika ili grupe figuriše UID, odnosno GID. Datoteke bez vlasnika ili grupe mogu se lako pronaći pomoću sledeće komande:

```
# find / -nouser -o -nogroup
/usr/share/doc/apache/manual/programs/suexec.html.html
/usr/share/doc/apache/manual/programs/suexec.html.en
/usr/share/doc/apache/manual/programs/suexec.html.ja.jis
# ls -l /usr/share/doc/apache/manual/programs/suexec.html.h*
-rw-r--r--  1  847  800  1659  jun 18 2002  suexec.html.en
```

Nakon pretrage potrebno je pregledati sadržaj ovih datoteka. Ukoliko je sadržaj datoteke "normalan" (na primer, tekstualna datoteka ili slika), datoteci se dodeljuju vlasnik i grupa. Na primer:

```
# cd /usr/share/doc/apache/manual/programs
# less suexec.html.html
# chown root suexec.html.html
# chgrp root suexec.html.html
```

Ukoliko je datoteka binarna, tj. njen sadržaj nije čitljiv, preporučuje se da je obrišete, jer postoji mogućnost da je napadač podmetnuo trojanskog konja ili neki drugi zlonamerni program.

SUID i SGID bitovi

Prilikom izvršavanja nekih programa postoji potreba za privremenim prisvajanjem identiteta drugog vlasnika ili grupe. To se na UNIX sistemima realizuje postavljanjem korisničkog bita (SUID) i grupnog bita (SGID).

Identifikator (UID) korisnika koji je pokrenuo izvršnu datoteku sa

postavljenim SUID bitom menja se identifikatorom vlasnika izvršnog programa u kontekstu procesa u kome se taj program izvršava. Ukoliko je SUID bit izvršnog programa postavljen, u kategoriji vlasnika se na mestu prava "x" nalazi oznaka "s". Bit se postavlja komandom chmod, tako što se kategoriji vlasnika dodaje pravo "s", a uklanja oduzimanjem prava "s" kategoriji vlasnika.

```
$ ls -l tar
-rwxr-xr-x  1  root  root  215K  dec 23 15:25  program
$ chmod u+s tar
$ ls -l tar
-rwsr-xr-x  1  root  root  215K  dec 23 15:25  program
```

Grupni bit (SGID) postavlja se i funkcioniše na sličan način – identifikator realne grupe se u procesu zamenjuje identifikatorom grupe kojoj izvršni program sa postavljenim SGID bitom pripada. Ukoliko je SGID bit izvršnog programa postavljen, u kategoriji grupe se na mestu prava "x" nalazi oznaka "s".

Regularnim korisnicima se na ovaj način može dozvoliti da pokrenu neke komande sa root ovlašćenjima bez otkrivanja lozinke root korisničkog naloga. Na primer, dovoljno je postaviti SUID bit programu za arhiviranje (tar) i time obezbediti da obični korisnici mogu da pročitaju sve datoteke i kreiraju rezervnu kopiju podataka. Međutim, ukoliko je vlasnik programa root, postavljanje SUID bita može biti krajnje opasno. Na primer, ako komanda rm (čiji je vlasnik root) ima postavljen SUID, svaki korisnik će moći da obriše bilo koju datoteku, jer komandu izvršava pod root identitetom. Zato je potrebno SUID i SGID bit ukloniti sa svih programa na kojima on nije apsolutno neophodan. Programi sa postavljenim SUID ili SGID bitom se mogu pronaći sledećom komandom:

```
# find / -type f \( -perm -04000 -o -perm -02000 \)
  -exec ls -l {} \;
```

Nakon toga se bit s lako može ukloniti sa programa kojima nije neophodan (funkcija programa može se lako odrediti komandom man program name, nakon čega se izvodi zaključak). Na primer, komanda rm realno ne zahteva s bit, tako da se on može ukloniti:

```
# chmod a-s /bin/rm
```

ext2/ext3 – specijalni atributi i fleg nepromenljivosti

Na ext2 i ext3 sistemima datoteka postoji nekoliko specijalnih atributa koji se po potrebi mogu dodeliti datotekama. Izdvajamo tri atributa koji su vezani za zaštitu datoteka:

- a (Append only) – dozvoljava isključivo dodavanje novih podataka u datoteku, ali ne i izmenu ili brisanje starih; samo root može postaviti ili ukinuti ovaj atribut; pogodan je za postavljanje na datoteke u kojima se vodi dnevnik događaja (engl. *log*)

- i (Immutable) – datoteka ne može biti modifikovana ili obrisana, ne može joj se promeniti ime niti se može kreirati link koji ukazuje na tu datoteku

- s (Secure deletion) – Prilikom brisanja datoteke u sve blokove koji čine datoteku upisuju se nule; ovo prihvatite sa zadrškom jer su algoritmi za sigurno brisanje mnogo složeniji

Specijalni atributi mogu se postaviti ili ukinuti komandom chattr. Tekući skup atributa može se pogledati komandom lsattr. Atributi se zadaju simbolički, slično kao što se i pristupna prava dodeljuju u simboličkom režimu komandom chmod.

```
$ chattr +i my_file
$ lsattr my_file
---i---------- my_file
```

Datoteka /etc/fstab

Aktivno UNIX stablo može biti sačinjeno od nekoliko sistema datoteka (na primer root, /home, /tmp) koji su montirani na odgovarajuće *mount-point* direktorijume. Datoteka /etc/fstab sadrži informacije o sistemima datoteka, njihovim podrazumevanim *mount-point* direktorijumima i podrazumevanim opcijama koje se koriste prilikom aktiviranja. Opcije koje se mogu navesti u datoteci /etc/fstab, a koje se tiču sigurnosti su:

- defaults – na sistemu datoteka je dozvoljeno sve (kvote, upis i suid),

- quota – dozvoljava se primena kvota,
- noquota – zabranjuje se primena kvota,
- suid – SUID i SGID bitovi postavljeni na datotekama imaće efekta,
- nosuid – SUID i SGID bitovi postavljeni na datotekama neće imati efekta,
- nodev – specijalne datoteke se ne interpretiraju kao uređaji,
- noexec – zabranjuje se izvršavanje datoteka bez obzira na pravo "x",
- ro – dozvoljava pristup sistemu datoteka isključivo u režimu čitanja,
- rw – dozvoljava pristup sistemu datoteka u režimu za čitanje i upis.

Pomoću navedenih opcija lako se može povećati sigurnost aktiviranih sistema datoteka. Na primer: direktorijum /boot je mesto na kome se čuva jezgro operativnog sistema. Na velikom broju Linux sistema ovaj direktorijum se nalazi na posebnom sistemu datoteka, koji se podrazumevano aktivira u režimu čitanja i pisanja. Ovo je radi povećanja sigurnosti potrebno promeniti, odnosno dozvoliti aktiviranje /boot sistema datoteka isključivo u režimu čitanja. Najpre je potrebno u datoteci /etc/fstab načiniti sledeću izmenu:

```
/dev/hda2 /boot ext2 defaults,ro 1 2
```

Nakon toga se ovaj sistem datoteka mora ponovo aktivirati kako bi promene postale validne:

```
# mount /dev/hda2 -oremount
```

Administrator može proveriti da li je boot sistem datoteka sada aktiviran u režimu čitanja komandom cat /proc/mounts koja prikazuje sve aktivne sisteme datoteka i opcije sa kojima su aktivirani:

```
# cat /proc/mounts
/dev/hda1 / ext2 rw 0 0
/proc /proc proc rw 0 0
/dev/hda2 /boot ext2 ro 0 0
/dev/hdb1 /home ext2 rw,nodev 0 0
/dev/hdb2 /tmp ext2 rw,nodev,noquota 0 0
none /dev/pts devpts rw 0 0
```

9.4 Linux na mreži

Prethodna razmatranja odnose se na svaki Linux sistem – umrežen ili neumrežen. Ukoliko se Linux nalazi na mreži, onda je poželjno da primenite neke dodatne mehanizme zaštite, kao što je instalacija i konfigurisanje iptables mrežne barijere (opisana u poglavlju 10). Sledeće zaštitne mere takođe mogu biti korisne. Čitaocu se preporučuje da, ukoliko nije upoznat sa osnovnim pojmovima vezanim za računarske mreže, TCP/IP skup protokola i IP adresiranje i konfiguraciju Linux mrežnog okruženja, najpre pročita uvodni deo poglavlja o mrežnim barijerama.

Datoteke /etc/hosts.allow i /etc/hosts.deny

Datotekama /etc/hosts.allow i /etc/hosts.deny određuje se kojim računarima je pristup lokalnom računaru dozvoljen, odnosno zabranjen. Datoteke su realizovane u vidu liste, pri čemu se u jednoj liniji navodi ime jednog računara, domena ili ALL-EXCEPT konstrukcija. Na primer, sledeća datoteka /etc/hosts.allow dozvoljava pristup svim računarima domena mydomain.org i svim računarima domena yourdomain.com, osim računaru spyserver.

```
ALL: .mydomain.org
ALL: .yourdomain.com EXCEPT spyserver.yourdomain.com
```

xinetd – praćenje aktivnosti i kontrola pristupa

Mrežni servisi prihvataju zahteve za uspostavljanje konekcije na određenom portu. Imena servisa su određena datotekom /etc/hosts, koja povezuje servis sa odgovarajućim protokolom (TCP ili UDP) i brojem porta. Funkcionisanje većine servisa je pod kontrolom *wrapper daemon* programa, kao što su inetd (Internet dispatch Daemon) i xinetd. Na Linux sistemima postoji veliki broj mrežnih servisa koji pružaju različite usluge korisnicima sistema:

- otvaranje udaljene interaktivne sesije (telnet) - zahtevi za otvaranjem telnet sesije šalju se na TCP port 23, a prihvataju ih telnetd ili

odgovarajući wrapper programi (inetd ili xinetd),

- transfer datoteka (ftp - file transfer protocol) - zahtevi za otvaranjem ftp sesije šalju se na TCP port 20, a prihvataju ih ftpd ili odgovarajući wrapper programi,
- zahtev za razrešavanjem imena (DNS) - šalje se na UDP port 53,
- pristup Web stranicama preko http protokola (servis obezbeđuje hppd iz Apache paketa),
- pristup elektronskoj pošti (servis obezbeđuju Sendmail i Postfix),
- pristup mrežnom sistemu datoteka (NFS),
- centralizovanu autentifikaciju (NIS).

Program xinetd osluškuje zahteve za uspostavljanjem konekcije i prima UDP datagrame, nakon čega pokreće odgovarajući servis i predaje mu kontrolu nad zahtevom. Konfiguracione datoteke programa xinetd su:

- /etc/xinetd.conf – xinetd globalna konfiguraciona datoteka
- /etc/xinetd.d/ime_servisa – direktorijum u kome se nalaze sve informacije o konkretnom servisu

Datoteka /etc/xinetd.conf sadrži podešavanja koja se tiču svih servisa koji su pod kontrolom programa xinetd. Datoteka se čita samo prilikom pokretanja xinetd programa. Ukoliko se sadržaj datoteke modifikuje, xinetd se mora ponovo pokrenuti kako bi izmene bile validne. U nastavku teksta dat je isečak datoteke /etx/xinetd.conf:

```
defaults
{
   instances        = 60
   log_type         = SYSLOG authpriv
   log_on_success   = HOST PID
   log_on_failure   = HOST
   cps              = 25 30
}
includedir /etc/xinetd.d
```

Linije u datoteci imaju sledeće značenje:

- instances – određuje maksimalan broj zahteva kojima xinetd može

rukovati istovremeno,

- log_type – određuje tip praćenja aktivnosti; ovako konfigurisan, dnevnik generiše syslog u datoteci /var/log/secure[52],
- log_on_success – u slučaju uspešnog ostvarivanja konekcije, u dnevnik se upisuje IP adresa računara s kojim je konekcija ostvarena i PID servisa koji upravlja konekcijom,
- log_on_failure – u slučaju neuspešnog ostvarivanja konekcije, u dnevnik se upisuje IP adresa računara koji je pokušao da ostvari konekciju,
- cps – xinetd će dozvoliti najviše 25 konekcija u sekundi ka svakom servisu. Kada se dostigne limit, servis postaje nedostupan za otvaranje novih konekcija narednih 30 sekundi.

Direktiva includedir /etc/xinetd.d/ specificira xinetd programu da pročita sve konfiguracione datoteke u /etc/xinetd.d direktorijumu. U ovim datotekama su navedeni konfiguracioni parametri specifični za konkretne servise kojima upravlja xinetd. Imena datoteka u /etc/xinetd.d su u korelaciji sa imenima servisa. Kao i datoteka /etc/xinetd.conf, i ove datoteke se čitaju samo prilikom pokretanja programa xinetd; ukoliko se njihov sadržaj modifikuje, xinetd se mora ponovo pokrenuti da bi izmene bile validne. Format datoteka u direktorijumu /etc/xinetd.d/ sličan je formatu datoteke xinetd.conf. Osnovni razlog zbog koga se konfiguracija svakog servisa čuva u posebnoj datoteci je mogućnost postavljanja specifičnih parametara za određeni servis koji neće imati uticaja na druge servise.

Na sistemima na kojima funkciju obvojnice mrežnih servisa (engl. *wrapper*) obavlja servis xinetd, kontrola pristupa sa udaljenih sistema može se obaviti sledećim metodama[53]:

- korišćenjem datoteka /etc/hosts.allow i /etc/hosts.deny (metod koji koristi i inetd obvojnica),
- korišćenjem xinetd konfiguracionih datoteka,

52 Ukoliko se navede direktiva log_type = FILE /var/log/xinetdlog, dnevnik će se čuvati u datoteci /var/log/xinetdlog

53 Sve promene u kontroli pristupa, kao i sve druge promene u xinetd konfiguracionim datotekama, postaju validne posle ponovnog pokretanja xinetd daemon procesa.

- kombinovanim korišćenjem ovih metoda (primenjuje se unija restrikcija).

Za razliku od datoteka hosts.allow i hosts.deny, kojima se reguliše pristup svim servisima, xinetd dozvoljava da se za svaki servis u odgovarajućoj datoteci definiše sa kog se računara može, odnosno ne može pristupiti servisu. Xinetd opcije koje se tiču kontrole pristupa su sledeće:

- only_from – pristup servisu se dozvoljava samo sa navedenih računara (ili IP mreža), pri čemu se računari (mreže) navode pomoću imena ili IP adresa,
- no_access – zabranjuje se pristup servisu sa navedenih računara (ili IP mreža), pri čemu se računari (mreže) navode pomoću imena ili IP adresa,
- access_times - specificira se vreme kada se specifični servis sme koristiti. Vreme se navodi u 24-časovnom formatu HH:MM-HH:MM.

Sledeći primer ilustruje datoteku /etc/xinetd.d/telnet kojom se blokira pristup Telnet servisu sa mreže čija je IP adresa 123.45.0.0, a svim ostalim dozvoljava upotreba Telnet servisa u periodu 8:15-17:30h.

```
service telnet
{
        disable         = no
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
        no_access       = 123.45.0.0
        log_on_success  += PID HOST EXIT
        access_times    = 08:15-17:30
}
```

Klijentu sa mreže 123.45.0.0, koji pošalje zahtev za otvaranjem Telnet sesije, sistem šalje sledeću poruku:

```
Connection closed by foreign host.
```

Dodatno, pokušaj prijavljivanja na sistem preko telnet sesije se beleži u

datoteci /var/log/secure:

```
Sep 15 14:18:42 boo xinetd[11002]: START: telnet pid=11006
from=166.60.25.20
Sep 15 14:18:42 boo xinetd[11006]: FAIL: telnet address
from=166.60.25.20
Sep 15 14:18:42 boo xinetd[11002]: EXIT: telnet status=0
pid=16256
```

Datoteka /etc/exports

Svaki sistem čiji je deo aktivnog stabla dostupan na mreži je NFS server. Na NFS serveru se definiše koji su direktorijumi dostupni na mreži (export-points) i sa kojih se računara može tim direktorijumima pristupiti. Jednostavno rečeno, definiše se sadržaj NFS sistema datoteka, odnosno "deljeni direktorijumi" i kontrola pristupa. Na BSD UNIX i Linux sistemima sadržaj NFS sistema datoteka se definiše u datoteci /etc/exports. Ovu datoteku čita program exportfs koji ažurira sadržaj tekućeg NFS servera. U nastavku teksta dat je primer datoteke /etc/exports:

```
/projects       proj*.conwex.org(rw)
/usr            *.conwex.org(ro) @trusted(rw)
/home/joe       pc001(rw,all_squash,anonuid=150,anongid=100)
/pub            (ro,insecure,all_squash)
```

Svaka linija sadrži putanju i ime direktorijuma i prateću listu opcija. Korisnik root može ažurirati sadržaj tekućeg NFS servera (pokretanjem komande exportfs -a) ili zaustaviti deljenje nekog direktorijuma. NFS se sa udaljenih računara (klijenata) aktivira kao i svaki drugi sistem datoteka – montiranjem na *mount-point* direktorijume:

```
# mount -t nfs fileserver1://share/doc /doc
```

Osnovna razlika u odnosu na lokalne sisteme datoteka je u tome što NFS dozvoljava da se: montira ceo deljeni direktorijum ili deo deljene hijerarhije, odnosno neki poddirektorijum. Time je, na primer, omogućena realizacija home direktorijuma korisnika putem NFS sistema datoteka.

Što se NFS sigurnosti tiče, datoteka /etc/exports mora biti konfigurisana sa najrestriktivnijim dozvolama, što podrazumeva sledeće:

- zabranite root pristup preko NFS montiranih direktorijuma, tj. ne koristite opciju no_root_squash u datoteci /etc/exports. Vaš NFS server će root korisnika koji pristupa udaljeno tretirati kao običnog korisnika, osim ako neki direktorijum izvezete sa opcijom no_root_squash. U tom slučaju će on ostvariti root pristup NFS direktorijumu.
- dozvolite aktiviranje NFS sistema datoteka isključivo u režimu čitanja kad god je to moguće

Na primer, sledeća datoteka /etc/exports dozvoliće pristup direktorijumu /share/project sa svih računara iz domena mydomain.com isključivo u režimu čitanja (opcija ro). Root korisniku koji direktorijumu pristupa preko mreže će takođe biti zabranjen upis (root_squash).

```
/share/project mydomain.com (ro,root_squash)
```

Datoteke .rhosts

Datoteke ~/.rhosts obezbeđuju mehanizam udaljene autentifikacije za rlogin, lpd i rsh procese. Svaka .rhosts datoteka specificira računare i korisnike kojima se veruje. Tim korisnicima je dozvoljen pristup sistemu bez navođenja lozinke. Kao takve, .rhosts datoteke predstavljaju ozbiljan sigurnosni problem svakog servera; uklanjanje .rhosts datoteka sa neadekvatnim sadržajem je važan postupak. Iako nakon instalacije Linux sistema, ovih datoteka obično nema u aktivnom stablu, pretraga za .rhosts datotekama spada u regularne periodične administrativne poslove i treba se automatizovati. Na primer, u direktorijumu /etc/cron.daily može se kreirati skript datoteka rhosts_audit sa sledećim sadržajem:

```
find /home -name .rhosts | mail -s "rhosts file report" root
```

Zatim skript teba učiniti izvršnim i dodeliti mu vlasnika i grupu root:

```
# chmod 755 /etc/cron.daily/rhosts_audit
# chown root /etc/cron.daily/rhosts_audit
# chgrp root /etc/cron.daily/rhosts_audit
```

Nakon toga, root će dobijati jednom dnevno mail sa izveštajem o pronađenim rhosts datotekama. Root nakon toga može proveriti sadržaj i po

potrebi obrisati te datoteke.

Zaštita Apache Web servera

Web sajtovi su jedna od omiljenih meta napadača, s obzirom na to da obično imaju veliku bazu korisnika, i samim tim pružaju mogućnost sticanja publiciteta na mreži. U ozbiljnije posledice napada na Web sajtove spadaju zloupotreba tuđeg imena, neovlašćena upotreba važnih informacija, krađa značajnih podataka (kao što su brojevi kreditnih kartica članova nekog kluba). Naročito su u opasnosti sajtovi sa dinamičkom sadržinom (CGI, PHP) koja na određeni način vrši interakciju sa korisničkim računarom.

Neki problemi potiču od samog operativnog sistema koji predstavlja okruženje na kojem se Apache izvršava. U opštem slucaju, Apache je pokrenut od strane root korisnika, a zahteve opsužuje kao korisnik koji je definisan User direktivom (na primer, korisnik www). Kao što je slučaj sa svakom komandom koju izvršava root, potrebno je sprečiti modifikaciju sadržaja programa od strane neprivilegovanih korisnika. Dozvolu za upis u direktorijume, poddirektorijume i datoteke sme imati samo root. Na primer, ako je za ServerRoot izabran /usr/local/apache onda se preporučuje da root pre instalacije samog Apache paketa kreira direktorijume na sledeći način:

```
# mkdir /usr/local/apache
# cd /usr/local/apache
# mkdir bin conf logs
# chown 0 . bin conf logs
# chgrp 0 . bin conf logs
# chmod 755 . bin conf logs
```

Nakon instalacije httpd izvršne datoteke, potrebno je preduzeti slične mere zaštite:

```
# cp httpd /usr/local/apache/bin
# chown 0 /usr/local/apache/bin/httpd
# chgrp 0 /usr/local/apache/bin/httpd
# chmod 511 /usr/local/apache/bin/httpd
```

Takođe, može se kreirati poddirektorijum htdocs, u kom i drugi korisnici mogu vršiti izmene, s obzirom na to da root nikad ne izvršava datoteke sa te

lokacije, niti ih na toj lokaciji kreira.

Prethodno navedeni koraci zabranjuju neprivilegovanim korisnicima da menjaju datoteke u navedenim direktorijuma. Ukoliko se ovi koraci preskoče, mogući su sledeći scenariji zloupotrebe:

- zamena httpd izvršne datoteke drugom datotekom, koja može da izvede zlonamerne akcije prilikom pokretanja,
- dozvola upisa u log poddirektorijum omogućava nekom neprivilegovanom korisniku da log datoteku zameni linkom na neku sistemsku datoteku, koju superuser može nepažnjom da obriše,
- ako je upis u log datoteke dozvoljen, moguće je u njih uneti lažne podatke, sakrivajući time tragove nekih drugih akcija[54].

Korišćenje .htaccess datoteka predstavlja dodatnu zaštitnu meru. Pri tome, potrebno je zabraniti korišćenje svih .htaccess datoteka, osim onih čije je korišćenje eksplicitno dozvoljeno.

```
<Directory />
AllowOverride None
</Directory>
```

Sledeća sekcija ce onemoguciti navigaciju korisnika kroz strukturu direktorijuma u sistemu datoteka:

```
<Directory />
Order Deny,Allow
Deny from all
</Directory>
```

Pored ovih osnovnih, ali veoma bitnih mera zaštite, potrebno je stalno pratiti rad web servera pregledanjem log datoteka. Iako ove datoteke prikazuju podatke samo o onome što se već dogodilo, njihovim pregledanjem može se steći predstava o najčešće izvođenim tipovima napada na web server i ideja o protivmerama zaštite koje treba poduzeti.

Na primer, neka je u access.log datoteci prisutna linija slična sledećoj:

54 Specijalni atribut append only (postavlja se komandom chattr +a) rešava ovaj problem, međutim, samo ako se log datoteka nalazi na ext2 ili ext3.

```
www.intrudersgrotto.com - [1/Apr/2004:10:29:59 +0100] "GET
/.htpasswd HTTP/1.1"
```

To znači da postoji sigurnosni problem sa .ht* datotekama. Neovlašćeni korisnici ne smeju preuzeti ove datoteke posredstvom web servera, tako da se kao protivmera u httpd.conf unosi sledeće:

```
<Files ~ "^\.ht">
Order allow,deny
Deny from all
<Files>
```

Nakon toga će pokušaj preuzimanja ovih datoteka prouzrokovati upis linije slične sledećoj u access.log datoteci:

```
[Thu Apr 1 11:01:39 2004] [error] [client www.intruders
grotto.com] client denied by server configuration:
/usr/local/apache/htdocs/.htpasswd
```

"chroot jail"

Najekstremnija sigurnosna mera podrazumeva smeštanje Apache-a u takozvani chroot zatvor (engl. *chroot jail*). Time se ograničava područje sistema datoteka koje je dostupno Apache web serveru. Ovim premeštanjem se kreira nova struktura direktorijuma koja sadrži samo Apache, njemu potrebne datoteke i minimalan broj drugih programa. Bilo kakav sigurnosni propust u samom Apacheu može da ugrozi samo ovu strukturu, bez naročitog uticaja na ostatak sistema.

Kreiranje *chroot jail* okoline za Apache izvodi se u nekoliko koraka:

- dodavanje novog korisnika (www, UID=80) i grupe (www, GID=80)
- kreiranje chroot direktorijumske strukture

```
# mkdir /chroot/httpd
# mkdir /chroot/httpd/dev
# mkdir /chroot/httpd/lib
# mkdir /chroot/httpd/etc
# mkdir /chroot/httpd/home
# mkdir /chroot/httpd/tmp
```

```
# chmod 777 /chroot/httpd/tmp/
# chmod +t /chroot/httpd/tmp/
# mkdir -p /chroot/httpd/usr/sbin
# mkdir -p /chroot/httpd/var/run
# mkdir /chroot/httpd/var/log
```

- prebacivanje konfiguracione i izvršne datoteke u novu chroot strukturu i kreiranje specijalnih datoteka /dev/null i /dev/urandom koje su potrebne za normalno funkcionisanje sistema

```
# mv /etc/httpd /chroot/httpd/etc/
# mv /home/httpd /chroot/httpd/home/
# mv /var/log/httpd /chroot/httpd/var/log/
# mv /usr/sbin/httpd /chroot/httpd/usr/sbin/
# mknod /chroot/httpd/dev/null c 1 3
# chmod 666 /chroot/httpd/dev/null
# mknod /chroot/httpd/dev/urandom c 1 9
```

- pronalaženje deljenih biblioteka koje httpd koristi

```
# ldd /chroot/httpd/usr/sbin/httpd
libpam.so.0 => /lib/libpam.so.0 (0x4001b000)
libdl.so.2 => /lib/libdl.so.2 (0x40023000)
...
libjpeg.so.62 => /usr/lib/libjpeg.so.62 (0x4024a000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

- kopiranje potrebnih bibilioteka na chroot lokacije

```
# cp /lib/libpam.so.0 /chroot/httpd/lib/
# cp /lib/libdl.so.2 /chroot/httpd/lib/
...
# cp /usr/lib/libjpeg.so.62 /chroot/httpd/usr/lib/
# cp /lib/ld-linux.so.2 /chroot/httpd/lib/
# strip -R .comment /chroot/httpd/usr/lib/*
```

- kopiranje passwd i group datoteka u kojima se ostavljaju samo www korisnik i www grupa

```
# cp /etc/passwd /chroot/httpd/etc/
# cp /etc/group /chroot/httpd/etc/
# vi /chroot/httpd/etc/passwd
# vi /chroot/httpd/etc/group
# less /chroot/httpd/etc/passwd
```

```
www:x:80:80:Apache Server:/home/httpd:/bin/false
# less /chroot/httpd/etc/group
www:x:80:
```

- kopiranje konfiguracionih datoteka /etc/hosts, /etc/resolv.conf, /etc/nsswitch.conf i /etc/localtime

```
# cp /etc/resolv.conf /chroot/httpd/etc/
# cp /etc/nsswitch.conf /chroot/httpd/etc/
# cp /etc/localtime /chroot/httpd/etc/
# cp /etc/hosts /chroot/httpd/etc/
```

- promenu syslog i httpd skriptova

```
# vi /etc/rc.d/init.d/syslog
### daemon syslogd -m 0
daemon syslogd -m 0 -a /chroot/httpd/dev/log
# vi /etc/rc.d/init.d/httpd
### daemon httpd
/usr/sbin/chroot /chroot/httpd/ /usr/sbin/httpd
### rm -f /var/run/httpd.pid
rm -f /chroot/httpd/var/run/httpd.pid
```

Nakon toga je potrebno proveriti rad Apache servera u chroot-jail okolini:

```
# /etc/rc.d/init.d/syslog restart
# /etc/rc.d/init.d/httpd start
```

9.5 Praćenje događaja i nadzor sistema

Praćenje događaja (engl. *auditing*) i nadzor sistema (engl. *monitoring*) su veoma važni zadaci u administraciji mrežnog servera ukolko želite da obezbedite visok nivo sigurnosti. Tragovi uspešnih i neuspešnih napada na sistem beleže se u sistemskim log datotekama – realno, retko ko može napasti tuđi sistem tako da za njim ne ostane nikakav trag. Redovno pregledanje dnevnika predstavlja jedan od fundamentalnih koraka detekcije napada. Svaki administrator u log datotekama traži znakove učestalih neuspešnih pokušaja autentifikacije, neobrazloživog obaranja sistema (komanda shutdown) i sličnh događaja koji su neobični ili neočekivani. Na primer, jedan neuspešni pokušaj prijavljivanja na sistem može se objasniti

pogrešno unešenom lozinkom. Nekoliko neuspešnih pokušaja prijavljivanja sa istim korisničkim imenom mogu se objasniti sindromom "debelog prsta". Međutim, uzastopni pokušaji prijavljivanja sa različitim korisničkim imenom (ili sa imenom root) su očigledan pokazatelj napada na sistem.

Praćenje događaja – syslog alat

Linux nudi administratorima centralizovani sistem za praćenje događaja – syslog. Većina Linux distribucija nakon instalacije automatski konfiguriše syslog, shodno podrazumevanim podešavanjima za tu distribuciju i pokreće syslogd proces. Sistem administrator specificira u datoteci /etc/syslog.conf koje poruke želi da zabeleži, a syslog preusmerava poruke u odgovarajuću log datoteku. Linux najčešće generiše nekoliko log datoteka (koje rotira[55] pomoću /etc/logrotate.d/syslog skripta; parametri rotacije se definišu u datoteci /etc/logrotate.conf) i u njima beleži sledeće događaje:

- /var/log/messages – najveći broj sistemskih događaja
- /var/log/secure – neuspeli pokušaji autentifikacije klijenata ka mrežnim servisima koje je pokrenula inetd mrežna obvojnica
- /var/log/maillog – događaji vezani za sendmail server (preopširni)
- /var/log/spooler – događaji vezani za ostale servise (UUCP, NNTP)
- /var/log/boot.log – poruke koje se prikazuju na ekranu pri podizanju operativnog sistema

Format svake linije u datoteci syslog.conf je sledeći:

```
izvporuke.prioritet[;izvporuke.prioritet][TAB]akcija
```

Parametar izvporuke se odnosi na podsistem koji generiše poruku koju želite da zabeležite. U ove podsisteme, između ostalog, spadaju:

- auth, authpriv – sigurnosni događaji i autentifikacija,

55 Rotacija log datoteka znači promenu imena trenutne log datoteke logfile u logfile.1 i kreiranje prazne datoteke logfile u kojoj se čuvaju informacije o događajima od poslednje rotacije. U sledećoj rotaciji sistem će preimenovati datoteku logfile.1 u logfile.2, datoteku logfile u logfile.1 i kreirati novu praznu datoteku logfile. Zavisno od toga koliko log datoteka želite da čuvate, poslednja datoteka (npr. logfile.11) se briše.

- cron – događaji vezani za cron raspoređivač poslova,
- daemon – događaji vezani za razne servise, tj. *daemon* procese,
- kern – poruke iz Linux jezgra,
- lpr – događaji vezani za podsistem za štampu,
- mail – događaji vezani za podsistem za elektronsku poštu,
- uucp – događaji vezani za Unix-to-Unix Copy podsistem,
- sysloguser – interne syslog poruke.

Polje prioritet ukazuje na prioritet, tj. vrstu poruke iz specificiranog podsistema. Navešćemo moguće vrednosti ovog parametra, uređene po prioritetu (tj. prema nivou kritičnosti) u rastućem redu:

- debug, info – poruke informacionog karaktera,
- notice – normalne poruke od većeg značaja,
- warning – poruke koje ukazuju na moguć otkaz podsistema,
- error, crit – poruke o greškama i kritičnim greškama,
- alert – poruke o stanju koje zahteva hitnu intervenciju administratora,
- emmerg – poruke o grešama koje će dovesti do kraha sistema.

Simbol "*" može se koristiti kao džoker koji menja bilo koji prioritet. Parametar akcija određuje odredište za smeštaj poruke određene selektorom. Ovaj parametar može biti regularna datoteka u koju će se smestiti poruka, terminal ili konzola, imenovani međuprocesni kanal (engl. *named pipe*, poruke se koriste kao standarni ulaz), udaljeni računar (poruke se šalju syslogd servisu udaljenog računara) ili lista korisnika (poruke se šalju na sve terminale sa kojih su ti korisnici prijavljeni).

Navodimo nekoliko mogućih linija ove konfiguracione datoteke (koje pokrivaju najčešće korišćene slučajeve) sa objašnjenjem.

- Sve poruke sa prioritetom crit (kritične greške), osim kritičnih grešaka jezgra, čuvaju se u datoteci /var/adm/critical:

```
*.=crit;kern.none              /var/adm/critical
```

- Poruke jezgra sa prioritetom crit ili višim prikazuju se na konzoli, smeštaju u datoteku /var/adm/kernel-critical i šalju udaljenom računaru logserver. Sve informativne poruke jezgra (prioriteta info. notice i warning) smeštaju se u datoteku /var/adm/kernel-info:

```
kern.crit              @logserver
kern.crit              /dev/console
kern.crit              /var/adm/kernel-critical
kern.info;kern.!err    /var/adm/kernel-info
```

- Poruke vezane za mail podsistem, prioriteta manjeg od warning smeštaju se u datoteku /var/adm/mail. Poruke vezane za mail i news podsistem prioriteta debug, info ili notice beleže se u datoteci /var/adm/info:

```
mail.*;mail.warning    /var/adm/mail-err
mail,news.!warning     /var/adm/mail-info
```

- Sve poruke (osim poruka mail i news podsistema) prioriteta info i notice beleže se u datoteci /var/log/messages:

```
*.=info;*.=notice; mail,news.none    /var/log/messages
```

- Sve poruke sa prioritetom emerg prikazuju se svim korisnicima pomoću komande wall (Write All):

```
*.=emerg    *
```

- Sve poruke sa prioritet alert ili emerg prikazuju se na terminalima sa kojih su prijavljeni korisnici root i johndoe:

```
*.alert     root,johndoe
```

Zaštita syslog servera

Syslog, kao i većina mrežnih servisa, predstavlja pogodnu tačku za napad na sistem. Direktiva iz datoteke /etc/syslog.conf koja ima oblik: "izvporuke.prioritet @logserver" ukazuje lokalnom syslogd procesu da poruke o greškama šalje udaljenom syslog serveru logserver na UDP port 514. Ovakav način vođenja dnevnika ima svoje prednosti, ali istovremeno

otkriva mogućnost napada na sistem na kom se prate događaji. Na primer, moguće je najpre izvršiti DoS napad na syslog server slanjem velikih datoteka na UDP port 514. Ukoliko se disk syslog servera napuni a sve poruke sistema, koji je meta napadača, beleže na serveru, napadač će ostati neprimećen.

Zbog toga je potrebno obezbediti syslog servere na sledeći način:

- uvodi se kontrola pristupa UDP portu 514 – upis se dozvoljava samo autorizovanim računarima čiji rad želimo da pratimo,
- odvojite syslog server na poseban mrežni segment ukoliko je to moguće,
- direktorijume u kojima se smeštaju log datoteke realizujte kao odvojene sisteme datoteka; čak i ako se napune, ostatak syslog servera neće pretrpeti posledice tog prepunjenja.

Nadzor sistema – pregledanje log datoteka

Najveća mana syslog alata je obim log datoteka – log datoteke su isuviše detaljne, a samim tim i isuviše velike, da bi bile od neke koristi. Na primer, na srednje opterećenom serveru datoteka /var/log/messages se dnevno uveća za oko 500KB (što je isuviše velika količina teksta za analizu). Zbog toga relativno mali broj sistem administratora ume da iskoristi usluge ovog alata. U svakom slučaju, "ručno" pregledanje log datoteka ima dva nedostatka:

- zadatak je isuviše komplikovan i mora ga obaviti osoba koja je obučena da prepozna znakove napada (ovakve usluge su obično veoma skupe),
- ljudski faktor – ljudi su skloni greškama, naročito kada je u pitanju velika količina teksta čiji se delovi mnogo puta ponavljaju.

Zbog toga su nastali posebni alati koji analiziraju log datoteke i u njima traže određene ključne reči ili šablone koji ukazuju na moguće napade. Ovi programi zahtevaju da administrator najpre definiše log datoteke od značaja i skup okidača (engl. *triggers*), tj. regularnih izraza koji se traže u log datotekama. Program izvršava specificiranu akciju (slanje e-mail poruke

administratoru ili izvršenje neke komande) onda kada pronađe okidač u log datoteci. Primer takvih programa su swatch i logcheck.

Swatch (Simple WATCHer) je alat koji se sastoji od Perl skripta koji obavlja nadzor i konfiguracione datoteke (najčešće .swatchrc) u kojoj možete definisati okidače i skup akcija koje se izvršavaju kada swatch nađe taj okidač u log datoteci. Navodimo nekoliko primera konfiguracije swatch alata.

- Swatch će na svakih 30 minuta (direktiva throttle) poslati poruku o neuspešnim pokušajima autentifikacije[56] u proteklih 30 minuta sistem administratoru (admin@conwex.org) i u poruci priložiti isečak log datoteke (echo) koji se odnosi na te pokušaje:

```
watchfor /authentication failure/
  echo
  throttle 00:30
  mail=admin@conwex.org,subject=Neuspešna_autentifikacija
```

- Swatch će u slučaju punog sistema datoteka poslati poruku sistem administratoru; poruka će takođe biti poslata osobi koja održava sistem u periodu od 17-21h radnim danima (backadmin@conwex.org):

```
watchfor /filesystem full/
  echo
  mail=admin@conwex.org,subject=Fajl_sistem_je_pun
  mail=backadmin@conwex.org,when=1-5:17-21
```

- Swatch će u slučaju punog sistema datoteka poslati poruku sistem administratoru i pokrenuti skript CleanUp_FS:

```
watchfor /filesystem full/
  echo
  mail=admin@conwex.org,subject=Čistim_stare_datoteke
  exec "Clean_Up_FS"
```

Nakon konfigurisanja potrebno je pokrenuti swatch:

```
swatch –config-file=~/.swatchrc –tail-file=/var/log/messages
```

56 Okidač za događaj je regularni izraz – u ovom slučaju niz karaktera "authentication failure"

Program logcheck nakon instalacije upisuje u direktorijum /etc/logcheck nekoliko datoteka koje koristi za traženje uzoraka u log datotekama. Jedna od datoteka sa najzanimljivijim sadržajem je logcheck.hacking, u koju je autor smestio regularne izraze koji opisuju dokumentovane napade na Linux servere, kao na primer:

```
kernel: Oversized packet received from
login.*: .*LOGIN FAILURE.* FROM .*root
```

U istom direktorijumu se nalazi i datoteka logcheck.violations koja sadrži regularne izraze koji opisuju sumnjive situacije koje se ne mogu definisati kao napad u toku. Na primer, linija “RETR passwd” ukazuje na pokušaj da je neko pokušao da preuzme /etc/passwd datoteku preko FTP protokola. Ovo nije akcija koja direktno ukazuje na napad jer je nepoznato šta osobe koja je tu datoteku preuzela želi sa njom da radi.

Logcheck sa podrazumevanom konfiguracijom proverava poruke u datoteci /var/log/messages, tako da je potrebno da obezbedite da syslog beleži sve poruke u toj datoteci[57]. Logcheck nema nikakvu konfiguracionu datoteku, ali je sam skript prilično razumljiv i na samom početku ima nekoliko promenljivih čije vrednosti možete izmeniti. Na primer, preporučuje se da vrednost promenljive SYSADMIN promenite u ime korisnika koji treba da primi izveštaje koje generiše logcheck. Logcheck se oslanja na cron za izvršavanje periodične provere log datoteka. Nakon instalacije logcheck RPM distribucije u direktorijumu /etc/cron.hourly/ smešta se datoteka logcheck, što znači da se provera log datoteka obavlja na svakih sat vremena.

9.6 Pitanja i zadaci

1. Prijavite se na sistem kao root korisnik i kreirajte novi korisnički nalog "proba". Postavite parametre lozinke naloga “proba” tako da:
   - nalog nikad ne ističe, ali se zaključava nakon 7 dana neaktivnosti,
   - korisnik mora da promeni lozinku na svakih 40 dana,
   - 10 dana nakon promene lozinke korisnik ne sme ponovo da menja

57 Izmenite sadržaj datoteke /etc/syslog.conf u *.info /var/log/messages

lozinku,

- 20 dana nakon promene lozinke korisnik dobija upozorenje da će mu lozinka isteći za 20 dana.

2. Ograničite prijavljivanje root korisnika na sistem isključivo na sistemske konzole.

3. Kreirajte tri nova korisnička naloga "korisnik1", "korisnik2" i "korisnik3" i dodelite im sledeća ovlašćenja:
   - sva tri korisnika mogu da ugase računar komandom shutdown,
   - korisnik1 može da menja vlasničke odnose nad datotekama,
   - korisnik2 može da pokrene bilo koju komandu sa root privilegijama.

4. Prijavite se na sistem kao korisnik "proba" i kreirajte nekoliko datoteka. Odjavite se sa sistema i prijavite kao root. Uklonite nalog "proba" sa sistema. Pronađite sve datoteke bez vlasnika i obrišite ih.

5. Izmenom podataka u datoteci /etc/fstab zabranite korisnicima da koriste cd-rom uređaj.

6. Zabranite pristup telnet servisu na vašem računaru sa mreže 172.16.0.0. Obezbedite praćenje događaja o odbijenim konekcijama.

7. Kreirajte jedan direktorijum i objavite ga kao NFS export na mreži. Dozvolite aktiviranje NFS direktorijuma isključivo sa domena conwex.org i obezbedite da root udaljenog računara ne nasledi lokalne root privilegije nad tim direktorijumom.

8. Objasnite princip funkcionisanja chroot zatvora.

9. Probajte nekoliko puta da se prijavite na sistem koristeći pogrešnu lozinku. Pronađite u log datotekama zapise koji ukazuju na neuspešne pokušaje autentifikacije. Instalirajte swatch i obezbedite da nakon svake neuspešne autentifikacije root korisnika swatch obavesti korisnika johndoe e-mail porukom o događaju.

10

Mrežne barijere

10.1 Osnovni pojmovi o računarskim mrežama

Računarska mreža omogućava komunikaciju međusobno povezanih autonomnih računara. Da bi se formirala računarska mreža, potrebna su najmanje dva uređaja (radne stanice, štampači ili serveri) povezana bakarnim ili optičkim kablovima, ili bežičnom vezom, u cilju razmene informacija i/ili deljenja resursa (izlazni i ulazni uređaji, masovni memorijski medijumi, podaci i aplikacije, veza sa Internetom). Lokalne računarske mreže (engl. *Local area network,* LAN) povezuju računare (radne stanice i servere) i periferne uređaje na ograničenom geografskom području, poput jedne ili više zgrada. Komunikacija između uređaja na različitim lokalnim mrežama ostvaruje se vezivanjem lokalnih mreža na mreže širokog područja, tj. WAN mreže (engl. *wide area network*), kao što je Internet. WAN mreže pokrivaju geografska područja veličine grada i države.

Mrežno-komunikacioni zadaci su OSI referentnim modelom podeljeni na sedam manjih, lakše upravljivih celina (slojeva), od kojih svaki definiše određene funkcije mreže i može se zameniti drugim slojem istog nivoa, bez uticaja na ostatak sistema. Prednosti upotrebe OSI modela su smanjenje složenosti, standardizacija interfejsa, sprečavanje uticaja promene jednog sloja na druge slojeve (čime je olakšan razvoj pojedinačnih funkcija) i omogućavanje lakšeg izbora pravog mrežnog uređaja za željenu namenu. Funkcije slojeva su sledeće: aplikacioni sloj – pristup aplikacije mrežnom okruženju, prezentacioni sloj – formatiranje podataka, sloj sesije – dijalog između računara, transportni sloj – logička veza s kraja na kraj i kontrola toka, mrežni sloj – logičko adresiranje i rutiranje, sloj veze – fizičko adresiranje, kontrola toka i pristup medijumu, fizički sloj – prenos bitova.

Ukoliko sistemi nisu direktno povezani transmisionim medijumom, koriste se posredni uređaji, koji pripadaju kategoriji otvorenih sistema i implementiraju niže slojeve OSI modela. U zavisnosti od slojeva koji su implementirani, za te uređaje će od značaja biti bitovi, ramovi (engl. *frames*) i paketi (engl. *packets*), ali ne i jedinice podataka viših slojeva. To određuje funkciju uređaja: fizički sloj – obnavljač signala (engl. *repeater*) i hab (engl. *hub*), sloj veze – most (engl. *bridge*) i komutator (engl. *switch*), mrežni sloj – ruter (engl. *router*). S obzirom na to da nema nikakve modifikacije podataka viših slojeva postojanje ovih uređaja je za korisnika transparentno.

TCP/IP skup protokola

TCP/IP je skup protokola koju je razvila agencija DARPA (*Defense Advanced Research Projects Agency*), a koja je kasnije uključena u Berkeley distribuciju UNIX sistama (BSD). Internet je zasnovan na TCP/IP skupu protokola, koji je postao de-facto standard za povezivanje računara i mreža.

TCP/IP model čine sloj pristupa mreži, Internet sloj, transportni sloj i aplikacioni sloj. TCP/IP model ne specificira sloj veze podataka i fizički sloj, već koristi različite protokole (SLIP, PPP) i tehnologije (*Ethernet*) na tom sloju. SLIP (*Serial Line Internet Protocol*) je adaptacija TCP/IP skupa protokola za računare koji su na mrežu povezani preko serijskog porta. Smatra se zastarelim i zamenjen je PPP protokolom (*Point-to-Point Protocol*), tj. adaptacijom TCP/IP skupa protokola za računare koji su na mrežu povezani modemom. PPP koristi kompresiju podataka u cilju povećanja propusnog opsega i metode autentifikacije (PAP i CHAP) za povećanje sigurnosti.

Internet sloj odgovara sloju mreže u OSI modelu. Bavi se IP adresiranjem i rutiranjem paketa, čime obezbeđuje vezu između računara koji se ne moraju nalaziti na fizički istoj mreži. Na ovom sloju prisutni su sledeći protokoli:

- IP (*Internet Protocol*) – fundamentalni protokol koji obezbeđuje transfer informacija od računara do računara,
- ICMP (*Internet Control Message Protocol*) – obezbeđuje kontolne poruke IP protokolu, kao što je "Destination Unreachable". Najčešća upotreba ICMP protokola je slanje ICMP ECHO paketa (ping) kojim se proverava da li je udaljeni računar dostupan,
- ARP (*Adress Resolution Protocol*) – osim logičke adrese (IP), svaki mrežni uređaj se karakteriše i fizičkom adresom (MAC) dužine 48 bita. MAC adrese dodeljuju proizvođači mrežnih adaptera i one su, uslovno rečeno, nepromenljive. MAC adrese se koriste prilikom transporta ramova podataka po fizički istoj mreži. ARP protokol razrešava IP adrese u MAC adrese. RARP je inverzan protokol ARP-u i pomoću njega se vrši određivanje IP adrese hosta na osnovu fizičke adrese.

Transportni sloj preuzima podatke sa višeg nivoa, po potrebi vrši

segmentaciju podataka u datagrame ili uspostavljanje virtuelnih veza i prenosi podatke do odredišta koristeći usluge Internet sloja. Na transportnom sloju prisutni su:

- TCP (*Transmission Control Protocol*) – protokol koji obezbeđuje pouzdanu vezu između dva procesa, otkriva i ispravlja greške,
- UDP (*User Datagram Protocol*) – protokol koji ne uspostavlja virtuelne veze niti obezbeđuje mehanizam za detekciju grešaka.

Aplikacioni sloj omogućava aplikacijama, odnosno korisnicima da pristupe servisima Internet mreže. Na aplikacionom sloju između ostalih prisutni su i sledeći protokoli:

- HTTP (*HyperText Transport Protocol*) – pristup Web stranicama,
- FTP (*File Transport Protocol*) – transfer datoteka,
- SMTP (*Simple Mail Transport Protocol*) – dolazeća pošta,
- POP3 (*Post Office Protocol v3*) – odlazeća pošta,
- DNS (*Domain Name System*) – razrešavanje imena u IP adrese.

IP adresiranje i podmrežavanje

Svaki računar i ruter na Internetu ima svoju jedinstvenu IP adresu (ili više IP adresa). IP adrese su 32-bitne, sastoje se od 4 okteta i obično se predstavljaju u decimalnoj notaciji sa tačkom (na primer: 192.198.3.1). Svaka IP adresa ima dva dela: deo koji predstavlja adresu IP mreže, koji je isti za sve računare na jednoj IP mreži, i deo koji predstavlja adresu računara, koji je jedinstven za svaki računar na istoj IP mreži

Na osnovu broja okteta koji pripadaju adresi mreže, odnosno adresi računara, IP adrese se dele u klase A, B, C, D i E.

- U binarnom obliku IP adrese klase A počinju sa 0. Prvi oktet predstavlja adresu mreže, a sledeća tri okteta adresu računara. Kako su adrese 0.x.y.z i 127.x.y.z rezervisane, IP adrese klase A se nalaze u opsegu 1.0.0.0 do 126.255.255.255,
- U binarnom obliku IP adrese klase B počinju sa 10. Prva dva okteta

pretstavljaju adresu mreže, a sledeća dva adresu računara. IP adrese klase B nalaze se u opsegu 128.0.0.0 do 191.255.255.255,

- U binarnom obliku IP adrese klase C počinju sa 110. Prva tri okteta predstavljaju adresu mreže, a poslednji adresu računara. IP adrese klase C nalaze se u opsegu 192.0.0.0 do 223.255.255.255,
- IP adrese klase D (u binarnom obliku počinju sa 1110) se dodeljuju grupi računara i namenjene su za *multicast* saobraćaj,
- IP adrese klase E (počinju sa 11110) su eksperimentalne i ne koriste se za adresiranje mreža i računara.

U posebne slučajeve IP adresa spadaju:

- 127.0.0.1 – adresa lokalne petlje (engl. *local loopback adress*),
- x.0.0.0 adresa mreže u klasi A,
- x.y.0.0 adresa mreže u klasi B,
- x.y.z.0 adresa mreže u klasi C,
- 255.255.255.255 – *broadcast* adresa,
- x.y.255.255.255 – *broadcast* adresa mreže x.y.0.0.

IP adrese se dalje mogu podeliti na javne i privatne. Javne adrese dodeljuje InterNIC i mogu se koristiti na Internetu. Privatne adrese su namenjene mrežama koje nisu direktno povezane na Internet i ne mogu se koristiti na Internetu. U privatne adrese spadaju:

- 10.0.0.0 – 10.255.255.255,
- 172.16.0.0 – 172.31.255.255,
- 192.168.0.0 – 192.168.255.255.

Maska podmreže je 32-bitni broj koji se formira tako što se umesto bitova koji u IP adresi pretstavljaju adresu mreže i podmreže stavi 1, a umesto bitova koji predstavljaju adresu host računara stavi 0. Podrazumevane maske podmreže za mreže u klasi A, B i C su redom 255.0.0.0, 255.255.0.0 i 255.255.255.0.

Adresa neke mreže se uvek navodi sa maskom podmreže. Na primer,

adresa mreže 192.168.10.0 u klasi C zapisuje se kao 192.168.10.0 255.255.255.0 ili 192.168.10.0/16 (broj 16 ukazuje na broj bitova koji pripadaju adresi mreže; radi se o takozvanoj CIDR notaciji).

Podmreže su segmenti iste IP mreže koji komuniciraju preko rutera. Podmrežavanjem se smanjuje broj računar po segmentu, tj. *broadcast* domen IP mreža sa velikim brojem računara. Svaka podmreža ima svoju jedinstvenu adresu, koja se formira tako što se određen broj bitova pozajmi iz dela IP adrese koji predstavlja adresu računara.

Podmrežavanje se može jednostavno objasniti na primeru IP mreže u klasi C. U ovom slučaju, pozajmićemo tri bita iz četvrtog okteta. Ukoliko je adresa IP mreže 192.168.10.0, adrese podmreža će redom biti: 192.168.10.32/28, 192.168.10.64/28, 192.168.10.92/28, itd. Na ovaj način je kreirano 8 segmenata, tj. osam podmreža čija je maska podmreže 255.255.255.248. U zavisnosti od toga da li protokol za rutiranje na ruteru koji spaja podmreže podržava VLSM (variable lenght subnet masking) ili ne, upotrebljivo je 6, odnosno 8 podmreža. Svaki segment sadrži 2^5=32 adrese. Prva adresa u svakom segmentu (na primer, 192.168.10.32) je adresa podmreže, a poslednja (192.168.10.63) broadcast adresa za tu podmrežu. To znači da se na svakom segmentu može naći najviše 30 računara.

Osim IP adrese protokoli koriste i broj porta, odnosno 16 bitni kvantitet koji dozvoljava više istovremenih veza ka jednom računaru. Značajniji portovi su dati u sledećoj tabeli:

20, 21	FTP	119	NNTP
22	SSH	135	RPC Locator
23	Telnet	137	NetBIOS Name
25	SMTP	139	NetBIOS Session
53	DNS	143	IMAP
69	TFTP	443	HTTPS
80	HTTP	530	RPC
110	POP3	3389	Windows Terminal Service

Svaka veza se odnosi na određeni port, koji se dodeljuje određenom mrežnom servisu na korišćenje. Administratori mogu otvoriti ili zatvoriti određeni port, čime se omogućava, odnosno onemogućava uspostavljanje konekcije ka nekom tipu servisa.

Korisnici se svakom računaru mogu obratiti putem IP adrese. To, naravno, znači da korisnici koji koriste servise 150 različitih računara moraju znati 150 IP adresa. Da bi se komunikacija pojednostavila, koristi se sistem dodele logičkih imena IP adresama. Na primer, korisnici se mogu obratiti računaru čija je IP adresa 166.60.10.15 imenom myserver, ukoliko je ime myserver dodeljeno toj IP adresi.

Rutiranje

Prava upotrebna vrednost IP mreže iskazuje se njenom mogućnošću da kontaktira druge IP mreže. Da bi se paket poslao računaru koji se nalazi na drugoj mreži, potrebno je da u mreži postoji uređaj koji zna kako i gde isporučiti paket. Ovaj oblik isporuke paketa je poznat kao rutiranje. Ruter se definiše kao višeportni uređaj mrežnog sloja OSI modela koji posmatra mrežu u celini i na osnovu toga donosi odluke o najboljoj putanji za slanje paketa. Izbor putanje se svodi na izbor sledećeg skoka u mreži. Postoji nekoliko tipova rutiranja: standardno rutiranje (svi paketi koji nisu namenjeni mreži šalju se na podrazumevani izlaz), statičko (administrator specificira statičke rute u tabeli rutiranja) i dinamičko rutiranje.

10.2 Šta je mrežna barijera i koje su njene funkcije?

Mrežnim barijerama (engl. *firewall*) kreiraju se kontrolne tačke bezbednosti na granicama privatnih mreža (engl. *border security*). Na ovim kontrolnim tačkama mrežne barijere ispituju sve pakete koji prolaze između privatne mreže i Interneta. Jednostavno rečeno, *firewall* se nalazi na granici privatne mreže i filtrira saobraćaj na relaciji lokalna mreža – Internet. Mrežne barijere održavaju vezu sa spoljnom mrežom što je moguće bezbednijom tako što ispituju i nakon toga odobravaju ili odbijaju svaki pokušaj povezivanja privatnih mreža i spoljnih mreža. *Firewall*, takođe, štiti kućne računare sa stalni pristupom Internetu.

Mrežne barijere omogućuju centralizaciju svih bezbednosnih servisa na računarima koji su optimizovani i posvećeni zadatku zaštite. Mrežne barijere štite mrežu na mrežnom, transportnom i aplikacionom sloju OSI referentnog modela:

- mrežni sloj – filtriranje paketa na osnovu IP adresa i prevođenje privatnih u javne IP adrese (*engl. network adress translation*, NAT),
- transportni sloj – kontrola pristupa TCP servisima, tj. dozvola ili zabrana pristupa TCP/IP portovima u zavisnosti od izvorišnih i odredišnih IP adresa,
- aplikacioni sloj – prihvatanje zahteva za pristup određenoj aplikaciji koji se dalje upućuju ka odredištu ili blokiraju.

Kao mrežna barijera može se koristiti skup hardverskih uređaja i/ili servera od kojih svaki obavlja samo jednu od navedenih funkcija. Na primer, ruter, kao zaseban hardverski uređaj, filtrira pakete na osnovu IP adresa i broja porta, dok se proksi server nalazi na posebnom računaru unutar mreže.

Filtriranje paketa

Mrežne barijere analiziraju pakete i upoređuju ih sa prethodno definisanim skupom pravila. Filtriranje je moguće na osnovu bilo kog dela zaglavlja paketa, a većina filtara donosi odluku na osnovu:

- tipa protokola – na ovaj način se može izvršiti diskriminacija čitavih skupova protokola, kao što su UDP, TCP, ICMP, IGMP),
- IP adrese – prihvatanje ili odbijanje paketa na osnovu IP adrese je najjači oblik zaštite koji se može postići prostim filtriranjem paketa,
- TCP/UDP porta – na primer, svim računarima se može dozvoliti da pristupe TCP portu 80 (HTTP), dok je pristup TCP portu 22 (ssh) ograničen računarima koji pripadaju određenom opsegu IP adresa.

Na osnovu definisanih pravila i zaglavlja konkretnog IP paketa, filter paketa može da odluči da:

- prihvati određeni paket,

- odbaci paket,
- odbaci paket i obavesti pošiljaoca da njegov paket nije prihvaćen.

Navodimo neke preporuke za konfigurisanje filtratora paketa:

- eksplicitno zabranite sve osim onog što treba da bude dozvoljeno,
- napravite demilitarizovanu zonu za servere koji trebaju da budu dostupni računarima sa Interneta,
- zabranite sve ulazne konekcije, tj. konekcije spolja ka računarima u lokalnoj mreži (time se sprečava mogućnost povezivanja spolja na prethodno instalirane trojanske konje na računarima u lokalnoj mreži),
- zabranite odgovore na ICMP echo ili ICMP redirect pakete,
- zabranite slanje update protokola za rutiranje ka ruterima na unutrašnjoj mreži.

Sofisticirani filtri proučavaju sve konekcije koje prolaze kroz njih i pri tom traže oznake koje ukazuju na moguće ”hakerisanje”, kao što je navođenje tačne putanje puta (engl. *source routing*), preusmeravanje ICMP paketa (ICMP redirect) i lažiranje IP adresa. Konekcije koje prikazuju ovakve karakteristike bivaju odbačene.

Ne oslanjajte se samo na filtriranje paketa! Filtriranje paketa ne rešava u potpunosti problem bezbednosti lokalnih mreža. Na primer, filtri ne ispituju HTTP poruke, sadržane u TCP paketima kako bi utvrdili da li oni sadrže elemente kojima se eksploatišu slabe tačke nekog Web servera kog taj filtar štiti.

Mrežne barijere sa i bez uspostave stanja

Postoje dve vrste filratora paketa:

- mrežne barijere bez uspostave stanja (engl. *stateless firewall*) i
- mrežne barijere sa uspostavom stanja (engl. *statefull firewall*).

Mrežna barijera bez uspostave stanja odbacuje paket ukoliko nema dovoljno informacija šta bi sa njim trebalo da uradi. Većina mrežnih barijera

ovog tipa ostavlja portove veće od 1024 otvorene, kako bi omogućila slanje odgovora računaru koji je poslao zahtev. Trojanski konji mogu da iskoriste ove portove i to predstavlja ozbiljan sigurnosni propust.

Mrežne barijere sa uspostavom stanja su fleksibilnije jer prate stanje na mrežnom sloju (pamte zahteve za uspostavljanjem veze) i to koriste prilikom donošenja odluka. Karakterišu se postojanjem tabele stanja, tj. tabele u kojoj *firewall* vodi evidenciju o trenutnim stanjima konekcija. Barijere ovog tipa dozvoljavaju slanje odgovora ka računarima koji su uspostavili konekciju, a potencijalne rupe ostaju otvorene samo onoliko dugo koliko je potrebno. Obajsnićemo to na sledećem primeru.

Pretpostavite da je računarima na lokalnoj mreži dozvoljeno da uspostave konekcije ka određenim portovima računara na spoljnoj mreži. Računar na lokalnoj mreži, koji odluči da inicira TCP konekciju, šalje TCP paket na IP adresu i broj porta javnog servera. U ovoj poruci, računar navodi udaljenom serveru svoju IP adresu i broj porta na kom očekuje odgovor. Mrežna barijera dozvoljava paketu da prođe na spoljašnju mrežu i pamti relevantne informacije iz zaglavlja paketa. Nakon primanja paketa, spoljašnji server šalje odgovor na specificirani port. Mrežna barijera proverava sve podatke koji su razmenjeni između ta dva računara i, budući da zna da je konekciju inicirao računar sa lokalne mreže, dozvoljava računaru sa spoljašnje mreže da odgovori na taj zahtev.

Rad mrežne barijere sa uspostavom stanja ilustrovaćemo primerom. Između klijenta koji pripada unutrašnjoj mreži (192.168.0.1) i servera koji pripada spoljašnjoj mreži (10.0.0.1) nalazi se *firewall* sa uspostavom stanja konfigurisan tako da propušta sav odlazeći saobraćaj:

- klijent šalje serveru zahtev na port 80 i zahteva odgovor na portu 1220,
- *firewall* prosleđuje paket dodaje u tabelu stanja pravilo: server 10.0.0.1 može slati pakete računaru 192.168.0.1 na port 1220,
- server prima zahtev i šalje odgovor na 192.168.0.1:1220,
- firewall proverava tabelu stanja i utvrđuje da server 10.0.0.1 može slati pakete računaru 192.168.0.1 na port 1220,
- računar prima odgovor na portu 1220.

Ukoliko napadač sa IP adresom 10.0.0.2 pokuša da odgovori na zahtev klijenta, firewall neće proslediti odgovor jer u tabeli stanja ne postoji zapis koji dozvoljava računaru 10.0.0.2 da šalje podatke na 192.168.0.1:1220.

Kada učesnici u sesiji zatvore TCP konekciju, mrežna barijere briše zapise u svojoj tabeli stanja i time ukida mogućnost računaru sa spoljašnje mreže da dalje komunicira sa računarom na lokalnoj mreži. Ukoliko računar na lokalnoj mreži prestane da odgovara računaru na internetu pre zatvaranja TCP konekcije (na primer, zbog prekida veze) ili ako protokol koji je u pitanju ne podržava sesije (na primer, UDP), mrežna barijera će ukloniti zapis iz tebele stanja nakon određenog vremenskog intervala.

Prevođenje mrežnih adresa

NAT skriva informacije o računarima u privatnoj mreži od napadača sa Interneta. Prilikom prolaza paketa kroz mrežnu barijeru NAT skriva IP adrese računara iz privatne mreže prevodeći ih u adresu mrežne barijere. Mrežna barijera, zatim, ponovo šalje podatke koji se u tom paketu nalaze sa svoje adrese, koristeći pritom tablicu prevođenja adresa. Osim zaštitne funkcije, NAT omogućava uštedu javnih IP adresa, jer se jedna javna IP adresa, koristeći različite brojeve porta, može prevesti u veći broj privatnih IP adresa.

Postoji nekoliko vrsta prevođenja IP adresa:

- statičko – blok javnih IP adresa se na osnovu fiksne tablice prevođenja prevodi u blok privatnih IP adresa, tako da jednoj javnoj IP adresi odgovara jedna privatna IP adresa. Na ovaj način se skriva identitet računara u lokalnoj mreži;
- dinamičko – blok javnih IP adresa se dinamički prevodi u blok privatnih IP adresa. Na ovaj način se skriva identitet računara u lokalnoj mreži;
- dinamičko sa preopterećenjem (engl. *port address translation*, PAT) – jedna ili više javnih IP adresa se na osnovu broja porta prevodi u veći broj privatnih IP adresa. Na ovaj način se skriva identitet računara u lokalnoj mreži. Ovaj način prevođenja adresa se najčešće koristi.

Kao posebni slučajevi dinamičkog prevođenja mogu se izdvojiti slučajevi prevođenja radi raspodele opterećenja sa:

- unutrašnje strane mrežne barijere – pretpostavite da sa unutrašnje strane mrežne barijere imate troslojnu klijent-server arhitekturu (četiri Web servera sa aplikacionom logikom povezana na istu bazu podataka). U tom slučaju, javna IP adresa se prevodi u jednu od privatnih IP adresa (dodeljenih serverima) po round-robin algoritmu kako bi se raspodelilo opterećenje između Web servera.
- spoljašnje strane mrežne barijere – pretpostavite da je Vaša mrežna barijera povezana sa Internetom pomoću nekoliko veza. *Firewall* bira rutu i povezuje računar iz lokalne mreže na Internet shodno opterećenju i dostupnosti javnih mreža.

Mrežne barijere na aplikacionom sloju OSI modela

Ova vrsta mrežnih barijera uspostavlja vezu između računara na visokom aplikacionom nivou, čime se prekida konekcija na mrežnom sloju između računara u privatnoj mreži i računara iz spoljne mreže. U mrežne barijere aplikacionog nivoa spadaju proksi serveri koji prihvataju zahteve za pristup određenoj aplikaciji koje dalje upućuju ka odredištu ili ih blokiraju. Mrežne barijere na aplikacionom sloju realizuju se softverski.

Prednost barijere na aplikacionom nivou je odsustvo prostog prosleđivanja paketa (engl. *forwarding*), povećana kontrola uspostavljanja veze i napredno praćenje događaja. Loša strana ovakvih barijera je brzina; svaki paket se prihavata, proverava, prevodi i prosledjuje dalje, pa su ovakva rešenja najčešće sporija od prostog filtriranja paketa. U mrežne barijere na aplikacionom sloju spadaju različita složenija (*Thrusted Information Systems Firewall Toolkit*) i jednostavnija rešenja (*Kerio Personal Firewall*).

Demilitarizovane zone

Opasnost od napada sa Interneta može se znatno smanjiti korišćenjem dvonivovske zaštite. Na primer:

- jedan *firewall* štiti Web server od napada sa Interneta, ali dozvoljava pristup Internet servisima koje pruža taj deo mreže,
- drugi *firewall* sa jačom bezbednosnom polisom ne dozvoljava pristup privatnoj mreži sa Interneta i skriva identitete računara iz privatne mreže.

Primenom ove tehnike mreža se deli na tri domena:

- Internet – krajnje nepoverljiv i nesiguran domen,
- demilitarizovana zona (DMZ), tj. javni deo privatne mreže,
- lokalna mreža.

Po pravilu, dozvoljeno je uspostavljanje veze između Interneta i DMZ, kao i veze između lokalnih računara i Interneta (pod uslovom da računar sa lokalne mreže inicira uspostavljanje veze). Uspostavljanje veze između Interneta ili DMZ i računara u lokalnoj mreži je strogo kontrolisano (ili još češće, zabranjeno). Najveći broj hardverskih i softverskih mrežnih barijera dozvoljava primenu različitih bezbednosnih polisa na svakom interfejsu. Na taj način jednom mrežnom barijerom sa tri interfejsa može se postići funkcionalnost dva *firewall* uređaja.

10.3 iptables

U jezgra Linux sistema, počev od verzije 2.4, ugrađen je sistem za filtriranje paketa poznat kao Netfilter. Da bi Netfilter radio kako treba, potrebno je da se:

- pri izgradnji kernela uključi opcija CONFIG_NETFILTER,
- sistem konfiguriše kao ruter (zbog prosleđivanja paketa), tj da se ip_forward postavi na 1. To možete uraditi tako što ćete u datoteku /etc/rc.d/rc.local dodati sledeću liniju:
  - `echo 1 > /proc/sys/net/ipv4/ip_forward`

  To će se izvršiti svaki put kada se podigne sistem. Ukoliko koristite Debian distribuciju, u konfiguracionu datoteku za postavljanje sistemskih promenljivih (/etc/sysctl.conf) dodajte sledeću liniju:

- `net/ipv4/ip_forward=1`

Filtriranje paketa se obavlja na Internet sloju TCP/IP skupa protokola, tj. na mrežnom sloju OSI referentnog modela, odnosno na osnovu zaglavlja IP paketa. Dodatno, pravila filtriranja se mogu definisati i na osnovu zaglavlja transportnog sloja (broj porta) i sloja veze (izvorišna MAC adresa) OSI modela. Prilikom filtriranja, sadržaj paketa se ne tumači, jer je to isuviše procesorski zahtevno.

Netfilter lanci za filtriranje paketa rade u zaštićenom režimu rada (engl. *kernel mode*). U korisničkom režimu rada radi poseban alat – iptables, koji zahteva privilegije root korisnika i služi za konfigurisanje:

- filterskih lanaca,
- NAT tabela i
- mangle tabele.

Filtriranje paketa

Netfilter koristi tri filtra (INPUT, OUTPUT i FORWARD), realizovanih u formi lanaca pravila (engl. *chains*). Svaki lanac sadrži pravila koja se primenjuju jedno za drugim na svaki paket koji prolazi kroz lanac. Ako paket zadovolji neko pravilo, izvršava akcija, poput prihvatanja paketa (ACCEPT) ili odbacivanja paketa (REJECT). Ako paket ne zadovolji pravilo, prosleđuje se dalje kroz lanac, tj. proverava se da li će zadovoljiti sledeće pravilo u lancu. Na kraju lanca se nalazi polisa koja određuje šta treba da se uradi sa paketima koji ne zadovoljavaju ni jedno pravilo.

Kada paket stigne u jezgro operativnog sistema, Netfilter najpre analizira odredišnu adresu paketa:

- Paket namenjen procesu na lokalnom računaru prosleđuje se INPUT lancu. Ako pravila lanca prihvate paket, on se prosleđuje procesu koji ga očekuje. Ova situacija je tipična za računare na kojima je instaliran neki mrežni servis (npr. Apache Web Server), koji je zaštićen mrežnom barijerom.

- Dolazeći paket koji nije namenjen lokalnim procesima predaje se

FORWARD lancu. Ako u jezgru nije uključena podrška za prosleđivanje IP paketa, ili jezgro ne može da odredi kako da prosledi paket na osnovu tabele rutiranja, paket se odbacuje. Ako je paket prihvaćen pravilima FORWARD lanca, prosleđuje se interfejsu, prethodno određenom na osnovu tabele rutiranja.

- Odlazeći paket koji potiče od lokalnog procesa predaje se OUTPUT lancu. Ako pravila lanca prihvate paket, paket se šalje na interfejs, koji je određen tabelom rutiranja.

Filtarski lanci konfigurišu se pomoću iptables alata (ne zaboravite, potrebne su vam root privilegije). Sintaksa komande je sledeća:

```
iptables –A lanac pravilo [opcije] –j akcija
iptables –[RI] lanac red_br pravilo [opcije] [–j akcija]
iptables –D lanac red_br [opcije]
iptables –[LF] [lanac] [opcije]
iptables –P lanac odrednica [opcije]
```

Komande imaju sledeće značenje:

- -A (dodaj pravilo na kraj lanca), -D (obriši pravilo iz lanca),
- -R (zameni pravilo u lancu), -I (ubaci numerisano pravilo u lanac),
- -L (izlistaj pravila u lancu), -F (zbriši sva pravila iz lanca),
- -P (definiši polisu koja će se primenjivati na pakete koji se ne slažu ni sa jednim pravilom u lancu).

U slučaju da podrazumevana polisa na kraju lanca odbacuje sve pakete koji nisu prihvaćeni ni jednim pravilom lanca, preporučuje se da pravila u lance unosite od najspecifičnijih ka najopštijim. Tako je, na primer, moguće da dozvolite pristup svim računarima sa podmreže 10.1.1.0, osim računara 10.1.1.8 i 10.1.1.9.

Parametar “lanac” može biti input, output ili forward lanac.

Parametrom “pravilo” specificira se jedno ili više pravila koja se primenjuju na pakete, zavisno od lanca kroz koji se paket propušta. Pravila se specificiraju na sledeći način:

- ! označava negaciju,
- -p [!]protocol

  numerička ili simbolička oznaka protokola iz datoteke /etc/services (na primer: tcp, udp, icmp, ...),
- -s [!]address/mask i -d [!]address/mask

  izvorišna, odnosno odredišna IP adresa paketa i maska podmreže (na primer, 38.24.12.0/24),
- --src-range StartIP–EndIP i --dst-range StartIP–EndIP

  opseg izvorišnih, odnosno odredišnih IP adresa paketa (na primer, 38.24.12.1–38.24.12.24),
- --sport [!]port[:port] i --dport [!]port[:port]

  izvorišni, odnosno odredišni port ili opseg portova,
- -i [!]interface i -o [!]interface

  izvorišni, odnosno odredišni interfejs na računaru (mrežna kartica).

Ukoliko paket odgovara nekom pravilu, na paket se primenjuje akcija specificirana parametrom "-j akcija":

- ACCEPT – prihvati paket,
- DROP – odbaci paket,
- REJECT – odbaci paket i obavesti pošiljaoca ICMP porukom,
- LOG – čuvanje detaljnih informacija o paketu,
- MASQ – maskiraj izvorišnu IP adresu paketa adresom mrežne barijere i prosledi dalje paket (samo za forward lanac).

Primer 10.1. Na Linux serveru d-fens.security.org pokrenut je Netfilter. Server, takođe, obavlja funkciju rutera. Server ima dva mrežna adaptera: eth0 (64.10.10.1) i eth1 (172.16.32.1). Interfejs eth0 obezbeđuje vezu sa Internetom. Na eth1 interfejs Linux servera vezani (pomoću komutatora) Web i mail server iz lokalne mreže 172.16.32.0/24, čije su IP adrese 172.16.32.2 i 172.16.32.3 respektivno. Koristeći program iptables obezbedite filtriranje saobraćaja bez uspostave stanja na serveru i konfigurišite odgovarajuće lance kojima se:

- dozvoljava HTTP i mail saobraćaj sa Interneta ka serverima,
- dozvoljava ssh saobraćaj ka mrežnoj barijeri sa adrese 172.16.32.4 (radna stanica administratora),
- zabranjuje *spoofing*, tj. osigurava da sav saobraćaj koji napušta lokalnu mrežu ima odgovarajuće adrese,
- sve ostalo zabranjuje.

Pre nego što počnete da dodajete pravila u lanac potrebno je da obrišete sva postojeća pravila iz svih lanaca i definišete polisu kojom se zabranjuje sve što nije eksplicitno dozvoljeno.

- brisanje lanaca:

```
# iptables –F INPUT
# iptables –F OUTPUT
# iptables –F FORWARD
```

- definisnje podrazumevane polise:

```
# iptables –P INPUT DROP
# iptables –P OUTPUT DROP
# iptables –P FORWARD REJECT
```

Potrebno je da dozvolite računarima sa Interneta da pristupe http i mail servisima na vašim serverima; takođe potrebno je da omogućite i serverima da odgovore na zahtev klijenata. Pošto se zahteva da mrežna barijera radi bez uspostave stanja, konfigurisaćete forward lance na sledeći način:

- dozvolićete http, pop3 i smtp zahteve ka odrgovarajućim serverima (proverava sa odredišni port i odredišna adresa)

```
# iptables –A FORWARD –s 0.0.0.0/0 –d 172.16.32.2
          --dport http –j ACCEPT
# iptables –A FORWARD –s 0.0.0.0/0 –d 172.16.32.3
          --dport smtp,pop –j ACCEPT
```

- dozvolićete serverima da odgovore na zahteve (proverava se izvorišni port i izvorišna adresa)

```
# iptables –A FORWARD –s 172.16.32.2 –d 0.0.0.0/0
```

```
            --dport http –j ACCEPT
# iptables –A FORWARD –s 172.16.32.3 –d 0.0.0.0/0
            --sport smtp,pop –j ACCEPT
```

Pošto se radna stanica administratora nalazi na lokalnoj mreži, ssh saobraćaj se može ograničiti na interfejs eth1 i IP adresu radne stanice administratora. Potrebno je da dozvolite ssh zahtev sa radne stanice administratora (konfiguriše se INPUT lanac) ka mrežnoj barijeri i odgovor mrežne barijere na ssh zahtev (konfiguriše se OUTPUT lanac).

```
# iptables –A INPUT –s 172.16.32.4 -i eth1 --dport ssh
            –j ACCEPT
# iptables –A OUTPUT –d 172.16.32.4 -o eth1 --sport ssh
            –j ACCEPT
```

Ostaje još da zabranite spoofing, tj. da obezbedite da izvorišne adrese svih paketa koji napuštaju lokalnu mrežu i ulaze u interfejs eth1 pripadaju mreži 172.16.32.0/24, tj. da odbacite sve pakete sa neispravnom IP adresom.

```
# iptables –A FORWARD –i eth1 –s !172.16.32.0/24 –j REJECT
```

Kreiranje novih lanaca

Komanda iptables dozvoljava administratorima da kreiraju lanac specifičnog imena i da pakete sa INPUT, OUTPUT i FORWARD lanaca preusmere na dalje ispitivanje u korisnički definisane lance. Preusmeravanje se obavlja ukoliko paket zadovolji neko pravilo u INPUT. OUTPUT ili FORWARD lancu, ili podrazumevanom polisom, ukoliko paket ne zadovolji ni jedno pravilo u lancu.

Sledeće naredbe redom služe za kreiranje, brisanje i promenu imena korisnički definisanih lanaca:

```
iptables –N kor_def_lanac
iptables –N kor_def_lanac
iptables –E staro_ime novo_ime
```

Paket se može preusmeriti na korisnički definisani lanac sa tekućeg lanca navođenjem imena lanca umesto akcije. Na primer, sledeće pravilo će

sve dolazeće TCP pakete tipa SYN preusmeriti na lanac TCP_SYN_PACKETS.

```
# iptables –A INPUT –p tcp --syn –j TCP_SYN_PACKETS
```

Paket se preusmerava na korisnički definisani lanac pomoću podrazumevane polise navođenjem imena lanca umesto akcije. Na primer, sledeća komanda će sve pakete koji ne zadovolje ni jedno pravilo u FORWARD lancu preusmeriti na lanac UNMATCHED_PACKETS.

```
# iptables –P FORWARD UNMATCHED_PACKETS
```

Pravila se u korisnički definisane lance dodaju na isti način kao i u INPUT, OUTPUT i FORWARD lance.

Prošireni skup pravila

Kao što je već rečeno, parametrom “pravilo” specificira se jedno ili više pravila koja se primenjuju na pakete. Prilikom definisanja filtara možete primeniti i neka od sledećih poređenja:

- -f – drugi ili neki od sledećih fragmenata većeg paketa u nizu,
- --tcp-flags [!] mask comp

  flegovi TCP paketa (SYN, ACK, FIN, RST, URG, PSH, ALL, NONE),
- --syn – paketi sa postavljenim SYN i resetovanim ACK i FIN bitovima,
- --icmp-type [!] type – ICMP poruka tipa “type”,
- -m limit --limit rate

  najveća učestalost paketa; parametar “rate” se zadaje kao najveći mogući broj paketa po jedinici vremena, na primer: 2/second, 5/minute, 30/hour, 100/day,
- -m state

  trenutno stanje konekcije (INVALID – asocirana sa nepoznatim tokom, ESTABLISHED – dozvoljena uspostavljena konekcija, NEW – paket je započeo ili će započeti novu konekciju , RELATED – paket započinje novu konekciju i asociran je sa već uspostavljenom konekcijom),
- -m owner --uid-owner UID, -m owner --gid-owner GID,

-m owner --pid-owner PID

atributi lokalnog procesa (PID, GID, user ID) koji je kreirao paket i poslao ga na mrežu; poređenje je pogodno za sprečavanje dejstva trojanaca.

Primer 10.2. Dozvolite "pingovanje", tj. dozvolite svim računarima da komandom ping provere dostupnost mrežne barijere.

Potrebno je dodavanjem pravila u INPUT lanac dozvoliti prijem ICMP echo-request paketa i dodavanjem pravila u OUTPUT lanac dozvoliti slanje ICMP echo-reply paketa. Znači, u INPUT i OUTPUT lanac se dodaje po jedno pravilo koje proverava protokol i tip ICMP poruke.

```
# iptables –A INPUT –p icmp –icmp-type echo-request –j ACCEPT
# iptables –A OUTPUT –p icmp –icmp-type echo-reply –j ACCEPT
```

Primer 10.3. Napišite pravilo kojim se otvaranje FTP konekcije ka FTP serveru ograničava na svakih 15 sekundi.

Potrebno je u FORWARD lanac dodati pravilo kojim se najviše četiri puta u minutu dozvoljava prosleđivanje TCP SYN paketa usmerenih na port 21[58].

```
# iptables –A FORWARD –p tcp --syn --dport 21
           –m limit –limit 2/minute –j ACCEPT
```

Primer 10.4. Na Linux serveru guardian.security.org pokrenut je Netfilter. Server, takođe, obavlja funkciju rutera. Server ima dva mrežna adaptera: eth1 (172.16.32.254) i eth0 (11.22.33.44). Deset računara sa IP adresama u opsegu 172.16.32.1 – 172.16.32.10 vezano je u lokalnu mrežu 172.16.32.0/24. Lokalna mreža je pomoću komutatora vezana na interfejs eth1 servera. Veza sa Internetom obezbeđena je preko interfejsa eth0 Linux servera. Koristeći program iptables, obezbedite da filtriranje saobraćaja sa uspostavom stanja na serveru i konfigurišite odgovarajuće lance kojima se:

- dozvoljava SSH saobraćaj sa računara čije IP adrese pripadaju opsegu 172.16.32.1-10 – 172.16.32.10 ka mrežnoj barijeri

58 SYN paketi su paketi koji uspostavljaju TCP konekciju. U ovom slučaju kontrolišemo uspostavljanje FTP konekcije, pa zato ograničavamo frekvenciju prosleđivanja paketa ka portu 21.

- dozvoljava SSH, WWW, SMTP, DNS saobraćaj sa lokalne mreže ka Internetu,
- klijentima dozvoljava da “pinguju” računare na Internetu,
- sve ostalo zabranjuje.

Najpre ćete obrisati lance:

```
# iptables –F INPUT
# iptables –F OUTPUT
# iptables –F FORWARD
```

Zatim ćete blokirati saobraćaj namenjen mrežnoj barijeri sa Interneta (eth0) i saobraćaj koji procesi sa mrežne barijere generišu i šalju na internet.

```
# iptables –A INPUT –i eth0 –j DROP
# iptables –A OUTPUT –o eth0 –j DROP
```

Potrebno je da omogućite ssh sa računara iz navedenog opsega ka mrežnoj barijeri:

```
# iptables -A INPUT -p tcp --dport ssh -i eth1
           --src-range 172.16.32.1-172.16.32.10 -j ACCEPT
# iptables -A OUTPUT -p tcp --sport ssh -o eth1
           --dst-range 172.16.32.1-172.16.32.10 -j ACCEPT
```

Blokirajte sve što nije dozvoljeno na kraju FORWARD lanca:

```
# iptables –P FORWARD DROP
```

Dozvolite prosleđivanje fragmentisanih paketa i paketa na već uspostavljenim konekcijama:

```
# iptables –A FORWARD –f -j ACCEPT
# iptables –A FORWARD –p tcp –m state
           –state ESTABLISHED, RELATED –j ACCEPT
```

Dozvolite SSH, WWW i SMTP saobraćaj sa lokalne mreže ka Internetu (odgovore ne dozvoljavate zbog klauzule koja kontoliše prosleđivanje paketa na uspostavljenim konekcijama):

```
# iptables –A FORWARD –p TCP –i eth1
     --dport ssh,www,smtp -j ACCEPT
```

Dozvolite DNS saobraćaj sa lokalne mreže ka Internetu i odgovore na DNS zahteve sa Interneta:

```
# iptables –A FORWARD –p UDP –i eth1 --dport 53 -j ACCEPT
# iptables –A FORWARD –p UDP –i eth0 --sport 53 -j ACCEPT
```

Dozvolite ping, tj. prosleđivanje echo-request poruka iz lokalne mreže ka Internetu i echo-reply poruka sa Interneta ka lokalnoj mreži:

```
# iptables –A FORWARD –p icmp –i eth1 –s 172.16.32.0/24
           --icmp-type echo-request -j ACCEPT
# iptables –A FORWARD –p icmp –i eth0 –d 172.16.32.0/24
           --icmp-type echo-reply –j ACCEPT
```

Zaštita od čestih napada

Navešćemo neke klasične vrste napada i objasnićemo kako možete da iskoristite iptables da zaštitite mrežu od tih napada:

- Lažiranje izvorišne IP adrese (engl. *address spoofing*)

  Zaštita: odbacite sve pakete koji stižu na javni interfejs a imaju adresu lokalne mreže

```
iptables –A FORWARD –s intranet_IP –i public_IP -j DROP
```

- Podmetanje žrtve (engl. *smurf attack*)

  Opis: napadač šalje *broadcast* ICMP echo-request pakete računarima u intranetu, a kao izvorišnu adresu navodi adresu žrtve. Žrtva trpi bombardovanje echo-reply paketima.

  Zaštita: odbacite sve broadcast echo-request pakete.

```
iptables –A FORWARD –p icmp –d intranet_broadcast -j DROP
```

- Bombardovanje SYN paketima (engl. *SYN-flood)*

  Opis: slanje velikog broja SYN paketa (TCP connection request)

Zaštita: ograničite broj SYN paketa u jedinici vremena

```
iptables –A FORWARD –p tcp –syn –m limit –limit 1/s –j ACCEPT
```

- Skeniranje portova (engl. *port scanner*)

  Opis: identifikacija otvorenih portova; napadač šalje SYN ili FIN pakete opsegu portova i očekuje SYN+ACK pakete za otvorene i RST pakete za zatvorene portove.

  Zaštita: dozvoliti ograničen broj paketa u jedinici vremena

```
iptables –A FORWARD –p tcp --tcp-flags SYN,ACK,FIN,RST
         –m limit --limit 1/s –j ACCEPT
```

- "*Ping-of-Death*"

  Opis: slanje velikog broja ICMP echo-request paketa može "ubiti" neke operativne sisteme

  Zaštita: dozvoliti ograničen broj ICMP echo-request paketa u jedinici vremena

```
iptables –A FORWARD –p icmp --icmp-type echo-request
         –m limit --limit 1/s –j ACCEPT
```

  Idealna zaštita: zabranite prosleđivanje echo-request paketa ka serverima

```
iptables –A FORWARD –p icmp --icmp-type echo-request –j DROP
```

NAT

NAT tabela se koristi za prevođenje IP adresa paketa, tj. za promenu vrednosti polja u zaglavlju paketa koja označavaju izvorišnu i odredišnu adresu. NAT tabela mrežne barijere sastoji se od tri lanca pravila:

- PREROUTING (izmena paketa pre odluke o rutiranju),
- POSTROUTING (izmena paketa nakon odluke o rutiranju),
- OUTPUT (izmena paketa pre slanja).

Odredišno (engl. *destination NAT*, DNAT) prevođenje služi za promenu odredišne adrese paketa, tj. za preusmeravanje konekcije sa mrežne barijere na drugo odredište. Obavlja se pre rutiranja, što znači da se DNAT pravilo dodaje u PREROUTING lanac. Na primer, pomoću DNAT-a se paket može preusmeriti sa mrežne barije sa javnom IP adresom na neki server u demilitarizovanoj zoni.

Izvorišno (engl. *source NAT*, SNAT) prevođenje služi za promenu izvorišne adrese paketa. Obavlja se posle rutiranja, što znači da se SNAT pravilo dodaje u POSTROUTING lanac. SNAT je pogodan za sakrivanje informacija o lokalnoj mreži ili o demilitarizovanoj zoni. Na primer, posmatrajte mrežnu barijeru sa poznatom javnom IP adresom. Pomoću SNAT-a, mrežna barijera će zameniti izvorišne adrese paketa (koje pripadaju opsegu privatnih adresa) u javne IP adrese i time omogućiti vezu računara na lokalnoj mreži sa Internetom.

Ukoliko želite da omogućite NAT prevođenje, potrebno je da sistem konfigurišete kao ruter, tj postavite vrednost promenljive ip_forward na 1 i da učitate iptables_nat modul u jezgro Linux operativnog sistema:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
# modprobe iptables_nat
```

Primer 10.5. Pretpostavite da mrežna barijera ima dva interfejsa:

- eth0, koji služi za vezu sa Internetom; interfejsu eth0 dodeljeni su odgovarajući IP alijasi 97.158.253.26 i 97.158.253.29,
- eth1, koji je vezan na lokalnu mrežu 192.168.1.0/24.

Mrežna barijera treba da obezbedi:

- 1:1 prevođenje privatne adrese Web servera 192.168.1.100 sa lokalne mreže u javnu IP adresu 97.158.253.26,
- 1:N prevođenje adresa sa privatne mreže 192.168.1.0/24 u javnu IP adresu 97.158.253.29.

Mrežnu barijeru ćete najpre konfigurisati dodavanjem pravila za prevođenje adresa u PREROUTING i POSTROUTING lance:

- PREROUTING pravilo za 1:1 DNAT prevođenje (paketi upućeni Web serveru sa Interneta)

```
# iptables -t nat -A PREROUTING -d 97.158.253.26 -i eth0
           -j DNAT --to-destination 192.168.1.100
```

- POSTROUTING pravilo za 1:1 SNAT prevođenje (paketi upućeni Internetu sa Web servera, tj. odgovori na http zahteve)

```
# iptables -t nat -A POSTROUTING -s 192.168.1.100 -o eth0
           -j SNAT --to-source 97.158.253.26
```

- POSTROUTING pravila za N:1 NAT prevođenje

```
# iptables -t nat -A POSTROUTING -s 192.168.1.0/24
           -j SNAT -o eth0 --to-source 97.158.253.29
```

Dodatno, potrebno je konfigurisati i filtarske lance, koji će dozvoliti saobraćaj između lokalne mreže i Interneta:

- dozvolite prosleđivanje paketa Web serveru sa Interneta (obratite pažnju da ovde koristite privatnu, tj. stvarnu IP adresu servera)

```
# iptables -A FORWARD -i eth0 -o eth1 -d 192.168.1.100
           -p tcp -m multiport --dport 80,443
           -m state --state NEW -j ACCEPT
```

- dozvolite prosleđivanje svih novih i uspostavljenih SNAT konekcija sa lokalne mreže i već uspostavljenih DNAT konekcija sa Interneta

```
# iptables -A FORWARD -t filter -o eth0 -m state
           --state NEW,ESTABLISHED,RELATED -j ACCEPT
```

- dozvolite prosleđivanje svih paketa koji pripadaju 1:1 NAT konekcijama koje potiču sa Interneta i već su uspostavljene

```
# iptables -A FORWARD -t filter -i eth0 -m state
           --state ESTABLISHED,RELATED -j ACCEPT
```

Maskiranje

Kao što je već rečeno (i ilustrovano primerom), statički NAT tipa više-na-jedan može se konfigurisati dodavanjem pravila za SNAT prevođenje u POSTROUTING lanac ukoliko je javnom interfejsu mrežne barijere dodeljen IP alijas.

Maskiranje (engl. *masquerading*) je N:1 NAT prevođenje izvorišnih adresa paketa koji potiču sa lokalne mreže javnom IP adresom mrežne barijere. Maskirana IP adresa je, podrazumevano, javna IP adresa mrežne barijere, što znači da administrator ne mora da specificira NAT IP adresu, tj. adresu koja će se dodeliti paketima koji napuštaju lokalnu mrežu. To takođe znači da je uz pomoć maskiranja mnogo jednostavnije konfigurisati NAT ukoliko se koristi DHCP.

Maskiranje se konfiguriše:

- dodavanjem pravila za prevođenje u POSTROUTING lanac NAT tabele,
- dodavanjem pravila u FORWARD lanac filter tabele koje će omogućiti prosleđivanje paketa između interfejsa; na primer dozvolićete.

Primer 10.6. Linux server ima dva mrežna interfejsa:

- eth0 (111.22.33.44), koji služi za vezu sa Internetom,
- eth1 (192.168.1.1), koji je vezan na lokalnu mrežu 192.168.1.0/24.

Maskiranje IP paketa iz lokalne mreže 192.168.1.0/24 adresom javnog interfejsa mrežne barijere konfigurisaćete na sledeći način:

- dodaćete pravilo o maskiranju u POSTROUTING lanac NAT tabele

```
# iptables -A POSTROUTING -t nat -o eth0
           -s 192.168.1.0/24 -d 0/0 -j MASQUERADE
```

- dozvolićete prosleđivanje paketa koji pripadaju NEW, ESTABLISHED i RELATED konekcijama sa lokalne mreže ka Internetu

```
# iptables -A FORWARD -t filter -o eth0 -m state
           --state NEW,ESTABLISHED,RELATED -j ACCEPT
```

- prosleđivanje paketa koji pripadaju ESTABLISHED i RELATED konekcijama sa Interneta ka lokalnoj mreži

```
# iptables -A FORWARD -t filter -i eth0 -m state
          --state ESTABLISHED,RELATED -j ACCEPT
```

Mangle tabela

Mangle tabela se sastoji od tri lanca pravila (INPUT, OUTPUT i FORWARD) pomoću kojih se mogu postaviti neka svojstva paketa i pomoću kojih se paketi mogu obeležiti (markirati). Mangle tabele mogu se koristiti za promenu sledećih polja IP paketa:

- TOS – podešavanje ili promena polja Type Of Service (tip servisa),
- TTL – promena Time To Live polja paketa,
- MARK – markiranje paketa; korisno je ukoliko je potrebno da se izvrši neka operacija nad određenim paketom van iptables barijere. Markirane pakete prepoznaju programi poput iproute2 koji na osnovu vrednosti MARK polja izvršavaju različito rutiranje paketa. Takođe, na osnovu ovih obeležja može se obaviti kontrola dodele propusnog opsega određenim tipovima paketa.

Kao primer upotrebe mangle tabele, navodimo komandu kojom se može obaviti markiranje svih zaglavlja SMTP paketa brojem 1:

```
# iptables –t mangle –p tcp --dport 25 –j MARK –mark 1
```

Put paketa kroz Netfilter

Paket namenjen procesu (aplikaciji), koji se izvršava na računaru na kom funkcioniše i iptables, prolazi kroz sledeće korake pre nego što se isporuči aplikaciji:

- paket ulazi u interfejs → mangle prerouting → nat prerouting → odluka o rutiranju → mangle input → filter input → paket se šalje pravoj aplikaciji ili drugom računaru.

Paket sa lokalnog računara, odnosno sa računara na kom funkcioniše i iptables, prolazi kroz sledeće korake pre nego što se pošalje na mrežu:

- lokalni proces generiše paket → odluka o rutiranju → mangle output → nat output → filter output → mangle postrouting → nat postrouting → paket se šalje na mrežni interfejs.

Paket koji dolazi sa mreže, a koji je namenjen drugom računaru na mreži, prolazi kroz sledeće korake:

- paket ulazi u mrežni interfejs → mangle prerouting → nat prerouting → odluka o rutiranju → mangle forward → filter forward → mangle postrouting → nat postrouting → paket se šalje na odredišni mrežni interfejs.

Paket se može zaustaviti na bilo kom lancu mrežne barijere. Lanci i tabele su jedinstveni za sve mrežne interfejse.

Snimanje tekuće konfiguracije mrežne barijere

Osnovni problem konfigurisanja mrežne barijere sa velikim skupom pravila pomoću shell skriptova je brzina. Svaka iptables komanda u konfiguracionim skriptu će najpre izvući potpuni tekući skup pravila iz zaštićenog režima (engl. *kernel mode*), zatim dodati pravilo u određemi lanac, i, na kraju, vratiti skup pravila iz korisničkog (engl. *user mode*) u zaštićeni režim. Ukoliko u Netfilter lance dodate 10000 pravila, skript će 10000 puta prebaciti podatke iz zaštićenog u korisnički režim i vraćati ih natrag, što je vremenski zahtevna operacija.

Problem konfigurisanja pomoću skriptova možete rešiti pomoću komandi iptables-save i iptables-restore. Komanda iptables-save iščitava tekući skup pravila iz zaštićenog režima i prikazuje ih u specifičnom formatu na standardnom izlazu. Ukoliko konfiguraciju želite da snimite u datoteku, potrebno je da koristite redirekciju izlaza:

```
# iptables-save > /etc/iptables.ruleset
```

Komanda iptables-save, takođe, omogućava da se iz tekuće konfiguracije

mrežne barijere izvezu samo filterski lanci, nat ili mangle tabela:

```
# iptables-save -t tebela > ime_datoteke
```

Parametar “tabela” može imati vrednosti “filter”, “nat” ili “mangle”.

Komanda iptables-restore prihvata skup pravila u specifičnom formatu sa standardnog ulaza i prenosi ih u zaštićeni režim. Ukoliko konfiguraciju želite da ičšitate iz datoteke, potrebno je da koristite redirekciju ulaza:

```
# iptables-restore [--noflush] < /etc/iptables.ruleset
```

Komanda iptables-restore će obrisati kompletnu tekuću konfiguraciju mrežne barijere, pre nego što formira novu na osnovu datoteke koju ste naveli kao parametar. Ukoliko komandu navedete sa parametrom --noflush, prethodna konfiguracija neće biti obrisana, nego će se nadograditi pravilima pročitanim iz datoteke.

Prednost korišćenja ovih komandi u odnosu na konfiguraciju pomoću shell skripta je u tome što se sva pravila odjednom prenose iz korisničkog u zaštićeni režim.

Praćenje događaja

Ukoliko sumnjate da Vašu mrežnu barijeru niste dobro konfigurisali, poželjno je da pratite događaje vezane za Netfilter, tj. da povremeno pročitate i analizirate zapise o odbačenim paketima u datoteci /var/log/messages i na osnovu toga rekonfigurišete mrežnu barijeru. Praćenje zapisa o paketima koji prolaze kroz iptables lance pravila omogućeno je akcijom LOG. Obratite pažnju na sledeće dve činjenice:

- akcija LOG beleži u datoteci /var/log/messages događaj o paketu koji je zadovoljio pravilo koje je završeno akcijom LOG,
- ne beleži se da li je paket prihvaćen ili odbačen; u jednom pravilu se nalazi samo jedna akcija, akcija LOG ne odbacuje i prihvata pakete, već samo beleži informacije o njima u dnevniku događaja,
- sledeće pravilo u lancu se izvršava, bez obzira na to da li je paket zadovoljio ili nije zadovoljio pravilo sa akcijom LOG.

Pošto akcija LOG ne beleži informacije o tome da li je paket prihvaćen ili odbijen, ukoliko nešto sa njim ne uradite u sledećem pravilu u lancu, dnevnik će biti pun informacija i o prihvaćenim i o odbijenim paketima. Ukoliko, na primer, želite da pratite događaje vezane samo za odbačeni saobraćaj, onda je potrebno da nakon pravila koje beleži neželjene pakete (akcija LOG) dodate i pravilo koje odbacuje te pakete (akcija DROP).

