# SIGURNOST RAČUNARSKIH MREŽA (SRM)

**Tema 12:**

## Sigurnosni aspekti programiranja

# URLs:



- Zvanična Web strana: www.viser.edu.rs/predmeti.php?id=122
- Dodatni resursi: www.conwex.info/draganp/teaching.html
- Knjige:

  www.conwex.info/draganp/books.html
- Teme za seminarske radove:

  www.conwex.info/draganp/SRM_seminarski_radovi.html

# Napomena



- Ovo je skraćena verzija prezentacije / predavanja na temu **“Sigurnosni aspekti programiranja”**

# Sigurnosni aspekti programiranja



- Sadržaj poglavlja i predavanja:
  - 12.1 Uvodne napomene
  - 12.2 C/C++ i problem prekoračenja bafera
  - 12.3 Sigurnosni aspekti programiranja na jeziku Java
  - 12.4 .NET tehnologija i Security Development Lifecycle
  - 12.5 Zaštita softvera

# Quote



*"To subdue the enemy without fighting is the acme of skill"*

– Sun Tzu

# Potrebna predznanja



- Programiranje
- Za primenu:
  - Računarske mreže i protokoli
  - Operativni sistemi
  - Sistemsko programiranje
  - Strukture i modeli podataka, baze podataka

# Uopšteno o problemu



- Pojam sigurne aplikacije odnosi se na njenu otpornost prema različitim vrstama napada. Poslednjih godina statistike ukazuju na to da se najviše sigurnosnih propusta odnosi na aplikacioni sloj. Drugim rečima, pisanje sigurnog kôda je od vitalnog značaja za održavanje sistema sigurnim. Princip zatvorenog kôda ne predstavlja ozbiljnu prepreku za napadača: upotrebom disasemblera može se ponovo doći do osnovnog kôda. Nezaštićen program je tempirana bomba – on funkcioniše dok su mu okolnosti naklonjene. Ovakvo izlaganje igri slučaja apsolutno je neprihvatljivo u bilo kojem ozbiljnom projektu. Sigurnost se ne sme zasnivati na spoljnoj sigurnosti mreže na štetu unutrašnje sigurnosti kôda.

# 12.1. Uvodne napomene



- Pravila „sigurnog programiranja“ podrazumevaju određena odstupanja od programske jednostavnosti, a često i od njegove efikasnosti. Argument za to je prilično ubedljiv: u slučaju pada sistema, do tada efikasan kôd više nikome ne koristi. Siguran kôd, između ostalog, korisnika štiti i od njega samog. Neko može biti zloupotrebljen bez svog znanja i pristanka, a da toga nije ni svestan. Takođe, sigurnost često rezultira umanjenjem efikasnosti i performansi, jednostavnosti i komfora, kao što je recimo i slučaj sa sigurnošću u drugim oblastima (vezivanje pojasa u automobilu, nošenje pancir prsluka ili šlema i slično).

# Kategorije napada



- **Subverzija aplikacije.** Akcija koja prouzrokuje izvršavanje nenameravane funkcionalnosti.
- **Subverzija sistema i spoljnih aplikacija.** Iskorišćavanje sigurnosnih propusta aplikacije utiče na druge pokrenute aplikacije ili sistemske resurse. Napad ne ugrožava samo pokrenutu aplikaciju, nego se ta aplikacija često koristi i kao kanal za napad na druge sisteme, aplikacije i operativni sistem. Napadi ovog tipa često uključuju i izvršavanje drugih aplikacija, kao što je komandni interpreter u operativnom sistemu UNIX, ili iskorišćavanje veze koju je druga aplikacija ostvarila sa mrežom.
- **Prekidi funkcionalnosti.** Svaki oblik odbijanja usluga, u šta spada i grubo rušenje aplikacije.

# 12.2 C/C++ i problem prekoračenja bafera



- **Bafer** je memorijski prostor ograničenog kapaciteta, predviđen za skladištenje podataka.
- **Prekoračenje bafera** je anomalija do koje dolazi kada proces u bafer upiše podatke koji su zbirno veći od veličine tog bafera.
- **Posledica prekoračenja** je prepisivanje susednih memorijskih lokacija, tj. izmena vrednosti upisanih u susedne memorijske lokacije.