Pretpostavite da ste mrežnu barijeru konfigurisali pravilima koja eksplicitno prihvataju određene pakete i da u lancima pravila nemate ni jedno pravilo sa DROP ili REJECT akcijom. Ukoliko želite da pratite događaje o odbačenim paketima, dodaćete sledeća pravila na krajeve lanaca:

```
# iptables -A OUTPUT -j LOG
# iptables -A OUTPUT -j DROP
# iptables -A INPUT -j LOG
# iptables -A INPUT -j DROP
# iptables -A FORWARD -j LOG
# iptables -A FORWARD -j DROP
```

Analizom dnevnika lako ćete utvrditi koje portove bi trebalo dodatno da otvorite kako biste, na primer, klijentima pružili usluge koje su im uskraćene neispravnom konfiguracijom mrežne barijere. Na primer, sledeći zapisi ukazuju na to da mrežna barijera blokira:

- odgovore na DNS upite (UDP port 53) upućene serveru 192.168.1.1 na lokalnoj mreži

```
Apr 11 10:11:21 secure kernel: IN=wlan0 OUT= MAC=00:06:25:09:
69:80:00:a0:c5:e1:3e:88:08:00 SRC=192.2.93.30 DST=192.168.1.1
LEN=220 TOS=0x00 PREC=0x00 TTL=54 ID=30485 PROTO=UDP SPT=53
DPT=32820 LEN=200
```

- NTP protokol (Network Time Protocol, UDP port 123)

```
Apr 11 11:21:43 secure kernel: IN= OUT=wlan0 SRC=192.168.1.1
DST=207.200.81.113 LEN=76 TOS=0x10 PREC=0x00 TTL=64 ID=0
DF PROTO=UDP SPT=123 DPT=123 LEN=56
```

10.4 Skeniranje portova - provera firewall konfiguracije

Najčešće korišćene tehnike napada na umrežene računare su skeniranje portova i analiza mrežnog saobraćaja. Ove tehnike, takođe, upotrebljavaju i administratori kako bi otkrili potencijalne sigurnosne propuste ili neželjeni saobraćaj na mreži. Skeniranje portova (engl. *port scanning*) je tehnika slanja ispravnih ili neispravnih (loše formatiranih) ICMP, UDP i TCP paketa računaru čiju sigurnost ispitujemo. Na osnovu odgovora računara na te pakete mogu se odrediti otvoreni portovi, dostupni mrežni servisi i vrsta operativnog sistema. Pomoću ove tehnike možete da proverite da li ste mrežnu barijeru ispravno konfigurisali, tj. da li su na njoj zatvoreni svi portovi osim onih koji moraju biti otvoreni.

Za skeniranje portova najčešće se koriste TCP paketi. 16-bitno polje Flags u zaglavlju TCP paketa sadrži numeričke podatke o uspostavljanju i praćenju jedne TCP konekcije. TCP konekcija se uspostavlja takozvanom *three-way handshake* procedurom:

- klijent najpre šalje SYN paket serveru,
- server odgovara SYN+ACK paketom ukoliko može da prihvati novu konekciju,
- klijent šalje ACK praket.

Nakon toga klijent i server mogu slati i primati podatke.

Napadač koji skenira portove šalje SYN pakete opsegu portova žrtve. Svi portovi za koje žrtva odgovori SYN+ACK paketom su otvoreni. Ukoliko želite da žrtva vaš pokušaj skeniranja ne zapiše u dnevnik događaja, pošaljite joj RST paket kojim se konekcija raskida – većina servera ne beleži ovakve događaje u dnevnik. TCP FIN (*stealth*) skeniranje je slanje FIN paketa opsegu portova žrtve; svi zatvoreni portovi zbog greške u implementaciji TCP protokola odgovaraju RST paketom, dok otvoreni portovi ne odgovaraju.

Za skeniranje se često koriste i neki loši ili drugačije formatirani TSR paketi. Na primer, napadač može poslati paket sa ACK paket žrtvi pre nego što je uopšte uspostavljena TSR konekcija. Na ovakve loše formatirane pakete različiti operativni sistemi različito reaguju pa se ovaj metod koristi

za određivanje tipa operativnog sistema na računaru koji je meta napada. Ovaj postupak se naziva i popisivanje (engl. *fingerprinting*). Ovaj metod je koristan potencijalnom napadaču jer, u zavisnosti od tipa operativnog sistemam može upotrebiti i odgovarajuću tehniku ili iskoristiti postojeći sigurnosni propust.

nmap

Jedan od najpoznatijih programa za skeniranje portova je nmap[59]. Nmap obezbeđuje različite metode skeniranja; pri tome možete odrediti opseg portova koje želite da skenirate, pojedinačne IP adrese, opseg adresa i vreme skeniranja. Detaljan spisak parametara možete dobiti ukoliko pogledate man stranicu za komandu map.

Program nmap se različito ponaša, u zavisnosti od toga da li je korisnik koji ga pokreće privilegovan (root, administrator) ili nije. Ako nmap pokreće privilegovani korisnik, podrazumevani način skeniranja je *TCP SYN stealth*; u suprotnom se koristi *TCP connect*, ilustrovan sledećim primerom:

```
root@agressor# nmap –sT 172.16.32.1
...
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
443/tcp  open  https
Nmap run completed -- 1 IP address scanned in 0.941 seconds
```

U primeru je skeniran računar sa IP adresom 172.16.321, a pronađeni su otvoreni TCP portovi 22, 80 i 443.

Opšti oblik komande nmap je:

```
nmap [opcije] [način_skeniranja] [-r opseg_portova] IP
```

Pri tome je moguće odabrati sledeće načine skeniranja:

59 Nmap je besplatan i isporučuje se uz većinu Linux distribucija. Ukoliko ga nema u distribuciji koju Vi koristite, preuzmite ga sa adrese http://www.insecure.org/nmap. Nmap postoji i u verzijama za druge operativne sisteme. Verzija za Windows ne omogućava skeniranje portova računara sa kog je pokrenut Nmap.

- –sS – TCP SYN stealth način skeniranja,
- –sT – TCP connect način skeniranja,
- –sU – UDP skeniranje,
- –sP – samo se proverava da li je udaljeni računar uključen,
- –P0 – NMAP ne proverava dostupnost računara pre skeniranja

  Ukoliko se navede parametar -P0, Nmap neće proveriti da li je udaljeni računar dostupan pre skeniranja portova (opcija je uvedena jer mnogi računari ne odgovaraju na ICMP echo request pakete).

Ukoliko se opseg portova ne navede, Nmap standardno skenira sve TSR ili UDP portove (što se preporučuje ukoliko testirate konfiguraciju mrežne barijere). Ukoliko Vas interesuju samo neki portovi, iskoristite opciju -r da odredite opseg portova od značaja. Sledeći primer ilustruje skeniranje portova u opsegu 60-40 svih računara koji pripadaju podmrežama 172.16.0.0 – 172.16.10.0:

```
root@agressor# nmap –r 60-400 172.16.0-10.*
```

Ukoliko želite da nmap odredi vrstu operativnog sistema na računaru koji se skenira, navedite opciju -O pre adrese ili imena računara. U tom slučaju, osim spiska portova biće navedena i vrsta operativnog sistema, kao i verzija jezgra, ako je u pitanju Linux.

```
root@agressor# nmap -O 172.16.32.5
...
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux Kernel 2.4.0 - 2.5.20
Uptime 0.010 days (since Fri Aug 13 19:49:02 2004)
```

10.5 Squid proksi server

U najširem smislu, proksi server (engl. proxy) je sloj između lokalne i spoljašnje mreže koji omogućava većem broju računara da dele jednu vezu ka Internetu i skladišti, tj. kešira podatke kako ubrzao pristup tim podacima sa lokalne mreže. Proksi serveri rade na aplikacionom sloju OSI modela, što

znači da svaki klijent mora biti pojedinačno konfigurisan (moraju se navesti adresa proksi servera i port na kome proksi servis radi pruža usluge).

Squid je nastao na osnovu projekta Harvest koji se, između ostalog, bavio i keširanjem pristupa objektima na mreži. Licenciran je GNU opštom javnom licencom, što znači da je besplatan. Podržava FTP, gopher i HTTP protokole, SSL, kontrolu pristupa i praćenje događaja, tj. vođenje evidencije o zahtevima. Za razliku od uobičajenih programa za keširanje, Squid razrešava sve zahteve u jednom neblokirajućem procesu koji spaja ulaz i izlaz. Squid, takođe, kešira DNS upite, meta podatke i "vruće" objekte u radnoj memoriji računara. Keširanje internet objekata (engl. *Internet object caching*) je način skladištenja traženih objekata[60] na sistemu koji je bliži mestu sa koga je zahtev upućen nego samom izvoru. Tako web pretraživači mogu da koriste podatke koji su lokalno uskladišteni.

Squid se sastoji od glavnog servisa (squid), programa za razrešavanje IP adresa na osnovu imena domena (dnsserver), nekih opcionih programa i upravljačkih alata. Prilikom pokretanja, squid podiže odredjeni broj dnsserver procesa (broj procesa se može konfigurisati), od kojih svaki može da izvrši jedan blokirajući DNS upit.

Squid se konfiguriše pomoću sledećih datoteka: /etc/squid/squid.conf, /etc/sysconfig/squid, /etc/logrotate.d/squid (rotacija dnevnika koji Squid generiše) i /etc/rc.d/init.d/squid (inicijalizaciona datoteka).

Proxy-caching režim rada i kontrola pristupa

Squid radi u jednom od dva režima: kešira podataka sa lokalnog Web servera, tj. ubrzava rad httpd servisa (*httpd-accelerator mode*), ili kešira podatke sa Interneta (*proxy-caching mode*).

U *httpd-accelerator* režimu rada Squid kešira podatake u smeru ka klijentu koji se nalazi van lokalne mreže (engl. *reverse web proxy cache*). Squid prima zahteve klijenata, daje ima podatke iz skladišta ukoliko je to moguće ili ih zahteva od Web servera. Ovim se može postići veliko rasterećenje, a samim tim i ubrzanje rada Web servera. Web server se, ukoliko je pokrenut na istom računaru na kom je pokrenut i Squid, mora

60 Podaci dostupni pomoću HTTP, FTP i gopher protokola.

konfigurisati tako da usluge pruža na portu 81[61].

U proksi-keš režimu rada svi korisnici mreže koristiće Squid za pristup Internetu. Squid u proksi-keš režimu rada usluge pruža na portu 3128; ako postoji potreba da se koristi neki drugi port (na primer, 8080), potrebno je u datoteku /etc/squid/squid.conf dodati parametar "http_port 8080" i u skladu sa tim podesiti klijente. Proksi-keš konfiguracija omogućava kontrolu pristupa, tj. mogu se primeniti posebna pravila koja određuju šta se može videti, čemu se može pristupiti, koji se podaci mogu preuzeti. Može se, takodje, kontrolisati i korisničko zauzeće protoka, trajanje veze, itd.

Pristup Internetu se ograničava kreiranjem lista za kontrolu pristupa, u kojima se najčešće koriste sledeći parametari:

- izvorišna IP adresa

```
acl ime_liste src izvorišna_IP_adresa/maska_podmreže
```

- odredišna IP adresa

```
acl ime_liste dst odredišna_IP_adresa/maska_podmreže
```

- izvorišni domen

```
acl ime_liste srcdomain izvorišni_domen
```

- odredišni domen

```
acl ime_liste dstdomain odredišni_domen
```

Navodimo jedan isečak iz datoteke /etc/squid/squid.conf koji se tiče kontrole pristupa:

```
acl lokalna_mreza src 192.168.1.0/255.255.255.0
acl proksi_server src 127.0.0.1/255.255.255.255
acl dozvoljeni_portovi port 80 443 21 1025-65535
acl CONNECT method CONNECT
acl sve_izvorisne_adrese src 0.0.0.0/0.0.0.0
http_access allow lokalna_mreza
```

61 Na primer, ukoliko se kao Web server koristi Apache, port se menja izmenom linije Port 80 u Port 81 u datoteci httpd.conf.

```
http_access allow proksi_server
http_access deny !dozvoljeni_portovi
http_access deny CONNECT
http_access deny sve_izvorisne_adrese
```

Ovakva konfiguracija servera će dozvoliti proksi serveru i svim računarima iz lokalne mreže 192.168.1.0/24 da pristupe Internetu na portovima 80 (http), 443 (https), 21 (ftp). Direktivom "http_access deny CONNECT" sprečen je spoljašnji pristup proksi serveru. Poslednjom direktivnom je blokirano sve što nije eksplicitno dozvoljeno.

10.6 Kućna rešenja – mrežne barijere za Windows XP

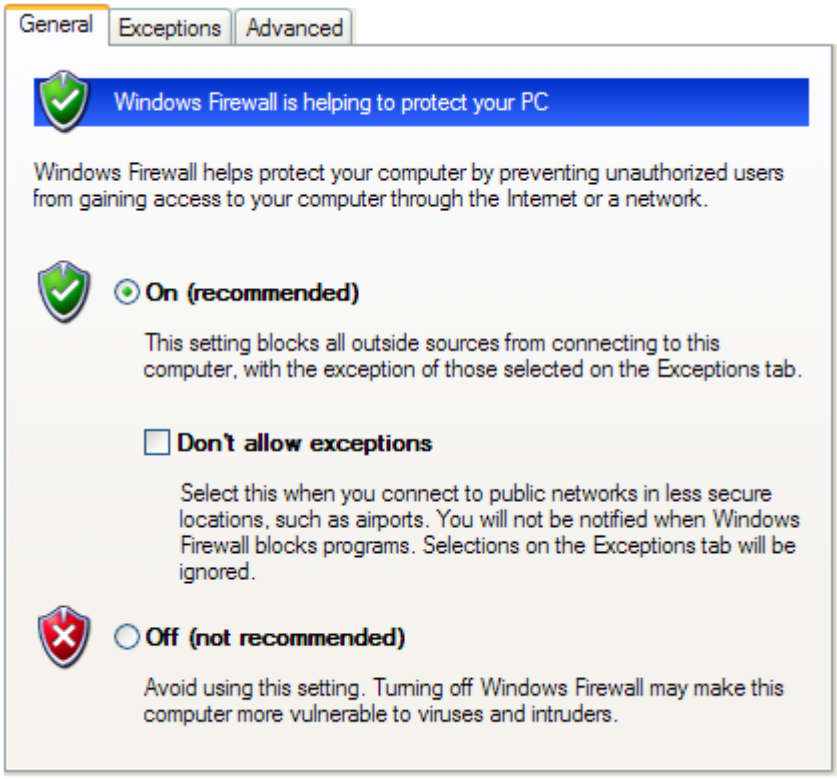
Programi kao što su iptables i squid imaju očiglednu upotrebnu vrednost u zaštiti lokalnih računarskih mreža. Osim zaštite lokalnih mreža, mrežne barijere obezbeđuju usluge zaštite i za korisnike radnih stanica. Na primer, iptables možete pokrenuti na svakom računaru koji radi pod Linux operativnim sistemom, bez obzira na to da li ga koristite kao server ili radnu stanicu. Mrežna barijera u tom slučaju sprečava napadača da pristupi podacima i servisima na Vašem računaru ili da izvrši neku zlonamernu akciju (na primer, skeniranje portova ili prekoračenje bafera).

Ukoliko kao operativni sistem na radnim stanicama koristite Windows XP, mrežna barijera vam je neophodna (prisetite se problema koje je svojevremeno velikom broju korisnika naneo Sasser crv[62] i shvatićete zašto). Microsoft je u Windows XP Service Pack 2 integrisao mrežnu barijeru; osim nje, veliki broj proizvođača nudi besplatne i komercijalne barijere koje vam mogu biti od pomoći. Opisaćemo neke koje se najčešće koriste.

62 Sasser koristi ranjivost Local Security Authority podsistema (LSASS), tj. prepunjenje bafera koje omogućava izvršavanje udaljenih procedura (RPC), čime se napadaču omogućava kontrola nad žrtvom. Crv izaziva prepunjenje bafera u LSASS.EXE; svaki put kada to toga dođe, operativni sistem se mora ponovo podići (*reboot*). Crv se širi pomoću AgoBot mobilnog agenta koji skenira portove ranjivih računara na mreži – svaki Windows računar na kome je otvoren TCP port 135, 139, 445, 593 ili UDP port 135, 137, 138 i 445 je potencijalna žrtva. Korisnici se prema preporuci kompanije Symantec mogu zaštititi filtriranjem saobraćaja na ovim portovima, kao i na svim portovima većim od 1024.

Windows firewall

Počev od Service Pack 2, Windows XP se isporučuje sa integrisanom mrežnom barijerom. Windows Firewall se pokreće pomoću odgovarajuće ikonice u Control Panel-u; nakon pokretanja dobićete dijalog sa slike 10.1.



Slika 10.1 – *Windows Firewall*

Windows Firewall obavlja sledeće funkcije:

- blokira dolazni saobraćaj, tj. otvaranje konekcija sa udaljenog računara (osim ka specificiranim portovima),
- sprečava programe sa vašeg računara da se ponašaju kao mrežni servisi (osim onih programa kojima Vi to eksplicitno dodelite u kartici Exceptions).

- nudi mogućnost da dozvolite prijem i slanje ICMP paketa,
- beleži događaje o odbačenim paketima (podrazumevano, u datoteci %systemroot%\pfirewall.log, osim ako administrator to ne promeni).

Realno, Windows Firewall nije dovoljno dobro rešenje jer, ako zanemarite ICMP, filtrira samo dolazni saobraćaj. Na Windows Firewall-u se ne može obaviti restrikcija odlazećih paketa koje lokalni računar generiše niti odrediti koje aplikacije mogu, odnosno ne mogu da pristupe Internetu. Ukoliko je računar inficiran, crv ili trojanski konj mogu iskoristiti otvorene portove i preuzeti zlonamerni kod sa Interneta. Windows Firewall je jednostavan filter dolazećih paketa i kao takav je koristan relativno maloj kategoriji kućnih korisnika koji "znaju šta rade", a iz nekih razloga ne žele da instaliraju dodatni softver. Ukoliko možete, koristite neku ozbiljniju mrežnu barijeru.

(Sunbelt) Kerio Personal Firewall

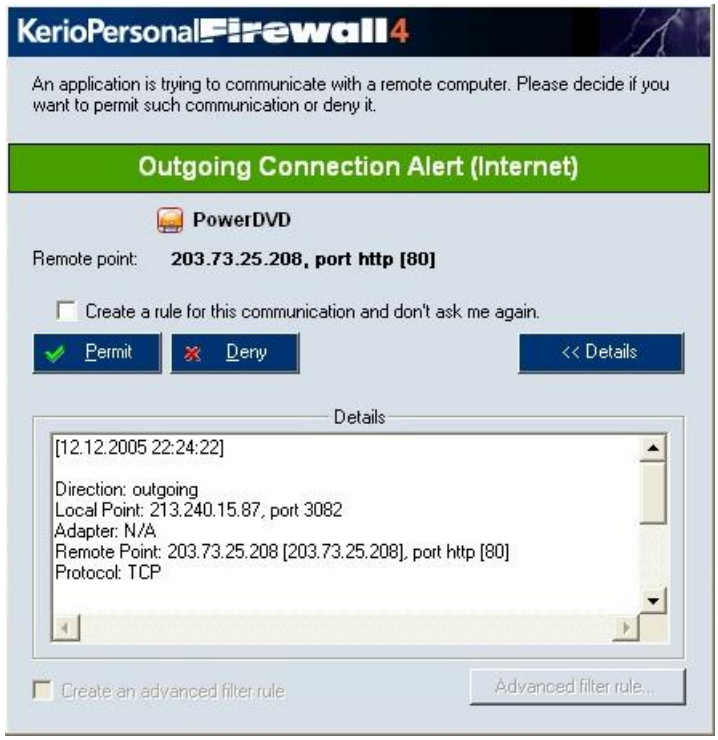
Kerio Personal Firewall (KPF) je kvalitetan hibrid mrežne barijere i sistema za detekciju napada namenjen radnim stanicama. KPF je karakterističan po prilagođenosti manje iskusnim korisnicima, uz istovremeno zadržavanje naprednih opcija za konfigurisanje paketa namenjenih iskusnijim korisnicima. KPF je dostupan u besplatnoj verziji, koja zadržava svu funkcionalnost mrežne barijere, sistema za detekciju upada u sistem i praćenja događaja, i komercijalnoj verziji[63], koja je proširena dodacima za udaljenu administraciju i filtriranje sadržaja (na primer, skriptova i reklamnih poruka).

KPF zahteva klijentski operativni sistem (Windows 9x, ME ili XP) i ne može se pokrenuti na serverskoj platformi (na primer, Server 2003). Korisniku se nude dva režima rada – jednostavan (*simple*) ili napredni (*advanced*). U jednostavnom režimu, KPF ne zahteva nikakva posebna podešavanja – dozvoljen je sav odlazeći saobraćaj, a dolazeći saobraćaj se blokira (isključujući odgovore na zahteve koji su poslati sa računara). Nivo zaštite se donekle povećava (što je očigledno), a korisnik može da vidi ko i šta želi da pristupi njegovom računaru. Ovaj režim rada je pogodan za korisnike koji se tek susreću sa mrežnim barijerama i nemaju iskustva u administraciji mrežnog okruženja. U svakom trenutku se može preći

63 "Probni rad" KPF u punoj verziji je 30 dana. Posle isteka ovog roka program nastavlja da funkcioniše kao besplatni KPF, tj. gubi funkcionalnost filtriranja sadržaja.

napredni režim rada, što je i preporuka svima koji ovladaju osnovama programa.

U naprednom načinu rada Kerio Personal Firewall morate naučiti kako da se ponaša (*learning mode*). Najveća prednost KPF je jednostavnost i jasnoća poruka i upita pomoću kojih se korisnik obaveštava šta program od njega zahteva. Kod svake poruke je precizno naglašeno koja aplikacija pokušava da pristupi kom portu (slika 10.2) i da li je u pitanju odlazeći ili dolazeći saobraćaj. Korisnik, takođe, dobija na uvid i IP adresu i puno kvalifikovano ime računara, odnosno domena (ako je moguće razrešiti DNS upit).



Slika 10.2 *– Aplikacija želi da pristupi mreži*

Osim toga, KPF nudi sistem zaštite na nivou samog sistema. Naime, KPF će zahtevati od korisnika da dozvoli pokretanja svake nove aplikacije. Moguće je postaviti pravilo kojim se trajno dozvoljava, odnosno zabranjuje rad nekim aplikacijama. Takođe, KPF obaveštava korisnika ukoliko neka

aplikacija pokušava da pokrene neku drugu aplikaciju, ili u slučaju da je došlo do zamene ili modifikacije izvršnog programa.

Kerio nudi i sitem zaštite od napada na računar, tj. IDS (engl. *Intrusion Detection System*) koji prati i beleži kada neko ispituje portove spolja. Program poseduje bazu sa mnogim poznatim načinima napada, koju je moguće obnoviti preko na stranici proizvođača programa. Moguće je i filtrirati podatke po sadržaju i blokirati reklamne poruke (engl. *advertisement*), kolačiće (engl. *cookie*), iskačuće prozore (engl. *pop-up*) i izvršavanje skriptova (JavaScripts, VBScripts, ActiveX).

Sve podatke (pokušaje upada, blokiranje kolačića, skriptova i iskačućih prozora) KPF beleži u dnevnik događaja. Na primer, statistika filtriranja po sadražaju je prikazana na slici 10.3)

Slika 10.4 *– Statistika filtriranja po sadržaju*

KPF se može pokrenuti sa podizanjem sistema (poželjno), ili naknadno pre izlaska na Internet (ukoliko koristite dial-up i ne želite da opterećujete sistem). Takođe, prilikom pregledanja Web stranica, držanjem pritisnutog tastera CTRL privremeno zaobilazimo KPF, što može biti korisno kod nekih Web stranica koje zahtevaju korišćenje kolačića, kao i upotrebu skriptova. Sjajna opcija ovog programa je dugme koje potpuno zaustavlja sav saobraćaj, što može biti veoma korisno ukoliko korisnik posumnja na upad.

Ako posedujete napredno znanje o računarskim mrežama, KPF možete konfigurisati kompleksnim skupom pravila, čime ostvarujete potpunu kontrolu nad mrežnim saobraćajem.

U vreme dok pišemo ovu knjigu (decembar, 2005), kompanija Kerio Technologies se u potpunosti posvećuje razvoju serverskig softvera (WinRoute Firewall i Kerio Mail Server) i prestaje da radi na daljem razvoju KPF; dalji razvoj KPF preuzima Sunbelt Software. Kako je najavljeno, Sunbelt KPF će i dalje biti dostupan u besplatnoj (ograničenoj) i komercijalnoj (punoj) verziji.

Zone Alarm (Pro)

Kompanija Zone Labs proizvodi nekoliko proizvoda namenjenih zaštiti radnih stanica (Zone Alarm, ZoneAlarm Pro, ZoneAlarm Internet Security Suite, ZoneAlarm Anti-Spyware, ZoneAlarm Antivirus, Imsecure). Pošto se u ovom poglavlju ograničavamo na analizi proizvoda koje možemo svrstati u kategoriju mrežnih barijera[64], opisaćemo ukratko Zone Alarm.

ZoneAlarm je definitivno najčešće korišćena "kućna" mrežna barijera – besplatan je (ukoliko ga ne koristite u komercijalne svrhe) i može se naći na skoro svim diskovima koji se isporučuju uz računarske časopise. Ukoliko na to dodate činjenicu da većina korisnika ne obraća mnogo pažnju na problematiku zaštite[65], prilično je jasno zašto je ZoneAlarm popularan.

ZoneAlarm pruža osnovne usluge mrežne barijere:

- filtriranje paketa na mrežnom i transportnom sloju,
- takozvani *stealth* režim (korisnik je nevidljiv na Internetu dok sam ne

64 U ZoneAlarm Pro je ugrađen alat za uklanjanje špijunskih programa, mehanizmi za sprečavanje pop-up prozora, zaštitu elektronske pošte i zaštitu računara na bežičnim mrežama – što, u suštini, i nisu funkcije mrežne barijere.

65 Zaštita je vrlo slična rezervnim kopijama podataka. Niko ne pravi backup dok prvi put ne izgubi 1GB korisnih podataka. Slično je i sa zaštitom – korisnik ne obraća pažnju na mrežnu barijeru, redovno osvežavanje antivirusnih definicija i zaštitu od špijunsih programa dok ne oseti posledice "na svojoj koži" (na primer, otvara jednu instancu Internet Explorer-a, a dodatno se otvore još četiri sa pornografskim sadržajem; pri tome, mreža se neprekidno koristi, pa je protok korisnih podataka deset puta manji, a iskorišćenje procesora ne pada ispod 75%).

inicira saobraćaj),

- kontrola aplikacija – restrikcije koje određuju koje aplikacije mogu, odnosno ne mogu pristupiti Internetu,
- blokiranje Internet saobraćaj posle određenog vremena neaktivnosti ili po aktiviranju čuvara ekrana (engl. *screensaver*).

ZoneAlarm obezbeđuje takozvanu dinamičku barijeru koji omogućava korisniku da postavi različite nivoe zaštite za različite zone (lokalna mreža ili Internet). Na primer, možete omogućiti deljenje datoteka i štampača u lokalnoj mreži, što znači da ćete prema računarima u lokalnoj mreži imati otvorene portove, dok će svi portovi biti zatvoreni za računare koji pripadaju Internet zoni. Ako se, eventualno, desi da neko pokuša da "provali" u Vaš računar, dobićete obaveštenje da je pokušaj napada blokiran (sa podacima kao što su IP adresa napadača, port i protokol koji je koristio). Inače, postavljanje sigurnosnih nivoa je lako (pomerite klizač gore ili dole zavisno od željenog nivoa zaštite), i to je jedino što je potrebno da znate biste inicijalno konfigurisali ZoneAlarm (poznavanje portova i protokola nije neophodno).

Aplikacija je jednostavna za korišćenje. Nakon podešavanja parametara zaštite, posao se svodi na povremeno izdavanje dozvole za pristup (klikom na dugme u porukama upozorenja) ili na eventualnu izmenu pravila koja ste ranije definisali. Kontrolni centar nudi nekoliko opcija za konfigurisanje programa i praćenje događaja:

- podešavanje nivoa zaštite (nizak, srednji i visok, tj. *stealth*) prema različitim zonama (zona kojoj se veruje, Internet zona i zona blokiranog saobraćaja); zona kojoj se veruje se definiše nabrajanjem IP adresa računara koji pripadaju toj zoni,
- zabrana ili dozvola pristupa mreži za različite aplikacije,
- trenutno prekidanje veze sa Internetom u slučaju nekog ozbiljnijeg napada,
- evidencija o količini poslatih i primljenih podataka tog dana i praćenje događaja vezanih za napade koje je ZoneAlarm sprečio.

Ukoliko možete da birate između KPF i ZoneAlarm (za sada ne vidimo razlog zašto ne biste mogli da birate), odaberite Kerio.

10.7 Pitanja i zadaci

10.1 Na Linux serveru koji obavlja funkcije rutiranja pokrenut je Netfilter. Server ima dva mrežna adaptera:

- eth0 (11.12.13.14) i
- eth1 (192.168.254.1).

Interfejs eth0 obezbeđuje vezu sa Internetom. Na eth1 interfejs Linux servera vezani (pomoću komutatora) Web i mail server iz lokalne mreže 192.168.254.0/24, čije su IP adrese 192.168.254.2 i 192.168.254.3 respektivno.

Koristeći program iptables, obezbedite filtriranje saobraćaja sa uspostavom stanja na serveru i konfigurišite odgovarajuće lance kojima se dozvoljava:

- HTTP i mail saobraćaj sa Interneta ka odgovarajućim serverima u lokalnoj mreži i
- ssh/telnet/ping saobraćaj ka mrežnoj barijeri sa adrese 192.168.254.10 (radna stanica administratora).

Sve ostalo eksplicitno zabranite!

Sprečite *spoofing*, tj. osigurajte da sav saobraćaj koji napušta lokalnu mrežu ima odgovarajuće adrese i zabranite napade tipa podmetanja žrtve i "*ping of death*" servera u lokalnoj mreži, kao i skeniranje portova mrežne barijere.

10.2 Na Linux serveru koji obavlja funkcije rutiranja pokrenut je Netfilter. Server ima dva mrežna adaptera:

- eth1 (172.16.32.254) i
- eth0 (11.12.13.14).

Dvadeset računara sa IP adresama koje pripadaju opsegu 172.16.32.1–20 vezano je u lokalnu mrežu 172.16.32.0/24. Lokalna mreža je pomoću komutatora vezana na interfejs eth1 servera. Veza sa Internetom obezbeđena je preko interfejsa eth0 Linux servera.

Koristeći program iptables, obezbedite da filtriranje saobraćaja sa uspostavom stanja na serveru i konfigurišite najrestriktivnije moguće

lance kojima se:

- dozvoljava kompletan saobraćaj sa lokalne mreže ka Internetu
- konfigurisanje mrežne barijere sa radne stanice administratora (isključivo preko šifrovanog kanala).

Sve ostalo eksplicitno zabranite.

10.3 Izmenite konfiguraciju mrežne barijere iz prethodna dva zadatka tako da mrežna barijera beleži događaje o svim odbačenim paketima.

10.4 Na Linux serveru koji obavlja funkcije rutiranja pokrenut je Netfilter. Server ima dva mrežna interfejsa:

- eth0 (11.12.13.14), koji služi za vezu sa Internetom i
- eth1 (172.16.0.1), koji je vezan na lokalnu mrežu 172.16.0.0/16.

Sastavite potpuni skup pravila kojima se obezbeđuje maskiranje IP paketa iz lokalne mreže 172.16.0.0/16 adresom javnog interfejsa mrežne barijere.

10.5 Na Linux serveru koji obavlja funkcije rutiranja pokrenut je Netfilter. Server ima dva mrežna interfejsa:

- eth0, koji služi za vezu sa Internetom i
- eth1, koji je vezan na lokalnu mrežu 192.168.1.0/24.

Interfejsu eth0 dodeljeni su odgovarajući IP alijasi 11.12.13.14 i 11.12.13.15.

Sastavite potpuni skup pravila kojima se obezbeđuje 1:1 prevođenje privatne adrese Web servera 192.168.1.100 sa lokalne mreže u javnu IP adresu 11.12.13.14 i 1:N prevođenje adresa sa privatne mreže 192.168.1.0/24 u javnu IP adresu 11.12.13.15.

11

Kontrola pristupa na Cisco uređajima

11.1 Osnovna konfiguracija Cisco rutera

U prethodnom poglavlju ukratko je opisana jedna softverska mrežna barijera – iptables firewall na Linux sistemima. Ukoliko želite rešenje koje možete smestiti u *rack* (orman sa mrežnom opremom), verovatno ćete se odlučiti za kupovinu hardverske mrežne barijere, kao što su Cisco PIX ili Watchguard Firebox. Takođe se i ruter preko kog ste vezani na Internet može pretvoriti u filtrirator paketa konfigurisanjem listi za kontrolu pristupa. Taj postupak ćemo u ovom poglavlju ukratko opisati.

Kao što je već rečeno, komunikacija između uređaja na različitim geografski udaljenim lokalnim mrežama ostvaruje se vezivanjem lokalnih mreža na mreže širokog područja, tj. WAN mreže. WAN mreže pokrivaju geografska područja veličine grada i države. WAN standardi uobičajeno definišu protokole fizičkog sloja i sloja veze OSI referentnog modela. Mrežni uređaji koji se koriste u WAN mrežama su ruteri, WAN svičevi, modemi i komunikacioni serveri (koji koncentrišu ulazne modemske linije).

Ruter se može jednostavno opisati kao namenski dizajnirani računar sa operativnim sistemom posebne namene. Preciznije rečeno, ruter je višeportni uređaj trećeg sloja OSI referentnog modela čija je funkcija rutiranje, tj. usmeravanje paketa po WAN mrežama. Ruter sadrži jedan ili više procesora, RAM memoriju (za smeštaj tabele rutiranja, ARP tabele), fleš memoriju (sadrži sliku IOS operativnog sistema), NVRAM memoriju (sadrži inicijalnu konfiguraciju rutera), ROM memoriju (čuva dijagnostičke rutine i program za pokretanje operativnog sistema iz fleš memorije). Sadržaji ROM, NVRAM i fleš memorije ne gube se nakon isključenja napajanja. Ruter takođe sadrži nekoliko interfejsa koji mogu biti fiksirani ili modularni. Na WAN i LAN interfejse priključuje se WAN ili lokalna preža, dok konzolni i AUX interfejs služe za konfiguraciju rutera (preko serijskog porta računara ili preko modema). LAN interfejsi rutera najčešće su RJ-45 utičnice i povezuju se *straight-through* UTP kablom na komutator (engl. *switch*) ili koncentrator (engl. *hub*). Najčešći oblik WAN interfejsa je serijski port preko koga se ruter povezuje na DCE.

Šta je Cisco IOS ?

IOS (Internetworking Operating System) je operativni sistem na Cisco ruterima i komutatorima iz Catalyst serije koji obezbeđuje osnovne funkcije rutiranja i komutacije ramova, kao i pouzdan i bezbedan pristup mrežnim resursima. Korisnički interfejs na Cisco ruteru je alfanumerički (Cisco ovakvu vrstu interfejsa naziva CLI – Command Line Editing) i može mu se pristupiti preko konzolnog porta povezanog na serijski port PC računara i terminal emulatora (HyperTerminal), preko modema povezanog na AUX port i preko telnet servisa (virtuelni terminal)[66].

Struktura IOS-a je hijerarhijska – podeljen je na više režima u kojima se unose različite komande. U takozvanim izvršnim, tj. "exec" režimima, IOS odmah izvršava zadatu komandu i prikazuju njen rezultat. U ovom režimu mogu se zadati komande za proveru konfiguracije rutera i komande kao što su ping i traceroute. Iz sigurnosnih razloga, postoje dva izvršna režima: privilegovani (privileged exec mode, koji je najčešće zaštićen lozinkom) i korisnički režim (user exec mode), kao podskup komandi privilegovanog režima. Na sledećem primeru obasjnićemo neke osnovne komande i navigaciju između režima:

```
myrouter> enable            ulazak u privilegovani režim
Password:                    unošenje lozinke
myrouter# show version      komanda zadata u priv.režimu
myrouter# show run          komanda zadata u priv.režimu
myrouter# disable           povratak u korisnički režim
myrouter>
```

U privilegovanom režimu dostupan je veliki broj show komandi za pregledanje tekuće konfiguracije rutera:

- show interfaces – podaci o interfejsima,
- show protocol – podaci o konfigurisanim protokolima trećeg sloja,
- show running-config – prikazuje tekuću konfiguraciju,
- show startup-config – prikazuje start-up konfiguraciju.

66 Pristup CLI preko virtuelnog terminala moguć je ukoliko se ruteru može pristupiti preko IP protokola (tj. ukoliko je ruteru dodeljena IP adresa i ukoliko je dostupan preko mreže).

Iz privilegovanog korisničkog režima može se preći u globalni konfiguracioni režim, u kome su dostupne komande za izmenu osnovne konfiguracije rutera (na primer, postavljanje raznih lozinki). Iz globalnog konfiguracionog režima može se ući u posebne režime za konfigurisanje interfejsa (na primer, dodela IP adrese interfejsu), protokola za rutiranje, virtuelnih terminala itd.

```
myrouter# configure terminal
myrouter(config)# interface serial 0/0
myrouter(config-if)# exit
myrouter(config)# router rip
myrouter(config-router)# end
myrouter# exit
myrouter>
```

Postavljanje lozinki

Ukoliko želite da postavljene liste za kontrolu pristupa na Vašem ruteu imaju ikakvog smisla, potrebno je da Vaš ruter zaštitite lozinkama. U suprotnom, napadač nezadovoljan kontrolom pristupa prijaviće se na ruter i promeniti ili ukloniti ACL.

Na Cisco ruterima postoje tri vrste lozinki: lozinka za konzolni pristup, lozinka za Telnet pristup i lozinka za prelaz u privilegovani režim rada (takozvana “enable” lozinka). Lozinka za konzolni pristup se postavlja u konfiguracionom režimu konzole na sledeći način:

```
myrouter# configure terminal
myrouter(config)# line console 0
myrouter(config-line)# password nova_lozinka
myrouter(config-line)# login
myrouter(config-line)# exit
myrouter(config)#
```

Komada login označava da ruter zahteva autentifikaciju prilikom pristupa, a komada password specificira lozinku. Ako se zada komada login a ne i komanda password, pristup ruteru preko konzole biće onemogućen. Lozinke za Telnet pristup se postavljaju kao i lozinke za konzolni pristup, ali se umesto u konfiguracioni režim konzole prelazi u konfiguracioni režim virtuelnih terminala:

```
myrouter(config)# line vty 0 4
myrouter(config-line)# password novalozinka
myrouter(config-line)# login
myrouter(config-line)# exit
```

Lozinke za konzolni pristup i pristup preko virtuelnih terminala vidljive su kao otvoreni tekst prilikom pregledanja konfiguracije rutera (komanda show run zadata u privilegovanom režimu). Komanda service password-encryption, zadata u globalnom konfiguracionom režimu, sprečava da se tokom pregledanja konfiguracije rutera ove lozinke ispišu u formi otvorenog teksta.

```
myrouter(config)# service password encryption
```

Komanda enable secret zadata u globalnom konfiguracionom režimu specificira lozinku za prelaz iz korisničkog u privilegovani režim rada:

```
myrouter(config)# enable secret brisi-sa-mog-rutera
```

Dodela IP adrese mrežnom interfejsu

Interface configuration mode je režim rada u kome se definišu parametri određenog interfejsa. Iz globalnog konfiguracionog režima u ovaj režim prelazi se komandom: interface tip [modul/]broj_interfejsa. Na primer, komandom interface FastEthernet 0/1 prelazi se u režim za konfigurisanje interfejsa označenog sa 1 na prvom Fast Ethernet modulu (označen sa 0). IP adresa se mrežnom interfejsu dodeljuje u konfiguracionom režimu tog interfejsa komandom: ip address adresa maska_podmreže. Kada interfejsu dodelite IP adresu, možete da ga aktivirate komandom no shutdown. Ilustrovaćemo primerom kako se to radi:

```
myrouter(config)# interface fastethernet 0/1
myrouter(config-if)# ip address 172.16.32.1 255.255.0.0
myrouter(config-line)# no shut
myrouter(config-line)# exit
myrouter(config)# interface serial 0/0
myrouter(config-if)# ip address 201.1.0.1 255.255.255.0
myrouter(config-line)# no shut
myrouter(config-line)# exit
```

Konfigurisanje protokola za rutiranje

IP je protokol koji se rutira, tj. rutabilni (engl. *routed*) protokol. Svaki IP paket sadrži zaglavlje (engl. *header*) u kome je zapisana izvorišna IP adresa, tj. IP adresa računara koji taj paket šalje i odredišna IP adresa, tj. IP adresa računara koji taj paket prima. Na osnovu odredišne IP adrese određuje se put (ruta) paketa kroz IP WAN mrežu ka odredištu. Objasnićemo to što jednostavnije možemo. Kada računar pošalje paket ka računaru na nekoj drugoj mreži, paket se šalje na adresu podrazumevanog prolaza (engl. *default-gateway*). Podrazumevani prolaz je ruter koji računarima na LAN mreži obezbeđuje pristup WAN mreži (najčešće, pristup Internetu). U RAM memoriji rutera nalazi se tabela za rutiranje (engl. *routing table*) koja opisuje šta ruter treba da uradi sa paketom koji je namenjen nekoj udaljenoj mreži, tj. na koji interfejs treba da prosledi taj paket. Ruter će, dakle, odrediti odredišnu adresu IP paketa i pronaći odgovarajući zapis u tabeli za rutiranje i na osnovu toga odrediti interfejs na koji će poslati paket. Paket se šalje ka sledećem ruteru u IP mreži, koji će na osnovu svoje tabele rutiranja proslediti dalje paket. Nakon nekoliko ovakvih skokova sa rutera na ruter, paket će stići do odredišne mreže; poslednji ruter u ovom nizu proslediće paket ka odredišnom računaru.

Kako ruteri formiraju svoje tabele za rutiranje?

- Administrator mreže unosi statičke rute i na taj način formira tabelu za rutiranje. Ovaj način je pogodan ukoliko je Vaš ruter vezan za ruter Vašeg Intetnet provajdera – tada se sav saobraćaj prosleđuje ka ruteru provajdera, tako da vam je najlakše da konfigurišete jednu statičku rutu (takozvani *stub network* slučaj). Statičke rute se, takođe, konfigurišu kada iz sigurnosnih razloga ne želite da razmenjujete informacije sa ostatkom sveta pomoću protokola za rutiranje. Navodimo primer konfigurisanja statičke rute koji nalaže ruteru da sve pakete namenjene mreži 66.0.0.0 255.0.0.0 prosledi na interfejs Serial 0/0:

  ```
  myrouter(config)# ip route 66.0.0.0 255.0.0.0 Serial 0/0
  ```

- Administrator konfiguriše protokole za rutiranje (engl. *routing protocol*). Ruteri međusobno komuniciraju pomoću protokola za rutiranje i na taj način šalju jedni drugima obaveštenja o IP mrežama

na koje su vezani, te na taj način formiraju svoje tabele rutiranja.

U algoritme za rutiranje spadaju, na primer, RIP v1 i v2, IGRP, EIGRP i OSPF. Bez zalaženja u detalje, navešćemo jednostavan primer konfiguracije RIP protokola:

```
myrouter(config)# router rip
myrouter(config-router)# network 67.0.0.0
myrouter(config-router)# network 210.1.1.0
myrouter(config-router)# exit
```

Najjednostavnije rečeno, ovako konfigurisan ruter će ruter na interfejse koji pripadaju mrežama 67.0.0.0 i 210.1.1.0 poslati deo svoje tabele za rutiranje (konkretno, oglašavanje sadrži mreže koje su specificirane network komandama i mreže koje su dobijene RIP protokolom od drugih rutera). Susedni ruteri će primiti oglašavanje i na osnovu njega ažurirati svoje tabele rutiranja. Myrouter će oglašavanja susednih rutera primiti sa svih interfejsa i na taj način ažurirati svoju tabelu za rutiranje.

11.2 Liste za kontrolu pristupa (za IP protokol)

Lista za kontrolu pristupa (engl. *access control list*, *ACL*) je niz naredbi kojima se određuje koji će IP paketi biti primljeni, odnosno poslati sa određenog interfejsa rutera. ACL se koriste za filtriranje paketa na osnovu IP adresa (čime se blokira ili dozvoljava saobraćaj sa neke mreže ili ka nekoj mreži) i/ili broja porta (čime se blokira ili dozvoljava pristup ka određenim servisima). ACL se takođe koriste za ograničavanje pristupa virtuelnim terminalima rutera sa određenog opsega IP adresa.

Svaka naredba u ACL definiše jedan uslov koji može dozvoliti ili zabraniti slanje ili prijem paketa. Prilikom prolaska paketa kroz interfejs za koji je vezana ACL, podaci iz zaglavlja paketa (na primer, izvorišna i odredišna adresa) upoređuju se sa prvim pravilom liste. Ako paket zadovoljava uslov, i ako je prvo pravilo završeno “permit” klauzulom, paketu se dozvoljava prolaz. Ako paket zadovoljava uslov, a pravilo je završeno “deny” klauzulom, paket se odbacuje. Ako paket ne zadovoljava uslov, prelazi se na sledeće pravilo. Ako paket ne zadovolji ni jedno pravilo, odbacuje se.

ACL se kreiraju tako što se ruteru navodi jedna po jedna naredba tipa:

```
myrouter(config)# access-list brojliste {permit|deny} pravilo
```

Raspored redova je očigledno bitan i zavisi od toga šta želite da postignete tom listom. Pravila se zadaju od najspecifičnijih ka najopštijim – na taj način možete, na primer, zabraniti pristup celom opsegu adresa osim jednom manjem delu tog opsega (što može biti i nekoliko računara). Obratite pažnju na to da neimenovane ACL nemaju mogućnost izmene pravila. Stoga je najbolje da se ACL najpre napiše u nekom editoru teksta a zatim prenese u terminal emulator (copy-paste). Napisana ACL se vezuje za konkretan interfejs rutera zadavanjem sledeće komande u konfiguracionim režimu tog interfejsa:

```
myrouter(config-if)# ip access-group brojliste {in|out}
```

Tekuće ACL se mogu pregledati komandom show access-list [brojliste] ili zadatom u privilegovanom režimu i proverom tekuće konfiguracije rutera (komanda show run). Takođe, komandom show ip interface možete pregledati koje su ACL postavljene na određeni interfejs. Shodno broju liste, za filtriranje IP paketa mogu se koristiti standardne ili proširene liste. Ova dva tipa ćemo ukratko i opisati.

Standardne ACL

Standardna ACL filtrira pakete samo na osnovu izvorišne IP adrese. Broj standardne IP filtarske liste mora pripadati opsegu 1-99 ili 1300-1399 (noviji IOS). Sintaksa za navođenje pravila je sledeća:

```
myrouter(config)# access-list broj_liste {permit|deny} srcIP
                  [srcWildcard] [log]
```

Parametri su sledeći: broj_liste je identifikator standardne IP ACL (1-99, 1300-1399), srcIP je adresa mreže ili računara sa kojom se upoređuje izvorišna adresa paketa, a srcWildcard je maska koja određuje koji se bitovi adrese porede[67] (ukoliko se ne navede, podrazumeva se 0.0.0.0, tj.

67 Prilikom poređenja IP adrese paketa sa IP adresom navedenom u pravilu, porede se samo oni bitovi koji odgovaraju bitovima wildcard maske koji su postavljeni na 0. Bitovi adresa koji se nalaze na onim mestima gde wildcard maska ima bit "1" se ne porede.

računar). Ukoliko se navede i parametar log, na svakih pet minuta se na konzoli ispisuje statistika o broju dozvoljenih ili zabranjenih paketa koji zadovoljavaju to pravilo.

Navodimo tri primera standardnih listi za kontrolu pristupa:

- standardna ACL koja propušta pakete sa izvorišnom adresom koja pripada opsegu 66.1.0.0 – 66.1.255.255, a odbacuje pakete koji potiču sa podmreže 66.1.10.0 i računaru 62.1.11.1

```
myrouter(config)# access-list 1 deny 66.1.10.0 0.0.0.255
myrouter(config)# access-list 1 deny 66.1.11.1
myrouter(config)# access-list 1 permit 66.1.0.0 0.0.255.255
```

- standardna ACL koja propušta sve pakete sa izvorišnom adresom 11.12.13.14 ili adresom koja pripada opsegu 67.1.0.0 – 67.1.0.255

```
myrouter(config)# access-list 2 permit 67.1.0.0 0.0.0.255
myrouter(config)# access-list 2 permit 11.12.13.14
```

- standardna ACL koja propušta sve pakete osim onih čija izvorišna adresa pripada opsegu 68.1.0.0 – 68.1.0.255

```
myrouter(config)# access-list 3 deny 68.1.10.0 0.0.0.255
myrouter(config)# access-list 3 permit any
```

Uzevši u obzir da standardne ACL filtriraju saobraćaj samo na osnovu izvorišnih IP adresa, u mrežama sa više rutera standardne ACL se postavljaju na ruteru koji je najbliži mreži sa koje je paket stigao. Pretostavite da je ruter vezan na Internet preko interfejsa Serial 0/0 i da Vi kao administrator želite da filtrirate dolazeći saobraćaj na način opisan u ACL 1. Postavićemo na interfejs Serial 0/0 listu 1 u ulaznom smeru:

```
myrouter(config)# interface serial 0/0
myrouter(config-if)# ip access-group 1 in
```

Standardne ACL se mogu iskoristiti i za definisanje pojedinačnih IP i opsega IP adresa sa kojih je dozvoljeno otvaranje telnet sesije ka ruteru. ACL se postavlja na linije virtuelnog terminala sledećom komandom koja se zadaje u režimu za konfigurisanje virtuelnih terminala:

```
myrouter(config)# line vty 0 4
myrouter(config-line)# access-class brojliste in
```

Poželjno je da sve virtuelne terminale zaštitite pomoću ACL.

Proširene ACL

Za razliku od standardnih, proširene ACL filtriraju saobraćaj na osnovu izvorišne i odredišne IP adrese, broja porta, protokola i tipa ICMP poruke. Zbog složenosti filtriranja, proširene ACL se postavljaju na ruteru koji je najbliži odredišnoj mreži.