# Prekoračenje bafera



- Podaci koji su prepisani, tj. izmenjeni mogu biti drugi baferi, promenljive ili podaci koji obezbeđuju kontrolu toka programa. Prekoračenje bafera može izazvati krah ili nepravilno izvršenje programa; najčešća posledica je ranjivost kôda koju napadači mogu iskoristiti.
  - Na primer, prosleđivanjem određenog niza kao ulaznog podataka programu, napadač može prepuniti bafer, prepisati susedne memorijske lokacije i omogućiti izvršenje (najčešće zlonamernog) kôda podmetnutog u tom nizu. Da bi u tome uspeo, napadač pažljivo formira ulazni niz znakova i precizno smešta nove podatke (povratnu adresu i kôd) na određene adrese u memoriji, što dovodi do izvršavanja podmetnutog kôda. U najgorem slučaju, prekoračenje bafera može dovesti do povećanja privilegija napadača i kompromitovanja sistema (napadač preuzima kontrolu nad računarom).

# Vežbe



- Detalji vezani za prekoračenje bafera, kao i praktični primeri, se obrađuju na vežbama iz ovog predmeta

- Takođe, detaljniji opisi i objašnjenja se mogu naći u knjizi koja prati predmet i predavanja:

  - D. Pleskonjić, N. Maček, B. Đorđević, M. Carić: **"Sigurnost računarskih sistema i mreža"**, Mikro knjiga, Beograd, 2007., ISBN: 978-86-7555-305-2, knjiga – udžbenik
  - www.conwex.info/draganp/books_SRSiM.html
  - www.mk.co.yu/store/prikaz.php?ref=978-86-7555-305-2

# 12.3 Sigurnosni aspekti programiranja na jeziku Java



- Programski jezik smatramo sigurnim ako obezbeđuje sledeće:
  - otpornost prema zlonamernim programima koji opterećuju resurse ili kradu informacije, npr. trojanskim konjima ili virusima
  - proveru identiteta strana koje komuniciraju,
  - mogućnost šifrovanja primljenih i poslatih podataka,
  - programski zahtevi ne smeju previše okupirati raspoložive resurse,
  - onemogućavanje zloupotrebe regularnih naredbi jezika u cilju opstrukcije normalnog rada operativnog sistema.
- **Apleti** su Java programi koji se pozivaju iz Web čitača i namenjeni su za korišćenje na Internetu.
- Kada stignu u korisnikov sistem, apleti se smeštaju u poseban kontejner – takozvani ***sandbox*** – koji predstavlja okruženje u kome su dozvoljena samo određena prava.
- **Javina virtuelna mašina** (*Java Virtual Machine*, *JVM*) predstavlja Javino izvršno okruženje.
- **JVM** je deo okruženja **JRE** (*Java Runtime Environment*), koje sadrži i neke osnovne klase, tj. klase koje pripadaju jezgru Jave (engl. *core*).

# Vežbe



- Detalji vezani za sigurnosne aspekte programiranja na jeziku Java, kao i praktični primeri, se obrađuju na vežbama iz ovog predmeta

- Takođe, detaljniji opisi i objašnjenja se mogu naći u knjizi koja prati predmet i predavanja:

  - D. Pleskonjić, N. Maček, B. Đorđević, M. Carić: **"Sigurnost računarskih sistema i mreža"**, Mikro knjiga, Beograd, 2007., ISBN: 978-86-7555-305-2, knjiga – udžbenik
  - www.conwex.info/draganp/books_SRSiM.html
  - www.mikroknjiga.rs/store/prikaz.php?ref=978-86-7555-305-2

# 12.4 .NET tehnologija i Security Development Lifecycle



- Microsoft je bio predmet velikih kritika u pogledu sigurnosti njegove familije operativnih sistema Windows, kao i ostalih softverskih proizvoda i alata i integrisanih okruženja za razvoj softvera. Tokom 2002. godine osnivač i direktor Microsota, Bill Gates je pokrenuo takozvanu *Trustworthy Computing* inicijativu koja je trebala da doprinese poboljšanju sigurnosti Micorosoft proizvoda

# .NET tehnologija i upravljani kod



- Tehnologija upravljanog (engl. *managed*) kôda
- Microsoftov .NET Framework Common Language Runtime (CLR) ili neka druga implementacija CLI specifikacije (Mono projekat ili DotGNU projekat)
  - CLR je Microsoftova implementacija CLI specifikacije.
- Posredni jezik (engl. *Intermediate Language*, *IL*)