Pravila u proširenim ACL se kreiraju komandom koja ima sledeću sintaksu:

```
myrouter(config)# access-list brojliste {permit|deny} prot
                  srcIP srcWildcard dstIP dstWildcard
                  [dstOp dstPort] [established]
```

Parametri imaju sledeće značenje:

- brojliste je identifikator proširene IP ACL (100-199, 2600-2699),
- prot je protokol mrežnog ili transportnog sloja (ip, icmp, tcp, udp),
- srcIP i dstIP su izvorišna i odredišna IP adresa paketa,
- srcWildcard i dstWildcard odgovarajuće wildcard maske uz izvorišnu i odredišnu IP adresu,
- dstOp je logički operator (eq, neq, lt, le, gt, ge) koji poredi odredišni port sa argumentom dstPort. Na primer eq 53 znači tačno port 53, a ge 1024 se odnosi na port 1024 ili veći. Broj porta dstPort može se u nekim slučajevima zameniti ključnim rečima. Na primer: ftp-data (20), ftp (21), smtp (25), http (80) i pop3 (110),
- established identifikuje segmente već uspostavljene TCP konekcije.

Navodimo dva primera proširenih listi za kontrolu pristupa:

- proširena ACL koja propušta IP pakete, čija izvorišna adresa pripada opsegu 192.168.1.0/24, a odredišna opsegu 15.1.1.0/24

```
myrouter(config)# access-list 101 permit ip 192.168.1.0
                  0.0.0.255 15.1.1.0 0.0.0.255
```

- proširena ACL, koja ka računaru 10.1.1.1 dozvoljava samo HTTP, SMTP i POP3 saobraćaj, a ka ostatku mreže 10.1.1.0 bilo kakav saobraćaj

```
myrouter(config)# access-list 101 permit tcp any
                  host 10.1.1.1 eq 80
myrouter(config)# access-list 101 permit tcp any
                  host 10.1.1.1 eq 25
myrouter(config)# access-list 101 permit tcp any
                  host 10.1.1.1 eq 110
myrouter(config)# access-list 101 deny ip any
                  host 10.1.1.1
myrouter(config)# access-list 101 permit ip any
10.1.1.0 0.0.0.255
```

Imenovane ACL

Imenovane ACL (podržane počev od IOS v11.2) razlikuju se u odnosu na klasične ACL po tome što je identifikator liste niz karaktera, a ne broj. Za razliku od klasičnih, moguće je brisanje pojedinačnih redova, tj. pravila iz Imenovanih ACL. Imenovane ACL kreiraju se komandom:

```
myrouter(config)# ip access-list {standard|extended} imeliste
```

Nakon izvršenja ove komande prelazi se u režim za konfigurisanje imenovanih ACL, u kome se definišu pojedinačni redovi pomoću komandi permit i deny. Sintakse ovih komandi su iste kao kod i sintakse komandi za definisanje standardnih i proširenih ACL.

Kao primer, kreiraćemo listu Apache kojom se ka Web serveru dozvoljava jedino HTTP i HTTPS saobraćaj:

```
myrouter(config)# ip access-list extended Apache
myrouter(config-ACL)# permit tcp any host 172.16.32.1 eq 80
myrouter(config-ACL)# permit tcp any host 172.16.32.1 eq 443
```

11.3 Pitanja i zadaci

11.1 Jedan Cisco ruter ima dva mrežna interfejsa:

- Serial 0/0 (11.12.13.14) i
- FastEthernet 0/0 (192.168.254.1).

Serial 0/0 interfejs rutera obezbeđuje vezu lokalne mreže sa Internetom. Na FastEthernet 0/0 interfejs vezani su, pomoću komutatora, Web i mail server iz lokalne mreže 192.168.254.0/24, čije su IP adrese 192.168.254.2 i 192.168.254.3 respektivno, i radna stanica administratora.

Sastavite ACL kojima se dozvoljava:

- HTTP i mail saobraćaj sa Interneta ka odgovarajućim serverima u lokalnoj mreži i
- ssh/telnet/ping saobraćaj ka mrežnoj barijeri sa radne stanice administratora (čija je IP adresa 192.168.254.10).

Sve ostalo eksplicitno zabranite.

11.2 Jedan Cisco ruter ima dva mrežna interfejsa:

- FastEthernet 0/0 (172.16.32.254)
- Serial 0/0 (11.12.13.14)

Dvadeset računara sa IP adresama koje pripadaju opsegu 172.16.32.1–20 vezano je u lokalnu mrežu 172.16.32.0/24. Lokalna mreža je pomoću komutatora vezana na FastEthernet 0/0 interfejs rutera. Veza sa Internetom obezbeđena je preko Serial 0/0 interfejsa.

Sastavite ACL kojima se dozvoljava:

- kompletan saobraćaj sa lokalne mreže ka Internetu i
- konfigurisanje rutera preko virtuelnih terminala isključivo sa radne stanice administratora.

Sve ostalo eksplicitno zabranite.

12

Sigurnost bežičnih mreža

12.1 Vrste bežičnih mreža

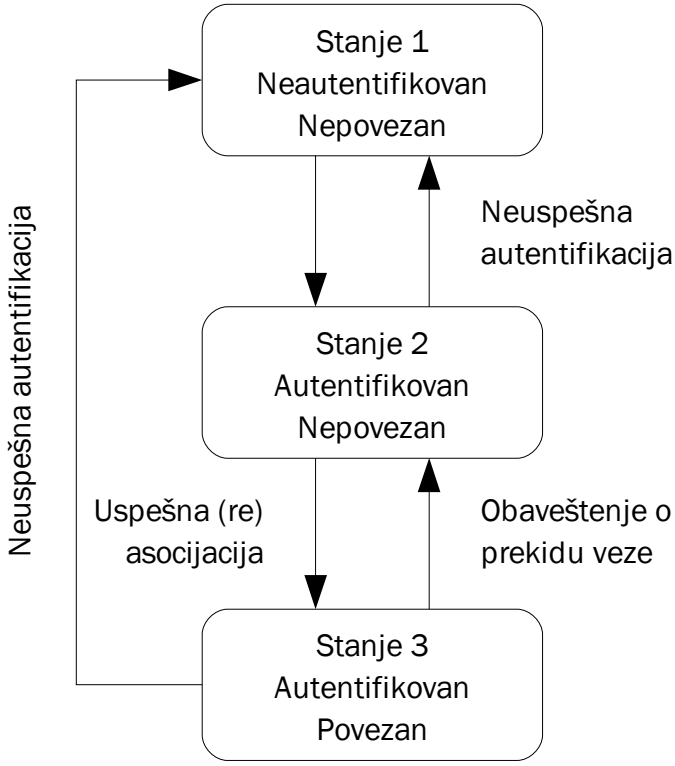
Postoje dve vrste bežičnih mreža:

- Ad hoc mreže (*Independent Basic Service Set*, IBSS) – mreža ovog tipa uspostavlja se direktno između dva ili više računara smeštenih u relativno malom prostoru zbog male snage njihovih antena. Mreže ovog tipa se, uglavnom, ne koriste.

- Strukturirane mreže (*Basic Service Set*, BSS) – klijenti komuniciraju preko pristupnih tačaka (engl. *acces point*), tj. preko uređaja koji omogućavaju pristup mreži. Mreže ovog tipa mogu imati veći broj korisnika na većem geografskom prostoru od Ad hoc mreža.

Mikroćelija je prostor pokriven signalom jedne pristupne tačke. Pristupna tačka upravlja kompletnim mrežnim saobraćajem u mikroćeliji. Područje pokrivanja mreže može se povećati dodavanjem drugih pristupnih tačaka koje su povezane u Eternet mrežu pomoću komutatora. Preporučuje se da proširena područja uključuju 10 – 15% preklapanja signala kako bi korisnici mogli da prelaze iz jedne u drugu ćeliju bez gubljenja signala.

Klijent mora uspostaviti vezu sa pristupnom tačkom kako bi mogao biti član mreže. Proces pristupanja mreži može se prikazati konačnim automatom na slici 12.1. Prelazak iz stanja u stanje obavlja se razmenom upravljačkih okvira (engl. *managment frames*) između klijenta i pristupne tačke. Sve pristupne tačke u fiksnim vremenskim intervalima šalju upravljački okvir *beacon* (što na engleskom jeziku znači svetionik, odnosno upadljiv znak) koji klijentima signalizira postojanje pristupne tačke. Klijent koji želi da pristupi mreži osluškuje signal i čeka upravljačke okvire koje šalju sve pristupne tačke koje su mu u dometu. Klijent bira pristupnu tačku kojoj želi da se pridruži i sa njom razmenjuje nekoliko upravljačkih okvira; nakon toga oni ulaze u proces pridruživanja. Ukoliko prođe autentifikaciju, klijent prelazi u drugo stanje konačnog automata. Klijent zatim šalje upravljački okvir kojim zahteva pridruživanje mreži (tj. ćeliji) i tek kada mu pristupna tačka odgovori sa drugim upravljačkim okvirom, on prelazi u treće stanje i pristupa mreži.



Slika 12.1 *– Pristupanje mreži: konačni automat*

12.2 Šta trenutno "štiti" bežične mreže ?

Iako su neki mehanizmi zaštite definisani standardima, činjenica je da su bežične mreže najslabija sigurnosna karika neke organizacije. Standardi ne mogu zadovoljiti tri osnovna sigurnosna zahteva: pouzdana autentifikacija korisnika, zaštita privatnosti i autorizacija korisnika. Na primer, *Wired Equivalent Privacy*[68] (WEP), kao osnovni zaštitni mehanizam, sadrži značajne sigurnosne propuste, a pitanja kao što su razmena ključeva i robusni način autentifikacije korisnika su još uvek otvorena, tj. nerešena. Većina organizacija koje imaju bežične mreže oslanjaju se na zaštitu

68 Wired Equivalent Priacy bi trebalo da znači "privatnost ekvivalentna onoj koju pružaju žičane mreže". Međutim, uzevši u obzir vrlo nivo zaštite koji pruža, može se prevesti i kao "ožičeno = privatno".

definisanu standardima ili čak i ne koriste nikakve sigurnosne mere.

Sigurnost na fizičkom nivou – ograničavanje propagacije signala

Računari se u bežične mreže povezuju preko antena – omnidirekcionih, ravnih ili usmerenih (Yagi). Antene različitog tipa pokrivaju signalom različita područja i utiču na veličinu ćelija. Pošto pristupne tačke imaju ograničeno područje pokrivanja, ćelije imaju ograničenu veličinu. Prilikom dizajniranja same bežične mreže u nekom području, potrebno je da projektant:

- detaljno analizira to područje i utvrdi optimalnu vrstu antena koje će se koristiti i potrebnu snagu, vodeći računa o svim ograničenjima,
- uzme u obzir da je frekvencija 2.4 GHz, koju koriste mreže 802.11b i 802.11g nelicencirana, tj. da može doći do interferencije signala sa bežičnim telefonima i drugim uređajima koji rade na istoj frekvenciji, što može dovesti do stanja koje podseća na posledicu DoS napada,
- pretpostavi da potencijalni napadač ima bolju i osetljiviju opremu (od opreme koja je propisana standardima) koja, praktično, proširuje područje mreže, a time i potencijalne opasnosti.

Ukoliko se o tome ne vodi računa, moguće je da signal dopre van fizičkih granica organizacije kojoj mreža pripada, što otvara mogućnosti za takozvani napad s parkirališta (eng. *parking lot attack*).

SSID

SSID je ime mreže koju pokriva jedna ili više pristupnih tačaka. Prema najčešće korišćenom načinu pristupa, klijent bira kojoj će se mreži pridružiti na osnovu SSID koji pristupna tačka šalje u okviru signalnog upravljačkog okvira (*beacon*). Drugi način pristupa omogućava da se SSID iskoristi kao zaštitni faktor - pristupne tačke se mogu podesiti tako da ne šalju SSID unutar kontrolnog okvira. Tada klijent koji želi da pristupi mreži mora imati isti SSID kao i mreža kojoj se želi pridružiti. Ukoliko klijent nema ispravan SSID, pristupna tačka odbacuje sve kontrolne okvire koje klijent šalje, što znači da on ne može proći postupak pridruživanja mreži.

SSID teoretski izgleda kao dobar način kontrole pristupa. U praksi, situacija je sledeća: svi kontrolni okviri se šalju kao otvoreni tekst. To znači da napadač može, osluškujući komunikaciju unutar mreže, da "uhvati" kontrolne okvire koje razmenjuju pristupne tačke u komunikaciji sa drugim korisnicima mreže. Napadač tako može saznati SSID mreže i neovlašteno se pridružiti mreži.

Autentifikacija korisnika mreže

Klijent mora da se autentifikuje ukoliko želi da pristupi mreži. Standardi definišu dva načina autentifikacije:

- autentifikacija otvorenog sistema (*Open System Authentication*) - podrazumevani način provere korisnika po standardu 802.11. Kako samo ime sugeriše, dopušta svakome ko zatraži da se pridruži mreži, tj. ne predstavlja nikakvu zaštitni mehanizam,

- autentifikacija zasnovana na deljenom ključu (engl. *Shared Key)* – zasniva se na pretpostavci da i klijent i pristupna tačka poseduju identičan deljeni ključ koji im je distribuiran sigurnim kanalom. Pristupna tačka šalje klijentu izazov (*challenge*) koji klijent šifruje svojim ključem i šalje pristupnoj tački. Pristupna tačka dešifruje primljenu poruku svojim ključem i ukoliko dobije identični tekst koji je i poslala, smatra da je klijent prošao proces autentifikacije i da se može pridružiti mreži. Ukoliko klijent želi da proveri pristupnu tačku, onda on čini isto samo u obrnutom smeru.

Veliki nedostatak autentifikacije zasnovane na deljenom ključu je slanje upravljačkih okvira u obliku otvorenog teksta preko nesigurnog medijuma. Napadač može presresti upravljačke okvire sa otvorenim tekstom i šifratom i na osnovu toga doći do ključa koji se koristio.

WEP

WEP (*Wired Equivalent Privacy*) je definisan u standardu 802.11 i za cilj ima sledeće:

- poverljivost poruka – osnovna namena je spečavanje prisluškivanja mrežnog saobraćaja (engl. *evesdropping*),
- kontrola pristupa – pristupne tačke mogu zabraniti klijentima pristup mreži ukoliko uspešno ne prođu proces autentifikacije,
- integritet poruka – dodatno polje u okviru služi za proveru integriteta samog okvira.

WEP se koristi radi zaštite podataka na sloju veze OSI modela.

Integritet poruke obezbeđuje se operacijom zaštitnog kodiranja CRC-32 algoritmom, čime se dobija čeksuma koja se dopisuje na kraj podatka koji se žele zaštititi. U osnovi algoritma je 32-bitni polinom, 0x04C11DB7 (zapisan heksadecimalno). Osnovna namena algoritma CRC-32 je očuvanje integriteta podataka u komunikacionom kanalu sa smetnjama i šumom. Međutim, CRC-32 je loš izbor ukoliko se primenjuje u kriptografske svrhe jer ne štiti u potpunosti integritet poruke (moguće je promeniti određene bitove tako da se to ne detektuje na prijemnoj strani). Prikladniji izbor bi bila neka heš funkcija, na primer SHA-1 ili MD-5.

Za šifrovanje tela okvira koristi se simetričan protočni algoritam RC4 (opisan u 4. polavlju). Da Vas podsetimo, sam algoritam se sastoji iz dva dela: raspoređivača ključeva i generatora pseudo-slučajnih brojeva. Algoritam za raspoređivanje ključeva pretvara slučajno generisani ključ u početnu permutaciju koju koristi generator pseudo-slučajnih brojeva kako bi proizveo pseudo-slučajan niz bitova na izlazu. Generator pseudo-slučajnih u petlji izvršava četiri jednostavne operacije u kojima je *i* brojač, dok se *j* povećava pseudoslučajno; nakon toga se u polju permutacija *S* zamenjuju dve vrednosti na koje pokazuju *i* i *j* i kao izlaz daje vrednost *S* na koju pokazuje S_i+S_j. Svaki član niza *S* se menja najmanje jednom, što znači da se cela permutacija *S* menja vrlo brzo.

Sigurnost WEP-a je zasnovana na tajnosti ključa pomoću kog se telo okvira poruke šifruje na relaciji pristupna tačka – klijent. Konkretno, snaga WEP-a se temelji na težini otkrivanja tajnog ključa pomoću napada grubom silom; međutim, napadi koji se izvode na WEP su znatno brži i uspešniji.

Upravljanje ključevima

Ovaj bitan detalj nije definisan standardom, već je njegovo rešavanje prepušteno proizvođačima mrežne opreme. Kao rezultat, samo nekoliko najvećih proizvođača mrežne opreme ugradilo je u svoje uređaje bilo kakav način upravljanja ključevima. Nažalost, proizvođači ne iznose dovoljno informacija o nivou sigurnosti koju su ugradili u svoje proizvode, a neki čak u opisu svojih rešenja navode da koriste protokole i metode sa dobro poznatim sigurnosnim propustima, kao što je, na primer, Diffie-Hellmanov protokol koji je osetljiv na napad tipa "čovek u sredini".

12.3 Sigurnosni propusti i napadi na WEP

Prilikom komunikacije klijenta i pristupne tačke, podaci se šalju u obliku kontrolnih okvira čija zaglavlja nisu šifrovana, što znači da napadač lako može doći do inicijalizacionog vektora koji je korišćen za šifrovanje. Napadač koji "uhvati" dva šifrata šifrovana istim inicijalizacionim vektorom dobiće informacije o samim porukama. Ako je *v* inicijalizacioni vektor, a *k* ključ, onda je:

- $C_1 \oplus C_2 = (P_1 \oplus RC4(v, k)) \oplus (P_2 \oplus RC4(v, k)) = P_1 \oplus P_2$.

Ukoliko je napadaču poznata jedna reč otvorenog teksta, drugu reč može automatski dobiti. Ukoliko otvoreni tekst sadrži dovoljno meta-informacija, napadač može otkriti P_1 i P_2 poznajući samo $P_1 \oplus P_2$. Postoji mnogo načina otkrivanja prikladnih kandidata za otvoreni tekst; na primer, mnogi protokoli mrežnog i transportnog sloja imaju jasno definisana i predvidljiva polja u zaglavlju paketa. Takođe, što je veći broj poznatih šifrovanih reči, veća je i verovatnoća da će napadač otkriti podatke. Znači, da bi napad ovog tipa uspeo, napadač mora imati podatke šifrovane istim inicijalizacionim vektorom i mora barem delimično poznavati otvoreni tekst.

Pošto zamena ključa nakon svakog poslatog okvira nije moguće rešenje ovog problema, WEP standard preporučuje (ali ne insistira) da se nakon svakog okvira promeni inicijalizacioni vektor. Mnogi proizvođači mrežne opreme slede ovu preporuku, ali su neki to učinili na veoma loš način. Na primer, većina PCMCIA bežičnih mrežnih kartica prilikom svakog pokretanja

postavlja inicijalizacioni vektor na nulu i povećava ga za jedan nakon svakog poslatog okvira. Napadaču je u tom slučaju dovoljno da zna samo deo vektora sa početka i na taj način može doći do nekih podataka.

Dodatno, u arhitekturi WEP-a postoji propust koji pogađa sve implementacije protokola i time izlaže korisnika ozbiljnoj opasnosti ponovne upotrebe ključa. Polje u kojem je upisana vrednost inicijalizacionog vektora je dužine 24 bita, što znači da postoji 2^{24}=16.777.216 različitih vrednosti tog vektora. Ukoliko uzmete u obzir činjenicu da će prosečna stanica koja šalje okvire veličine 1500 bajta pri prosečnoj brzini od 5 Mbps iscrpeti sve vektore za manje od pola dana, shvatićete da je ovo ozbiljan propust.

Što se napada tiče, postoji nekoliko vrsta napada na WEP, a oni se grubo mogu klasifikovati u dve kategorije:

- pasivni napadi – napadač samo prisluškuje komunikaciju korisnika sa mrežom. U ove napade spadaju analiza saobraćaja i pasivno prisluškivanje;

- aktivni napadi – napadač aktivno utiče na mrežni saobraćaj ubacivanjem svojih podataka, lažiranjem komunikacije između klijenta i pristupne tačke, zagušivanjem saobraćaja na mreži ili neovlašćenim korišćenjem mrežnih resursa. U ove napade spadaju ponavljanje inicijalizacionog vektora, obrtanje bitova, čovek u sredini, krađa sesije i napad ponavljanjem paketa.

Pasivni napadi na WEP

Analiza saobraćaja je najjednostavniji pasivni napad – napadač prisluškuje mrežu i prati broj i veličinu paketa u mreži. Za ovu vrstu napada napadaču je potrebna zadovoljavajuća antena, mrežna kartica koja radi u režimu slušanja (ne šalje nikakve pakete) i softver koji će analizirati veličinu i broj paketa. Pomoću toga napadač može saznati tri osnovne informacije:

- količinu mrežnog saobraćaja – pojava naglog povećanja saobraćaja na mreži obično ukazuje na neki bitan događaj,
- fizičku lokaciju pristupnih tačaka – pomoću usmerene, tj. Yagi antene u kombinaciji sa GPS (*Global Positioning System*) napadač metodom

triangulacije može doći do fizičke lokacije pristupne tačke ili centra bežične mreže,

- vrste protokola koji se koriste na mreži.

Pasivno prisluškivanje je takođe relativno jednostavan napad jer napadač samo osluškuje mrežu. Jedini uslov za uspešan napad ovog tipa je pristup signalu mreže, a tu do izražaja dolazi fizička sigurnost mreže, tj. koliko je projektant vodio računa o prostiranju signala pristupnih tački u prostoru. Prema nekom standardnom scenariju napada, napadač osluškuje mrežu i čeka da se ponovi isti inicijalizacijski vektor i tako, na prethodno opisani način, dolazi do vrednosti $P_1 \oplus P_2$. Nakon toga, napadač na osnovu poznatih reči jedne poruke može odmah doći do druge poruke. Ukoliko napadač ne zna ni jednu poruku, koristeći informacije o protokolima dobijene napadom analiza saobraćaja, može pretpostaviti neke konstantne delove poruka i tako doći do podataka. Na primer, izvorišna i odredišna IP adresa koje su fiksne dužine nalaze se na fiksnoj udaljenosti od početka paketa. TCP protokol, takođe, ima na tačno određenom mestu zapisan izvorišni i odredišni port. Isti se princip može primeniti i na zaglavlja raznih aplikacija koje imaju potpuno definisan oblik (na primer, HTTP protokol).

Aktivni napadi na WEP

Jedan od mogućih scenarija napada ponavljanjem inicijalizacionog vektora (engl. *IV replay attack*) je sledeći:

- napadač preko Interneta šalje poruku klijentu koga želi napasti,
- napadač zatim pažljivo prisluškuje mrežu i čeka da pristupna tačka pošalje klijentu poruku sa poznatim tekstom,
- napadač “skida” kriptografsku zaštitu sa poruke jer mu je poznat inicijalizacioni vektor šifrovane poruke.

Nakon toga napadač može dodavati svoje podatke u šifrovane pakete. Osnovna pretpostavka ovoga napada je da se inicijalizacioni vektor i WEP ključ mogu ponavljati sve dok mreža ne prihvati da je to ispravan paket. Jednom kada napadač dobije niz bitova kojim je paket šifrovan, on može taj niz primeniti na druge podatke koje će sam ubaciti u mrežu.

Napad obrtanjem bitova (engl. *bit-flipping attack*) iskorišćava slabost vektora integriteta poruke (ICV). Iako je veličina podatka koji šifrovani paket nosi promenljiva, mnogo elemenata se nalazi na fiksiranim mestima u paketu. Napad se može opisati sledećim koracima:

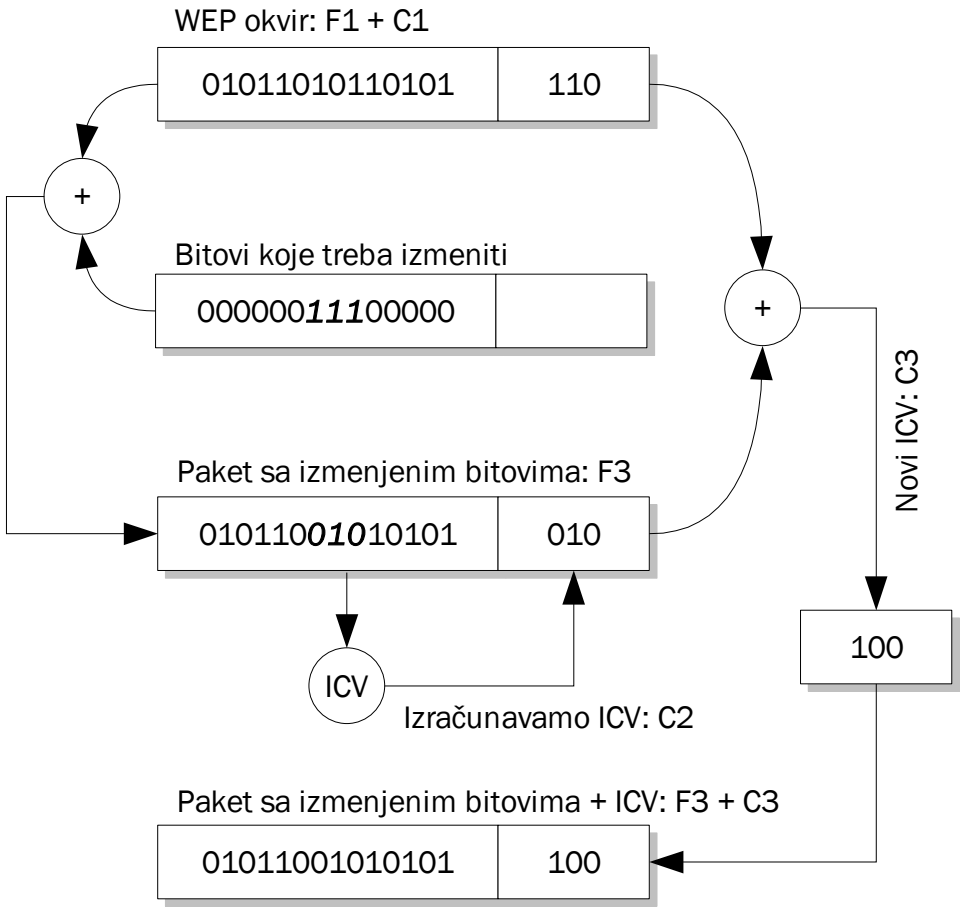
- Napadač (1) prisluškuje okvire na mreži, (2) preuzima jedan okvir sa mreže i menja vrednosti nekoliko slučajno odabranih bitova unutar polja IP paketa koje sadrži poruku, (3) menja sadržaj polja u kom se nalazi vektor integriteta poruke (ICV) i (4) šalje izmenjeni paket na mrežu.

- Prijemna strana (klijent ili pristupna tačka) prima paket i računa vektor integriteta poruke na osnovu podataka koji se nalaze u paketu. Prijemna strana upoređuje izračunatu i dobijenu vrednost vektora integriteta poruke (koja je u polju ICV paketa). Ukoliko su ta dva vektora ista, prijemna strana prihvata izmenjeni paket, skida enkapsulaciju i predaje ga višem, trećem, sloju OSI modela. Pošto je napadač zamenio bitove IP paketa, provera integriteta na mrežnom sloju nije uspešna i zbog toga se generiše predvidljiv izveštaj o greški.

- Napadač prisluškuje saobraćaj na mreži očekujući predvidljivi šifrovani odgovor. Kada primi odgovor, napadač dolazi do niza bitova ključa i može ga iskoristiti za prethodno opisan napad.

Uspeh ovog napada zasnovan je na propustu vektora integriteta. Pošto se ovaj vektor nalazi u šifrovanom delu paketa, postavlja se pitanje kako napadač može uspešno izmeniti vrednost bitova? Pogledajte sledeći algoritam i sliku 12.2:

- napadač "hvata" paket kom želi izmeniti ICV(C_1),
- napadač generiše paket jednake dužine sa postavljenim bitovima (F_2),
- treći paket se dobija kao rezultat XOR prva dva paketa: $F_3 = F_1 \oplus F_2$,
- napadač računa ICV za treći paket (C_2),
- vektor integriteta koji će se umetnuti dobija se pomoću XOR nad vektorima primljenog paketa i paketa koji se dobio kao XOR generisanog i originalnog paketa ($C_3 = C_1 \oplus C_2$).



***Slika 12.2** - Određivanje ICV paketa*

Napad čovek-u-sredini (engl. *man-in-the-middle attack*) može se iskoristiti za čitanje ili modifikaciju podataka. Napad je zasnovan na propustu u standardu koji ne omogućava obostranu autentifikaciju klijenta i pristupne tačke. Napadač se postavlja u komunikacioni kanal između klijenta i pristupne tačke i presreće njihovu komunikaciju, a zatim izvodi napad u nekoliko koraka:

- napadač prekida komunikaciju klijenta i pristupne tačke i ne dopušta klijentu da ponovno uspostavi vezu sa pristupnom tačkom,

- klijent nastoji uspostaviti vezu sa pristupnom tačkom, ali pošto to ne

može obaviti, uspostavlja vezu sa napadačevim računarom koje glumi pristupnu tačku. Napadač se predstavlja pristupnoj tački kao klijent i uspostavlja vezu s njom. Na ovaj način napadač uspostavlja dva šifrovana tunela: napadač – klijent i napadač – pristupna tačka.

Kao podvrsta ovog napada mogu se izdvojiti ARP napadi, koji se mogu iskoristiti i protiv računara koja nisu na bežičnoj mreži. Napadač ne mora uspostaviti vezu sa klijentom nego je dovoljno da se lažno predstavi pristupnoj tački i time dobije pristup mreži. Napadač šalje lažni odgovor na APR upit i tako menja način na koji se do tada povezivala određena MAC sa IP adresom. Dakle napadač nije promenio MAC adresu, nego samo način na koji se ona prevodi u IP adresu. Nakon toga, napadač se nalazi u sredini komunikacije između dva klijenta i može uticati na komunikaciju.

Krađa sesije (engl. *session hijacking*) je napad usmeren protiv integriteta sesije između korisnika i pristupne tačke – napadač krade sesiju autentifikovanom i autorizovanom korisniku mreže. Žrtva zna da je izgubila sesiju, ali ne zna da je tu sesiju preuzeo napadač; žrtvi se čini da je u pitanju normalni prestanak rada bežične mreže. Napadač koji je ukrao sesiju može da nastavi da radi u mreži proizvoljno dugo. Preduslovi za uspešan napad ovog tipa su mogućnost lažiranja paketa viših slojeva, korišćenje onih metoda autentifikacije i šifrovanja koje mreža zahteva i mogućnost da se žrtva spreči u daljoj komunikaciji sa pristupnom tačkom.

Napad ponavljanjem paketa (engl. *packet re-play attack*) je takođe usmeren na narušavanje integriteta informacija na mreži. Za razliku od prethodnog napada, ovde se ničim ne utiče na tekuće sesije - napad se odvija kada klijent završi svoju sesiju. Napadač snima jednu ili više sesija između klijenata i pristupne tačke kako bi ih kasnije iskoristio. Kada klijent završi svoju sesiju, napadač ponavlja njegove pakete i tako dobija pristup mreži. Bez daljih sigurnosnih prepreka, napadač može koristiti sve privilegije klijenta čiju je sesiju snimio. Ukoliko napadač ne može da zaobiđe šifrovanje koje se koristi na mreži, on je i dalje u mogućnosti da modifikuje pakete kako bi narušio integritet podataka.

Kako što se iz izloženog vidi, postoji veliki broj mogućih napada na WEP, a neki od njih su u praksi i uspešno izvedeni. To sve govori u prilog činjenici da je WEP, a samim tim i standard koji ga definiše, krajnje nesiguran i da bi kao takav, što pre trebao biti zamenjen nekim sigurnijim i boljim standardom koji bi u potpunosti uklonio navedene propuste.

12.4 Nadogradnje standarda 802.11

Postojeći standard očigledno ne pruža kvalitetnu zaštitu korisnika bežičnih mreža. IEEE je, uočivši propuste u standardu, započeo rad na novim predlozima i rešenjima kako bi se povećala sigurnost bežičnih mreža. Tako je nastao standard 802.1x.

802.1x

Fokus 802.1x standarda je unapređenje mehanizma autentifikacije čime se rešava dobar deo trenutnih problema sigurnosti bežičnih mreža. 802.1x radi na MAC podsloju drugog sloja OSI modela. Pridruživanje mreži izvodi se preko portova, koji u okvirima standarda označavaju pridruživanje klijenata pristupnoj tački. Standard 802.1x pruža radni okvir (engl. *framework*) za različite metode autentifikacije – pomoću lozinki, sertifikata i identifikacionih kartica (engl. *smartcard*).

Prema 802.1x, *supplicant* (mrežna kartica klijenta) je entitet koji koristi usluge autentifikatora (pristupne tačke) koje mu on nudi preko portova. Klijent se posredstvom autentifikatora predstavlja autentifikacionom servisu (bilo koji EAP server, najčešće RADIUS) koji nalaže autentifikatoru da *supplicantu* dozvoli pristup mreži. Pretpostavka je da svi autentifikatori komuniciraju sa istim centralnim servisom za autentifikaciju.

EAP

Standard 802.1x koristi EAP (*Extensible Autenthication Protocol*, slika 12.3) kao osnovu za korišćenje različitih autentifikacionih mehanizama. EAP je izgrađen na osnovu paradigme izazov-odgovor (engl. *challenge-response*). Prvobitno je namenjen za upotrebu u "žičanim", ali je kasnije implementiran i u bežičnim mrežama. EAP radi na drugom sloju OSI modela. Postoje četiri osnovna tipa poruka u EAP protokolu:

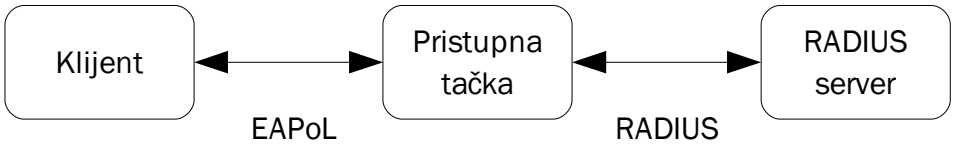
- *EAP Request* – izazov koji šalje autentifikator šalje supplicantu,
- *EAP Response* – odgovor supplicanta autentifikatoru,

- *EAP Success* – autentifikator prihvata supplicanta,
- *EAP Failure* – autentifikator odbija supplicanta.

U bežičnoj mreži EAP paket se enkapsulira u EAPoL (*EAP over LANs*). EAPoL paketi služe za komunikaciju između supplicanta i autentifikatora preko mreže. Postoji tri vrste EAPoL paketa:

- *EAPoL Start* – nalaže autentifikatoru da počne proces autentifikacije,
- *EAPoL Logoff* – obaveštava autentifikatora da se korisnik odjavljuje s mreže,
- *EAPoL Key* – nosi informaciju o WEP deljenom ključu.

EAP je proširiv u smislu da se unutar EAP zahteva i odgovora može enkapsulirati bilo koja metoda autentifikacije. EAP može da preusmeri sve zahteve za autentifikacijom ka centralnom RADIUS serveru (slika 12.3).



Slika 12.3 *– EAP i RADIUS*

Opisaćemo ukratko postupak jedne vrste EAP autentifikacije:

- u prvoj fazi, klijent (*supplicant*) šalje EAPoL Start zahtev pristupnoj tački, tj. autentifikatoru; autentifikator odgovara EAP Request/Identity porukom, a klijent EAP Response/Identity porukom,
- u drugoj fazi, autentifikator šalje RADIUS serveru Access Request poruku; RADIUS preko EAP-a šalje izazov (EAP Request) klijentu; klijent preko EAP-a odgovara serveru (EAP Response),
- u trećoj fazi, server šalje autentifikatoru Access Accept poruku; nakon toga autentifikator klijentu šalje EAP Accept poruku i deljeni ključ (EAPoL Key).

Vrste EAP-a

EAP je veoma fleksibilan standard koji se može implementirati na više različitih načina, čime je standardu 802.1x omogućeno da ispuni očekivane sigurnosne zahteve. Navodimo, uz kraći opis, spisak značajnijih metoda koje EAP koristi:

- MD5 – MD5 je EAP ekvivalent PPP CHAP protokolu koji koristi jednosmernu heš funkciju u kombinaciji sa deljenom tajnom i izazovom kako bi proverio da li *supplicant* zna deljenu tajnu. MD5 je donekle osetljiv na napad rečnikom – ukoliko napadač "presretne" izazov i odgovor koji je prošao kroz heš funkciju, on može, poznavajući tu funkciju, menjati tajnu reč dok ne dobije isti odgovor. Zato je bitno da korisnici za lozinku ne odabiraju reči koje se nalaze u rečniku.

- TLS (*Transport Layer Security*) – TLS nudi veoma siguran način autentifikacije; umesto lozinke, koriste se sertifikati i infrastruktura javnih ključeva. TLS podržava obostranu autentifikaciju, kao i dinamičke WEP ključeve. TLS je dobar izbor ukoliko se zahteva visok nivo sigurnosti, a već postoji razvijena PKI infrastruktura. Međutim, u odnosu na jednostavne lozinke, potrebno je obezbediti i prateću programsku podršku kao i obuku za korisnike.

- TTLS (*Tunneled Transport Layer Security*) – TTLS je proširenje TLS-a koje otklanja potrebu za klijentskim sertifikatima. Ovo je jedan od dva protokola koji podržavaju sigurni tunel preko mreže. Autentifikacija klijenata se obavlja pomoću heš funkcije i hibridnog kriptosistema koji obezbeđuje simetrično šifrovani tunel. Simetrični tunel postoji samo da bi se zaštitio proces autentifikacije klijenta i nakon toga je on nepotreban pa se uništava. Autentifikacija može biti EAP tipa (MD-5), a može se iskoristiti i neka druga, starija metoda (CHAP, PAP, MS CHAP, MS CHAP v2). Klijent dalje pomoću WEP ključa sa pristupnom tačkom stvara sigurni tunel.

- PEAP (*Protected Extensible Authentication Protocol*) – PEAP je drugi protokol koji, kao i TTLS, stvara sigurni tunel između klijenta i pristupne tačke koji se koristi za autentifikaciju klijenta. Za razliku od TTLS-a, PEAP ne dozvoljava stare, već samo EAP autentifikacijske metode.

- LEAP (*Light Extensible Authentication Protocol*) – LEAP je razvio Cisco za svoje proizvode koji su usklađeni sa 802.11 standardom. LEAP je vlasništvo Cisco-a i može se ugrađivati samo u Ciscove uređaje. LEAP je ranjiv na napade rečnikom – izazov i odgovor se šalju u obliku otvorenog teksta, tako da napadač može izvesti napad sličan napadu na MD5.

Sigurnosni propusti i napadi na 802.1x

EAP, kao najvažniji deo 802.1X standarda, prvenstveno je namenjen za upotrebu kao PPP protokol (*point-to-point protocol*) u "žičanim" lokalnim mrežama. Njegova upotreba u bežičnim mrežama dovela je do nekih novih sigurnosnih problema.

Glavni sigurnosni propust u EAP standardu je odsutnost mogućnosti da klijent, tj. *supplicant* autentifikuje pristupnu tačku, tj. autentifikatora. Prema standardu, autentifikator prihvata samo EAP Response*,* a klijentu šalje samo EAP Request poruke. Jednostrana autentifikacija otvara mogućnost napada čovek-u-sredini u kojem bi se napadač klijentu predstavio kao pristupna tačka, a pristupnoj tački kao klijent. To je propust čitavog okvira, jer ni viši slojevi ne podržavaju obostranu autentifikaciju.

EAP TLS pruža mogućnost obostrane autentifikacije ali se ne insistira na upotrebi TLS-a kao autentifikacionog mehanizma. Čak i kada se koristi, greška u dizajnu EAP-a omogućava napadaču uspješan napad. Na primer, EAP Success poruka se šalje klijentu kada autentifikator primi RADIUS Access Accept poruku od autentifikacionog servera (RADIUS)[69]. Poruka EAP Success bezuslovno prevodi konačni automat na strani klijenta i pristupne tačke u stanje Authenticated, bez obzira na to u kom se stanju automat pre nalazio. Koristeći ovu činjenicu, napadač može obmanuti autentifikatora i tako izvesti napad čovek-u-sredini. Napadač tada može videti sav saobraćaj između klijenta i pristupne tačke. Na ovaj način napadač zaobilazi i autentifikacijske metode viših mrežnih slojeva.

Još jedan od nedostataka 802.1x standarda je mogućnost krađe sesije. Napadač čeka da žrtva primi poruku EAP Success, a zatim, koristeći MAC

69 Poruka RADIUS Access Accept obaveštava autentifikatora da je autentifikacija uspešno obavljena.

adresu pristupne tačke, šalje klijentu upravljački okvir *disassociate managment frame*; klijent misli da je izgubio pristup mreži. Napadač dobija pun pristup mreži koristeći MAC adresu klijenta jer je 802.1x konačni automat u pristupnoj tački još uvek u stanju Authenticated. Krađa sesije je izvodljiva jer poruke između klijenta i pristupne tačke nisu prikladno zaštićene. Podaci na sloju veze su zaštićeni kada se koristi WEP šifrovanje (nakon završenog procesa autentifikacije); međutim, upravljački okviri u fazi autentifikacije nisu šifrovani, što ostavlja mogućnost lažiranja i izmene informacija.

WEP2

WEP2 je još jedan u nizu pokušaja povećanja sigurnosti bežičnih mreža. Kao što se iz imena standarda može naslutiti, on je nastao kao nadogradnja WEP-a, što znači da je nasledio neke fundamentalne slabosti u dizajnu. IEEE je menja dužinu ključa sa 40 na 128 bita i proširuje polje u kome se nalazi inicijalizacioni vektor sa 24 na 128 bita. Takođe, WEP2 podržava Kerberos V protokol.

Algoritam za šifrovanje i način upravljanja ključevima nisu izmenjeni, što znači da WEP2 ne donosi veliki pomak u povećanju sigurnosti. Dobra osobina je to što je WEP2 unazad kompatibilan sa WEP protokolom, što znači da postojeća mrežna oprema uz određenu programsku nadogradnju može koristiti WEP2 protokol.

12.5 Novi (i postojeći) standardi

Uvidevši nedostatke postojećih standarda vezanih za bežične mreže međunarodne organizacije za standardizacijsku nastavile su rad na boljim i sigurnijim standardima. Kao rezultat toga su nastala dva standarda: WPA (*Wi-Fi Protected Access*) i 802.11i.

Wi-Fi Alliance[70] dizajnirala je WPA (*Wi-Fi Protected Access*) u nameri da otkloni nedostatke uočene u WEP standardu, a da se pritom zadrži kompatibilnost sa postojećom mrežnom opremom. Isplativiji je od današnjih

70 Organizacija koju čine proizvođači mrežne opreme koji sarađuju sa IEEE.

IPsec rešenja, jer radi na drugom sloju OSI modela.

WPA koristi:

- TKIP protokol (*Temporal Key Integrity Protocol*) za šifrovanje,
- 802.1x standard i neki od uobičajenih EAP autentifikacionih protokola,
- MIC (*Message Integrity Check*, pominje se i pod imenom "*Michael*") za sprečavanje lažiranja paketa.

WPA predviđa mogućnost autentifikacije pomoću deljenih ključeva, što je pogodno za manje mreže, i pomoću RADIUS servera, što je pogodno za veće bežične mreže. WPA se može, bez većih troškova, ugraditi i u sadašnju mrežnu opremu – dovoljno je u pristupnim tačkama i klijentskim mrežnim karticama instalirati nove drajvere, odabrati tip EAP-a i po potrebi instalirati RADIUS server.

WPA2 je nadogradnja WPA koja, kao algoritam za šifrovanje, umesto RC4 koristi AES algoritam u CBC režimu rada. WPA2 je upotrebljiv i u IBSS (*Independent Basic Service Set*) režimu rada bežičnih mreža, kada klijenti komuniciraju jedan s drugim bez posredovanja pristupne tačke. Međutim, osnovni nedostatak WPA2 je potreba za ulaganjem u novu mrežnu opremu koja može da obezbedi funkcionisanje AES algoritma bez većeg pada performansi. Zato je pre prelaska na ovaj standard potrebno proveriti da li je ulaganje u WPA2 isplativo.

TKIP (T*emporal Key Integrity Protocol*) je kompatibilan sa WEP-om, što znači da je prilikom prelaska na novi standard dovoljno programski nadograditi postojeću mrežnu opremu. Problemi karakteristični za WEP razrešeni su u TKIP:

- povećanjem inicijalizacionog vektora sa 24 na 48 bita (sprečava se mogućnost šifrovanja dva paketa istim inicijalizacionim vektorom),
- generisanjem 128-bitnog ključa koji se koristi za šifrovanje jednog i samo jednog paketa RC4 algorimom.

Ključ za šifrovanje podataka se generiše mešanjem u dva koraka takozvanog vremenskog ključa (engl. *temporal key*), MAC adrese klijenta i inicijalizacionog vektora. Ukoliko se brojač paketa (veličine 16 bita) iscrpi, a

u međuvremenu se klijent i pristupna tačka ne dogovore oko novog vremenskog ključa, veza se prekida. Ukoliko se vremenski ključ ne obnovi, komunikacija se zaustavlja ili trajno prekida.

Upotreba pravih heš funkcija za zaštitu integriteta podataka nije izvodljiva bez značajnijeg pada performansi mreže. MIC algoritam definiše pogodan i efikasan mehanizam za sprečavanje aktivnih napada. Glavna pretpostavka je da paketi na prijemnu stranu stižu po redu – prijemna nakon prijema paketa koji ne očekuje odbacuje taj paket, menja vremenski ključ i postavlja na nulu inicijalizacioni vektor. Iako je u kriptografskom smislu još uvek loše rešenje u odnosu na heš funkcije, MIC algoritam je mnogo bolje rešenje od CRC32 funkcije koja se koristi u WEP-u.

802.11i

CCMP se smatra boljim i trajnijim rešenjem problema zaštite podataka u bežičnim mrežama. CCMP je zasnovan na AES algoritmu (*Advanced Encription Standard*) koji radi u takozvanom CCM režimu rada (*Counter Mode Encryption with CBC-MAC Data Origin Authenticity*). Upotreba CCMP-a je obavezna u svim implementacijama standarda 802.11i.

Za šifrovanje i zaštitu integriteta podataka u CCM režimu rada koristi se 128 bitni vremenski ključ (engl. *temporal key*), poznat klijentu i pristupnoj tački. CCM je specijalno dizajniran za 802.11i standard i predviđen je samo za blokovsko šifrovanje; trenutno ne postoje planovi da se prilagodi protočnom šifrovanju, tj. tokovima podataka. CCM radi sa 128 bitnim blokovima podataka. Najpre se u CBC-MAC[71] režimu generiše MIC kojim se štiti integritet paketa. Nakon toga se korisni podaci sa dodatnom zaštitom integriteta šifruju AES algoritmom u CTR[72] režimu rada (*counter mode*).

CCMP je značajno povećanje nivoa sigurnosti u odnosu na TKIP, a

71 Najpre se šifruje prvi blok podataka, zatim se izvodi operacija ekskluzivno ILI nad šifratom i drugim blokom podataka i rezultat ponovo šifruje; nastavlja se do kraja poruke i dobija rezultat.

72 Proizvoljno odabrana vrednost, koja se naziva brojač (engl. *counter*), šifruje se AES algoritmom. Zatim se nad dobijenim rezultatom i blokom podataka koji se šifruje izvodi operacija ekskluzivno ILI. Nakon svakog šifrovanog bloka, brojač se povećava za 1. Za šifrovanje bitova zaštite integriteta uzima se 0 za vrednost brojača. CTR režim rada ima funkciju zaštite privatnosti podataka.

naročito u odnosu na WEP. Jedini nedostatak CCMP-a je to što se ne može implementirati u postojeću mrežnu opremu, nego se ona mora zameniti novijom opremom, koja će biti dovoljno sposobna da pokreće AES algoritam bez značajnije degradacije performansi.

12.6 Pitanja i zadaci

12.1 Čime je omogućen "napad s parkinga"?

12.2 Šta je SSID?

12.3 Opišite ukratko zašto je zaštita koju WEP pruža slaba. Kako se može doći do WEP ključa?

12.4 Opišite ukratko kako funkcioniše napad ponavljanjem inicijalizacionog vektora.

12.5 Šta EAP koristi kako bi obezbedio centralizovanu autentifikaciju? Šta je glavni sigurnosni propust u EAP standardu?

12.6 Zašto se prave heš funkcije ne koriste za zaštitu integriteta podataka u bežičnim mrežama? Šta se koristi umesto heš funkcija?

12.7 Opišite ukratko zaštitne mehanizme implementirane u WPA, WPA2 i 802.11i standardima. Koji od ovih standarda zahtevaju izmenu postojeće WEP opreme i zašto?

13

Sistemi za detekciju i sprečavanje napada

13.1 Osnovni pojmovi i podele prema različitim kriterijumima

U novije vreme pokazalo se da zaštita kontrolom pristupa, mrežnom barijerom i softverom za sprečavanje infekcije zlonamernim programima (virusi, crvi, trojanski konji, špijunski programi) nije dovoljna. Pojavljuje se potreba da se sistem odbrani i od napada koji su dobro maskirani i sakriveni i koji su uspeli da prođu kroz sve ostale sisteme. Takođe, značajan je skup problema vezan za napade koji potiču od ljudi koji rade iznutra i koji legalno koriste pojedine delove sistema (engl. *insider*). Oni samostalno ili u saradnji sa nekim ko je izvan sistema, pokušavaju da ugroze sigurnost sistema iz najrazličitijih motiva. Kao posledica ove potrebe, pojavili su se:

- sistemi za detekciju upada u mreže (engl. *Intrusion Detection Systems*, IDS) i
- sistemi za sprečavanje upada u mreže (engl. *Intrusion Prevention Systems*, IPS).

Ove dve grupe zaštitnih sistema predmet su ovog poglavlja knjige. Upad može biti definisan kao bilo koji skup akcija koji pokušava da kompromituje integritet, poverljivost ili raspoloživost resursa. Sistem za detekciju upada proverava dolazeći (engl. *inbound*) ili odlazeći (engl. *outbound*) saobraćaj i identifikuje sumnjive uzorke koji mogu da indikuju napada na mrežu ili računarski sistem ili da kompromituju sistem. Primeri upada mogu biti izvedeni uz korišćenje dodatnih elemenata ili u celini sa: virusom, prekoračenjem bafera, odbijanjem usluge ili lažiranjem IP adrese.

Problem poznat pod imenom prelivanje (prekoračenje) bafera jedan je od zanimljivih primera koji je sve češće eksploatisan. On je najčešće posledica programerske greške. U okviru zloupotrebe ovog propusta, može se koristiti dodatni kod koji je napravljen da pokrene posebne akcije koje će poslati instrukciju napadnutom računaru da izvrši vrlo štetne akcije kao što su: uništavanje ili izmena podataka, otkrivanje poverljivih informacija ili narušavanje funkcionisanja rada računara. Neki tvrde da su ovakvi napadi prekoračenja bafera nastali nakon što je u okviru programskog jezika C stvorena takva mogućnost tj. okvir, a programeri i njihove loše prakse ili neznanje stvorilo ranjive tačke. Bilo je dosta propusta uzrokovanih prekoračenjem bafera u poznatim programskim proizvodima. Odbrana od

ove vrste opasnosti sastoji se u tome da programeri poznaju moguće probleme i da se drže pravila prilikom programiranja. Posebna tema u oblasti kurseva iz oblasti sigurnosti (i posebno poglavlje ove knjige), obrađuje ovu tematiku.

Jedan poseban slučaj opasnosti koja se pojavljuje je rootkit. Rootkit je skup alata ili alat koji se sastoji od malih i korisnih programa koji omogućavaju napadaču da ima pristup „root“ korisniku tj. korisniku sa najviše ovlašćenja u sistemu. Drugim rečima, rootkit je skup programa i koda koji omogućava permanentno i konzistentno, neprimetno prisustvo na kompjuteru. Da bi se zaobišao IDS / IPS softver, postoje dva pristupa: aktivni i pasivni. Oba pristupa moraju biti kombinovana da se napravi robustan rootkit. Aktivan pristup se koristi u vreme izvršavanja i dizajniran je da se predupredi otkrivanje. Samo ako neko postane sumnjičiv, pasivni pristup se aktivira iza scene kako bi se potražnja i otkrivanje učinili što težim. Više na ovu temu možete pročitati u knjizi: Rootkits: Subverting the Windows Kernel, koju su napisali Greg Hoglund, James Butler, izdanje Addison Wesley Professional, 2005.

Problem detekcije upada u računarske sisteme obrađuju mnogi autori, a značajan za intenziviranje istraživanja u ovoj oblasti je Jim Anderson, koji je objavio svoje radove u ranim osamdesetim godinama dvadesetog veka. Anderson definiše upad kao bilo koji neautorizovani pokušaj da se pristupi, manipuliše izmeni ili uništi informacija ili da se sistem učini nepouzdanim ili neupotrebljivim. Sistem za detekciju upada pokušava da detektuje ovakav tip aktivnosti.

U okviru ovog teksta, pokušavamo da ustanovimo temeljne principe i metode za detekciju ovih tipova zlonamernih aktivnosti. Pokušaćemo da pokažemo gde su pojedine tehnike za detekciju upada jake tj. odgovarajuće, u čemu su sadržani njihovi problemi i gde postoji potreba za unaređenjima. Postoji nekoliko tipova sistema za detekciju upada. Podele su definisane na osnovu različitih kriterijuma.

Sistemi za detekciju upada su klasifikovani na više načina, zavisno od kriterijuma za klasifikaciju.

Kriterijum podele: šta se detektuje ?

Jedna od najranijih podela je definisala dve kategorije:

- detekcija zloupotreba (engl. *misuse intrusion detection*) i
- detekcija anomalija (engl. *anomaly intrusion detection*).

Detekcija zloupotreba podrazumeva otkrivanje poznatih napada koje eksploatišu poznate slabosti tj. ranjivosti sistema. Detekcija anomalija se koncentriše na neuobičajenu aktivnost uopšte govoreći, koja može indicirati upad. Ako se aktivnost korisnika, koju posmatramo, razlikuje od uobičajene aktivnosti tj. ima devijaciju u odnosu na uobičajeno ponašanje, onda možemo reći da se dešava anomalija. Detekcija zloupotreba može biti veoma snažna za one tipove napada koji su programirani u sistemi za detekciju upada. Međutim, nemoguće je da se predvide svi mogući napadi koji se mogu desiti – čak i pokušaj da se to uradi uzeo bi puno truda bez značajnog rezultata. Na osnovu toga zaključujemo da je neka vrsta detekcije anomalija sasvim nužna. Problem sa detekcijom anomalija je, međutim, to što će ovakav sistem vrlo verovatno generisati mnogo lažnih pozitivnih uzbuna (engl. *false alarms*). Ovo su situacije u kojima sistem za detekciju upada podigne alarm u situaciji kada upada nije bilo. Neobična, ali legitimna upotreba sistema ili akcija korisnika može biti ocenjena kao anomalija tj. upad, te može dovesti do alarma. Izazov je napraviti takav model koji će prihvatiti nove (neuobičajene) načine upotrebe tj. ponašanja kao legitimnu upotrebu.

Teško je napraviti takav model iz istog razloga iz kojeg je teško napraviti sveobuhvatan i potpun sistem za detekciju zloputreba: nije moguće predvideti sve moguće varijacije takvog ponašanja. Ovaj zadatak može biti ostvariv na tri načina:

- Umesto generalnog sistema, koji opisuje legitimnu upotrebu, može biti definisano tj. modelirano ponašanje pojedinačnih korisnika u pojedinom sistemu. Zadatak karakterisanja tj. opisivanja regularnih uzoraka ponašanja pojedinačnog korisnika je lakši zadatak nego da se to uradi za sve korisnike generalno tj. simultano.

- Umesto da se paterni ponašanja ručno kodiraju za sve moguće

slučajeve upotrebe, oni mogu biti naučeni za primere regularne upotrebe.

- Detekcija upada u relanom vremenu, dok korisnik kuca komande ili izvodi određene operacije, vrlo je teška jer redosled operacija može mnogo varirati. U mnogim slučajevima je dovoljno da se prepozna da je redosled operacija tokom jedne login sesije ili preko celodnevne aktivnosti razlikuje u određenoj meri od uobičajene.

Postoje pristupi koji koriste neuralne mreže ili fuzzy logiku (neki ovo prevode kao „rasplinutu“ ili „nejasnu logiku“), da bi se detektovali nepoznati načini napada ili oni koji su veoma retki ili neuobičajeni.

Detekcija zloupotrebe

Primer: `If (ip.source == ip.destination) then "land attack"`

Kao što je ranije rečeno, problem sa ovim pristupom je u tome što on ne može otkriti nove tipove napada, već samo one poznate. Sistem je baziran na poznatim uzorcima tj. paternima (šablonima, oznakama) poznatih napada. Obično se oni retko ažuriraju u odnosu na frekvenciju kojom se novi tipovi napada izmišljaju od strane napadača. U međuvremenu, sistem je ranjiv, tj. otvoren prema novotkrivenim tipovima napada i zloupotrebama na sličan način kao što je neki sistem, koji koristi antivirusni softver, nesiguran od trenutka kada se pojavio novi virus do trenutka kad antivirusne definicije budu raspoložive i ažurirane. Taj period je poznat kao rizičan tj. nesiguran period. U ovom trenutku je frekvencija izdavanja tj. ažuriranja “potpisa” i pravila za nove tipove upada za sisteme za detekciju od strane kompanija koje prave ove alate znatno manja nego za nove viruse. Na osnovu ovog, može se zaključiti da nije obezbeđen odgovarajući nivo sigurnosti tj. zaštite od napada.

Detekcija anomalija

Mana sistema sa detekcijom anomalija je relativno visok broj lažnih alarma, tj. ovakav sistem vrlo često ukazuje na nepostojeći napad. Ovi sistemi koriste merenje aktivnosti i potrebno je da odluče da li je neka

aktivnost možda napad. Ovo podrazumeva postavljanje profila ponašanja: normalnog i abnormalnog profila.

Mnogi sistemi za detekciju anomalija i zloupotreba bazirani su na opštem modelu koji je predložio Denning (1987). Ovaj sistem je nezavistan od platforme, sistemskih ranjivosti i tipa upada. On održava skup istorijskih profila korisnika, uparuje pregledni tj. nadzorni zapis sa odgovarajućim profilom, ažurira profil kada je nužno i izveštava o otkrivenim anomalijama. Druga komponenta je skup pravila koji se koristi da se detektuje zloupotreba.

Kriterijum podele: gde je sistem smešten, tj. kako radi ?

Na osnovu domena ili područja u kome rade, odnosno na osnovu područja za koje su razvijeni, sistemi za detekciju upada mogu biti podeljeni u tri grupe:

- host bazirane (engl. *host based IDS*, HIDS),
- mrežno bazirane (engl. network based *IDS*, NIDS) i
- aplikativno bazirane.

Neki autori razmatraju samo prvu i drugu grupu, dok se aplikativno bazirani sistemi ponekad smatraju podgrupom (varijantom) host baziranih sistema. Takođe, neki autori navode kao posebnu vrstu i kombinovane (hibridne) sisteme za detekciju upada.

Host bazirani napadi obično se trude da osvoje pristup privilegovanim resursima na računaru. Mrežno bazirani napadi obično otežavaju legitimnim korisncima da pristupe različitim mrežnim uslugama.

Host bazirani sistemi za detekciju upada obično skeniraju i analiziraju sistem, njegovu aktivnost i logove aplikacija. Oni, takođe, osmatraju stanje sistema, izveštavajući o sumnjivim aktivnostima. Host bazirani sisemi koriste nadzorne logove sa: radnih stanica, servera, komutatora i logove specifične za određene proizvode. Najčešći problem sa host baziranim sistemima je to što na analizu dobijaju samo one podatke koje su aplikacije već upisale u logove, a takođe se može desiti da ne razumeju nameru neke

akcije, tj. da akciju pogrešno protumače.

Mrežno bazirani sistemi za detekciju upada najčešće su bazirani na praćenju mrežnog saobraćaja koji teče kroz čvor – senzor. Ovi sistemi mogu biti smešteni u rutere, pristupne tačke bežičnih mreža i druge mrežne komponente. Probleme mrežnim sistemima za detekciju upada predstavljaju šifrovani saobraćaj, opterećenje mreže i određivanje, tj. procenjivanje namere. Mrežno bazirani sistemi su u osnovi oslonjeni na prenosni medijum (žični ili bežični). Neprimetni su i nezavisni o operativnom sistemu.

Aplikacija najčešće odlučuje šta da raportira aplikativno baziranom sistemu za detekciju upada. Aplikativno bazirani IDS su obično bazirani na polisama tj. politici sigurnosti. Mana je ta što oni zahtevaju učešće aplikacije, a same aplikacije neće izveštavati o događajima. Funkcije autorizacije, kao što su GAA-API, mogu pomoći da se reši ovaj problem. Dobit je to što apliakcija razume objekte i entitete na koje se polise tj. politika sigurnosti odnosi.

Kriterijum podele: kada je napad otkriven ?

Na osnovu trenutka u kome se napad detektuje, ovi sistemi se dele na sisteme koji napad detektuju:

- u realnom vremenu (engl. *real time*) i
- naknadno, tj. nakon dešavanja (engl. *after the fact*).

Većina današnjih sistema za detekciju upada će alarmiraće onda kada se pojavi napad, zatim će odbaciti opasne pakete i terminirati sesiju za TCP i UDP bazirane napade, dinamički postaviti firewall pravila koja će zadržati izvor napada neograničeno ili za predefinisani period vremena. Poznati izvori napada mogu biti zaustavljeni ili će im pristup mreži biti zabranjen tako što će biti stavljeni na “crnu listu” (engl. *black list*), dok je mrežama kojima se veruje uvek dozvoljen pristup preko takozvanih “belih lista” (engl. *white list*).

Kriterijum podele: reakcija na napad

Na osnovu tipa reakcije kada se napad desi, IDS sistemi se dele na pasivne i reaktivne.

Pasivni sistemi za detekciju upada jednostavno detektuju i alarmiraju. Ako se otkrije sumnjiv ili zlonameran saobraćaj ili ponašanje, generiše se alarm i šalje administratoru li korisniku, a do njih je da li će preduzeti akciju da blokiraju aktivnost ili da odgovore na neki način.

Reaktivni sistemi za detekciju upada ne samo da otkrivaju sumnjivi i zlonameran saobraćaj ili ponašanje i alarmiraju administatora, već će preduzeti predefinisanu proaktivnu akciju da odgovore na opasnost. Tipično, to može biti blokiranje daljeg mrežnog saobrćaja od izvorne IP adrese ili korisnika ili zaustavljanje zlonamerne aktivnosti.

Faze odgovora na napad i sistemi za sprečavanje upada

Različiti sistemi za detekciju upada koji postoje na tržištu u ovo vreme dolaze "iz kutije" prekonfigurisani tako da umanje rizike od poznatih napada. Oni obično sadrže biblioteke sa opisom opasnosti, njihovima karakteristikama i mogu da pomognu u sprečavanju hiljada poznatih napada, kao i u ojačavanju odbrane.

Faze odgovora na napad prema Bishopu su sledeće:

- priprema (engl. *preparation*),
- identifikacija (engl. *identification*),
- ogradjivanje – okruživanje (engl. *containment*),
- iskorenjivanje (engl. *eradication*),
- oporavak (engl. *recovery*),
- nastavak (engl. *follow-up*).

Postoji "linija" tj. razlika između mrežne barijere i sistema za detekciju upada. Postoji, takođe, tehnologija zvana Sistemi za sprečavanje upada u mreže (*Intrusion Prevention Systems* - IPS). Neki autori kažu da je IPS, u

osnovi, mrežna barijera koja kombinuje filtriranje na nivou mreže i aplikacije sa reaktivnim IDS kako bi proaktivno zaštitila mrežu. Izgleda da, kako vreme odmiče, firewall-ovi, IDS-ovi i IPS-ovi uzimaju sve više osobina i atributa jedni od drugih i da granica između njih, i neku ruku, postaje sve nejasnija.

U osnovi, mrežna barijera je prva linija perimetarske odbrane. Najbolje prakse preporučuju da firewall bude eksplicitno konfigurisan da odbije (DENY) sav dolazeći saobraćaj i da onda otvori prolaze tamo gde je to neophodno. Na primer, može zatrebati da otvorite port 80 da biste držali Web sajt, tj. da bi mu korisnici mogli pristupiti, ili port 21 za FTP server. Svaki od ovih "otvora" može biti nužan sa jedne tačke gledišta, ali sa druge strane oni reprezentuju moguće vektore kroz koje zlonamerni saobraćaj može proći u mrežu tj. koji mrežna barijera neće blokirati.

Ovo je razlog zašto vam treba IDS tj. sistem za detekciju upada. Bez obzira na to da li implementirate NIDS preko cele mreže ili HIDS na posebnom uređaju (hostu), IDS mora nadzirati dolazni i odlazni saobraćaj i identifikovati sumnjivi ili zlonamerni saobraćaj koji će možda nekako da zaobiđe mrežnu barijeru i koji može, takođe, biti poreklom od spoljašnjeg napdača, ali takođe može poticati i od nekoga iznutra (engl. *insider*). Takođe, postoji mogućnost da spoljašnji napadi i napadi iznutra budu međusobno u saradnji tj. sinhronizovani.

IDS može biti odličan alat za proaktivno nadziranje i zaštitu Vaše mreže od zlonamerne aktivnosti. Međutim, on je, takođe, sklon lažnim alarmima. Kada se IDS implementira, biće neophodno da se fino podesi nakon instalacije. Potrebno je da se IDS adekvatno konfiguriše i podesi da raspoznaje normalni saobraćaj na mreži, za razliku od saobrćaja koji može biti zlonameran i administratori koji su odgovorni za reagovanje na IDS alarme, moraju da razumeju šta konkretni alarmi znače i kako se na njih efikasno odgovara.

13.2 Postojeći sistemi za detekciju upada

Najednostavniji sistemi za detekciju upada su monitori log datoteka (engl. *Log File Monitors*). Ovi sistemi pokušavaju da detektuju upad parsirajući logove sistemskih događaja. Na primer, najednostavniji monitor

datoteka može pretraživati (grep) access.log datoteku Apache Web servera po karakterističnim /cgi-bin/ zahtevima. Ova tehnologija je ograničena time što detektuje zapisane događaje, koje napadač može relativno lako izmeniti. Takođe, takav sistem će propustiti sistemske događaje niskog nivoa (engl. *low level events*), zato što je logovanje događaja operacija relativno visokog nivoa.

Monitori log datoteka su primitivni primer host baziranih sistema za detekciju upada, s obzirom na to da nadziru samo jednu mašinu. Za razliku od njih, mrežno bazirani sistemi obično skeniraju mrežu na paketskom nivou, direktno na prenosnom medijumu (žici ili vazduhu za bezičnu mrežu) kao što to radi sniffer. Mrežno bazirani sistemi za detekciju upada mogu biti koordinasani preko više hostova. Ovaj tip primene može imati prednosti u različitim situacijama. Jedan poznati monitor log datoteka je Swatch, što je skraćenica od Simple WATCHer[73]. Budući da većina alata ovog tipa skenira logove periodično, Swatch ima tu prednost da aktivno skenira zapise u realnom vremenu.

Monitor integriteta (engl. *integrity monitor*) osmatra promene vezane za ključne sistemske strukture. Na primer, osnovni monitor integriteta koristi sistemske datoteke ili ključeve u Registry bazi kao "mamac" koji prati promene koje je učinio napadač. Iako su limitiranih mogućnosti, monitori integriteta mogu biti dodatni sloj zaštite drugim formama detekcije upada. Najpopularniji monitor integriteta je Tripwire[74]. Ovaj alat je raspoloživ za Windows i Unix i može nadzirati veliki broj atributa, uključujući i dodavanje novih datoteka, brisanje i izmene sadržaja postojećih, vreme, veličinu i heš datoteka. Tripwire može biti prilagođen individulanim karakteristikama pojedinih mreža. Faktički, Tripwire može biti korišćen da nadzire bilo kakve promene sistema. Prema tome, to može biti snažan alat u arsenalu alata za detekciju upada.

Skeneri „potpisa" (engl. s*ignature scanners*), slično tradicionalnim virus skenerima baziranim na skeniranju hex vrednosti, pokušavaju da detektuju upade na osnovu baze "potpisa" poznatih napada. Kada napadač pokuša da iskoristi poznati napad, sistem za detekciju upada pokušava da upari taj napad tj. njegove značajne karakteristike sa onima koje ima u svojoj bazi. Na primer, Snort (http://www.snort.org) je besplatan sistem za detekciju

73 Web stranica posvećena swatch projektu je http://www.oit.ucsb.edu/~eta/swatch/
74 Više o Tripwire možete naći na http://www.tripwire.com

upada baziran na potpisima i pravilima, i postoji u verzijama i za Unix i za Windows. Pošto je Snort softver sa otvorenim kodom, on ima potencijal da njegova baza potpisa raste brže nego patentiran sistem nekog proizvođača. Snort se sastoji od dekodera paketa, podsistema za detekciju (engl. *detection engine*), podsistema za logovonje, tj. beleženje i podsistema za upozoravanje tj. uzbunjivanje (engl. *alerting subsystem*). Snort spada u grupu *stateful* sistema, što znači da on može prespojiti i pratiti fragmentirane TCP napade.

Klasični primer sistema za detekciju, baziranog na potpisima, uključuje CGI skriptove. U ovom slučaju, haker koristi alat za skeniranje koji obično uključuje CGI skener koji isprobava Web server na poznate CGI bagove. Npr. dobro poznati phf exploit omogućava napadaču da vrati bilo koji fajl umesto odgovarajućeg HTML-a. Da bi detektovao phf napad, mrežni skener za detekciju upada treba da pretraži pakete prema delu sledećeg niza znakova tj. stringa:

```
GET /cgi-bin/phf?
```

13.3 IDS softver za Windows operativne sisteme

Na tržištu je raspoloživ veliki broj komercijalnih i besplatnih IDS sistema za Windows i Linux. Mi ćemo ovde opisati neke host bazirane IDS za koje smatramo da vam mogu poslužiti za dodatnu zaštitu Windows radnih stanica i servera bez značajnijeg opterećenja sistema. Kompletniji spisak IDS programa možete naći na sledećim Web stranicama:

- http://force.coresecurity.com/ (firewall + IDS)
- http://www.sunbeltsoftware.com (firewall + IDS)
- http://netsecurity.about.com/od/intrusiondetectionid1/
- http://www.winpcap.org/misc/links.htm

Takođe, besplatne IDS možete naći i na stranicama www.softpedia.com i sličnim pretraživačima za besplatan softver.

Fortego All-Seeing Eye

"Svevideće oko" je besplatan host bazirani after-the-fact IDS kompanije Fortego Security namenjen nadzoru svih značajnih delova operativnog sistema. Fortego ASE se isporučuje sa krajnje jednostavnim grafičkim korisničkim interfejsom što ga čini jednostavnim za upotrebu za sve kategorije korisnika.

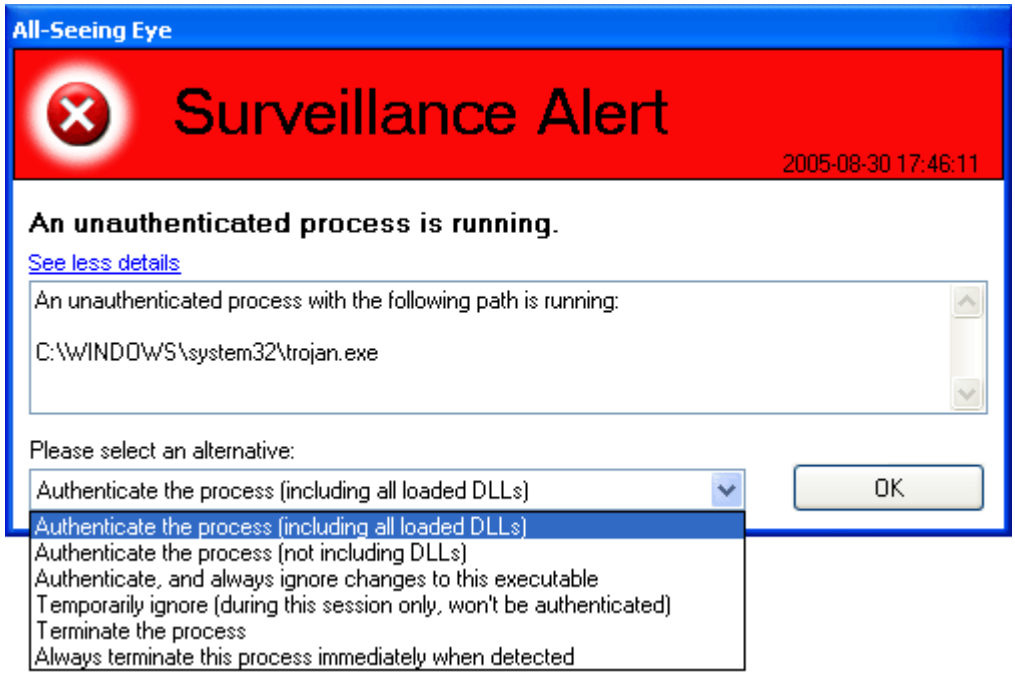
Fortego ASE nije skener potpisa niti je njegovo funkcionisanje zasnovano na crnim i belim listama procesa, DLL biblioteka ili drajvera. ASE inicijalno snima stanje i traži od korisnika da ga "nauči" šta je dozvoljeno, a šta nije dozvoljeno. Na primer, ukoliko napadač kasnije hoće da vam podmetne program keylogger.exe u neku od auto-start sekcija Windows sistema (koristeći, na primer, trojansku instalacionu rutinu drugog softvera), dobićete upozorenje prikazano na slici 13.1. Vi dalje odlučujete da li je ta akcija legitimna ili nije – shodno tome, ASE će dozvoliti ili ukloniti tu rutinu.

Slika 13.1 *– ASE upozorava o novom zapisu u auto-start sekcijama*

Ovaj način zaštite pruža mogućnost da sprečite izvršenje novog trojanca, crva, spyware komponente za koje još uvek ne postoje definicije i kao takav je savršena dopuna uz mrežnu barijeru, antivirusni i antispyware softver. Međutim, ASE zahteva izvestan nivo poznavanja procesa koji se izvršavaju na Vašem sistemu, kao i poznavanje sadržaja autostart sekcija, DLL biblioteka, drajvera i slično.

ASE je sačinjen od sledećih modula:

- Process Tracker – alat koji nadzire i autentifikuje procese, upozorava korisnika na postojanje neautentifikovanih ili izmenjenih procesa i nudi mu mogućnost da takve procese prekine, autentifikuje (uključujući ili ne uključujući DLL-ove koje proces poziva) ili privremeno ignoriše (slika 13.2).



Slika 13.2 *– Fortego ASE Process Tracker*

- DLL Tracker – alat koji nadzire i autentifikuje dinamičke biblioteke, tj. DLL datoteke. Na primer, maliciozna podmetnuta DLL datoteka koju

poziva Internet Explorer (kome je u firewall pravilima dozvoljen pristup Internetu) može uspostaviti konekciju ka nekom serveru kome će slati vaše podatke, na primer, brojeve kreditnih kartica ili e-mail kontakte (zbog spam-a). Ovo je česta tehnika infekcije koju koriste virusi i špijunski programi.

- Driver Tracker – drajveri su, jednostavno rečeno, specijalna vrsta izvršnih datoteka sa višim privilegijama od normalnih procesa. Ovaj modul autentifikuje drajvere, upozorava korisnika ukoliko sistem učita nepoznati drajver ili ukoliko je došlo do izmene autentifikovanog drajvera.

- Event Log Tracker – prati upis u Windows log datoteke i upozorava korisnika na sve upise ili na upise koje korisnik označi kao značajne.

- Autostart Guard – prati i obaveštava korisnika o novim zapisima u Windows auto-start sekcijama, tj. u mestima koja omogućavaju pokretanje programa prilikom podizanja operativnog sistema

- Service/Driver Guard – modul za nadzor pokrenutih servisa, sprečava pokretanje zlonamernih programa u formi servisa

- ActiveX Object Guard – ActiveX objekti su izvršni kod, koji se izvršava u Internet Exploreru. Shodno Security podešavanjuma u Internet Exploreru, korisnik se upozorava ili ne upozorava na instaliranje ActiveX objekata. ActiveX je pogodan način za obavljanje infekcije računara virusom, keyloggerom ili špijunskim programom. ASE obaveštava korisnika o svim dodatim, izmenjenim ili uklonjenim ActiveX objektima na računaru i daje korisniku mogućnost da te objekte ukloni odmah nakon instalacije.

- Browser Helper Object (BHO) Guard – BHO je izvršni kod, namenjen izvršavanju u Internet Exploreru. Koristeći sigurnosne propuste u Internet Exploreru, BHO se može neprimetno instalirati na računaru koji je nezaštićen. Nakon instalacije, BHO može inficirati računar virusom, instalirati keylogger ili neku spyware/adware komponentu. Zbog toga je potrebno da, ukoliko već koristite Internet Explorer, vodite računa o tome koji su BHO instalirani na Vašem računaru. Ovaj modul obaveštava korisnika o svim dodatim, izmenjenim ili uklonjenim BHO i

daje korisniku mogućnost da ukloni neželjene objekte.

- Winsock Layered Security Provider (LSP) Guard – jedna od novijih metoda koju "laki pisci" špijunskih programa i hakeri koriste za tajno izvršavanje koda na udaljenim računarima je instalacija Winsock LSP na žrtvi. Osim što omogućava tajno izvršavanje koda na umreženom računaru na kom je instaliran, Winsock LSP omogućava napadačima da prate i izmene sav odlazeći i dolazeći saobraćaj. ASE vam nudi mogućnost da pratite koji su sve Winsock LSP instalirani na Vašem računaru i da uklonite one koji su neželjeni. Ovaj ASE modul će Vas obavestiti o svim dodatim, modifikovanim i uklonjenim Winsock LSP.

- Hosts File Guard – blokira izmene hosts datoteke (razrešavanje imena u IP adresu) i sprečava krađu Web browsera.

- File System Guard – modul za nadzor pojedinačnih datoteka, direktorijuma i kompletnih direktorijumskih stabala. Obaveštava pri svakoj izmeni sadržaja datoteke ili direktorijuma koji se čuva.

- Registry Guard – modul za nadzor Registry baze. Obaveštava pri izmeni sadržaja onog dela Registry baze koji čuva (pojedinačni ključevi i stabla), kao i prilikom brisanja postojećih ili dodavanju novih ključeva u stablo koje se čuva.

Fortego All-Seeing Eye je besplatan program. Možete ga preuzeti sa Web stranice www.fortego.com/ase.

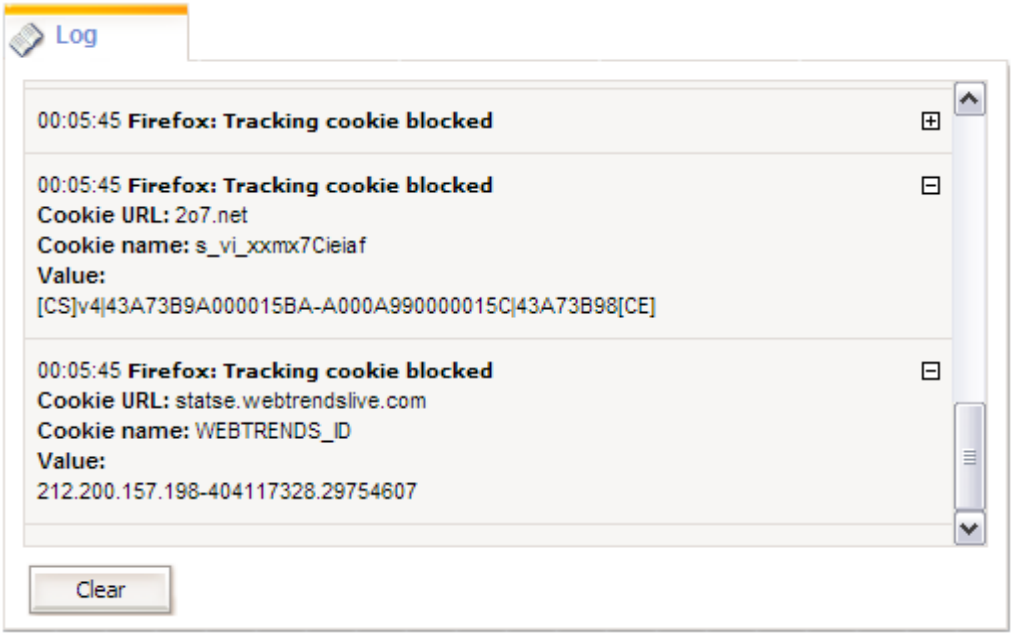
Arovax Shield

Arovax Shield je još jedan besplatan host bazirani IDS. Arovax Shield sprečava izmene nekih bitnih delova Windows operativnog sistema bez saglasnosti korisnika:

- krađu Web browsera (izmena početne stranice, pretraživača, itd),
- instalaciju izvršnog koda koji se izvršava u Web browseru (kao što, na primer, BHO),
- auto-start sekcije,

- asocijacije ka programima u kojima se otvaraju datoteke sa određenom ekstenzijom. Na primer, mp3 datoteke otvaraju programi poput Creative MediaSource Player ili Winamp; neki špijunski program lako može podmetnuti novu asocijaciju mp3 datoteka ka izvršnoj datoteci c:\windows\system32\trojan.exe ili nekoj sličnoj.

Arovax Shield takođe blokira špijunske kolačiće (engl. *tracking cookies*) u Internet Explorer i Mozilla Firefox Web browserima (slika 13.3).



Slika 13.3 *– Arovax Shield sprečava špijunske kolačiće*[75]

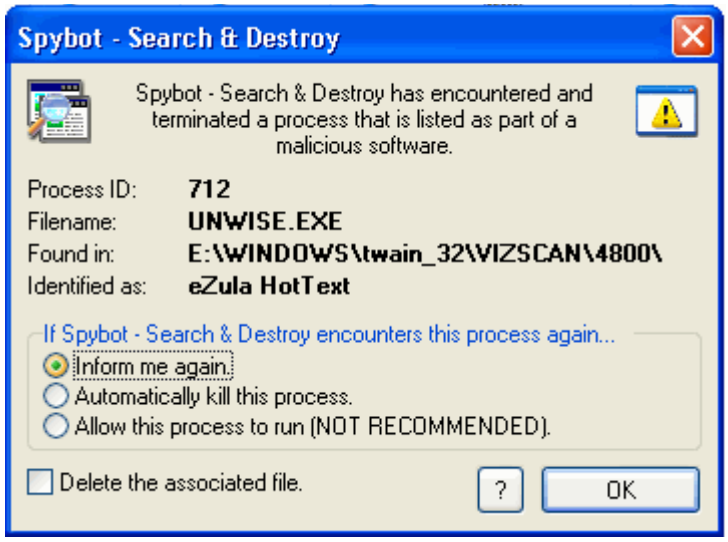
TeaTimer

Program Spybot Search & Destroy pominjemo kao veoma dobrog čistača špijunskih programa u sledećem poglavlju koje se bavi zlonamernim softverom. Ovde ćemo ukratko opisati funkcionalnost koju pruža rezidentni

75 Da li na slici vidite kolačić sa domena 207.net? Ukoliko vam je tematika špijuniranja i privatnosti zanimljiva, pogledajte kako velike kompanije špijuniraju svoje klijente na stranici http://www.safer-networking.org/en/news/2006-01-13.html.

modul Tea-timer.

TeaTimer nadzire pokrenute procese i sprečava izvršavanje onih koji se nalaze na crnoj listi (procesi koje pokreće zlonamerni softver). TeaTimer Vam, zatim, nudi da odaberete šta dalje želite da radite sa tim procesom, tj. da li želite da budete obavešteni onda kada se proces ponovo pokrene ili da taj proces bude automatski ubijen, da li želite da dozvolite izvršenje tog procesa ili da obrišete izvršnu datoteku na osnovu čijeg pokretanja je taj proces nastao.

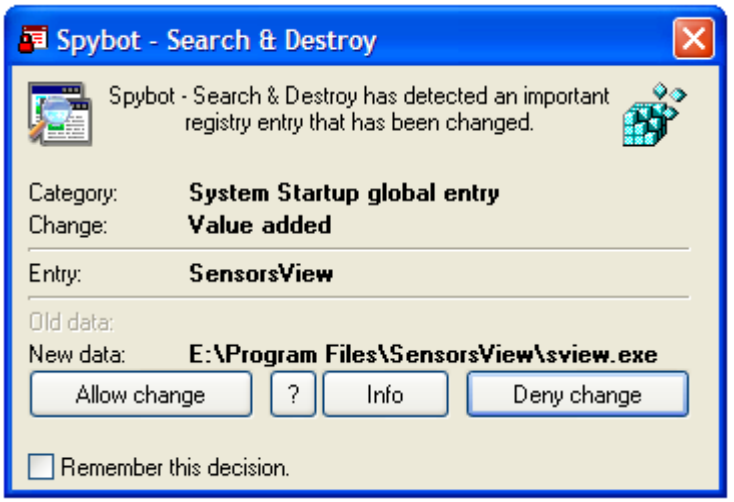


Slika 13.4 *– TeaTimer uništava proces sa crne liste*

Za razliku od Fortego ASE (after-the-fact detekcija neautentifikovanih procesa), TeaTimer odmah uništava procese sa crne liste, zato što je uništenje nekih parazita vremenski kritična operacija. Na primer, zamislite situaciju da ste otišli na ručak i da je neki dialer koga ste upravo zaradili preko modema pozvao telefonski broj u Australiji (na primer. tačno vreme u Sidneju ili pornografski server na Havajima). Ukoliko TeaTimer ne ubije takav proces odmah, dobićete jako visok telefonski račun koji nećete moći da opravdate.

Dodatno, TeaTimer će detektovati izmene kritičnih ključeva u Registry bazi (na primer, dodavanje novog ključa u neku od auto-start sekcija) i

ponuditi vam mogućnost da te promene dozvolite ili zabranite (slika 13.5).



Slika 13.5 *– Provera upisa u auto-start sekcije*

13.4 Pitanja i zadaci

13.1 Fortego ASE je after-the-fact IDS. Objasnite šta to znači. U čemu je razlika između IDS sa detekcijom anomalije i IDS sa detekcijom zloupotrebe.

13.2 Instalirajte ASE i pokušajte da ga naučite pravilima (šta je dozvoljeno, a šta nije). Probajte zatim da pokrenete neki proces, dodate neki program u auto-start sekciju ili pokrenete neki servis. Sprečite izvršenje tih akcija.

13.3 Pokušajte da instalirate Snort i odgovarajuću listu pravila. Pronađite na Internetu uputstva za dodavanje novih pravila. Dodajte pravilo koje sprečava napade tipa: *smurf*, *syn-flood* i *ping-of-death* (napadi su opisani u poglavlju o mrežnim barijerama).

14

Antivirusna zaštita i zaštita od špijunskih programa

14.1 Šta je zlonamerni softver?

Stroga i precizna definicija zlonamernog softvera ne postoji. U zlonamerni softver (engl. *malware*, *malicious software*) jednostavno se ubraja svaki program dizajniran u nameri da na bilo koj način ošteti, oteža ili onemogući korišćenje umreženog ili neumreženog računara. Ovakvi programi postoje u različitim oblicima – neki oblici zahtevaju nosioce, tj. korisne programe, dok su neki samostalni; neki se repliciraju, a drugi ne. Zlonamerni programi mogu raditi neprimetno u pozadini, ili usporiti računar i periodično izazivati zakucavanje ili obaranje sistema. Zavisno od vrste i namene, zlonamerni softver može obavljati jednu ili više aktivnosti:

- obaranje performansi sistema, dovođenje sistema u nestabilno stanje, generisanje "neobičnog" ponašanja sistema,
- preusmeravanje zahteva za otvaranjem Web stranica,
- opsluživanje pop-up prozora ili banera sa reklamnim sadržajem,
- praćenje aktivnosti i uznemiravanje korisnika,
- krađa ili uništavanje poverljivih informacija,
- izvršavanje DoS napada na određeni server,
- preuzimanje dodatnog izvršnog koda sa Interneta i instalacija drugih zlonamernih programa,
- smanjivanje sigurnosti sistema, isključivanje sigurnosnih aplikacija,
- modifikacija lozinki na sistemu i dodela prava udaljenog pristupa računaru,
- modifikacija podataka i registry baze,
- oštećenje hardvera.

Iako je najveći broj zlonamernih programa namenjen Microsoft Windows platformama (Windows XP omiljena meta pisaca zlonamernih programa), ugrožene su i UNIX, Linux, Mac OS i Novell Netware platforme. Zlonamerni softver se distribuira pomoću:

- deljenih direktorijuma na mreži,

- priloga elektronske pošte (engl. *e-mail attachment*),
- otvorenih TCP i/ili UDP portova koji omogućavaju izvršenje udaljenog koda na žrtvi (setite se Sasser crva),
- peer-to-peer (P2P) mreža,
- internet messaging (IM) servisa,
- drugih zlonamernih programa koji će preuzeti maliciozni kod sa neke lokacije na Internetu i instalirati ga na žrtvi.

Inficirani sistemi čiste se specijalnim programima za uklanjanje virusa, crva, trojanaca i špijunskih programa. Međutim, veliki problem predstavljaju zlonamerni programi koji su sposobni da svoje komponente prikriju (na primer, koristeći ključeve u registry bazi) i time otežaju njihovo uklanjanje sa inficiranog sistema. Programi za uklanjanje virusa i špijunskog softvera će sistem samo prividno očistiti od takvih parazita; nakon čišćenja, zlonamerni softver će se reinstalirati pomoću prikrivenih komponenti. Zbog toga bolji zaštitni programi primenjuju taktiku "bolje sprečiti nego lečiti", pružaju funkcionalnost zaštite sistema u realnom vremenu.

Zlonamerni softver se može podeliti na crve, trojanske konje, viruse, špijunske programe i ostale zlonamerne programe.

14.2 Crvi

Crvi (engl. *worms*) su samostalni (engl. *stand-alone*) programi koji se šire po principu s računara na računar. Uobičajene metode propagacije na žrtvu su upotreba elektronske pošte i Internet servisa (FTP, HTTP). Crv eksploatiše ranjivost žrtve ili koristi metode prevare i obmanjivanja, poznate kao sociološki inženjering (engl. *social engineering*), kako bi naterao korisnika da ga pokrene. Crvi se mogu se klasifikovati prema metodama propagacije, načinu instalacije i pokretanja i prema karakteristikama kojima se opisuje zlonamerni softver (na primer, polimorfnost, upotreba *stealth* tehnika). Crvi koji su uspeli da naprave veću štetu koristili su više različitih metoda propagacije.

Prema načinu propagacije crvi se mogu podeliti u sledeće kategorije:

- e-mail crvi,
- instant messaging (IM) crvi,
- Internet crvi,
- file-sharing i P2P crvi.

E-mail crvi - MyDoom

E-mail crvi (kao što su Netsky, MyDoom, Sasser.a, Lirva i Gibe) šire se preko inficiranih e-mail poruka kao prilozi (izvršne datoteke) ili linkovi ka inficiranim Web stranicama. Ukoliko se crv širi preko linkova, korisniku se šalje "udica" (engl. *hook*) koja nakon pokretanja otvara u Web čitaču (engl. *browser*) inficirani sajt koji će instalirati crva na žrtvu. E-mail crvi koriste metode socioloških inženjeringa kako bi naterali korisnika da pokrene program u prilogu ili odgovarajući link. Korisnik dobija poruku sa naslovom tipa "an important thing about you", "critical windows update" ili "meet the love of your life". Ukoliko korisnik pokrene link ili program u prilogu, crv će se aktivirati (ili će ga instalirati neka Web stranica) i inficirati računar. Nakon infekcije računara crv se širi slanjem kopija samog sebe na e-mail adrese koje pribavlja iz resursa inficiranog računara. Crv do ovih adresa dolazi:

- čitanjem podataka iz programa Address Book (karakterističan za Outlook i Outlook Express)[76],
- pretraživanjem sadržaja datoteka sa odgovarajućom ekstenzijom; crv traži nizove karaktera koji odgovaraju e-mail adresama (nizovi oblika you@yourname.com),
- odgovaranjem na svu poštu u poštanskom sandučetu).

Primer e-mail crva je W32/MyDoom.a (poznat i kao Novarg) koji napada Windows 95, NT 4.0, 98, ME, 2000, XP i Server 2003 platforme. Efekti MyDoom.A crva su sledeći: pokušaj izvođenja DoS napada na SCO Web sajt, generisanje neobičnog ponašanja sistema (otvaranje Notepad-a sa gomilom besmislenih karaktera), modifikacija registry baze, omogućavanje udaljenog pristupa računaru (TCP portovi 3127-3198). Crv se širi putem elektronske pošte i P2P mreža. Veličine je 22,528 bajtova.

76 Crv se izvršava pod korisničkim nalogom korisnika koji ga je pokrenuo, tako da ima pristup svim podacima tog korisnika, uključujući i Address Book.

Zašto je MyDoom izazvao epidemiju?

- Za razliku od većine crva, MyDoom ne pokušava da "obradi" naivnog korisnika porukama u kojima se spominju pornografske fotografije javnih ličnosti. MyDoom koristi metode sociološkog inženjeringa – na primer, poruka "The message contains Unicode characters and has been sent as a binary attachment" ima tehnički prizvuk; naivni korisnici će pomisliti da je prilog legitiman i pokrenuće ga.

- "Tajming" – crv je počeo da se širi iz Sjedinjenih Američkih Država u vreme kada je protok elektronske pošte najveći, takozvani "business hours on Monday". Prema informacijama iz kompanije MessageLabs koja skenira e-mail na viruse, jedna u 12 poruka je sadržala crva u peridu epidemije. Drugi crvi, čije je širenje brzo suzbijeno, nisu imali ovakav "tajming", tako da su kompanije koje se bave anti-virusnom zaštitom mogle da generišu dodatke antivirusnih baza i tako spreče širenje crva.

Prilikom širenja, crv šalje poruke na adrese sastavljene od korisničkih imena i imena domena pronađenih u datotekama na sistemu i onih skrivenih u samom kodu crva. Budući da su ovako sastavljene adrese u većini slučajeva nepostojeće, slanje poruka će prouzorkovati veliki broj upozorenja o neisporučenoj elektronskoj pošti na zaraženom računaru sa kojeg se crv širi. MyDoom ne šalje zaražene poruke na određeni skup predefinisanih domena i računara (kao što su root, info, nobody i njima slični).

Prilikom širenja, MyDoom generiše poruke sa sledećim naslovima: "Mail Delivery System", "Mail Transaction Failed", "Server Report", "Status", "Error". Telo poruke sadrži jedan od sledećih tekstova:

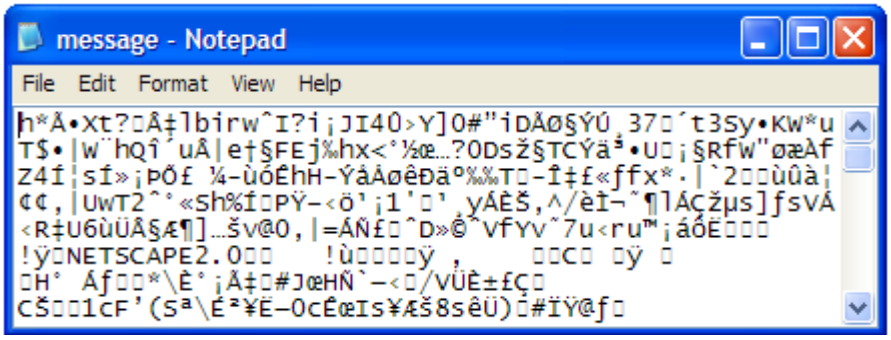
- "The message can not be represented in 7-bit ASCII encoding and has been sent as a binary attachament",
- "The message contains Unicode characters and has been sent as a binary attachment",
- "Mail transaction failed. Partial message is available."

Naravno, poruka se isporučuje žrtvi sa prilogom u kome se nalazi crv. Prilog je datoteka sa imenom "document", "readme", "doc", "text", "file",

“data”, test”, “message” ili “body” i ekstenzijom “.pif”, “.scr”, “.exe”, “.cmd”, “.bat” ili “.zip”. Napominjemo da otvaranjem datoteka sa nastavkom .zip sistem neće biti ugrožen. Crv se aktivira tek kada korisnik na žrtvi pokuša da pokrene izvršnu datoteku unutar .zip arhive.

U slučaju širenja MyDoom-a putem KaZaA mreže, crv će kopirati svoju izvršnu datoteku u direktorijum sa deljenim datotekama pod nekim od sledećih imena: “winamp5”, “icq2004-final”, “activation-crack”, “rootkitXP”, “office_crack” ili “nuke2004” i ekstenzijom “.bat”, “.exe”, “.pif”, ili “.scr”. Ovo se, takođe, može ubrojiti u tehniku socioloških inženjeringa, jer se korisnicima očigledno nude programi koji se često upotrebljavaju ili “crack” za Office paket. Korisnici koji sa inficiranih računara preuzmu ovakve datoteke, inficiraće svoj računar.

Nakon pokretanja, crv otvara Notepad, u kome je ispisana gomila slučajnih znakova (slika 14.1).



Slika 14.1 *– Jedan od efekata My Doom crva*

Izvršna datoteka MyDoom-a pritom se kopira u sistemski direktorijum %System% (najčešće C:\Winnt\System32) kao datoteka taskmon.exe. Kako bi se osiguralo pokretanje izvršne datoteke crva tokom svakog ponovnog pokretanja računara, crv u Registry bazu upisuje sledeći ključ:

- [HKLM\Software\Microsoft\Windows\Current Version\Run]
  “Taskmon”=%System%\taskmon.exe

ili, u slučaju neuspeha, tj. ukoliko je crva pokrenuo korisnik koji nema

administratorske privilegije:

- [HKCU\ Software\Microsoft\Windows\Current Version\Run]
  "TaskMon"=%sysdir%\taskmon.exe.

Uz izvršnu datoteku, u %system% direktorijum smešta se i datoteka shimgapi.dll, čiji je cilj osluškivanje dolazećih konencija na TCP portovima 3127 – 3198. U Regitry bazu crv dodaje sledeću liniju:

- [HKCR\CLSID\{E6FB5E20-DE35-11CF-9C87-00AA005127ED}\
  InprocServer32],

koja obezbeđuje da se datoteka shimgapi.dll učita u memoriju kao Windows Explorer dodatak. To znači da je taj proces nevidljiv u programu Task Manager. Pomoću ulaza u sistem otvorenog na ovaj način, neovlašteni korisnik je u mogućnosti da ubaci dodatne izvršne datoteke na sistem i pokrene ih ili da inficirani sistem jednostavno upotrebi kao TCP proksi server.

Osim širenja porukama elektronske pošte i ostavljanjem ulaza u sistem, MyDoom.A je programiran da 1 februara u 16:09 časova (prema sistemskom vremenu) pokrene DDoS napad na Web stranicu www.sco.com. Svi inficirani računari će u zadato vreme uputiti 64 simultana zahteva za otvaranjem glavne stranice ovog Web servera, pokušavajući da ga na taj način preopterete. Aktivnost crva prestaje 12.februara, ali pomenuti "*backdoor*" ostaje otvoren i posle tog datuma.

Iako je delovanje crva vremenski ograničeno, veliku pretnju po integritet inficiranog sistema predstavljaju trajno otvoreni mrežni portovi pomoću kojih neovlašteni korisnik može, sa udaljenog računara, na sistem ubaciti proizvoljni programski kod. Neki mrežni crvi za svoje širenje koriste upravo ovaj propust. Osim uklanjanja MyDoom-a sa inficiranih računara, preporučuje se i detaljna analiza sistema koja bi otkrila eventualne promene nastale naknadnim delovanjem neovlaštenih korisnika. Ukoliko je to moguće, svakako se preporučuje reinstalacija operativnog sisitema.

IM crvi – CoolNow i Funner

Instant messaging (IM) crvi se šire pomoću standardnih servisa za poruke kao što su Microsoft MSN and AOL AIM. Crv šalje svim korisnicima link koji ukazuje na inficirani sajt ili datoteku. Kada žrtva pokrene link, računar preuzima i pokreće crva. Crv se zatim instalira na računaru, proverava listu kontakata i svim korisnicima u listi šalje sličnu poruku. Primeri IM crva su JS/CoolNow i Funner.

CoolNow je crv koji se širi po Internetu koristeći MSN Messenger-a. Crv je realizovan kao Java Script u HTML dokumentu. Crv nakon pokretanja čita listu kontakata u MSN Messengeru i šalje svima sledeću poruku: "URGENT - Go to http://www.rjdesigns.co.uk/cool Now". Adresa ukazuje na HTML dokument koji sadrži telo crva. Nakon slanja inficiranih e-mail poruka, crv koristi skript koji se nalazi na samom sajtu da na adresu jonathansmith288@hotmail.com pošalje poruku koja sadrži IP adresu inficiranog računara i da preusmeri korisnika na sledeću adresu http://www.rjdesigns.co.uk/cool/go.htm. Crv ne sadrži nikakav “payload” tj. ne sadrži nikakav destruktivni kod.

Funner je crv koji se širi po Internetu koristeći MSN Messenger. Napisan je Visual Basic-u, a spakovan pomoću ASP-a. Nakon izvršenja, crv se kopira u Windows sistemski direktorijum (%system%\) u datoteke sa sledećim imenima: IEXPLORE.EXE, explorer.exe i userinit32.exe. Crv se, takođe, kopira u Windows root direktorijum (%WinDir%\), kao datoteka sa imenom rundll32.exe, i kreira log datoteku %System%\bsfirst2.log. Crv, zatim, modifikuje registry bazu, i time obezbeđuje da će biti izvršen svaki put kada se sistem pokrene:

- [HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run],
- [HKCU\Software\Microsoft\Windows\CurrentVersion\Run], "MMSystem" = "%Windir%\rundll32.exe "%System%\mmsystem.dll"", RunDll32",
- [HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon] "Userinit" = "%System%\userinit32.exe".

Ukoliko se na žrtvi izvršava Windows 95/98/ME operativni sistem, crv će izmeniti boot sekciju u “system.ini” datoteci:

- [boot] Shell = %System%\explorer.exe

Nakon pokretanja crv pristupa MSN Messenger kontakt listi i šalje kopiju sebe pod imenom “funny.exe” svim kontaktima. Kao efekat, crv u datoteku "%System%\drivers\etc\hosts" upisuje sledeći tekst:

```
222.89.98.219 001wz.com
222.89.98.219 07007.com
222.89.98.219 114.com.cn
222.89.98.219 ... veliki broj drugih sajtova
222.89.98.219 yymp3.com
222.89.98.219 zhao99.com
222.89.98.219 zzkan.com
```

To znači da se browser preusmerava na adresu 222.89.98.219 ukoliko korisnik želi da vidi sadržaj bilo kog od navedenih sajtova.

Internet crvi – Sasser.b

Tužno ali istinito: da bi se računar inficirao crvom, dovoljno je da bude povezan na Internet. Neki crvi na Internetu traže ranjive računare, tj. računare na kojima nisu instalirani sigurnosne zakrpe, računare koji nemaju mrežnu barijeru ili računare na kojima su otvoreni neki portovi koje crv može da iskoristi. Kada pronađe takav računar, crv će pokušati da iskoristi propust, tj. da se iskopira i instalira na žrtvi.

Tehnike za propagaciju internet crva su:

- Kopiranje crva na mrežne resurse. Crv koristi usluge operativnih sistema kako bi na mreži pronašao direktorijume koji su otvoreni za čitanje i pisanje.

- Eksploatacija slabosti operativnih sistema. Crv na Internetu traži računare koji su pogodni za eksploataciju – najčešće se radi o računarima koji nemaju poslednje sigurnosne zakrpe. Crv šalje pakete koji će instalirati ili celog crva ili samo jedan deo – udicu (engl. *hook*).

Udica nakon instalacije preuzima i instalira celog crva.

- Korišćenje javnih mreža. Crv najpre napada HTTP i FTP servere, npr. statičke web stranice, a zatim čeka da klijenti pristupe inficiranim fajlovima i napada klijent računare.

- *Piggy-backing* (korišćenje drugog zlonamernog softvera kao nosioca). Crv najpre identifikuje trojanskog konja ili drugog crva koji je instalirao *backdoor* na žrtvi. Ova funkcionalnost u većini slučajeva dozvoljava crvu da šalje komande žrtvi – na primer, da preuzme i izvrši neku datoteku, najčešće, novog crva.

Početkom maja meseca 2004. godine pojavila su se dva crva iz familije Sasser. Sasser.A se širi kao e-mail crv i prošao je nezapaženo. Međutim, Sasser.B je malo složeniji, jer koristi LSASS RPC siguirnosni propust nekih Windows operativnih sistema (na primer, Windows XP i Windows 2000 Professional) kako bi zarazio računare povezane na Internet. Po načinu širenja Sasser.B sličan je crvu Blaster[77] koji je zapamćen kao jedan od većih sigurnosnih incidenata vezanih za crve. Nastao je kao modifikacija već dobro poznatog i još uvek aktivnog crva NetSky.

Iako nije destruktivan, crv Sasser.B je definisan kao crv s visokim potencijalom distribucije. Kao što je već rečeno, Sasser.b iskorišćava sigurnosni propust u LSASS (*Local Security Authority Subsystem Service*) koji omogućava udaljenom korisniku (napadaču) da izvrši proizvoljni kod. Sigurnosni nedostatak obeležen je prepisivanjem bafera[78] u LSASS procesu. Ova vrsta eksploatacije sistema ne zahteva autentifikaciju korisnika, a može dovesti do potpunog kompromitovanja sistema, što napadaču omogućuje instalaciju programa, pregled, promenu i brisanje datoteka, te kreiranje novih korisničkih naloga sa administratorskim pravima. Ovaj nedostatak zabeležen je u operativnim sistemima Windows 2000 Professional i XP i na njima se može iskoristiti; nedostatak postoji i na

77 Internet crv koji eksploatiše ranjivost prepunjenjea bafera DCOM RPC-a (Distribted Component Object Model) u Microsoft Windows operativnom sistemu. Crv na Internetu traži računare kod kojih ova ranjivost postoji i nad njima izvršava napad (TCP port 135). Udica je EXE datoteka veličine 11KB, kompresovana pomoću UPX arhivera na veličinu 6KB. Udica preuzima i izvršava datoteku msblast.exe koja izaziva pojavljivanje neočekivanih poruka o grešci i automatski obara i ponovo podiže sistem.

78 Prepunjenje bafera napadač izvodi tako da zadaje dugačak argument poruke koristeći lsasrv.dll funkciju DsRoleUpgradeDownlevelServer() kako bi se pažljivo kreirala poruka koja se šalje određenom baferu.

Windows Server 2003 i 64-bitnim Windows XP operativnim sistemima, ali se može iskoristiti samo ako je na računar prijavljen lokalni administrator. Kod operativnih sistema Windows verzije 98, 98 SE i NT nije uočen ovaj sigurnosni nedostatak, ali računari sa tim operativnim sistemima mogu poslužiti kao posrednici za širenje crva na računarima sa ranjivim sistemima u lokalnoj mreži. Microsoft je objavio zakrpu koja razrešava LSASS sigurnosni propust; zakrpa menja način na koji LSASS procenjuje dužinu poruke pre nego što je prosledi određenom baferu. Međutim, iako je Microsoft izdao sigurnosnu zakrpu nekoliko dana pre pojave crva, veliki broj računara povezanih na Internet zaražen je u vrlo kratkom vremenskom periodu.

Za širenje među računarima koji koriste Windows 2000 Professional i XP operativne sisteme crv kreira 128 izvršnih niti, odnosno serija poruka, koje generišu slučajne IP adrese. Nakon toga, crv šalje specijalno oblikovane pakete slučajno odabranih IP adresa na TCP port 445, pri tome zaobilazeći rezervisane IP adrese: 1.0.0.0, 127.0.0.0, 169.254.0.0, 172.16.0.0 – 172.31.0.0, 192.168.0.0 i 213.191.74.19. Paketi izazivaju prepisivanje bafera ranjivih sistema u datoteci lsass.exe, što rezultira izvršenjem udaljene komandne linije koja otvara port 9996 za sledeće udaljene naredbe. Sa udaljene lokacije crv šalje naredbe za generisanje i izvršenje FTP skripta cmd.ftp. Ovaj skript preuzima kopiju crva sa inficiranog računara sa kog se crv širi na računar na kom je skript pokrenut korišćenjem TCP porta 5554. Preuzeta kopija crva čuva se na lokalnom računaru pod imenom <siučajan_broj>_up.exe (na primer, 34567_up.exe), a veličine je 15.872 bajta. Po završetku preuzimanja kopije crva, datoteka cmd.ftp se briše sa novog inficiranog računaraa, a kreira se bezopasna tekstualna datoteka win2.log u root direktorijumu (C:\). Datoteka C:\win2.log sadrži broj sistema koji su inficirani i IP adrese tih računara. Nakon pokretanja izvršne datoteke <slučajan_broj>_up.exe, crv se samostalno kopira u Windows direktorijum kao datoteka avserve2.exe. Kako bi se osiguralo automatsko pokretanje izvršne datoteke prilikom svakog sledećeg pokretanja Windows operativnog sistema, crv kreira sledeći zapis u registry bazi:

- [HKLM\Software\Microsoft\Windows\CurrentVersion\Run]
  avserve2.exe = %Windows%\avserve2.exe

Takođe, crv kreira mutex semafor koji omogućava nitima da dele isti

resurs pod imenima: Jobaka3, JumpaIlsNlsTillt. Ukoliko je instanca JumpaIlsNlsTillt mutex-a pronađena na računaru, crv prekida svoje izvršenje jer prepoznaje svoje postojanje na računaru. Po izvršenju izvodi se prepisivanje bafera sistema u datoteci lsass.exe, što izaziva grešku u pomenutoj datoteci (vidite sliku 14.2), nasilno rušenje i ponovno podizanje operativnog sistema. Pri tome se korisniku prikazuje obaveštenje o tome da će sistem biti oboren, ali mu se ne daje mogućnost da to prekine.

Slika 14.2 *– Dijalog koji ukazuje na grešku u datoteci lsass.exe*

Pre uklanjanja crva potrebno je da instalirate zakrpu kako ne bi došlo do ponovne zaraze računara, a zatim i da blokirate TCP portove 445, 9996 i 5554 kako bi se sprečilo dalje širenje crva na računarima na kojima još nije instalirana zakrpa. Ukoliko se tokom tog postupka operativni sistem prisilno obara i pokreće i time onemogućava postupak nadgradnje, potrebno je da učinite sledeće:

- prekinute vezu računara sa Internetom ili lokalnom mrežom (čitaj: izvucite kabl),
- oborite i ponovo pokrenite operativni sistem,
- pokrenite komandu shutdown -i (start ⟶ run ⟶ shutdown -i),
- u Remote Shutdown Dialog izmenite broj sekundi (sa 20 na 9999) koji

određuje koliko će dugo korisnicima biti prikazano obaveštenje o gašenju računara pre samog gašenja,

- povežite računar na Internet ili lokalnu mrežu (čitaj: priključite kabl),
- instalirajte sigurnosnu zakrpu.

Preporučuje se da pre uklanjanja crva privremeno onemogućite System Restore.

Da biste uklonili crva, potrebno je da u memoriji računara pronađete procese koji ga sadrže. Procesi su u Task Manager programu imenovani kao izvršne datoteke. Dakle, procese koji sadrže crva možete detektovati pomoću bilo kog antivirusnog alata sa ažuriranom bazom (jednostavno "skenirajte" diskove). Zatim je potrebno u Task Manager programu (možete ga pozvati pomoću kombinacije tastera <Ctrl>+<Shift>+<Esc>) pronaći odgovarajuće procese i prekinuti ih (opcija End Process). Nakon zaustavljanja pokrenutih procesa možete preći na ručno uklanjanje crva sa zaraženog računara (brisanje datoteka koje je crv kreirao i registy ključa [HKLM\Software\Microsoft\Windows\CurrentVersion\Run] avserve2.exe = %Windows%\avserve2.exe). Detekciju i uklanjanje crva možete obaviti i pomoću namenskih programa kao što je Symantec W32-Sasser.Worm FixTool 1.0.4. Ovaj alat detektuje i uklanja datoteke i ključeve koje crv postavlja.

File-sharing i P2P crvi – Benjamin

Crv može pokušati da se iskopira u potencijalno deljeni direktorijum pod imenom koje navodi korisnika da pomisli da se radi o uslužnoj aplikaciji. Datoteka će tako postati vidljiva svakom ko ima pristup tom direktorijumu. Korisnici će preuzeti datoteku sa mreže i pokrenuti je, misleći da je to korisna aplikacija, što će rezultovati infekcijom rečunara.

P2P crv se kopira u neki P2P deljeni direktorijum. Nakon toga, P2P mreža odrađuje dalji posao oko propagacije crva: informiše druge korisnike o postojanju novog resursa i obezbeđuje infrastrukturu za preuzimanje datoteke. Složeniji P2P crvi oponašaju protokole P2P mreža – ovi crvi odgovaraju potvrdno na sve zahteve i, umesto prave datoteke, korisnicima nude telo crva.

Primer P2P crva je Benjamin. Benjamin koristi Kazaa P2P mrežu za razmenu datoteka za svoju propagaciju. Kazaa mreža dozvoljava korisnicima da međusobno razmenjuju datoteke koristeći Kazaa klijentski softver. Benjamin je napisan u Borland Delphi razvojnom okruženju, veličine je prosečno 216 Kb i komprimovan je pomoću AsPack alata. Veličina crva varira jer crv završava svaki fajl besmislenim karakterima (takozvani "*dust*") kako bi se zamaskirao. Prilikom instalacije crv prikazuje prozor sa izveštajem: "Error: Access error #03A:94574 Invalid Pointer Operation. File possibly corrupted". Crv se kopira u datoteku %WinDir%\system\explorer.scr i kreira dva registry ključa:

- [HKLM\Software\Microsoft\Windows\CurrentVersion\Run]
  "System-Service"="C:\\WINDOWS\\SYSTEM\\EXPLORER.SCR"
- [HKEY_LOCAL_MACHINE\Software\Microsoft]
  "syscod"="0065D7DB20008306B6A1"

Crv se izvršava nakon ponovnog podizanja sistema. Efekat crva je otvaranje sajta benjamin.xww.de u nameri da prikaže "reklamu".

Propagacija Benjamin crva je moguća samo ako je Kazaa klijent instaliran. Crv kreira direktorijum %WinDir%\Temp\Sys32 kog registruje kao direktorijum dostupan svim Kazaa mrežnim korisnicima. Zatim popunjava direktorijom sa raznoimenim kopijama samog sebe koje se razlikuju na osnovu dodatih besmislenih karaktera. Imena datoteka se generišu na osnovu liste koja se nalazi u telu crva. Korisnik neinficirane mašine pretražuje Kazaa i pronalazi listu dostupnih datoteka na inficiranoj mašini. Korisnik preuzima datoteku, pokreće je i na taj način inficira svoj računar.

14.3 Trojanski konji

Sličnost između trojanskog konja opisanog u udžbeniku iz istorije i trojanskog konja opisanog ovde je metodika napada. Trojanski konji (ili kraće, trojanci) su zlonamerni programi koji se maskiraju i reklamiraju kao korisni programi kako bi se korisnici prevarili, tj. naterali da te programe pokrenu (sociološki inženjering je očigledno prisutan). Na primer, trojanac može da bude zlonamerni crv upakovan u formu programa za instalaciju

neke manje aplikacije (na primer, setup.exe). Dovoljno je da se ovakva datoteka postavi na neku Web stranicu i da na nju upućuje link poput "besplatan program za ...". Trojanac će najčešće pokušati da zloupotrebi inficirani sistem u cilju krađe poverljivih informacija, slanja crva u elektronskoj pošti radi izvršavanja DoS napada ili korišćenja resursa računara (na primer, procesorskog vremena). Alternativna metoda za širenje trojanskih konja je upotreba crva kao nosioca. Na primer, Baggle je email crv koji je spoofing tehnikom "From" polja u zaglavlju paketa pokušavao da instalira *backdoor* na žrtvi. Kao posebna vrsta trojanaca mogu se izdvojiti logičke bombe (kod ubačen u legitimne programe) koje se aktiviraju pod specijalnim uslovima i mogu oštetiti sistem.

Iako se ne mogu precizno klasifikovati u određene kategorije, u praksi se najčešće pojavljuju sledeće vrste trojanaca: trojanci koji otvaraju "zadnja vrata", nosioci zlonamernog softvera, kradljivci informacija, proksi serveri i programi koji pozivaju telefonske brojeve.

- Trojanac koji otvara "zadnja vrata" (engl. *backdoor*) je program koji omogućava udaljenom korisniku da pristupi inficiranom računaru i to najčešće tako da vlasnik računara nije ni svestan "posetioca". Nakon infekcije, napadač može da koristi resurse inficiranog računara, u koje spadaju i poverljive informacije, kao što su lozinke. Kao primer navodimo čuveni Back Orifice 2K (BO2K)[79] koji omogućava da se sa udaljenog računara na računaru na kom se nalazi BO2K server prate pritisnuti tasteri, prenose datoteka, upravlja deljenim direkrorijumima, modifikuje registry baza, pristupi konzolnim programima (kao što je komandni interpreter) preko Telnet servisa, kontrolišu procesi ili preusmere mrežnih zahtevi.

- Postoji nekoliko specifičnih vrsta kradljivaca informacija. PSW[80] trojanac pokušaće da pretraži inficirani računar kako bi došao do poverljivih informacija kao što su lozinke[81], privatni i javni ključevi,

79 BO2K možete preuzeti sa stranice http://www.bo2k.com/. Pazite, Back Orifice možete koristiti zlonamerno i dobronamerno, na Vama je da odlučite kako ćete ga koristiti.

80 Skraćeno od reči "password"

81 Ovakve informacije možete "pribaviti" i pecanjem (engl. *phishing*), tj. postavljanjem Web stranice (za elektronsku trgovinu) i metodama sociološkog inženjeringa kojim ćete ubediti posetioca sajta da popuni formular koji zahteva navođenje poverljivih podataka (na primer, broja kreditne kartice). Dodatno, žrtvi možete da pošaljete poruku, lažno se predstavite i zatražite od primaoca da poseti tu stranicu. Začuđujuće, ljudi se lako "upecaju"...

sertifikati i/ili detalji vezani za kreditne kartice. Nakon sakupljanja korisnih informacija, PSW trojanac će svom tvorcu poslati e-mail koji sadrži prikupljene podatke, ili će te podatke preneti na neku drugu lokaciju radi skladištenja, kako bi tvorac trojanca kasnije mogao da ih pročita. Slične aktivnosti obavljaju i špijuni (engl. *trojan spy*) i takozvani "obaveštajci" (engl. *trojan notifiers*). Špijun miruje na inficiranom računaru i beleži pritisnute tastere[82] (engl. *keylogging*), snima ekrane, ili na neki drugi način omogućava napadaču da prati rad korisnika. Obaveštajac šalje informacije kao što su IP adrese, e-mail adrese i status portova autoru trojanca. Često se koristi kao deo zlonamernog softverskog paketa koji obaveštava autora o uspešnoj instalaciji crva ili *backdoor* trojanca.

- Nosioci softvera su, obično, realizovani u vidu trojanskih konja koji se nakon instalacije ponašaju kao magnet za drugi zlonamerni softver. *Downloader* obično miruje na računaru i pokušava sa Interneta da preuzme i instalira drugi zlonamerni softver. Slično se ponaša i *dropper*, koji u izvršnoj datoteci sadrži kod drugih zlonamernih programa. Prilikom pokretanja, dropper će izvršiti ekstrakciju drugog softvera na računar iz svoje izvršne datoteke.

- Trojanski proksi server (engl. *trojan proxy*) će pokušati da preobrati inficirani računar u proksi server, čime se udaljenim korisnicima dozvoljava da preko inficiranog računara anonimno pristupe Internetu. Ovim se inficirani računar efikasno pretvara u zombija koji se može iskoristiti za slanje spam poruka ili za učestvovanje u DoS napadu.

- Cilj programa koji pomoću modema pozivaju "egzotične" telefonske brojeve (engl. *dialers*) je da žrtvi (tj. vlasniku računara na kome se izvršavaju) obezbede sumanuti telefonski račun. Nakon toga ih korisnici obično primete. Ukoliko modem nije vezan na telefonsku liniju, ovi programi su bezopasni.

Trojanski konji, u opštem slučaju, inficiraju računar, ali ne i datoteke, što omogućuje njihovu jednostavnu detekciju i uklanjanje sa računara. Oni često kreiraju zapise u registry bazi, kako bi omogućili svoje izvršenje prilikom svakog pokretanja sistema.

82 Keylogging se koristi, na primer, radi krađe detalja vezanih za kreditne kartice.

14.4 Virusi

Verovatno su najpodmuklija vrsta od svog raspoloživog zlonamernog softvera. Česti efekti infekcije virusima su brisanje vežnih datoteka i/ili dovođenje sistema u stanje u kome ne može normalno da se koristi. Za razliku od crva, virusi ne koriste mrežne resurse za širenje, ali se mogu širiti preko mreže kao deo nekog crva. Virusi se šire oslanjajući se na činjenicu da korisnik ne zna da je poslao inficiranu datoteku kao prilog e-mail poruke ili da je prijatelju polonio CD na kome se nalazi aplikacija zaražena virusom.

Virusi se mogu klasifikovati prema okruženju u kome virus može da obavi inficiranje drugih objekata, i prema metodama infekcije, tj. tehnikama za umetanje virusa u neki objekat. Okruženje u kome se virusi mogu naći su sistemi datoteka i okruženja za izvršenje makroa i skriptova (engl. *script host*).

Virusi u sistemu datoteka

Virusi ove vrste koriste jednu ili više vrsta fajl sistema za širenje. Najveći broj ovih virusa inficira izvršne fajlove. Prema metodama infekcije fajl sistem virusi se mogu podeliti na:

- viruse koji prepisuju postojeći kod (engl. *overwriting*),
- parazitske viruse (engl. *parasitic*),
- pridužujuće viruse (engl. *companion*),
- virusi startnog zapisa (engl. *boot-sector*).

Virusi koji prepisuju postojeći kod koriste najjednostavniji metod infekcije: virus menja deo koda inficirane datoteke svojim kodom, a datoteka nakon toga postaje neupotrebljiva. Ovi virusi se, zbog toga, lako detektuju, ali se teško čiste jer antivirusni softver ne zna kako da rekonstruiše originalni kod datoteke. Metoda zaštite koja donekle može obezbediti rekonstrukciju uništenih datoteka je povremeno kopiranje svih izvršnih datoteka u neki zaštićeni direktorijum, tj. čuvanje kopija zdravih izvršnih datoteka u karantinu.

Parazitski virusi dodaju svoj kod u datoteku tako da datoteka ostane delimično ili potpuno funkcionalna. Virus može upisati svoj kod:

- na početak datoteke (engl. *prepending*)

  Ovi virusi mogu dodati svoj kod pomeranjem zdravog koda sa početka na kraj izvršne datoteke i upisivanjem malicioznog koda na početak. Alternativno, virus može dodajti kod zdrave datoteke na svoj kod. U oba slučaja, nakon pokretanja inficirane datoteke, najpre se izvršava kod virusa. Kako bi održao integritet aplikacije, virus može privremeno očistiti inficiranu datoteku, dozvoliti joj da se normalno izvrši, a zatim je ponovo inficirati. U tom slučaju, virus može da koristi privremene datoteke za skladištenje čistih verzija inficirane datoteke, ili da očisti aplikaciju u memoriji i sredi potrebne memorijske adrese.

- na kraj datoteke (engl. *appending*)

  Većina virusa pripada ovoj kategoriji. Virus najpre dopisuje svoj kod na kraj inficirane datoteke, a zatim modifikuje ulaznu tačku (engl. *entry-point*) u zaglavlju datoteke kako bi osigurao da će se pre izvršenja samog programa izvršiti maliciozni kod. Ovi virusi se lako detektuju i čiste sa izvršnih datoteka na osnovu antivirusnih definicija.

- unutar postojećeg koda (engl. *inserting*)

  Za dodavanje malicioznog koda unutar postojećeg koda virus koristi dve tehnike: pomera originalni, zdravi kod na kraj datoteke, ili upisuje svoj kod u šupljine zdravog koda (engl. *cavity virus*), na primer, u šupljine između sekcija .exe datoteka. Ukoliko je virus ovog tipa loše napisan (jednostavno prepisuje sekciju koda koja je neophodna za izvršenje aplikacije), aplikacija neće funkcionisati i virus će najverovatnije biti brzo detektovan.

EPO virusi (engl. *entry point obscuring*) mogu se izdvojiti kao posebna kategorija parazitnih virusa. Ovoj kategoriji pripada manji broj virusa koji svoj kod upisuju na kraj datoteke ili unutar postojećeg koda. EPO virusi su karakteristični po tome što ne modifikuju adresu ulazne tačke u zaglavlju .exe datoteke. EPO virus upisuje rutinu koja izvršava telo virusa negde pri sredini datoteke. Telo virusa se izvršava samo ako je rutina koja sadrži virus pozvana. Ukoliko se ova rutina retko kada koristi (npr. poruka o grešci koja

se retko javlja), EPO virus može biti neaktivan dugo vremena. Pisac virusa mora da odabere ulaznu tačku pažljivo – loše odabrana ulazna tačka može dovesti do oštećenja nosioca ili do toga da virus ostane dovoljno dugo neaktivan (u tom slučaju se može desiti da korisnik jednostavno obriše inficiranu datoteku jer mu ona više ne treba).

Kao primer parazitskog virusa (ranije su bili jako popularni, jer su se širili preko disketa sa programima) navodimo Pcvrs.1904. To je parazitski virus koji ostaje rezidentan u memoriji i presreće interapte INT 21h i 16h (tastatura). Dopisuje se na početak .COM datoteka (osim COMMAND.COM) i na kraj .EXE datoteka koje su izvšene. Posle nekog vremena (nakon izvršenja) virus počinje da briše datoteke koje su otvorene. Svakog 23. u mesecu (ukoliko je to ponedeljak), virus formatira hard disk i prikazuje poruku: “PcVrsDs Version 1.00 Copyright (C) VirOP 1990.”

Pridružujući virusi ne menjaju sadržaj originalne datoteke, već samo njeno ime i kreiraju novu datoteku pod originalnim imenom koja sadrži virus. Umesto zdrave datoteke, prvo se izvršava virus, a zatim zdrava datoteka. Na primer, virus može iskopirati datoteku notepad.exe u notepad.dat, a zatim upisati svoj kod u notepad.exe. Svaki put kada korisnik otvori notepad, izvršava se telo virusa koje zatim pokreće notepad.dat. Postoje i druge vrste pridružujućih virusa – na primer, path-companion koji smešta svoje kopije u Windows sistemski direktorijum, pretpostavljajući da je on naveden prvi u promenljivoj PATH.

Virusi startnog zapisa svoj kod upisuje u glavni startni zapis hard diska (engl. *master boot record*) ili startni zapis (engl. *boot sector*) aktivne particije na disku. Po potrebi, virus obog tipa može upisati svoj kod u neki sektor na disku, a zatim promeniti vrednost u MBR-u. Zasnovani su na principima na kojima radi rutina za podizanje operativnog sistema.

Makro virusi

Najčešće su napisani i ugrađeni u dokumente koji se otvaraju aplikacijama iz Microsoft Office paketa koje koriste OLE2 tehnologiju (Object Linking and Embedding). Makro virusi za druge aplikacije su relativno retki. Lokacija virusa u MS Office dokumentu zavisi od formata datoteke, koji je kod MS aplikacija najčešće vrlo složen. Svaki Word ili Excel dokument snima

se kao sekvenca blokova podataka (svaki blok ima poseban format) povezanih meta-podacima (engl. *service data*). Mesto virusa u dokumentu objasnićemo simbolički:

- zaglavlje – meta podaci – tekst – fontovi – makroi - MAKRO VIRUS – ostali podaci

Prilikom rada sa dokumentima, MS Word izvršava razne akcije: aplikacija otvara, snima, štampa ili zatvara dokument. MS Word pri tome traži i izvršava adekvatne makroe. Na primer, komanda File → Save će pozvati FileSave makro pod pretpostavkom da je taj makro ispravno definisan i konfigurisan. Takođe postoje i auto-makroi (engl. *auto-macros*), koji se automatski izvršavaju u određenim situacijama, tj. ne zahtevaju da ih korisnik eksplicitno pozove. Na primer, prilikom otvaranja dokumenta, MS Word će proveriti da li u dokumentu postoji AutoOpen makro. Ukoliko makro postoji, Word će ga izvršiti. Prilikom zatvaranja dokumenta, Word će izvršiti AutoClose makro, ukoliko isti postoji. Prilikom pokretanja, Word će izvršiti AutoExec makro. Takođe, u Word-u postoje makroi koji se izvršavaju u određeno vreme, određenog datuma, ili onda kada korisnik pritisne odgovarajuću kombinaciju tastera.

Po pravilu, makro virusi koji napadaju Office dokumente koriste jednu od prethodno opisanih tehnika. Virus može:

- da sadrži auto-makro,
- da se oslanja na redefinisanje standardnih sistemskih makroa (asociranih sa stavkom u meniju) i
- da sadrži macro koji se poziva kada korisnik pritisne odgovarajuću kombinaciju tastera.

Kada se makro virus izvrši i preuzme kontrolu, pokušaće da prenese svoj kod na druge datoteke, najčešće na one koje su otvorene u aplikaciji. Takođe, virus može na disku da traži druge datoteke.

Kao primer makro virusa, opisaćemo ukratko Melissa virus. Melissa se propagira putem elektronske pošte koja sadrži inficirani Word dokument kao prilog. Poruka nosilac obično ima sledeće osobine:

- naslov (*subject*): “Important Message From <puno_ime_pošiljaoca>”,

- tekstualni sadržaj: “Here is that document you asked for ... don't show anyone else ;-)”,
- U prilogu se nalazi datoteka "list.doc" koja sadrži reference ka pornografskim Web stranicama i makro virus.

Pod izvesnim uslovima, virus može da generiše poruke sa drugim prilozima koje je kreirala žrtva, a koji su, takođe, inficirani. Makro virus se izvršava čim korisnik otvori inficiranu .doc datoteku u Microsoft Word-u 97/2000 (pod uslovom da su makroi dozvoljeni). Nakon izvršenja, virus obara nivo sigurnosnih podešavanja koja se tiču makroa (*macro security settings*) kako bi dozvolio svim makroima da se nesmetano izvrše prilikom otvaranja dokumenata. To znači da korisnik neće biti obavešten sledeći put kad se makro virus izvrši.

Makro virus proverava da li u registry bazi postoji ključ HKCU\Software\Microsoft\Office\Melissa? i da li taj ključ ima vrednost "... by Kwyjibo". Ukoliko ključ ne postoji ili nema tu vrednost, virus pokušava da se proširi. Virus se širi tako što prethodno opisanu e-mail poruku šalje pedesetorici prvih korisnika koje virus pronađe u svakoj Microsoft Outlook MAPI *address book* datoteci koja je čitljiva korisniku pod čijim se SID-om makro izvršava. Da bi se virus širio dalje, Microsoft Outlook mora biti instaliran. Virus ne može slati e-mail poruke sa sistema koji rade po Mac OS operativnim sistemom, ali se može čuvati na Mac OS. Virus zatim postavlja vrednost prethodno pomenutog ključa na "... by Kwyjibo". To znači da se virus propagira samo jednom sa inficirane mašine. Ukoliko se ključ izbriše, virus će se opet propagirati preko elektronske pošte.

Virus zatim inficira Normal.dot šablon (engl. *template*), koji, ukoliko se drugačije ne naglasi, koriste svi Word dokumenti. To znači da će svi novokreirani dokumenti koji koriste taj šablon biti inficirani. Ukoliko se minut u satu poklapa sa danom u mesecu, makro upisuje u tekući dokument poruku "Twenty-two points, plus triple-word-score, plus fifty points for using all my letters. Game's over. I'm outta here."

Ukoliko zabranite izvršavanje makroa i otvorite dokument, ni Word 97 ni Word 2000 neće izlistati makro. Melissa je parče VBA koda asocirano sa “document.open” methodom. Kod je vidljiv iz Visual Basic editora.

Skript virusi

Script virusi su podskup fajl sistem virusa, pisani u script jezicima (VBS, JavaScript, BAT, PHP). Script virusi su sposobni da inficiraju datoteke u drugom formatu, kao što je HTML format, ukoliko datoteke tog formata omogućavaju i dozvoljavaju izvršavanje skriptova. Ovi virusi mogu da funkcionišu kao deo složenog višedelnog virusa, ili samostalno (virus će inficirati druge Windows ili Linux skriptove).

Primer skript virusa je JS/Fortnight. Fortnight je realizovan kao Java Script, koji se širi koristeći tehnike karakteristične za e-mail crve (preko email poruka sa skrivenim linkovima ka Web stranici sa infektivnim kodom). Crv koristi sigurnosni propust Microsoft virtualne mašine (VM) vezan za izvršavanje ActiveX kontrola. Ova ranjivost dozvoljava da se ActiveX kontrole kreiraju i koriste iz Web stranice, ili iz HTML kodirane e-mail poruke. Ovaj propust dozvoljava automatsko pokretanje skrivenog linka iz poruke nosioca i izvršavanje koda na lokalnom računaru. Microsoft je naknadno objavio zakrpu koja razrešava ovaj nedostatak. Fortnight crv koristi kolačić (engl. *cookie*) nazvan "TF" za obeležavanje inficiranih računara. Ukoliko kolačić nije prisutan, Fortnight menja adresu podrazumevane Web stranice Internet Explorera na pornografski sajt. Crv dalje kopira podrazumevani potpis pošte za Outlook Express 5.0 u datoteku C:\Program Files\sign.htm sa linkom ka telu crva. Sve poruke poslate sa inficiranog računara nosiće u sebi taj link. Fortnight, zatim, kreira tri linka u Favorites folderu: "SEXXX. Totaly Teen.url", "Make BIG Money.url" i "6544 Search Engines Submission.url". Sajt koji sadrži telo crva je ugašen.

14.5 Špijunski programi

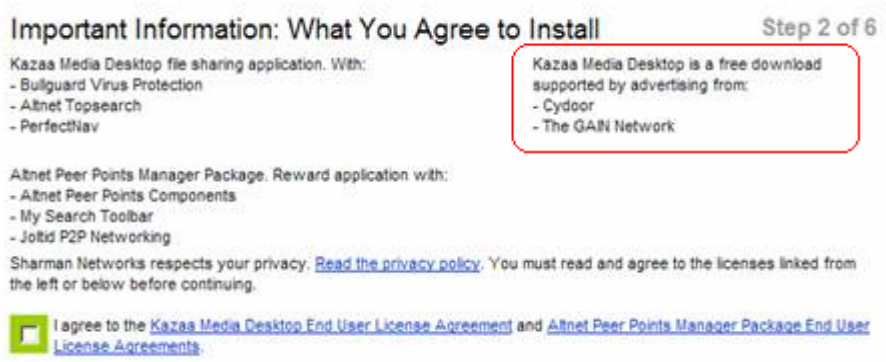
Špijunski softver (engl. spyware) je neželjeni program, instaliran na računaru bez znanja (ili odobrenja) korisnika, koji prikuplja informacije o aktivnosti korisnika (na primer, softver koji se koristi i posećene Web stranice), lozinke i finansijske informacije. Takođe, u špijuske programe mogu se ubrojiti i trojanski konji iz kategorije kradljivaca informacija (na primer, *keyloggeri*). Posebna vrsta špijunskih programa – reklamni špijunski programi (engl. *adware*) ove informacije prikuplja i šalje kompanijama koje se bave posebnom vrstom marketinga (*behavioural marketing*, zasnovan na

praćenju Vaših navika pri pretraživanju Weba i oglašavanjem preko iskačućih prozora i banera ugrađenih u aplikativni softver). Simptomi infekcije špijunskim prozorima su:

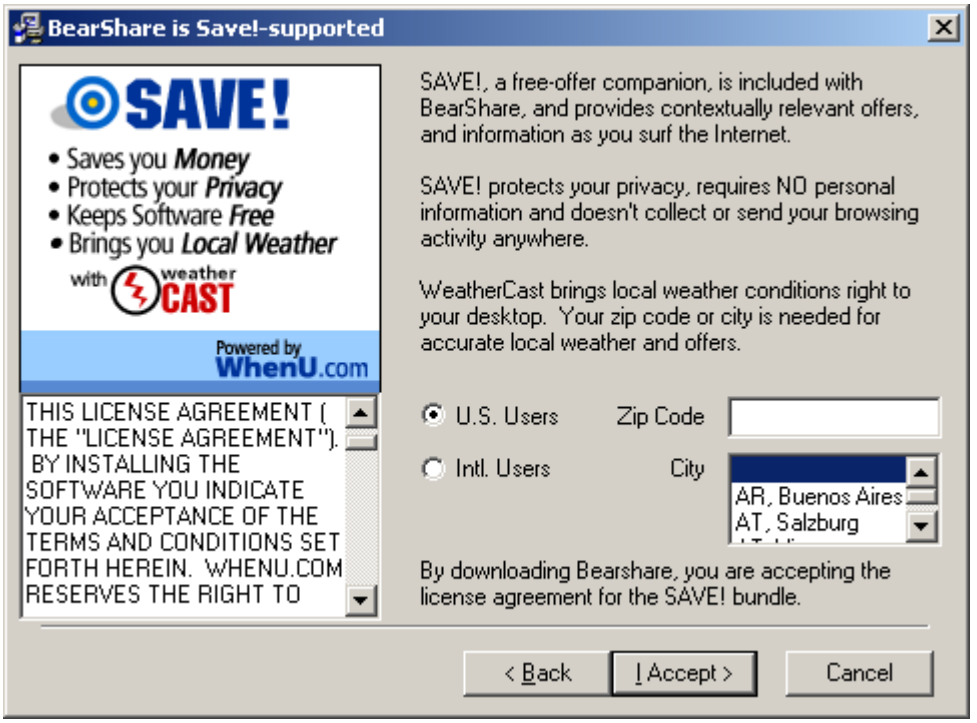
- neželjeni pop-up prozori sa reklamama koji se pojavljuju dok pretražujete Internet, otvaranje nove instance Web browsera sa neželjenim reklamama,
- promene na Web browseru - novi toolbar, promenjene ikonice ili baneri, promena podrazumevane Web stranice u browseru (engl. *home page*),
- računar se automatski, bez znanja korisnika, povezuje na Internet (koristeći Dialer trojanske konje) kako bi prikazao reklame,
- Web zahtevi su preusmereni, a veza sa Internetom je znatno sporija,
- računar radi sporije i nestabilno; operativni sistem se često blokira.

Osim krađe informacija (koja se u većini zemalja se smatra prekršajem), špijunski programi, takođe, kradu i resurse Vašeg računara i propusni opseg Vaše veze sa Internetom (za slanje prikupljenih informacija i prenos reklama).

Firme i pojedinci koji kodiraju špijunske programe najčešće ugrađuju svoj kod u razne datoteke i besplatni (engl. freeware) ili probni (engl. *shareware*) softver koji se može preuzeti sa Interneta. Špijunski program se ugrađuje kao deo izvršne datoteke korisnog programa (*hard-coded*) ili kao posebna aplikacija koju će rutina za instaliranje programa instalirati na Vaš računar. Ovakav softver je obično praćen veoma dugim ugovorom o korišćenju (engl. *End User Licence Agreement*, EULA) u kome je navedeno da se aplikacija isporučuje sa pratećim kodom koji će "najverovatnije" informisati neku marketinšku kompaniju o Vašim aktivnostima na Internetu; za uzvrat, dobićete gomilu šarenih banera (ne preporučujemo da "klinknete" na njih) i iskačuće prozore sa reklamama. Ukoliko u EULA pronađete ovakve tekstove, potrudite se da ne instalirate program (ili ga, ukoliko već morate, instalirajte na nekoj virtuelnoj mašini). Veoma popularni programi koji se isporučuju sa *spyware* i/ili *adware* komponentama su: Kazaa Media Desktop (dijalog na slici 14.3 ukazuje na instaliranje CyDoor i GAIN Gator *adware* komponenti), e-Donkey, BearShare (dijalog na slici 14.4 ukazuje na instaliranje WhenU SAVE! *adware* komponente), FlashGet, Daemon Tools, kao i razni čuvari ekrana i desktop teme upakovani u rutine za instaliranje.



Slika 14.3 *– Dijalog koji ukazuje na postojanje adware komponenti*

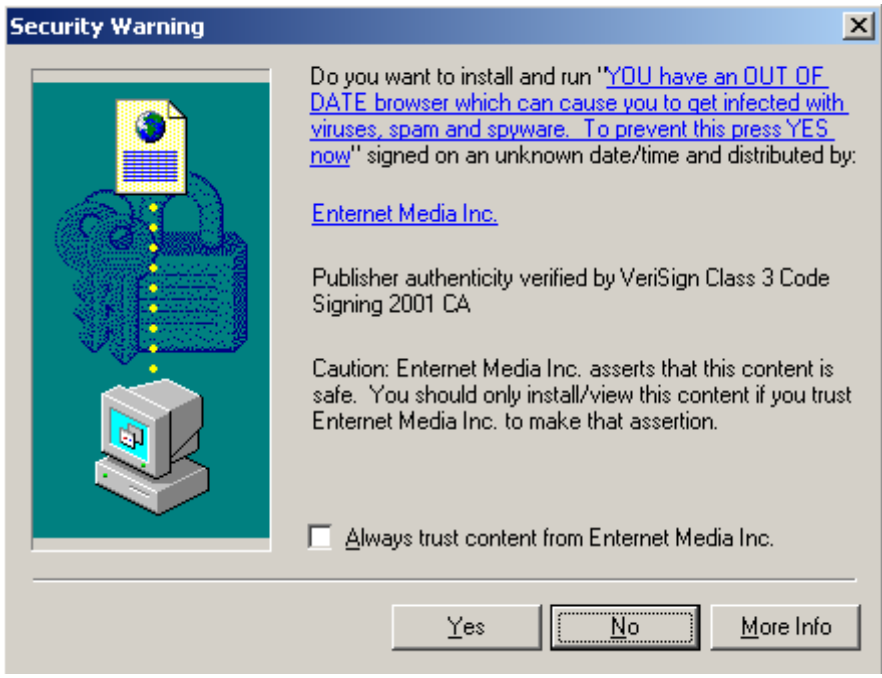


Slika 14.3 *– Instalacija špijunskih programa koji prate besplatni softver*

Kao primer, navodimo slučaj korisnika koji je preuzeo čuvar ekrana, tj. screensaver sa web stranice www.freeze.com u obliku izvršne datoteke.

Izvršna datoteka (navodno namenjena instalaciji screensavera) je na računar, osim screensaver datoteke, instalirala sledeće: Shop@Home i Bargain Buddy (vidljivi su u System Tray-u), New.net, WhenU, VX2, VirtuNet, SearchSquire, Kontiki plug-in, BackWeb, 180solutions i Cydoor. Računar je postao praktično neupotrebljiv – najjednostavnije i najsigurnije rešenje bilo je u reinstaliranju operativnog sistema. Freeze.com je ostao zakonski "čist", jer je u EULA napomenuto da instalacioni program može instalirati dodatan softver na računaru koji će reklamnom serveru dostavljati podatke o Web stranicama koje ste posetili kako bi vam server dostavio reklamni sadržaj.

Špijunski programi se takođe mogu instalirati na računarima naivnih korisnika koristeći metodiku širenja karakterističnu za e-mail crve. Još jedan od načina za instalaciju špijunskih programa je korišćenje ActiveX kontrola, koje dozvoljavaju tvorcima zlonamernih programa da na vaš računar instaliraju neželjeni softver pod maskom legitimnog dodatka (engl. *plug-in*) za Vaš Web browser. Na slici 14.5 prikazan je dijalog koji pokušava na računar žrtve da instalira špijunsku komponentu



Slika 14.5 *– Traženje dozvole za instalaciju špijunske komponente*

Špijunski programi se najšećće instaliraju kao softver koji nije registrovan u "Add/Remove Programs" sekciji Control Panel-a, što znači da ga je teško detektovati i ukloniti. Konkretno, detekcija špijunskih programa je najveći problem softverskih kompanija koje prave razna komercijalna ili besplatna *anti-spyware* rešenja. Najjednostavniji vid potencijalne špijunaže su špijunski kolačići (engl. *tracking cookies*) – tekstualne datoteke koje marketinške kompanije koriste kako bi pribavile informacije o tome kada je koji korisnik posetio neku Web stranicu. Ovakav vid špijunskog softvera se najlakše uklanja. Međutim, pravi kradljivci informacija (poput *keyloggera*), koji se koriste za ozbiljnije namene su znatno komplikovaniji od običnih kolačića. Ovakav kod se najčešće čuva na nekoliko lokacija, formira zapise u registry bazi i omogućava reinstalaciju nakon uklanjanja, i najčešće ima mogućnost da sprečava ili ometa rad anti-spyware sotvera.

Najjednostavniji način zaštite od špijunskih programa je instaliranje programa koji sprečava instalaciju zlonamernog koda (vidite 14.6), a sledeći postupci vam takođe mogu biti od pomoći:

- pazite šta od softvera instalirate – pročitajte EULA i proverite na Web stranici www.softpedia.net da li je taj softver označen labelom "100% free – no spyware, no adware, no viruses",

Slika 14.6 *– Softpedia označava čist softver*

- koristite alternativni Web browser (na primer, Mozilla Firefox ili Opera),
- ukoliko baš "morate" da koristite Internet Explorer, sprečite mogućnost instalacije ActiveX kontrola (ili instalirajte samo one kojima verujete),
- ne "klikćite" po pop-up prozorima i reklamnim banerima na kojima piše da ste osvojili novac ili dobili besplatan iPod – tamo ionako nećete naći ništa zanimljivo (novac verovatno niste osvojili, a iPod

skoro sigurno nećete dobiti besplatno, ali ćete zato dobiti besplatno nekoliko novih ActiveX kontrola koje sigurno ne želite),

- pripazite šta je od elektronske pošte legitimno (verovatnoća da će vam Microsoft na mail poslati kritičnu zakrpu za Windows je veoma mala).

Top lista špijunskih programa

Navešćemo i dati kraći opis špijunskih programa koji se najčešće sreću:

- Gator

  Adware komponenta kompanije Claria (poznatija kao Gator). Pominje se i pod imenima eWallet, GAIN ili Claria. Osnovne karakteristike: rezidentan je, povezuje se preko Interneta na odgovarajuće servere i prikazuje reklame.

  Prema zvaničnom opisu, Gator je "softverski proizvod" koji automatski popunjava lozinke i razna polja u formama na Web stranicama. Ovaj servis se isporučuje zajedno sa OfferCompanion reklamnim modulom koji prikazuje pop-up reklame prilikom posete nekih Web stranica (reklame su jasno označene, a dostavlja ih marketinška kompanija GAIN Publishing). Gator naznačava da je softver, pošto je rezidentan u memoriji, sposoban da usmeri korisnike ka "specijalnim ponudama prilikom kupovine" na osnovu imena domena i sadržaja stranice koju korisnik posećuje. Trenutno, Calria naznačava da se kompaniji GAIN Publishing šalje samo ime, poštanski broj i država korisnika.

  Trenutno postoji (bolje reći, "kruži") nekoliko verzija Gatora sa različitim polisama o privatnosti korisnika. Ove polise i prateći EULA možete naći na: http://gator.com/help/privacy_statement.html. Ako baš želite da instalirate Gatora (ne preporučujemo), pročitajte prvo EULA i polisu. Gator reklamne komponente prisutne su i u drugom softveru (kao što je, na primer, 4Arcade PBar toolbar).

- CoolWebSearch (CWS)

  Veoma opasna adware komponenta (prema www.spywareguide.com, CWS ima SG indeks 10[83]). Poznta je i pod sledećim imenima CWS,

83 Indeks 10 opisuje izuzetno opasne špijunske programe, koji su polimorfni i koriste stealth tehnike instalacije. Ovakvi programi su sposobni da otvore portove, koriste

CoolSearcher. BootConf, MSInfo, SvcHost, DNSRelay, DataNotary, Trojan.Norio, Jetseeker, winlink, Xplugin, coolwwwsearch, Aze Search Toolbar, Trojan.StartPage (Sunbelt), WinRes Spyware.CHM.A.

Jednostavno rečeno, CWS je kradljivac browsera sa sledećim karakteristikama: rezidentan je, napada zaštitini softver, koristi stealth metode, instalira drugi softver, povezuje se na Internet i prikazuje reklame, menja podešavanja Internet Explorera.

Ukratko ćemo opisati neke varijante:

- CWS/DataNotary: kopira CSS stylesheet datoteku u Windows direktorijum i postavlja je podrazumevanom za sve stranice. Ova datoteka uključuje Java skript koji pokušava da odredi kada korisnik posećuje stranice sa pornografskim sadržajem, a zatim preusmerava browser na datanotary.com.
- CWS/BootConf: radi slično kao i DataNotary, s tim što se preusmerava na www.coolwebsearch.com. Menja podrazumevanu početnu stranicu i pretraživač na coolwebsearch. Takođe, menja sadržaj DNS hosts datoteke kako bi sve zahteve za pretraživanjem u MSN search-bar preusmerio na coolwebsearch.com. Program bootconf.exe koji održava ta podešavanja se pokreće prilikom svakog podizanja operativnog sistema.
- CWS/SvcHost: menja sadržaj hosts datoteke i krade browser na sledeći način: Yahoo Search, MSN Search i sve verzije Google pretraživača preusmeravaju se na adresu lokalne petlje 127.0.0.1. CWS pretpostavlja da na računaru nije podignut Web server, te da će preusmeravanje zahteva na adresu lokalne petlje prouzrokovati stranicu o greški. CWS krade stranicu o greški, tj. preusmerava browser na slawsearch.com.
- CWS/PnP: instalira se pomoću oemsyspnp.inf datoteke, preko sekcija RunOnce, AudioPnP, VideoPnp, IdePnP ili SysPnP. Preusmerava ka www.adulthyperlinks.com i www.allhyperlinks.com i dodaje stranicu activexupdate.com u listu bezbednih stranica za instalaciju ActiveX kontrola.
- CWS/MSSPI: kradljivac rezultata pretrage implementiran kao

metode za sprečavanje rada zaštitnih programa i napadaču obično omogućavaju potpunu kontrolu nad žrtvom. Ukoliko se bave reklamiranjem, to rade neusmereno i na veoma agresivan način.

Winsock2 Layered Service Provider (mrežna komponenta niskog nivoa koja se teško uklanja). Preusmerava stranice rezultata pretrage Google, Yahoo i Altavista pretraživača ka reklamama koje generiše stranica unipages.cc.

- CWS/Winres: često menja početnu Web stranicu i sa Inteneta preuzima i instalira *adware* kao što je 2020 search, isearch, itd.

CoolWebSearch je jedna od najozloglašenijih familija špijunskih programa. Ukoliko detektujete CWS na računaru, uklonite ga pomoću programa kao što su SpyBot ili Adware Away.

- 180search Assistant

Adware komponenta koju proizvodi 180Solutions, Inc, koja se pominje i pod imenima SVAPlayer, 180solutions i MetricsDirect. Prema stranici www.spywareguide.com ima SG indeks 7[84]. Više o ovom programu možete pročitati na www.180searchassistant.com/home.html.

Isporučuje se uz besplatne programe koji se finansiraju preko reklamiranja (taktika poznatija kao "*bundling*"). Prema zvaničnoj polisi o privatnosti, 180 Solutions prikuplja samo informacije o posećenim Web stranicama. URL ili ključna reč se šalje sa jedinstvenim identifikatorom ka reklamnom serveru (engl. *advertising server*) koji nakon toga prikazuje reklamu na ekranu korisnika.

Prema izveštajima Bena Edelmana (www.bedelman.org), ovaj špijunski program se može instalirati pomoću sigurnosnih propusta Internet Explorera. Takođe, primećeno je da se različite 180 Solutions aplikacije instaliraju istovremeno sa drugim *adware* komponentama.

- Cydoor

Cydoor je adware komponenta kompanije Cydoor Technologies Ltd. sa relativno niskim SG indeksom 3 (prema www.spywareguide.com). Prema zvaničnom opisu, "Cydoor tehnologija dostavlja precizno usmerene reklame na desktop pomoću aplikacija koje podržavaju ovaj koncept". Više informacija možete naći na www.cydoor.com/Cydoor. Pazite, Cydoor koristi Vašu Internet vezu da preuzme reklame i

84 SG indeks 7 uukazuje na to da program prati aktivnosti i da se jako teško uklanja sa računara na kom je instaliran.

preproda statistike korišćenje. Uklanjanje ove komponente može da prouzrokuje prestanak rada nekih programa nosioca. Ukoliko vam te aplikacije nisu od kritičnog značaja, preporučujemo da obrišete Cydoor.

- ISTbar

  ISTbar je Internet Explorer toolbar, kradljivac početne stranice i rezultata pretrage kompanije Integrated Search Technologies/CDT Inc. SG index ISTbar komponente je 6[85]. Instalira se koristeći ActiveX sa Web stranica, najčešće onih koje se bave pornografijom. ISTbar takođe instalira druge parazite, kao što je RapidBlaster/Ip komponenta koja izbacuje pop-up prozorčiće sa pornografskim sadržajem.

  Varijanta ISTbar/AUpdate se instalira pomoću agresivnog Java skripta koji se ponovo otvara ukoliko korisnik odbije preuzimanje ActiveX kontrole. Varijanta ISTbar/XXXToolbar preusmerava ka kontrolišućem serveru xxxtoolbar.com ili ka stranici slotch.com.

- WhenU Desktop Bar

  Desktop toolbar koji omogućava brzu pretragu, prati Internet saobraćaj i omogućava izvršavanje koda sa udaljenog računara. Prikazuje reklamni sadržaj.

  Prema stranici www.spywareguide.com, WhenU Desktop Bar ima SG indeks 4[86].

Slika 14.7 *– WhenU Desktop-bar*

85 SG indeks 6 dodeljen je programima koji prate aktivnosti vezane za tastaturu i šalju informacije o sistemu.

86 Indeks 4 označava špijunski softver koji je gotovo nemoguće ručno ukloniti.

14.6 Programi za zaštitu od virusa i špijunskih programa

Bez detaljnijih opisa softvera, navešćemo prost scenario zaštite od virusa i špijunskih programa. Sav softver koji ovde opisujemo je sa otvorenim kodom, besplatan, ili, u najgorem slučaju, besplatan za ne-komercijalnu upotrebu. Kao što je već rečeno, najčešći načini da se inficirate virusom, crvom, ili špijunskim programom jesu:

- instaliranje softvera sumnjivog porekla (ili sa sumnjivim EULA) koji ste preuzeli sa neke neproverene Web stranice

- "kliktanje" po banerima ili pop-up prozorima koje pokreneće neki Java skript ili davanje dozvola Web stranicama (kao što je, na primer, www.xxxtoolbar.com) za instalaciju neke ActiveX kontrole).

Prvi problem ćete rešiti ukoliko softver koji instalirate potiče od onih proizvođača kojima verujete i ukoliko ga preuzmete sa Web stranice tog proizvođača. Takođe, potrebno je da obratite pažnju na EULA, što ponekad može da predstavi problem (ukoliko je EULA dugačak 15-20 stranica, a baš takvi su baš karakteristični za proizvode koji uključuju špijunski softver). Ovaj problem ima dva moguća rešenja:

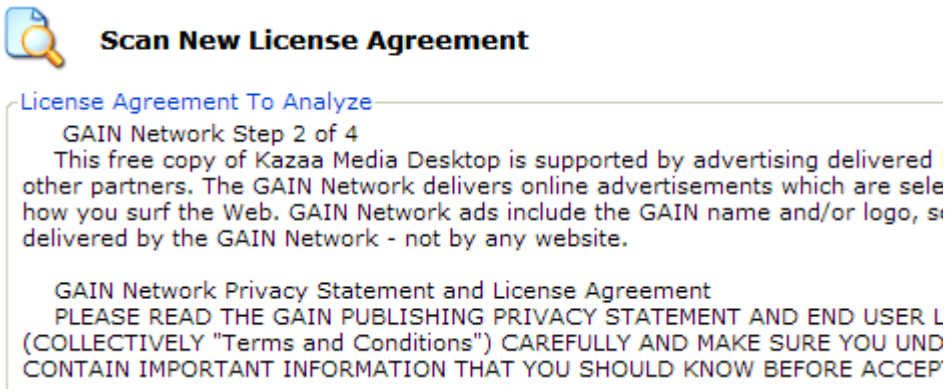
- proverite na stranici www.softpedia.com da li je taj proizvod obeležen kao "100% free, no viruses, no spyware, no adware" i

- preuzmite program EULAlyzer[87] i pomoću njega proverite da li EULA sadrži ključne reči koje ukazuju na postojanje špijunskih programa ili drugih parazita.

Na slikama 14.8 i 14.9 prikazani su isečci prozora programa EULAlyzer koji je iz kompletne GAIN licence koja prati Kazaa Media Desktop izvukao ključne reči (advertising, ads, pop-up, privacy, third party). Program upozorava da je licenca "preduga" i da sadrži jako veliki broj "interesantih" reči, i samim tim joj dodeljuje visok nivo rizika (Interest ID). Ukoliko pažljivo pročitate ovu licencu, videćete da su, ukoliko instalirate takav softver, proizvođači špijunske komponente zaštićeni zakonom (praktično, sami

87 EULAlyzer možete preuzeti sa Web stranice www.javacoolsoftware.com/eulalyzer.html. Besplatan je za ne-komercijalnu upotrebu i trenutno je raspoloživ u verziji 1.1.

pristajete na špijuniranje i "usmereno" reklamiranje i ne možete ih zbog toga kasnije tužiti). Ovakve licence često zabranjuju njuškanje paketa na relaciji klijent – reklamni server, kao i blokiranje komunikacije između klijenta i reklamnog servera.



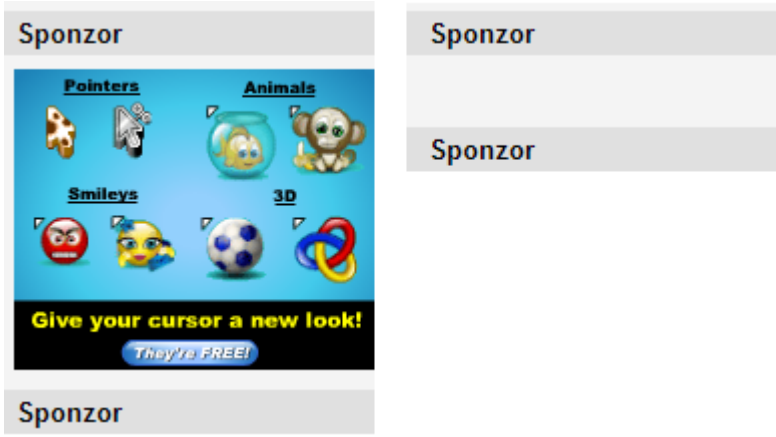
Slika 14.8 *– Kazaa Media Desktop EULA unešen u EULAlyzer*

Slika 14.9 *– Rezultati pretrage (očigledno da je EULA "interesantan")*

Ovim ćete donekle rešiti prvi problem (ukoliko niste isuviše radoznali). Ostaje još da rešite problem "brzog kliktanja" i instalacije ActiveX kontrola. Najjednostavnije rešenje – pokušajte da ne koristite Internet Explorer. Mozilla Firefox (www.mozilla.com) i Opera (www.opera.com) su besplatni browseri, brži i upotrebljiviji od Internet Explorera. Sama činjenica da Firefox ne podržava koncept instaliranja ActiveX kontrola znači da zaustavlja oko 90-95% špijunskih programa sa Interneta. Dodatno, Firefox možete nadograditi takozvanim Extension dodacima, a dva veoma korisna su:

- NoScript, koji dozvoljava da odaberete domene ili stranice čije ćete Java skriptove prihvatiti. Svi ostali Java skriptovi su podrazumevano zabranjeni. NoScript možete preuzeti sa adrese www.noscript.net.

- Adblock, koji filtrira Internet saobraćaj tako što imena stranica i domena poredi sa nizom regularnih izraza. Adblock umanjuje protok neželjenih podataka, što znači da će se koristan sadržaj brže preneti. Adblock je dobar način da ulonite reklame sa nekih stranica čime smanjujete i mogućnost slučajnog "klika" nakon kog sledi otvaranje neželjene stranice. Na slici 14.10 možete videti razliku – sa dela Web stranice (levo) Adblock je isfiltrirao reklame (desno). Adblock možete preuzeti sa adblock.mozdev.org a takođe vam preporučujemo da instalirate i Adblock Filterlist.G Updater koji automatski osvežava filtarske liste (može se preuzeti sa stranice www.pierceive.com).



Slika 14.10 *– Adblock filtrira reklame sa Web stranica*

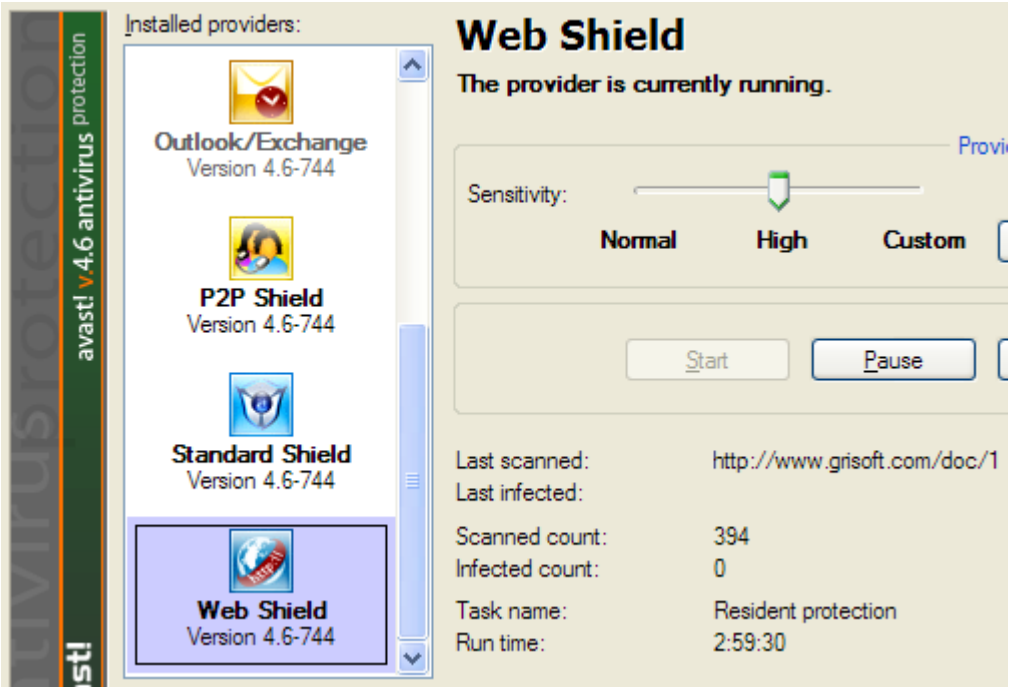
Ovim bi trebalo značajno da smanjite rizik zaraze neželjenim kodom. Sledeće što je potrebno da uradite jeste da instalirate antivirusni program. Ovakvih proizvoda imate mnogo – neki su besplatni, neki se daju na probu (na određeni period nakon koga morate kupiti licencu).

Dve osnovne komponente kvalitetnog antivirusnog programa su klasičan skener (proverava datoteke, MBR i registry bazu) i rezidentni skener koji obezbeđuje proveru u realnom vremenu (proverava izvršne datoteke koje pri pokretanju, elektronsku poštu, Web stranice i P2P mrežu). Zajedničko za oba skenera je da koriste bazu opisa zlonamernih programa koja se mora redovno obnavljati kako bi detekcija bika uspešna. Antivirusni program mora biti sposoban da aktiviranog trojanca ukloni iz liste procesa i očisti sa diska, pronađe virus u izvršnoj datoteci (bez obzira na ekstenziju datoteke), i proveri arhive. U principu, virus se mora pronaći pre aktiviranja jer ga je kasnije nekad nemoguce otkloniti, tj. zaražene datoteke vratiti u pređašnje stanje (ukoliko virus prepisuje zdravi kod).

Primer dobrog antivirusa koji je besplatan za nekomercijalnu upotrebu je Avast! koji proizvodi Alwil software. Potrebno je da sa stranice (www.avast.com) preuzmete paket Avast! Home Edition i zatražite serijski broj (važi godinu dana) koji će vam stići na mail. Nakon isteka ključa, jednostavno zatražite novi. Avast! obezbeđuje sledeću funkcionalnost:

- klasičan skener koji ispunjava gore navedene uslove, nudi mogućnost skeniranja celih diskova ili pojedinačnih direktorijuma i datoteka i prenosivih medijuma,
- skladište za zaražene datoteke,
- rezidenti skener koji proverava izvršne datoteke, poštu u Outlook i Outlook Express klijentima, podatke koji pristižu preko POP, SMTP, IMAP i NNTP protokola (osim ukoliko se koristi SSL, a klijent nije Outlook), IM servise (podržava veliki broj IM klijenata, kao što su ICQ, Trillian, Gaim, Yahoo i MSN Messenger) i P2P mrežu (većina klijenata za Bittorent, Gnutella i eDonkey2K mreže),
- besplatano inkrementalno osvežavanje antivirusnih definicija (update paketi su veličine 10-50KB, ponekad veći) i programa,

- generisanje baze za oporavak od infekcija virusom (VRDB, *Virus Recovery Database*) kada je računar u "mirovanju" ili kada se aktivira čuvar ekrana.



Slika 14.12 *– Deo ekrana za kontrolu modula rezidentnog skenera*

Ukoliko vam se Avast! ne sviđa, preostaju vam dve besplatne alternative: AVG (www.grisoft.com) ili Antivir Personal Edition (www.free-av.com) ili komercijalni proizvodi (Norton, McAffee, Kaspersky, Panda, NOD32 ili F-Prot Antivirus).

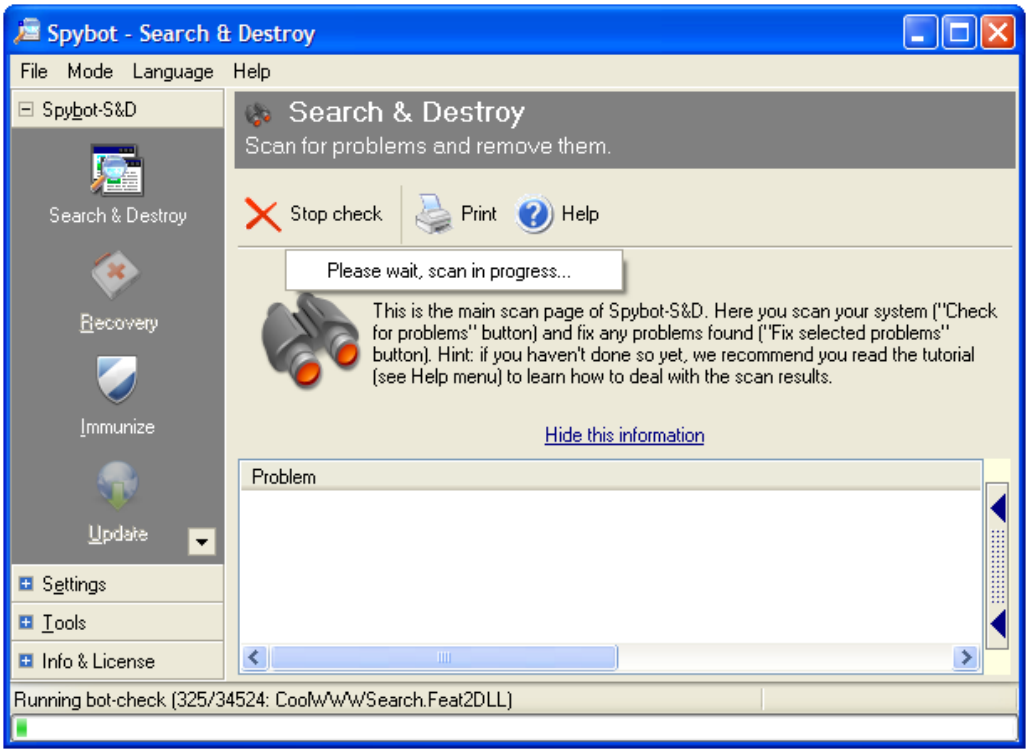
Preostalo je još da instalirate neki od programa za zaštitu od špijunskog softvera. Ovih programa takođe ima mnogo – besplatnih i komercijalnih. Bitno je samo da odaberete neki koji je okarakterisan kao pouzdan. Poželjno je da anti-spyware program mora da bude krajnje restriktivan po pitanju *adware*[88] i *spyware* komponenti, mora da obezbedi neke funkcije

88 Neki skeneri nisu dovoljno restriktivni po pitanju adware komponenti koje se u novije vreme isporučuju u okviru besplatnog bundleovanog softvera (kao što je, na primer, WhenU Save). Zaobiđite takve proizvode!

sistema za detekciju upada (IDS), kao što je zaštita hosts datoteke i registry ključeva koji određuju šta će od softvera biti pokrenuto prilikom podizanja operativnog sistema. Takođe, poželjno je da obezbedi mogućnost automatskog osvežavanja antispyware definicija. Relativno visok nivo zaštite od špijunskih programa postići ćete ukoliko instalirate:

- Spybot Search & Destroy (slika 14.13), koji obezbeđuje osnovnu funkcionalnost IDS-a, pronalaženje i uklanjanje već instaliranog špijunskog softvera. Može se preuzeti sa adrese www.safer-networking.org.

- Spyware blaster, koji sprečava pristup stranicama koje instaliraju špijunske programe. Besplatan je za ne-komercijalnu upotrebu. Može se besplatno preuzeti sa adrese www.javacoolsoftware.com.



Slika 14.13 *– Spybot Search & Destory*

Proizvođači oba pomenuta programa relativno često objavljuju nove definicije (dva do četiri puta mesečno). Ukoliko želite da budete sigurni, preporučujemo vam da instalirate još jedan antispyware skener sa rezidentnom zaštitom – na primer, SpyDefense (www.everestlabs.com, na slici 14.14 prikazan deo ekrana sa detektovanim spyware-om), Tenebril SpyCatcher Express (www.tenebril.com) ili MS Anti-Spyware. Ukoliko mislite da vam je dovoljan običan skener, instalirajte Lavasoft Adaware Personal (www.lavasoft.de)[89]. Takođe, na raspolaganju su vam i neki komercijalni proizvodi, kao što su Webroot Spysweeper, Aluria Spyware Eliminator i mnogi drugi.

Slika 14.4 *– SpyDefense (detektovan spyware)*

Neki programa ove vrste pripadaju klasi takozvanog "*rogue*" antispyware softvera (integrisan *adware*, programi prilikom skeniranja prijavljuju da na sistemu postoje špijunski programi kojih u stvari nema, tj. primenjuju sociološki inženjering kako bi Vas naterali da kupite licencu). Izbegavajte ovakav softver. Crnu listu antispyware programa možete naći na sledećoj stranici: http://www.spywarewarrior.com/rogue_anti-spyware.htm.

Postoji još nekoliko programa obezbeđuju zaštitu od potencijalnog "špijuniranja". Ove programe svrstavamo u posebnu klasu, koja sprečava neke funkcije Windows operativnog sistema, kao što su servisi koji obezbeđuju sinhronizaciju vremena sa time-serverom, automatsko skidanje

89 SpyDefense, SpyCatcher Express i Adaware Personal su besplatni za ne-komercijalnu upotrebu

zakrpa za operativni sistem, prijavljivanje grešaka Microsoftu, Messenger servis ili kompletan Windows Messenger, RPC locator i DCOM (koje su ranije iskorišćavali crvi), Windows XP calling home rutina, Remote Desktop i Remote Assistance, kao i DRM softver (Digital Rights Management) integrisan u Windows Media Player. Najpoznatiji programi koji pripadaju ovoj grupi su Xpy (xpy.whyeye.org), XP-antispy (www.xp-antispy.org), SafeXP (www.theorica.net/safexp.htm) i xpAntiSpy (www.xp-antispy.org).

14.7 Pitanja i zadaci

14.1 Instalirajte EULAlyzer i analizirajte EULA za, na primer, BearShare i neki neki GPL licenciran softver. Odredite razlike. Pokušajte "ručno" da analizirate EULA i otkrijete da li ima špijunskih programa.

INSTALIRAJTE VMWARE WORKSTATION NA VAŠEM RAČUNARU (www.vmware.com, besplatna probna verzija). KREIRAJTE WINDOWS VIRTUELNU MAŠINU, I OBEZBEDITE PRISTUP INTERNETU. ODRADITE SLEDEĆE ZADATKE NA VIRTUELNOJ MAŠINI !!!

14.2 Instalirajte Spyware blaster i preuzmite najnovije definicije. Pokušajte da otvorite www.xxxtoolbar.com u Internet Exploreru i instalirate xxx toolbar komponentu.

14.3 Instalirajte SpyDefense ili MS AntiSpyware i pokrenite rezidentan skener. Instalirajte bilo koji program koji instalira spyware (na primer, preuzmite neku Windows Desktop temu sa www.themexp.org – primetićete da su u pitanju *.exe datoteke). Da li je skener blokirao instalaciju spyware komponente ?

14.4 Zaustavite rezidentni skener i instalirajte što više špijunskih komponenti. Pokištajte da ih uklonite programom Spybot S&D ili Lavasoft Adaware. Jeste li uspeli? Šta je lakše - "sprečiti ili lečiti"?

14.5 Instalirajte Avast! antivirus i isprobajte funkcionalnosti skeniranja datoteka, Weba i elektronske pošte. Pokušajte da pronađete neku datoteku zaraženu virusom i da je pokrenete.

15

Programerske tehnike zaštite (problem prekoračenja bafera)

15.1 Šta je prekoračenje bafera?

Prepunjenje bafera je anomalija do koje dolazi kada proces u bafer upiše podatke koji su zbirno veći od veličine tog bafera; posledica toga je prepisivanje, tj. izmena vrednosti upisanih u susedne memorijske lokacije. Podaci koji su prepisani, tj. izmenjeni mogu biti drugi baferi, promenljive ili podaci koji obezbeđuju kontrolu toka programa. Do prepunjenje bafera ukoliko se prilikom upisa podataka u bafer nedovoljno ili uopšte ne vodi računa o veličini tog bafera, pa se podaci upišu na susedne memorijske lokacije. Do toga najčešće dolazi ukoliko se koriste funkcije za kopiranje jednog niza karaktera u drugi. Prepunjenje bafera može izazvati krah ili nepravilno izvršenje programa; najčešća posledica je ranjivost koda koju napadači mogu mogu iskoristi. Na primer, prosleđivanjem određenog niza kao ulaznog podataka može se prepuniti bafer, prepisati susedne memorijske lokacije i omogućiti izvršenje (najčešće zlonamernog) izvršnog koda podmetnutog u tom nizu.

Prva poznatija pojava prekoračenja bafera vezuje se za Roberta Morrisa čiji se crv 1988. godine širio po mreži, između ostalog iskorišćavajući prekoračenje u fingerd UNIX mrežnom servisu. (footnote: Morisov crv je osim ove slabosti iskorišćavao i propust u sendmail debug režimu i tranzitivnu relaciju poverenja kojom je omogućeno rexec/rsh prijavljivanje preko mreže bez navođenja lozinke). 1995. godine Thomas Lopatic je svoja zapažanja o prekoračenju bafera objavio na Bugtraq security mailing listi. Godine 1996, Elias Levy (pod pseudonimom Aleph One) objavio je u Phrack časopisu članak "Smashing the Stack for Fun and Profit" u kome je opisan postupak iskorišćavanja ranjivosti nastalih prekoračenjem bafera na steku. Ovaj članak je direktno ili indirektno zaslužan za veliki broj "iskorišćavača (exploit)" prepunjenog bafera. Godine 2001. crv poznat pod imenom Code Red slao je specijalne pakete ka serverima na kojima je pokrenut Microsoft Internet Information Services (IIS) 5.0, prouzrokujući prekoračenje bafera koje je crvu obezbeđivalo privilegije administratora (IIS5 je nespretno koristio administrativne privilegije neophodne za normalno pružanje servisa na TCP portu 80). Godine 2003. SQLSlammer crv kompromitovao je računare na kojima je pokrenut Microsoft SQL Server 2000 izazivajući prekoračenje koje omogućava izvršenje podmetnutog koda; ovaj crv je, prema proceni kompanije Symantec, zarazio najmanje 22.000 računara.

2004. godine Sasser je, iskorišćavajući prekoračenje u LSA sigurnosnom podsistemu (Local Security Authority Subsystem Service) zarazio veliki broj računara na Internetu.

Kako dolazi do prekoračenja?

Objasnićemo na prostom primeru kako dolazi do prekoračenja. Neka su u programu definisana dva podataka koja su smeštena u susedne memorijske lokacije: osmobitni bafer A (niz karaktera) i celobrojna vrednost B veličine dva bajta. U početnom stanju, A sadrži sve nule, a B broj 3.

A	A	A	A	A	A	A	A	B	B
0	0	0	0	0	0	0	0	0	3

Ako programer pokuša da u bafer A upiše niza karaktera "gang-bang" praćenu bajtom 0, koji označava kraj niza, doći će do prekoračenja bafera i prepisivanja vrednosti podatka B:

A	A	A	A	A	A	A	A	B	B
'g'	'a'	'n'	'g'	'–'	'b'	'a'	'n'	'g'	0

Do prekoračenja, očigledno dolazi zbog greške programera (osim ako programer nije imao nameru da u podatak B upiše karakter "g" praćen bajtom 0).

Osim izmene nasumičnih promenljivih u susednim lokacijama, prepunjenje bafera može dovesti do stanja čiju pojavu u normalnim okolnostima sam programski jezik neće dozvoliti. Do takvih stanja najčešće dolazi ukoliko je bafer smešten na steku. Stek je deo memorije u kome se podaci privremeno smeštaju (push)prilikom izvršenja neke funkcije. Kada funkcija započne svoje izvršenje, na vrhu steka se rezerviše jedan deo koji služi za smeštaj privremenih podataka koje će ta funkcija koristiti. U sledećem primeru, sa X su obeleženi podaci koji pripadaju glavnom programu, tj. podaci koji su se nalazili na steku kada je program započeo

izvršenje. Program poziva funkciju Y, koja zahteva jedan deo steka za interne potrebe, a funkcija Y dalje poziva funkciju Z, koja zahteva deo steka za smeštaj velikog bafera.

Z	Z	Z	Z	Z	Z	Y	X	X	X
						y	x	x	x

Ukoliko funkcija Z dovede do prepunjenje bafera, moguće je da će podaci koji pripadaju funkciji Y i/ili glavnom programu biti prepisani:

Z	Z	Z	Z	Z	Z	Y	X	X	X
z	z	z	z	z	z	z	z	x	x

Ovo može biti veoma opasno, jer se na na većini sistema na steku čuva i takozvana povratna adresa, odnosno lokacija u kojoj se nalazi onaj deo programa koji se izvršavao pre poziva funkcije. Kada se funkcija izvrši, deo steka dodeljen funkciji se oslobađa, a izvršenje programa se nastavlja sa povratne adrese. Međutim, ukoliko je povratna adresa prepisana usled prepunjenja bafera, program neće nastaviti izvršenje sa očekivane lokacije. Ukoliko je prepunjenje slučajno, povratna adresa ukazuje na slučajnu lokaciju i proces najčeše krahira.

Sledeći programski kod na jeziku C dovodi do prekoračenja bafera. Kao argument, u komandnoj liniji se navodi tekst koji će program iskopirati u bafer.

```
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[]) {
  char buffer[10];
  if(argc < 2)  {
    fprintf(stderr, "Upotreba: %s string\n", argv[0]);
    return 1;
  }
  strcpy(buffer, argv[1]);
  return 0;
}
```

Argumenti dužine do 9 karaktera neće izazvati prekoračenje. Niz dužine 10 ili više karaktera dovešće do prekoračenja (ovo ne implicira i grešku u segmentaciji). Očigledno da je u pitanju greška programera koji ne proverava dužinu bafera. Navodimo program koji pomoću funkcije strncpy obavlja identičnu aktivnost, ali ne dovodi do prekoračenja:

```
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[]) {
  char buffer[10];
  if(argc < 2) {
    fprintf(stderr, "Upotreba: %s string\n", argv[0]);
    return 1;
  }
  strncpy(buffer, argv[1], sizeof(buffer));
  buffer[sizeof(buffer) - 1] = '\0';
  return 0;
}
```

Tehnički obrazovan zlonamerni korisnik koji je upoznat sa strukturom programa može iskoristiti prekoračenje bafera i izmanipulisati program na nekoliko relativno jednostavnih načina:

- prepisivanjem vrednosti promenljive koja je smeštena blizu bafera (što je slučaj sa prvim primerom). Tako se može izmeniti, na primer, vrednost nekog numeričkog podatka ili nekog drugog bafera, što može dovesti do promene ponašanja programa u korist napadača.

- prepisivanjem povratne adrese na steku vrednošću koja pokazuje na adresu zlonamernog koda koji je podmenut. Sledeći primer ilustrovaće ovaj način iskorišćenja slabosti. Pretpostavite da napadač zna da će program prihvatiti bilo koji niz karaktera sa ulaza i proslediti ga funkciji X koja će ga dalje iskopirati u bafer dužine 100 bajta. Napadač podmeće niz karaktera dužine 104 bajta. Funkcija X će ovaj niz iskopirati u bafer i zahvaljujući četiri bajta viška, prepuniti ga i prepisati svoju povratnu adresu. Nova povratna adresa će ukazivati na podmetnuti izvršni kod, tj. na preostalih 100 bajta smeštenih na steku.

15.2 Prekoračenje na steku

Stek je kontinualan niz memorijskih lokacija u stek segmentu[90] koji funkcioniše po principu LIFO algoritma (Last In First Out – poslednji ušao prvi izašao). Promenjive koje se nalaze u pojedinim registrima mogu se na stek smestiti pomoću asemblerske instrukcije PUSH i preuzeti sa steka koristeći instrukciju POP. Prva sledeća dostupna memorijska lokacija na steku se naziva vrh steka. Pokazivač na vrh steka se nalazi se u ESP registru i sadrži pomeraj od početka steka. Stek raste u memoriji od viših adresa ka nižim. Kada se podatak "gurne" na stek, procesor umanjuje vrednost ESP registra, a zatim upisuje podatak na novi vrh steka. Kada je podatak "skida" sa steka, procesor čita podatak sa vrha steka, a zatim povećava vrednost ESP registra. Ukoliko program ili operativni sistem postavi više stekova (na primer, svaki zadatak koji se izvršava može dobiti svoj stek), trenutno je dostupan onaj na čiju osnovu ukazuje sadržaj SS (stack segment) registra.

Stek je često podeljen na delove, tj. okvire (frame) koji pripadaju različitm procedurama, a mogu sadržati lokalne promenljive, parametre koji će biti prosleđeni drugoj proceduri ili informacije o kontroli toka, tj. o povezivanju procedura. Procesor poseduje dva pokazivača koji obezbeđuju mehanizam za usaglašeno povezivanje procedura: pokazivač na osnovu steka (engl. *stack–base frame pointer*) i povratni pokazivač na instrukciju (engl. *return EIP*). Pre nego što se pozove prva instrukcije neke procedure, instrukcija CALL smešta adresu koja se nalazi u EIP registru[91] na trenutni stek. Ova adresa se naziva povratni pokazivač na sledeću instrukciju i pokazuje na instrukciju koja će se izvršiti po povratku iz procedure. Nakon vraćanja iz pozvane procedure, instrukcija RET "skida" povratni pokazivač instrukcije sa

90 Ukoliko se koristi segmentacija, program vidi memoriju kao grupu nezavisnih adresnih prostora koji se zovu segmenti. Memoriji se pristupa preko logičkih adresa, koja se sastoje od selektora segmenta i pomeraja unutar segmenta. Prevođenje logičke u fizičku adresu je transparentno. Kod, podaci i stek se sadrže u različitim segmentima – code, data i stack segment, na koje ukazuju selektoru CS, DS i SS.

91 EIP (Extended Instruction Pointer) sadrži relativnu adresu (pomeraj u trenutnom kod segmentu) sledeće instrukcije koja će se izvršiti. EIP registru se iz softvera ne može direktno pristupiti. Vrednost u EIP registru kontrolišu instrukcije koje služe za kontrolu toka (JMP, CALL i RET) i prekidi. Vrednost EIP registra može se pročitati indirektno, ukoliko se se izvrši instrukcija CALL, a zatim pročita vrednost vraćenog EIP-a sa steka. Vrednost EIP registra se može indirektno promeniti izmenom vrednosti vraćenog EIP na steku i izvšavanjem povratne instrukcije (RET ili IRET).

steka i vraća ga u EIP registar, a zatim se izvršenje procedure nastavlja. Procesor ne štiti vrednost na lokaciji povratnog pokazivača instrukcije niti zahteva da on ukazuje na proceduru. Pre nego što se izvrši RET instrukcija, povratni pokazivač instrukcije može se softverski modifikovati tako da pokazuje na bilo koju adresu u trenutnom ili nekom drugom kod segmentu. Ovo očigledno ostavlja dovoljno mesta za napad.

Šta najčešće izaziva prekoračenje bafera na steku ?

Programski jezik C programeru daje veliku slobodu na račun sigurnosti koda. Dovoljno je reći da u jeziku C ne postoji skoro nikakva automatska provere granica niza, a da je istovremeno omogućena aritmetika sa pokazivačima. Do prekoračenja bafera najčešće dolazi zbog lošeg rada sa nizovima karaktera, a sledeće funkcije ga najčešće izazivaju na steku:

- char *gets (char *buffer) – funkcija preuzima niz karaktera sa ulaza i smešta ga u promenjivu buffer,
- char *strcpy (char *strDestination, const char *strSource) - funkcija kopira sadržaj iz promenjive strSource u promenljivu strDestination,
- char *strcat (char *strDestination, const char *strSource) - funkcija dodaje jedan string na kraj drugog (koji se nalazi u baferu),
- int sprintf (char *buffer, const char *format [, argument] ...) - ponaša se slično funkciji printf, s tim što je izlaz bafer a ne ekran.

Primer na Windowsu

Virtuelna memorija svakog procesa je podeljena na adresni prostor jezgra i korisnički adresni prostor. Adresni prostor odvojen za korisnika i na Windows i na Linux operativnim sistemima sadrži stek segment, *heap*, kod programa i neke druge segmente.

Organizacija memorije na Windows sistemima je malo složenija nego na Linux sistemima. Na primer, procesi mogu imati više *heap* adresnih prostora, a svaki DLL poseduje sopstveni *heap* i stek. Najvažnija razlika je

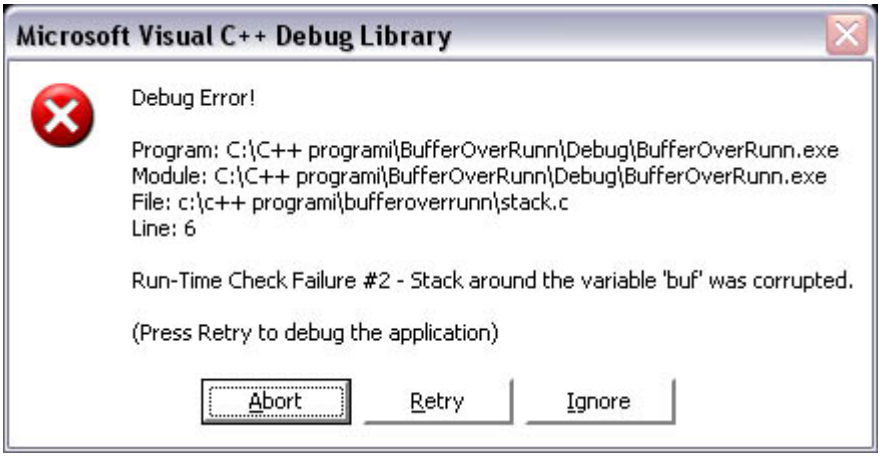
ta što pozicija steka na Windowsu nije tačno određena, što obično zadaje poteškoće u iskorišćavanju prekoračenja steka (čitaj: u pisanju exploita) u odnosu na Linux.

Razmotrimo sledeći program i njegov izlaz:

```
#include <string.h>

void func(char *str){
  char buf[3];
  strcpy(buf, str);
}

void main(int argc,char *argv[]){
  func(argv[1]);
}
```

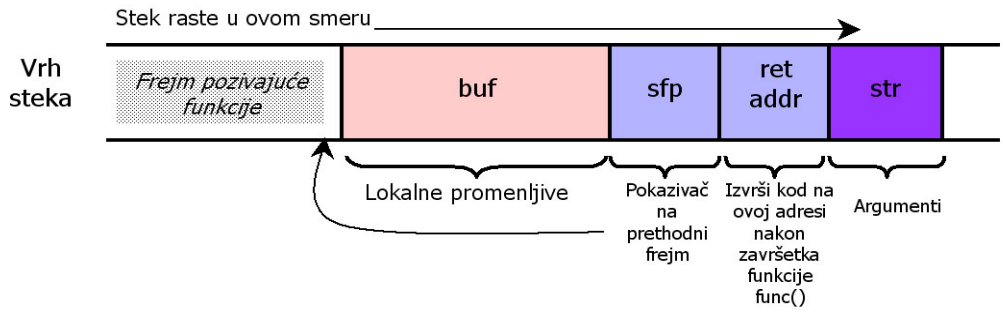
Ukoliko program pokrenemo iz komandne linije i kao prvi argument navedemo niz karaktera duži od tri, dobićemo sledeći dijalog sa koga se vidi da je stek prepunjen:

```
C:\C++Programi\BufferOverRunn\Debug\BufferOverRunn.exe aaaa
```



Slika 15.1 *– Prekoračenje bafera na steku (Windows)*

Na sledećoj slici jasno se vidi da je prekoračenje bafera izazvalo uništavanje informacija na steku od kojih zavisi dalji tok programa:



Slika 15.2 *– Šta je izazvalo krah programa*

Primer na Linuxu

Sledeći primer će pokazati da ni Linux nije imun na prekoračenje bafera.

U sledećem prepunjujemo bafer na steku i time menjamo vrednost EIP registra nizom karaktera AAAA (Ox41414141), čime praktično preuzimamo kontrolu nad daljim tokom programa. Ovakva vrednost EIP registra dovodi do kraha programa.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int bof(){
  char bafer[8];
  strcpy(bafer,"AAAAAAAAAAAAAAAAAAAA");
return 1;
}

int main(int argn, char **args){
  bof();
  printf("Nema prikaza!\n");
  /* poruka se nikada neće prikazati jer program
     nece dostici taj deo zbog pojave prekoracenja */
  return 1;
}
```

Analiziraćemo tok programa pomoću GDB debuggera:

```
.text:0804835C public bof
.text:0804835C bof proc near ; CODE XREF:main+10p
.text:0804835C
.text:0804835C bafer = dword ptr -8
;bof pocetak
.text:0804835C push ebp
.text:0804835D mov ebp, esp
;napravi prostor na steku za lokalne promenjive
.text:0804835F sub esp, 8
.text:08048362 sub esp, 8
;gurni drugi parametar za strcpy (20 karaktera A) na stek
.text:08048365 push offset "AAAAAAAAAAAAAAAAAAAA"
;gurni prvi parametar za strcpy na stek (adresa prom. bafer)
.text:0804836A lea eax, [ebp+bafer]
.text:0804836D push eax
;pozovi strcpy
.text:0804836E call _strcpy
;posle poziva ocisti stek
.text:08048373 add esp, 10h
;postavi povratnu vrednost u registru EAX
.text:08048376 mov eax, 1
;bof kraj (= move esp, ebp/pop ebp)
.text:0804837B leave
;vrati kontrolu glavnom programu
.text:0804837C retn
.text:0804837C bof endp
.text:0804837D public main
.text:0804837D main proc near
;main pocetak
.text:0804837D push ebp
.text:0804837E mov ebp, esp
;poravnaj stek, sto nije uvek slucaj
.text:08048380 sub esp, 8
.text:08048383 and esp, 0FFFFFFF0h
.text:08048386 mov eax, 0
.text:0804838B sub esp, eax
;pozovi "ranjivu" funkciju bof()
.text:0804838D call bof
.text:08048392 sub esp, 0Ch
;"gurni" argument za funkciju printf()
.text:08048395 push offset "Nema prikaza!"
;pozovi funkciju printf()
.text:0804839A call _printf
;ocisti stek posle poziva
.text:0804839F add esp, 10h
```

Iako se na ovaj način registar EIP ne može direktno kontrolisati, ovime su prikazani koncepti koji omogućavaju da se preuzme potpuna kontrola nad daljim izvršenjem programa. U ovom primeru prekoračenje bafera će dovesti do prekidanja toka programa, pošto će instrukcija povratka iz funkcije (RET) sa steka skinuti vrednost 0x41414141 (tj. AAAA), koja predstavlja ili pokazuje nevažeći deo u memoriji. Program će se završiti greškom u segmentaciji.

15.3 Prekoračenje na *heap*-u

Heap je oblast u memoriji vezana koja se dinamički dodeljuje procesu ukoliko potrebna količina memorije nije unapred poznata ili je veća od steka. Heap se sastoji od memorijskih blokova od kojih svaki ima zaglavlje koje, između ostalog, opisuje i veličinu tog memorijskog bloka. Prekoračenje bafera na heapu može dovesti do promene sadržaja sledećeg bloka. Ako se zaglavlje sledećeg bloka promeni, praktično se omogućava upis proizvoljnih podataka u memoriju.

Pojava prekoračenja bafera je česta u heap delu memorije. Za razliku od steka, heap memorija dodeljena jednoj funkciji ostaje dodeljena toj funkciji sve dok se eksplicitno ne oslobodi. To znači da će efekat prekoračenja biti primećen tek kada se deo heap memorije na kome je došlo do prekoračenja ponovo iskoristi.

Heap, za razliku od steka, ne sadrži povratne adrese funkcija, pa je preusmeravanje izvršenja programa teže ostvariti. Međutim, druge važne stvari smeštene na “heap”-u mogu da budu uništene i iskorišćene prekoračenjem dinamičkih bafera.

Dinamička dodela memorije

Dinamičku dodelu memorije, za razliku od dodele memorije za smeštaj statičkih promenljivih, kontroliše programer. Funkcije za rad sa heap delom memorije, tj. za dodelu, promenu veličine i oslobađanje memorije, na C jeziku su sledeće:

- void * malloc (size_t vel) – funkcija za dodelu memorije; vraća pokazivač na novi alocirani blok veličine vel bajtova ili NULL pokazivač ako se blok memorije ne može alocirati. Sadržaj memorije nije inicijalizovan, tj. ne popunjava se nulama niti bilo kakvim drugim vrednostima, već se to ostavlja programeru da sam uradi (alternativno, može se upotrebiti calloc)

- void * calloc (size_t n, size_t vel) – funkcija dodeljuje memoriju za niz od n elemenata veličine vel bajtova. Vrednost funkcije je generički pokazivač (tip void *) na dodeljeni prostor ili NULL ako zahtev ne može da bude zadovoljen. Dodeljeni prostor se popunjava nulama.

- void * realloc (void *ptr, size_t novavel) – funkcija menja veličinu bloka dinamički dodeljene memorije na koju pokazuje pokazivač *ptr u novavel bajtova. Nova veličina može da bude veća ili manja od stare. U slučaju smanjenja veličine dodeljene memorije, skraćivanje se vrši sa kraja, a sadržaj zadržanih bajtova se čuva. U slučaju povećavanja dodeljene memorije, novi bajtovi nedefinisanog sadržaja dodaju se na kraj. Vrednost funkcije je generički pokazivač (tip void *) na novo mesto dodeljene memorije ili NULL ako zahtev ne može da bude zadovoljen. Funkcija je korisna za rad sa strukturama podataka čija veličina nije unapred poznata, tj. čija se veličina menja tokom izvršenja programa.

- void free (void *ptr) – funkcija oslobađa blok memorije na koji pokazuje pokazivač *ptr. Pokazivač *ptr mora da sadrži vrednost koja je ranije dobijena kao vrednost neke od funkcija za dodelu memorije. Važno je da se prostor dodeljen podacima koji više nisu potrebni oslobodi, kako bi kasnije mogao da bude dodeljen drugim podacima u toku izvršenja programa.

Za rad sa dinamičkom memorijom na programskom jeziku C++ koriste se funkcije new() i delete() sa više ili manje istim efektima kao i ANSI C funkcije malloc() i free(). U Micorsoft Windows operativnom sistemu postoje ugrađene funkcije HeapAlloc() i HeapFree().

Primer prekoračenja

Sledeći primer će prikazati kako se koristi heap, a ujedno i prekoračenje bafera u heap delu memorije koje se može eksploatisati.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void main(int argc, char *argv[]){
  char *input = malloc(20*sizeof(char));
  char *output = malloc(20*sizeof(char));
  strcpy(output,"normalan izlaz");
  strcpy(input,argv[1]);
  printf("ulaz na adresi %p: %s\n",input,input);
  printf("izlaz na adresi %p: %s\n",output,output);
  printf("\n\n%s\n",output);
}
```

Posmatrajte šta se dešavanja prilikom izvršenja programa na Linux operativnom sistemu kada se ulazni argument koji je veći od 20 karaktera kopira u niz karaktera input dužine 20 karaktera. Pazite, ovako nešto je moguće, jer ne postoji provera veličine niza.

```
# ./heap-exploit 1234567890
ulaz na adresi 0x8049728: 1234567890
izlaz na adresi 0x8049740: normalan izlaz

normalan izlaz

# ./heap-exploit "12345678901234567890123456789012345678901234razbili smo heap!"
ulaz na adresi 0x8049728: 12345678901234567890123456789012345678901234Razbili smo heap!
izlaz na adresi 0x8049740: Razbili smo heap!

Razbili smo heap!
```

Ovo se, uz izmenu ulaznih parametara može izvesti i na Windows operativnom sistemu. Kao što se vidi, promena podataka na heap delu memorije je veoma lako ostvarljiva i ne izaziva uvek prekid programa, što je pogodno za eksploataciju.

Sličan slučaj iskorišćenja je moguć ako memorijskim prostor za promenljive statički dodeljen:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

static char input[20];
static char output[20];

int main(int argc, char *argv[]){
  strcpy (output, "normalan izlaz");
  strcpy (input, argv[1]);
  printf ("ulaz na adresi %p: %s\n", input, input);
  printf ("izlaz na adresi %p: %s\n", output, output);
  printf("\n\n%s\n", output);
}
```

Izvršavamo program sa različitim ulaznim parametrima:

```
# ./static-exploit 123456789012345678
ulaz na adresi 0x80496b8: 123456789012345678
izlaz na adresi 0x80496cc: normalan izlaz

normalan izlaz

# ./static-exploit "12345678901234567890Eto mene opet!"
ulaz na adresi 0x80496b8: hacks1hacks2hacks3haEto mene opet!
izlaz na adresi 0x80496cc: Eto mene opet!

Eto mene opet!
```

15.4 Sprečavanje prekoračenja

Najbolji način za sprečavanje prekoračenja bafera je pisanje pouzdanog koda u kome se proverava veličina svih ulaznih podataka prilikom poziva određenih funkcija.

Na primer, funkcija strcpy() ne obezbeđuje nikakvu sigurnost u kodu koji se piše, pa je zbog toga treba ređe koristiti. Deklaracija ove funkcije je

sledeća:

- char *strcpy(char *strDestination, const char *strSource);

Ako i jedan i drugi string, strDestination ili strSource, imaju vrednost NULL, program će se završiti u "exception handleru". Ako izvorišni niz nije završen NULL karakterom, rezultat je nedefinisan. Najveći problem nastaje ukoliko je veličina izvorišnog bafera strSource veća od veličine odredišnog bafera strDestination. Ova funkcija je bezbedna samo kada se koristi u trivijalnim slučajevima, kao što je kopiranje fiksne veličine stringa u bafer.

Sledeći kod prikazuje kako se ova funkcija može bezbedno koristiti:

```
bool HandleInput(const char* input) {
  char buf[80];
  if(input == NULL) {
    assert(false);
    return false;
  }
  if(strlen(input) < sizeof(buf)) strcpy(buf, input);
  else return false;
  return true;
}
```

Funkcija gets() funkcija je još jedna od funkcija koje nisu bezbedne za korišćenje. Njena deklaracija je sledeća:

- char *gets(char *buffer);

Ova funkcija će čitati preko glavnog ulaza podatke sve dok ne naiđe na "Line Feed" (0x10 – '\n') ili "Carriage Return" (0x13) karakter. Ne postoji način da se sazna kada će se prekoračenje bafera pojaviti. U tom slučaju ne treba koristiti ovu funkciju, nego umesto nje se može koristiti funkcija fgets() ili C++ objektne funkcije za rad sa nizovima karaktera.

Prekoračenje bafera se donekle može sprečiti i na druge načine. Na primer, postoje razni softverski alati za detekciju napada na sistem koji su sposobni da u nekim slučajevima otkrivaju pojavu prekoračenja i eksploatacije bafera. Novije 64-bitni procesorsi imaju mogućnost zaštite od prekoračenja bafera, ali se rutine za korišćenje takvih mehanizama moraju implementirati i u sam operativni sistem.

15.5 Pitanja i zadaci

15.1 Sastavite program koji ilustruje pojavu prekoračenja bafera na steku koristeći funkciju strcat za spajanje dva niza karaktera u novi niz koji se čuva u memoriji.

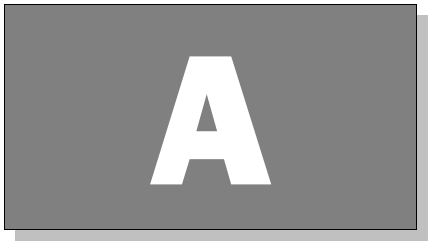
Objasnite kako se ovo prekoračenje može iskoristiti?

Objasnite šta treba učiniti i izmenite kod tako da do ovog prekoračenja ne dođe.

15.2 Sastavite program koji ilustruje prekoračenje bafera na heap delu memorije, koristeći funkcije malloc() ili new za dodelu memorije i funkcije strcpy za kopiranje niza karaktera u bafer.

Objasnite kako se ovo prekoračenje može iskoristiti?

Objasnite šta treba učiniti i izmenite kod tako da do ovog prekoračenja ne dođe.



Značajnije kriptografske tablice

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
...																									
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Tablica A.1 *– Vigenereov kvadrat*

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Tablica A.2 *– Inicijalna permutacija IP*

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Tablica A.3 *– Završna permutacija* IP^{-1}

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Tablica A.4 *– Funkcija proširenja EX*

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Tablica A.5 *– Supstitucijska kutija S_1*

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Tablica A.6 *– Supstitucijska kutija S_2*

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Tablica A.7 *– Supstitucijska kutija S_3*

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Tablica A.8 *– Supstitucijska kutija S_4*

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Tablica A.9 *– Supstitucijska kutija S_5*

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Tablica A.10 *– Supstitucijska kutija S_6*

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Tablica A.11 *– Supstitucijska kutija S_7*

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Tablica A.12 – *Supstitucijska kutija S_8*

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Tablica A.13 – *Permutacija P*

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Tablica A.14 – *Permutacija PK_1*

4	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Tablica A.15 *– Permutacija PK_2*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Tablica A.16 *– RIJNDAEL supstitucijska kutija (funkcija ByteSub)*

B

Izvorni kod

B.1 Cezarova šifra (Java)

```
import java.io.*;
public class Cezar {
   public static void main(String[] args) throws IOException {
        int kljuc=22; // vrednost ključa između 0 i 25
        BufferedReader ulaz = new
             BufferedReader(new InputStreamReader(System.in));
   System.out.println("Otvoreni tekst: ");
   String otvtekst=ulaz.readLine();
   System.out.println();
        for(int i=0; i<otvtekst.length(); i++) {
        int numekviv=otvtekst.charAt(i);
        if(numekviv>64 && numekviv<91) {
             numekviv=numekviv+kljuc;
             if(numekviv>90) numekviv=numekviv-26;
        }
        else if(numekviv>96 && numekviv<123) {
             numekviv=numekviv+kljuc;
                  if(numekviv>122) numekviv=numekviv-26;
        }
             System.out.print((char)(numekviv));
        }
   System.out.println();
}}
```

B.2 Jednostavan XOR algoritam (C++)

```
void main (int argc, char *argv[]) {
  FILE *otvtekst, *sifrat; char *kljuc;
  int tekuci; // tekući karakteri otv.teksta i šifrata
  if ((kljuc = argv[1]) && *kljuc!='\0') {
    if ((otvtekst = fopen(argv[2], "rb")) != NULL) {
      if ((sifrat = fopen(argv[3], "wb")) != NULL) {
        while ((tekuci = getc(otvtekst)) != EOF) {
          if (!*kljuc) kljuc = argv[1];
          tekuci ^= *(kljuc++);
          putc(tekuci,sifrat);
        }
        fclose(sifrat);
      }
      fclose(otvtekst);
}}}
```

B.3 AES

```
import java.io.*; // neophodno uvoženje potrebnih klasa

// AESencrypt: AES šifrovanje
class AESencrypt {
  private final int Nb = 4;
  private int Nk;
  private int Nr;
  private int wCount;
  private AESTables tab;
  private byte[] w;

// AESencrypt: konstruktor klase. Uglavnom proširuje ključ
  public AESencrypt(byte[] key, int NkIn) {
    Nk = NkIn; // reci u kljucu, = 4, or 6, or 8
    Nr = Nk + 6; // odgovarajući broj rundi
    tab = new AESTables();
    // klasa koja daje vrednost različitih funkcija
    w = new byte[4*Nb*(Nr+1)]; // niz za expanded key
    KeyExpansion(key, w); // dužina za w zavisi od Nr
  }

// Cipher: pozivi funkcija za šifrovanje
  public void Cipher(byte[] in, byte[] out) {
    wCount = 0; // broji bajtove u expanded key
    byte[][] state = new byte[4][Nb]; // state matrica
    Copy.copy(state, in); // kopiranje iz in u state
    AddRoundKey(state); // xor sa expanded key
    for (int round = 1; round < Nr; round++) {
      Print.printArray("Runda " + round + ":", state);
      SubBytes(state);
      ShiftRows(state);
      MixColumns(state);
      AddRoundKey(state);
      }
    Print.printArray("Runda " + Nr + ":", state);
    SubBytes(state);
    ShiftRows(state);
    AddRoundKey(state);
    Copy.copy(out, state);
  }


// KeyExpansion: pravi niz od konkatenacije svih ključeva
// radi sa bajtovima mada traži reči (4 bajta)
```

```
  private void KeyExpansion(byte[] key, byte[] w) {
    byte[] temp = new byte[4];
    // prvo se kopira ključ u w
    int j = 0;
    while (j < 4*Nk) w[j] = key[j++];
    // ovde je j == 4*Nk;
    int i;
    while(j < 4*Nb*(Nr+1)) {
      i = j/4;
      for (int iTemp = 0; iTemp < 4; iTemp++)
        temp[iTemp] = w[j-4+iTemp];
      if (i % Nk == 0) {
        byte ttemp, tRcon;
        byte oldtemp0 = temp[0];
        for (int iTemp = 0; iTemp < 4; iTemp++) {
          if (iTemp == 3) ttemp = oldtemp0;
          else ttemp = temp[iTemp+1];
          if (iTemp == 0) tRcon = tab.Rcon(i/Nk);
          else tRcon = 0;
          temp[iTemp] = (byte)(tab.SBox(ttemp) ^ tRcon);
        }
      }
      else if (Nk > 6 && (i%Nk) == 4) {
        for (int iTemp = 0; iTemp < 4; iTemp++)
          temp[iTemp] = tab.SBox(temp[iTemp]);
      }
      for (int iTemp = 0; iTemp < 4; iTemp++)
      w[j+iTemp] = (byte)(w[j - 4*Nk + iTemp] ^ temp[iTemp]);
      j = j + 4;
    }
  }

// SybBytes: primenjuje Sbox zamenu na svaki bajt u state
  private void SubBytes(byte[][] state) {
    for (int row = 0; row < 4; row++)
      for (int col = 0; col < Nb; col++)
        state[row][col] = tab.SBox(state[row][col]);
  }

// ShiftRows: prosto pomeranje vrsta 1, 2, 3 za 1, 2, 3
  private void ShiftRows(byte[][] state) {
    byte[] t = new byte[4];
    for (int r = 1; r < 4; r++) {
      for (int c = 0; c < Nb; c++) t[c] = state[r][(c+r)%Nb];
      for (int c = 0; c < Nb; c++) state[r][c] = t[c];
    }
  }
```

```
// MixColumns: komplikovanije operacije nad svakom kolonom
  private void MixColumns(byte[][] s) {
    int[] sp = new int[4];
    byte b02 = (byte)0x02, b03 = (byte)0x03;
    for (int c = 0; c < 4; c++) {
      sp[0] = tab.FFMul(b02, s[0][c]) ^
              tab.FFMul(b03, s[1][c]) ^ s[2][c] ^ s[3][c];
      sp[1] = s[0][c] ^ tab.FFMul(b02, s[1][c]) ^
              tab.FFMul(b03, s[2][c]) ^ s[3][c];
      sp[2] = s[0][c] ^ s[1][c] ^ tab.FFMul(b02, s[2][c]) ^
              tab.FFMul(b03, s[3][c]);
      sp[3] = tab.FFMul(b03, s[0][c]) ^ s[1][c] ^
              s[2][c] ^ tab.FFMul(b02, s[3][c]);
      for (int i = 0; i < 4; i++) s[i][c] = (byte)(sp[i]);
    }
  }

// AddRoundKey: xor delova expanded key sa state
  private void AddRoundKey(byte[][] state) {
    for (int c = 0; c < Nb; c++)
      for (int r = 0; r < 4; r++)
      state[r][c] = (byte)(state[r][c] ^ w[wCount++]);
    }
  }

// AEStables: konstrukcija različitih tablica
public class AEStables {
  public AEStables() {
    loadE(); loadL(); loadInv();
    loadS(); loadInvS(); loadPowX();
  }
  private byte[] E = new byte[256];
  private byte[] L = new byte[256];
  private byte[] S = new byte[256];
  private byte[] invS = new byte[256];
  private byte[] inv = new byte[256];
  private byte[] powX = new byte[15];

// Funkcije za dohvatanje sadržaja tablica
  public byte SBox(byte b) { return S[b & 0xff];}
  public byte invSBox(byte b) { return invS[b & 0xff]; }
  public byte Rcon(int i) { return powX[i-1]; }

// FFMulFast: brzo množenje polinoma korišćenjem tablica
  public byte FFMulFast(byte a, byte b){
    int t = 0;;
    if (a == 0 || b == 0) return 0;
```

```
    t = (L[(a & 0xff)] & 0xff) + (L[(b & 0xff)] & 0xff);
    if (t > 255) t = t - 255;
    return E[(t & 0xff)];
  }

// FFMul: sporo množenje, korišćenjem pomeraja bitova
  public byte FFMul(byte a, byte b) {
    byte aa = a, bb = b, r = 0, t;
    while (aa != 0) {
      if ((aa & 1) != 0) r = (byte)(r ^ bb);
      t = (byte)(bb & 0x80);
      bb = (byte)(bb << 1);
      if (t != 0) bb = (byte)(bb ^ 0x1b);
      aa = (byte)((aa & 0xff) >> 1);
    }
    return r;
  }

// loadE: kreiranje i punjenje E tablice (tablice stepena)
  private void loadE() {
    byte x = (byte)0x01;
    int index = 0;
    E[index++] = (byte)0x01;
    for (int i = 0; i < 255; i++) {
      byte y = FFMul(x, (byte)0x03);
      E[index++] = y;
      x = y;
    }
  }

// loadL: punjenje L tablice korišćenjem E tablice
  private void loadL() {
    // pazljivo: tablica L ima 254 vrednosti (nema za 00)
    int index;
    for (int i = 0; i < 255; i++) L[E[i] & 0xff] = (byte)i;
  }

// loadS: punjenje tablice S
  private void loadS() {
    int index;
    for (int i = 0; i < 256; i++)
      S[i] = (byte)(subBytes((byte)(i & 0xff)) & 0xff);
  }

// loadInv: punjenje tablice inv
  private void loadInv() {
```

```
    int index;
    for (int i = 0; i < 256; i++)
      inv[i] = (byte)(FFInv((byte)(i & 0xff)) & 0xff);
  }

// loadInvS: punjenje invS tablice korišćenjem S tablice
  private void loadInvS() {
    int index;
    for (int i = 0; i < 256; i++)
      invS[S[i] & 0xff] = (byte)i;
  }

// loadPowX: punjenje powX tablice množenjem polinoma
  private void loadPowX() {
    int index;
    byte x = (byte)0x02;
    byte xp = x;
    powX[0] = 1; powX[1] = x;
    for (int i = 2; i < 15; i++) xp = FFMul(xp, x);
    powX[i] = xp;
  }

// FFInv: multiplikativni inverz za bajt (novi bajt)
  public byte FFInv(byte b) {
    byte e = L[b & 0xff];
    return E[0xff - (e & 0xff)];
  }

// ithBit: vraćanje i-tog bita iz bajta
   public int ithBit(byte b, int i) {
    int m[] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
    return (b & m[i]) >> i;
  }

// subBytes: subBytes funkcija
  public int subBytes(byte b) {
    byte inB = b;
    int res = 0;
    if (b != 0) b = (byte)(FFInv(b) & 0xff);
    byte c = (byte)0x63;
    for (int i = 0; i < 8; i++) {
      int temp = 0;
      temp = ithBit(b, i) ^ ithBit(b, (i+4)%8) ^
             ithBit(b,(i+5)%8) ^ ithBit(b, (i+6)%8) ^
             ithBit(b, (i+7)%8) ^ ithBit(c, i);
      res = res | (temp << i);
```

```
    }
    return res;
  }
}

// GetBytes: stvaranje niza bajtova od heksadecimalnih cifara
class GetBytes {
  private String fileName; // ulazno ime fajla
  private int arraySize; // broj bajtova za citanje
  private Reader in;

// GetBytes: konstruktor, otvara ulazni fajl
  public GetBytes(String file, int n) {
    fileName = file;
    arraySize = n;
    try {
      in = new FileReader(fileName);
    } catch (IOException e) {
      System.out.println("Exception opening " + fileName);
    }
  }

// getNextChar: uzima sledeći karakter
  private char getNextChar() {
    char ch = ' ';
    try {
      ch = (char)in.read();
    } catch (IOException e) {
      System.out.println("Greška pri čitanju karaktera");
    }
    return ch;
  }

// val: vraća int vrednost heksadecimalne cifre
  private int val(char ch) {
    if (ch >= '0' && ch <= '9') return ch - '0';
    if (ch >= 'a' && ch <= 'f') return ch - 'a' + 10;
    if (ch >= 'A' && ch <= 'F') return ch - 'A' + 10;
    return -1000000;
  }

// getBytes: stvara niz bajtova od heksadecimalnih cifara
  public byte[] getBytes() {
    byte[] ret = new byte[arraySize];
    for (int i = 0; i < arraySize; i++) {
      char ch1 = getNextChar();
```

```
      char ch2 = getNextChar();
      ret[i] = (byte)(val(ch1)*16 + val(ch2));
    }
    return ret;
  }
}

// Copy: kopiranje niza bajtova
class Copy {
  private static final int Nb = 4;

// copy: kopiranje in u state
  public static void copy(byte[][] state, byte[] in) {
    int inLoc = 0;
    for (int c = 0; c < Nb; c++)
      for (int r = 0; r < 4; r++)
        state[r][c] = in[inLoc++];
  }

// copy: kopiranje state u out
  public static void copy(byte[] out, byte[][] state) {
    int outLoc = 0;
    for (int c = 0; c < Nb; c++)
      for (int r = 0; r < 4; r++)
        out[outLoc++] = state[r][c];
  }
}

// Print: ispis niza bajtova
class Print {
  private static final int Nb = 4;
  private static String[] dig = {"0","1","2","3","4","5","6",
                        "7","8","9","a","b","c","d","e","f"};

// hex: ispis bajta u obliku dve heksadecimalne cifre
  public static String hex(byte a) {
    return dig[(a & 0xff) >> 4] + dig[a & 0x0f];
  }

  public static void printArray(String name, byte[] a) {
    System.out.print(name + " ");
    for (int i = 0; i < a.length; i++)
      System.out.print(hex(a[i]) + " ");
    System.out.println();
  }
```

```
  public static void printArray(String name, byte[][] s) {
    System.out.print(name + " ");
    for (int c = 0; c < Nb; c++)
      for (int r = 0; r < 4; r++)
        System.out.print(hex(s[r][c]) + " ");
    System.out.println();
  }
}

// AEStestiranje: test za AES šifrovanje

public class AEStestiranje {
  public static void main(String[] args) {
  System.out.println("Šifrovanje...");
  GetBytes getInput = new GetBytes("plaintext1.txt", 16);
  byte[] in = getInput.getBytes();
  GetBytes getKey = new GetBytes("key1.txt", 16);
  byte[] key = getKey.getBytes();
  AESencrypt aes = new AESencrypt(key, 4);
  Print.printArray("Plaintext: ", in);
  Print.printArray("Key: ", key);
  byte[] out = new byte[16];
  aes.Cipher(in, out);
  try {
    FileWriter upis = new FileWriter("ciphertext1.txt");
    for(int i=0;i<out.length;i++)
      upis.write(Print.hex(out[i]),0,2);
    upis.close();
  }
  catch(Exception e){
    System.out.println("Neuspešan upis šifrata u datoteku!");
  }
  Print.printArray("Ciphertext: ", out);
  System.out.println("\nDešifrovanje...");
  getInput = new GetBytes("ciphertext1.txt", 16);
  in = getInput.getBytes();
  AESdecrypt aesDec = new AESdecrypt(key, 4);
  Print.printArray("Ciphertext: ", in);
  Print.printArray("Key: ", key);
  aesDec.InvCipher(in, out);
  Print.printArray("Plaintext: ", out);
}
```

B.4 SHA1 (Java)

```
import java.util.*;
public final class SHA1 {
  private int state[];
  private long count ;
  public byte[] digestBits ;

// Preklopljena metoda update za čitav niz;
// poziva update za svaki bajt
  public void update (byte input[]) {
    for (int i = 0; i < input.length; i++) update (input[i]);
  }

//Uzima karakter po karakter iz osnovne poruke,
//Tretira ih kao bajtove i za svaki poziva metodu update
  public void updateASCII (String input) {
    for (int i = 0; i < input.length(); i++) {
      byte x = (byte) (input.charAt(i));
      update (x);
    }
  }

//Niz block služi za skladištenje osnove poruke
  private int block [] = new int[16];
  private int blockIndex ;

   // Kružno pomeranje za value bitova u levo
  final int rol (int value, int bits) {
    int q = (value << bits) | (value >>> (32 - bits));
    return q;
  }

// Ispravno formiranje sadržaja prvih 16 blokova
  final int blk0 (int i) {
    block [i] = (rol (block [i],24)&0xFF00FF00) |
                (rol (block [i],8)&0x00FF00FF);
    return block [i];
  }

// Dalje punjenje blokova
  final int blk (int i) {
    block [i&15] = rol (block [(i+13)&15]^block [(i+8)&15]^
                   block [(i+2)&15]^block [i&15], 1);
    return (block [i&15]);
```

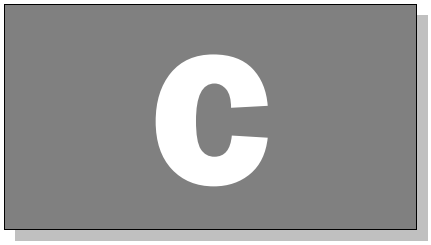
```
  }

// Implementacija samog algoritma
// Niz data predstavlja promenjive  A, B, C, D, E

  final void R0 (int data[], int v, int w,
                 int x , int y, int z, int i) {
    data[z] += ((data[w] & (data[x] ^ data[y])) ^ data[y]) +
               blk0 (i) + 0x5A827999 + rol (data[v] ,5);
    data[w] = rol (data[w], 30);
  }

  final void R1 (int data[], int v, int w,
                 int x, int y, int z, int i) {
    data[z] += ((data[w] & (data[x] ^ data[y])) ^ data[y]) +
                blk (i) + 0x5A827999 + rol (data[v] ,5);
     data[w] = rol (data[w], 30);
   }

  final void R2 (int data[], int v, int w, int x,
                 int y, int z, int i) {
    data[z] += (data[w] ^ data[x] ^ data[y]) +
               blk (i) + 0x6ED9EBA1 + rol (data[v] ,5);
    data[w] = rol (data[w], 30);
   }

  final void R3 (int data[], int v, int w, int x,
                 int y, int z, int i) {
    data[z] += (((data[w] | data[x]) & data[y]) | (data[w] &
                 data[x])) + blk (i) + 0x8F1BBCDC +
                 rol (data[v] ,5);
    data[w] = rol (data[w], 30);
  }

  final void R4 (int data[], int v, int w, int x,
                 int y, int z, int i)  {
    data[z] += (data[w] ^ data[x] ^ data[y]) +
                blk (i) + 0xCA62C1D6 + rol (data[v] ,5);
    data[w] = rol (data[w], 30);
  }

  int dd [] = new int[5];

// Generisanje sažetka (message digest)
  void transform () {
    dd [0] = state [0];
```

```
dd [1] = state [1];
dd [2] = state [2];
dd [3] = state [3];
dd [4] = state [4];
R0 (dd ,0,1,2,3,4, 0); R0 (dd ,4,0,1,2,3, 1);
R0 (dd ,3,4,0,1,2, 2); R0 (dd ,2,3,4,0,1, 3);
R0 (dd ,1,2,3,4,0, 4); R0 (dd ,0,1,2,3,4, 5);
R0 (dd ,4,0,1,2,3, 6); R0 (dd ,3,4,0,1,2, 7);
R0 (dd ,2,3,4,0,1, 8); R0 (dd ,1,2,3,4,0, 9);
R0 (dd ,0,1,2,3,4,10); R0 (dd ,4,0,1,2,3,11);
R0 (dd ,3,4,0,1,2,12); R0 (dd ,2,3,4,0,1,13);
R0 (dd ,1,2,3,4,0,14); R0 (dd ,0,1,2,3,4,15);
R1 (dd ,4,0,1,2,3,16); R1 (dd ,3,4,0,1,2,17);
R1 (dd ,2,3,4,0,1,18); R1 (dd ,1,2,3,4,0,19);
R2 (dd ,0,1,2,3,4,20); R2 (dd ,4,0,1,2,3,21);
R2 (dd ,3,4,0,1,2,22); R2 (dd ,2,3,4,0,1,23);
R2 (dd ,1,2,3,4,0,24); R2 (dd ,0,1,2,3,4,25);
R2 (dd ,4,0,1,2,3,26); R2 (dd ,3,4,0,1,2,27);
R2 (dd ,2,3,4,0,1,28); R2 (dd ,1,2,3,4,0,29);
R2 (dd ,0,1,2,3,4,30); R2 (dd ,4,0,1,2,3,31);
R2 (dd ,3,4,0,1,2,32); R2 (dd ,2,3,4,0,1,33);
R2 (dd ,1,2,3,4,0,34); R2 (dd ,0,1,2,3,4,35);
R2 (dd ,4,0,1,2,3,36); R2 (dd ,3,4,0,1,2,37);
R2 (dd ,2,3,4,0,1,38); R2 (dd ,1,2,3,4,0,39);
R3 (dd ,0,1,2,3,4,40); R3 (dd ,4,0,1,2,3,41);
R3 (dd ,3,4,0,1,2,42); R3 (dd ,2,3,4,0,1,43);
R3 (dd ,1,2,3,4,0,44); R3 (dd ,0,1,2,3,4,45);
R3 (dd ,4,0,1,2,3,46); R3 (dd ,3,4,0,1,2,47);
R3 (dd ,2,3,4,0,1,48); R3 (dd ,1,2,3,4,0,49);
R3 (dd ,0,1,2,3,4,50); R3 (dd ,4,0,1,2,3,51);
R3 (dd ,3,4,0,1,2,52); R3 (dd ,2,3,4,0,1,53);
R3 (dd ,1,2,3,4,0,54); R3 (dd ,0,1,2,3,4,55);
R3 (dd ,4,0,1,2,3,56); R3 (dd ,3,4,0,1,2,57);
R3 (dd ,2,3,4,0,1,58); R3 (dd ,1,2,3,4,0,59);
R4 (dd ,0,1,2,3,4,60); R4 (dd ,4,0,1,2,3,61);
R4 (dd ,3,4,0,1,2,62); R4 (dd ,2,3,4,0,1,63);
R4 (dd ,1,2,3,4,0,64); R4 (dd ,0,1,2,3,4,65);
R4 (dd ,4,0,1,2,3,66); R4 (dd ,3,4,0,1,2,67);
R4 (dd ,2,3,4,0,1,68); R4 (dd ,1,2,3,4,0,69);
R4 (dd ,0,1,2,3,4,70); R4 (dd ,4,0,1,2,3,71);
R4 (dd ,3,4,0,1,2,72); R4 (dd ,2,3,4,0,1,73);
R4 (dd ,1,2,3,4,0,74); R4 (dd ,0,1,2,3,4,75);
R4 (dd ,4,0,1,2,3,76); R4 (dd ,3,4,0,1,2,77);
R4 (dd ,2,3,4,0,1,78); R4 (dd ,1,2,3,4,0,79);
state [0] += dd [0];
state [1] += dd [1];
state [2] += dd [2];
state [3] += dd [3];
```

```
    state [4] += dd [4];
  }

// Početna inicijalizacija pre svakog testiranja
  public void init () {
    state  = new int[5];
    if (block  == null) block  = new int[16];
    state [0] = 0x67452301;
    state [1] = 0xEFCDAB89;
    state [2] = 0x98BADCFE;
    state [3] = 0x10325476;
    state [4] = 0xC3D2E1F0;
    count  = 0;
    if(digestBits == null) digestBits  = new byte[20];
    blockIndex  = 0;
  }

//Ubacuje bajt u niz block koji se dalje obrađuje
  public void update (byte b) {
    int mask = (8 * (blockIndex  & 3));
    count  += 8;
    block [blockIndex  >> 2] &= ~(0xff << mask);
    block [blockIndex  >> 2] |= (b & 0xff) << mask;
    blockIndex ++;
    if (blockIndex  == 64) {
      transform ();
      blockIndex  = 0;
    }
  }

//Kompletira i poslednji 512-bitni blok osnovne poruke
//i posle obrade, ubacuje u rezultujući niz DigestBits
  public void finish () {
    byte bits[] = new byte[8];
    int i, j;
    for (i = 0; i < 8; i++)
      bits[i] = (byte)((count  >>> (((7 - i) * 8))) & 0xff);
    update ((byte) 128);
    while (blockIndex  != 56) update ((byte) 0);
    update (bits);
    for (i = 0; i < 20; i++) digestBits [i] = (byte)
      ((state [i>>2] >> ((3-(i & 3)) * 8) ) & 0xff);
  }

//Ispis
  public String digout () {
    StringBuffer sb = new StringBuffer();
```

```
    for (int i = 0; i < 20; i++) {
      char c1, c2;
      c1 = (char) ((digestBits [i] >>> 4) & 0xf);
      c2 = (char) (digestBits [i] & 0xf);
      c1 = (char) ((c1 > 9) ? 'a' + (c1 - 10) : '0' + c1);
      c2 = (char) ((c2 > 9) ? 'a' + (c2 - 10) : '0' + c2);
      sb.append(c1);
      sb.append(c2);
      if (((i+1) % 4) == 0) sb.append(' ');
    }
    return sb.toString();
  }

  public static void main(String args[]) {
    int i, j;
    SHA1 s = new SHA1();
    System.out.println("SHA-1 Test PROGRAM.");

    /* ulazni podatak: "abc"
       heš: A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D */

    System.out.println("\nPrvi test je 'abc'");
    String z = "abc";
    s.init();
    s.update((byte) 'a');
    s.update((byte) 'b');
    s.update((byte) 'c');
    s.finish();
    System.out.println(s.digout());
    System.out.println("A9993E36 4706816A BA3E2571
                        7850C26C 9CD0D89D");

    /* ulazni podatak: "abcdbcdecdefdefgefghfghighij
                         hijkijkljklmklmnlmnomnopnopq"

       heš: 84983E44 1C3BD26E BAAE4AA1 F95129E5 E54670F1 */
    System.out.println("\nSledeci test je 'abcdbcdecdefdefge
                  fghfghighijhijkijkljklmklmnlmnomnopnopq'");
    z="abcdbcdecdefdefgefghfghighij
       hijkijkljklmklmnlmnomnopnopq";
    s.init();
    s.updateASCII(z);
    s.finish();
    System.out.println(s.digout());
    System.out.println("84983E44 1C3BD26E BAAE4AA1
                        F95129E5 E54670F1");
```

```
    /* ulazni podatak: milion ponavljanja "a"
       heš: 34AA973C D4C4DAA4 F61EEB2B DBAD2731 6534016F */

    System.out.println("\nPoslednji test je 1 milion
                          karaktera 'a'");
    s.init();
    for (i = 0; i < 1000000; i++) s.update((byte) 'a');
    s.finish();
    System.out.println(s.digout());
    System.out.println("34AA973C D4C4DAA4 F61EEB2B
                        DBAD2731 6534016F");
  }
}
```



Lozinke na nivou BIOS-a

BIOS lozinke sprečavaju napadača da izmeni podešavanja BIOS-a (ili da podigne operativni sistem) i kao takve predstavljaju dodatni nivo zaštite laptop i desktop računara. BIOS lozinke su, nažalost, i veliko opterećenje za korisnike koji ih zaborave, kao i za sve korisnike kojima je treće lice zlonamerno (ili šale radi) promenilo lozinku. Ukoliko pripadate toj klasi korisnika, preporučujemo vam da zaboravite mogućnost da Vašu osnovnu ploču pošaljete proizvođaču kako bi vam promenio BIOS lozinku – to je skupo i nije pokriveno garancijom. Međutim, nekoliko lozinki za oporavak i nekoliko trikova koje ćemo opisati može vam biti od pomoći.

Lozinke za oporavak

Neki proizvođači BIOS-a ugrađuju specijalne lozinke (ili kombinacije tastera) koje možete iskoristiti za pristup BIOS konfiguraciji ukoliko ste Vašu lozinku zaboravili. Na primer, na većini Toshiba laptop i na nekim Toshiba desktop računarima provera BIOS lozinke može se izbeći ukoliko prilikom pokretanja računara držite pririsnut levi <shift>.

Ove lozinke su osetljive na mala i velika slova, pa vam preporučujemo da isprobate nekoliko kombinacija ukoliko na ovaj način pokušate da obavite oporavak.

Međutim, neki sistemi će se zaključati ukoliko tri ili više puta unesete neispravnu BIOS lozinku. Zato je najbolje da pre nego što započnete oporavak pročitate dokumentaciju.

Ukoliko je u Vaš računar ugrađen Award BIOS (vrlo česta situacija), isprobajte sledeće lozinke:

- ALFAROME, ALLy, aLLy, aLLY, ALLY, aPAf, _award, AWARD_SW
- AWARD?SW, AWARD SW, AWARD PW, AWKWARD, awkward
- BIOSTAR, CONCAT, CONDO, Condo, d8on, djonet, HLT, J64
- J256, J262, j332, j322, KDD, Lkwpeter, LKWPETER, PINT, pint
- SER, SKY_FOX, SYXZ, syxz, <shift> + syxz, TTPTHA, ZAAADA, ZBAAACA
- ZJAAADC, 01322222, 589589, 589721, 595595, 598598

Lozinke za oporavak za BIOS-e raznih proizvođača date su u sledećoj tabeli.

BIOS proizvođač	Moguća lozinka za oporavak
AMI	AMI, BIOS, PASSWORD, HEWITT, RAND, AMI?SW, AMI_SW, LKWPETER, CONDO
Compaq	Compaq
Dell	Dell
Epox	central
Packard Bell	bell9
Phoenix	phoenix, PHOENIX, CMOS, BIOS
Siemens	SKY_FOX
Toshiba	Toshiba
VOBIS & IBM	merlin
Ostali proizvođači	ALFAROME, BIOSTAR, biostar, biosstar, CMOS, cmos, LKWPETER, lkwpeter, setup, SETUP, Syxz, Wodj

Upotreba "Clear CMOS" kratkospojnika na matičnoj ploči

Mnogi proizvođači matičnih ploča obezbeđuju set kratkospojnika, tj. "džampera" (engl. *jumper*) koji će obrisati CMOS i sva BIOS podešavanja, uključujući i BIOS lozinku. Mesto tih kratkospojnikaa na ploči se razlikuje zavisno od proizvođača i modela, pa savetujemo da pročitate dokumentaciju pre nego što bilo šta uradite. Ukoliko nemate[92] dokumentaciju, kratkospojnike možete naći uz ivicu matične ploče, blizu CMOS baterije, ili blizu procesora. Neki proizvođači će te džampere obeležiti imenima poput CLEAR, CLEAR CMOS, CLR, CLRPWD, PASSWD, PASSWORD ili PWD.

Procedura je sledeća: isključite Vaš računar, premestite kratkospojnike

92 Neki prodavci "žutih konfiguracija" ne isporučuju "detaljnu tehničku dokumentaciju", kao što su uputstva za instalaciju matičnih ploča, jer smatraju da je ona nepotrebna korisnicima.

prema uputstvu, zatim uključite računat i potvrdite da je lozinka izbrisana. Ukoliko jeste, isključite računar i vratite kratkospojnike u originalni položaj. Nakon toga preostaje vam da unesete novu lozinku i podesite sva neophodna BIOS podešavanja. Preporučuje se da pre nego što upotrebite kratkospojnike, zabeležite sva BIOS podešavanja u koja niste sigurni, jer ćete nakon oporavka lozinke morati da rekonfigurišete BIOS.

Uklanjanje CMOS baterije

Za čuvanje CMOS podešavanja na većini računara zadužena je mala baterija (slična baterijama za ručne časovnike) koja se nalazi na matičnoj ploči. Ukoliko isključite napajanje računara i uklonite CMOS bateriju na 10-15 minuta, CMOS će se resetovati, što znači da će i BIOS lozinka biti izbrisana. Neki proizvođači obezbeđuju dodatni nivo zaštite CMOS napajanja pomoću kondenzatora, koji će napon održati dovoljno visokim u nekom dužem intervalu. Ukoliko 15 minuta nije dovoljno za brisanje lozinke, ostavite računar bar 24 časa bez napajanja i bez CMOS baterije. Neke baterije su zalemljene za matičnu ploču što znači da je postupak vađenja baterije otežan. U tom slučaju preporučujemo vam da odustanete i kontaktirate servisera. Alternativno rešenje, koje takođe ne preporučujemo, je uklanjanje CMOS čipa na određeno vreme.

Uklanjanje baterije je postupak koji na nekim računarima neće imati efekta. Na primer, većina novih laptop računara čuva BIOS lozinke u specijalnim memorijama koje ne zahtevaju kontinualno napajanje. Drugi metod zaštite je primenjen na IBM Thinkpad laptop računarima: računar će zaključati hard disk ukoliko je administrativna BIOS lozinka postavljena. Ukoliko resetujete BIOS lozinku, ali ne i lozinku kojom je zaštićen hard disk, nećete moći da pristupite podacima na disku, čak i ako ga premestite u nov računar.

Rečnik termina

adresa lokalne petlje	*local loopback address*
auto makro (samoizvršavajući)	*auto-macro*
agent za oporavak	*recovery agent*
besplatan softver	*freeware*
bombardovanje SYN paketima	*SYN flooding*
chroot zatvor	*chroot jail*
crv	*worm*
čarobnjak	*wizard*
čekanje na novu CRL	*time-granularity*
čitač Web-a	*Web browser*
čuvar ekrana	*screensaver*
datoteka sa ključevima	*keyring*
datoteka za straničenje	*page file*
detekcija anomalija	*anomaly detection*
detekcija zloupotreba	*misuse detection*
dešifrovanje	*decryption, decipher*
digitalni potpis	*digital signature*
dolazeći saobraćaj	*inbound traffic*
“džamper”, kratkospojnik	*jumper*
NAT sa preopterećenjem	*port address translation, PAT*
dnevnik događaja, dnevnik aktivnosti	*log*
dnevnik transakcija	*journal, log*
efekt lavine	*avalanche*
Feistelova mreža	*Feistel network*
funkcija sažimanja	*message digest*
glavni startni zapis	*master boot record, MBR*
generator pseudoslučajnih sekvenci	*pseudorandom number generator, PRNG*
grupa	*group*

hab, koncentrator	*hub*
heš, heš funkcija	*hash*
host bazirani IDS	*host based IDS, HIDS*
identifikaciona kartica	*smartcard*
imenik	*direktorijum*
informaciono stablo direktorijuma	*directory information tree*
infrastruktura javnih ključeva	*public key infrastructure, PKI*
iskačući prozor	*pop-up window*
izazov-odgovor	*challenge-response*
javni ključ	*public key*
jedinstveno prepoznatljivo ime	*distinguished name*
jednokratna beležnica	*one-time pad*
jednosmerna funkcija	*one-way*
jednosmerna funkcija sa zamkom, privatna jednosmerna funkcija	*trapdoor one-way*
keširanje Internet objekata	*Internet object caching*
ključ	*key*
ključ-datoteka	*keyfile*
ključ za šifrovanje datoteka	*File Encryption Key, FEK*
ključna sekvenca	*keystream*
kolačić	*cookie*
kolizija	*collision*
komutator, mrežna skretnica	*switch*
konačno polje	*Galois field*
koncentrator, hab	*hub*
kontrolna tačka bezbednosti (na granici mreže)	*border security*
korisnički režim rada	*user mode*
korisničko ime	*username*
krađa sesije	*session hijacking*

kratkospojnik, “džamper”	*jumper*
lanac pravila	*(rule) chain*
lažiranje izvorišne adrese	*address spoofing*
lažna pozitivna uzbuna	*false alarm*
linearni difuzioni sloj	*linear mixing layer*
linearni pomerački registar sa povratnom spregom	*linear feedback shift register, LFSR*
lista kontrole pristupa	*Access Control List, ACL*
lista poništenih sertifikata	*certificate revocation list, CRL*
lokalna računarska mreža	*local area network, LAN*
lozinka	*password*
identifikator objekta	*object identifier, OID*
maskiranje	*masquerading*
međuprocesni kanal	*pipe*
mehanizmi autentifikacije poruka	*message authenticaton codes, MAC*
mehanizmi za uočavanje promena	*modification detection codes, MDC*
metoda “grube sile”	*brutal force*
metode samouništenja u slučaju napada	*tamper-proof*
monitor integriteta	*integrity monitor*
monitor log datoteka	*log file monitor*
montiranje	*mount*
most	*bridge*
mreža širokog područja	*wide area network, WAN*
mrežna barijera	*firewall*
mrežna barijera bez uspostave stanja	*stateless firewall*
mrežna barijera sa uspostavom stanja	*statefull firewall*
mrežna skretnica, komutator	*switch*
mrežno bazirani IDS	*network based IDS, NIDS*

nadzor sistema	*monitoring*
najmanje značajan bit, najniži bit	*least significant bit*
najznačajniji bit, najviši bit	*most significant bit*
napad “čovek u sredini”	*man-in-the-middle attack*
napad obrtanjem bitova	*bit-flipping attack*
napad “odabrani šifrat”	*chosen-ciphertext attack*
napad ponavljanjem inicijalizacionog vektora	*IV replay attack*
napad ponavljanjem paketa	*packet replay attack*
napad potkupljivanjem, krađom ili ucenom	*rubber-hose attack*
napad “poznat otvoreni tekst”	*known-plaintext attack*
napad “samo šifrat”	*ciphertext-only attack*
napad “susret u sredini”	*meet-in-the-middle attack*
nastavljanje poruke	*padding*
navođenje tačne putanje puta	*source routing*
nelinearni sloj	*non-linear layer*
objekat grupne polise	*Group Policy Object*
obnavljač signala	*repeater*
obvojnica mrežnih servisa	*wrapper, network daemon wrapper*
odlazeći saobraćaj	*outbound traffic*
okidač	*trigger*
okruženje za izvršavanje skriptova	*script host*
okvir, ram	*frame*
organizaciona jedinica	*organizational unit*
ostali (korisnici)	*others*
otisak prsta	*fingerprint*
otkrivanje napada nakon dešavanja	*after–the–fact detection*
otkrivanje napada u toku	*real–time detection*
otvoreni tekst	*plain text*

paket	*packet*
parazitski virus	*parasitic virus*
polisa oporavka podataka	*Data Recovery Policy*
podmetanje žrtve	*smurfing*
podobjekat	*child object*
podsistem za detekciju	*detection engine*
podsistem za upozoravanje	*alerting subsystem*
početna Web stranica u browseru	*home page*
pomerački registar sa povratnom spregom	*feedback shift register, FSR*
popisivanje	*fingerprinting*
poruka izazova (udaljenom ruteru)	*challenge*
praćenje događaja	*auditing*
pravila za izdavanje sertifikata	*certificate practices statement*
prekoračenje, prelivanje bafera	*baffer overrun, buffer overflow*
prethodno postavljeni ključ	*pre-shared keys*
prevođenje izvorišnih adresa	*source NAT, SNAT*
prevođenje mrežnih adresa	*network address translation, NAT*
prevođenje odredišnih adresa	*destination NAT, DNAT*
pridužujući virus	*companion virus*
prilog elektronske pošte	*e-mail attachment*
prisluškivanje mrežnog saobraćaja	*evesdropping*
pristupna tačka	*access point*
privatni ključ	*private key*
program koji beleži unos sa tastature, program za krađu lozinki	*keylogger*
program koji poziva telefonske brojeve bez znanja korisnika	*dialer*
prosleđivanje paketa	*forwarding*
protočna šifra	*stream cipher*
protokol koji se rutira	*routed protocol*

protokol za rutiranje	*routing protocol*
provera bioloških atributa	*biometrics*
radni okvir	*framework*
ram, okvir	*frame*
rečnik (za napad na šifrat ili heš)	*wordlist, dictionary*
registracioni centar	*registration authority, RA*
reklamne poruke	*advertisment, ad*
reklamni server	*advertising server, ad*
reklamni špijunski softver	*adware*
režim rada	*mode of operation*
ruter, usmerivač	*router*
samopotpisani sertifikat	*self-signed certificate*
samostalni program	*stand-alone program*
serifikat identiteta	*identity certificate*
sertifikaciona polisa	*certificate policy*
sertifikacioni centar	*certificate authority, CA*
sertifikat	*certificate*
sertifikat akreditiva	*credential certificate*
server ključeva	*keyserver*
server sertifikata	*certificate server*
sigurno brisanje datoteka	*shredding*
sigurnosni identifikator	*Security Identifier, SID*
sigurnosni opis objekta	*Security Descriptor*
sistem za detekciju napada	*intrusion detection system, IDS*
sistem za sprečavanje napada	*intrusion prevention system, IPS*
sistemski događaj niskog nivoa	*low-level event*
skladište sertifikata	*certificate repository*
skener potpisa	*signature scanner*
skeniranje portova	*port scan*
skriveni volumen	*hidden volume*

sloj dodavanja ključa	*key addition layer*
snimak mrežnjače oka	*retina scan*
sociološki inženjering	*social engineering*
stanje (AES bloka)	*state (AES)*
startni zapis particije	*boot sector*
supstitucijska kutija, S-kutija	*supstitution box, S-box*
šifrat	*ciphertext*
šifrovanje	*encryption, encipher*
šifrovanje celog diska	*whole disk encryption*
šifrovanje link-po-link	*link-by-link encryption*
šifrovanje s kraja na kraj	*end-to-end encryption*
šifrovanje sistema datoteka u datoteci	*volume disk encryption*
špijunski kolačići	*tracking cookies*
špijunski softver	*spyware*
tabela za rutiranje	*routing table*
trojanski konj “obaveštajac”	*trojan notifiers*
trojanski konj “špijun”	*trojan spy*
trojanski proksi server	*trojan proxy*
“udica”	*hook*
ugovoro o korišćenju	*End User Licence Agreement, EULA*
ugrađen nalog	*built-in account*
ukrštena sertifikacija	*cross-certification*
ulančavanje blokova šifrata	*cipher block chaining, CBC*
ulazna tačka u zaglavlju izvršnje datoteke	*entry-point*
ulazni podatak promenljive dužine	*pre-image*
upravljački okvir	*management frame*
usmerivač, ruter	*router*
uspostavljanje veze	*link establishment*

vlasnik	*owner*
virus koji prepisuju postojeći kod	*overwriting virus*
virus koji se dopisuje na kraj datoteke	*appending virus*
virus koji se dopisuje na početak datoteke	*prepending virus*
virus koji se dopisuje unutar postojećeg koda	*inserting*
virus koji se upisuje u šupljine zdravog koda	*cavity virus*
virusi startnog zapisa	*boot-sector virus*
vremenski ključ	*temporal key*
“zadnja vrata”	*backdoor*
zamka	*trapdoor*
zapis u ACL	*Access Control Entry, ACE*
zaštićeni režim rada	*protected mode*
zlonamerni softver	*malware, malicious software*
žeton za pristup	*access token*
“živa” Linux distribucija	*live Linux distribution*

Literatura

[1] B. Stroustrup, *Programski jezik C++*, Mikro knjiga (1991)

[2] D. Pleskonjić, *Analiza kriptografskih metoda zaštite podataka*, magistarski rad, Elektrotehnički fakultet u Zagrebu (1991)

[3] D. Stinson, *Cryptography – Theory and Practice*, CRC Press, Boca Raton, Florida (1995)

[4] B. Schneier, *Applied Cryptography, Second Edition*, John Wiley & Sons, Inc. (1996)

[5] A. S. Tannenbaum, *Computer Networks, Third Edition*, Prentice Hall (1996)

[6] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Inc. (1997)

[7] A. S. Tannenbaum, A. S. Woodhull, *Operating System Design and Implementation*, Second Edition, Prentice Hall (1997)

[8] W. Stallings, *Operating Systems*, Fourth Edition, Prentice Hall (2000)

[9] M. Živković, *Algoritmi*, Matematički fakultet, Beograd (2000)

[10] S. McClure, J. Scambray, G. Kurtz, *Sigurnost na mreži*, Kompjuter biblioteka (2001)

[11] G. Mourani, *Securing and Optimizing Linux: The Ultimate Solution*, Open Network Architecture Inc. (2001)

[12] *Security Complete*, Second Edition, Sybex Inc. (2002)

[13] Michael Howard, David LeBlanc, *Writing Secure Code*, 2nd Edition, Microsoft Press (2002)

[14] A. Silberschatz, P. B. Galvin, G. Gagne, *Operating System Concepts*, Sixth Edition (Windows XP Update), John Wiley & Sons, Inc (2003)

[15] N. Ferguson, B. Schneier, *Practical Cryptography*, Weily Publishing Inc. (2003)

[16] S. Obradović, *Osnovi računarske tehnike i programiranja*, četvrto izdanje, Viša elektrotehnička škola (2003)

[17] D. Pleskonjić, *Wireless Intrusion Detection Systems (WIDS)*, 19th Annual Computer Security Applications Conference, December 8-12, 2003, Las Vegas, Nevada, USA

[18] B. Đorđević, D. Pleskonjić, N. Maček, *Operativni sistemi: UNIX i Linux*,

Viša elektrotehnička škola, Beograd, 2004

[19] B. Đorđević, D. Pleskonjić, N. Maček, *Operativni sistemi: koncepti*, Viša elektrotehnička škola, Beograd, 2004

[20] V. Vasiljević, P. Gavrilović, B. Krneta, M. Krstanović, N. Maček, B. Bogojević, *Priručnik za administraciju računarskih mreža*, Viša elektrotehnička škola, Beograd, 2004

[21] V. Vasiljević, P. Gavrilović, B. Krneta, M. Krstanović, N. Maček, B. Bogojević, *Protokoli u računarskim mrežama – priručnik za laboratorijske vežbe*, Viša elektrotehnička škola, Beograd, 2004

[22] B. Đorđević, D. Pleskonjić, N. Maček, *Operativni sistemi: zbirka rešenih zadataka*, Viša elektrotehnička škola, Beograd, 2005

[23] B. Đorđević, D. Pleskonjić, N. Maček, *Operativni sistemi: teorija, praksa i rešeni zadaci*, Mikro knjiga (2005).

[24] D. Pleskonjić, *Protecting wireless computer networks by using intrusion detection agents*, IPSI 2005, Venice, Italy, November 10-13, 2005

[25] S. Mesarović, *Infrastruktura javnih ključeva*, diplomski rad, Tehnički fakultet "Mihajlo Pupin", Zrenjanin, 2005

[26] M. Bojović, *Squid proksi server*, diplomski rad, Viša elektrotehnička škola, Beograd, 2005

[27] A. Dujella, Kriptografija (skripta za predavanja), PMF, Sveučilište u Zagrebu, http://www.math.hr/~duje/kript.html

[28] *AES algoritam*, CCERT-PUBDOC-2003-08-37, revizija 1.1, CARNet CERT (Hrvatska akademska i istraživačka mreža), www.cert.hr.

[29] http://www.pmdtc.org/reference.htm, *International Traffic in Arms Regulations*, Directorate of Defense Trade Controls, U.S. Department of State

[30] http://www.pgpi.org/doc/whypgp/en/, P. Zimmermann, *Why do you need PGP?*

[31] http://www.spywareguide.com, On-line guide to Spy and Anti-Spy software

[32] http://www.softpedia.com, Encyclopedia of Free Software Downloads

[33] http://www.wikipedia.org, The Free Encyclopedia