- Neupravljani kôd (engl. *unmanaged code*)

# Security Development Lifecycle



- Trustworthy Computing Security Development Lifecycle (u slobodnom prevodu – životni ciklus razvoja softvera sa osvrtom na sigurnost) je proces razvoja softvera razvijen u Microsoftu. Autori knjige „*The Security Development Lifecycle*“, Michael Howard i Steve Lipner, radili su na konceptu koji je dobio ovo ime, a poznat je pod skraćenicom **SDL**.
- Ovaj koncept vodi kroz sve faze razvoja softvera uključujući i obrazovanje i usmeravanje projektanata i programera, projektovanje, razvoj, testiranje, isporuku (objavljivanje) i održavanje, tj. proces posle objavljivanja (engl. *post-release*).
- Koncept podrazumeva analizu mogućih napada i izloženosti i pre početka projektovanja aplikacije, kao i procenu rizika. SDL navodi i listu zabranjenih API funkcija koje mogu prouzrokovati sigurnosne probleme i koje treba vremenom ukloniti iz nasleđenog koda.

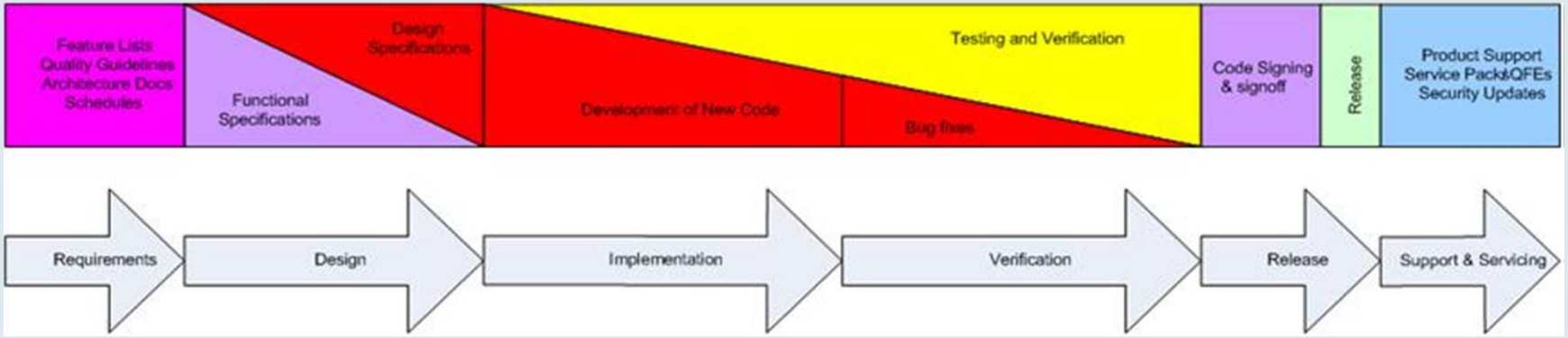
# Standardni Microsoft proces razvoja softvera



- Standardni Microsoft proces razvoja softvera obuhvata sledeće faze:
  - **analiza zahteva** (spisak zahteva, smernice za obezbeđivanje kvaliteta, dokumentovanje arhitekture projekta i formiranje preliminarnog rasporeda)
  - **projektovanje** (specifikacije strukture i funkcionalnosti)
  - **razvoj** (razvoj novog kôda uz povremene provere i testiranje koda)
  - **testiranje** (provera kôda i ispravka grešaka)
  - **objavljivanje** (potpisivanje kôda i isporuka gotovog proizvoda)
  - **održavanje i servisiranje** (tehnička podrška krajnjim korisnicima, sigurnosne zakrpe i servisni paketi)

# Software Development Lifecycle






*Source: Microsoft Web*

# SDL unapređenja



- **SDL – Security Development Lifecycle** - unapređenja standardnog Microsoftovog procesa razvoja (po fazama) su sledeća:

- **Analiza zahteva.** Obuka članova razvojnog tima, dodela savetnika za sigurnost, razmatranje kako će sigurnost biti implementirana u razvojni proces, identifikacija ključnih sigurnosnih ciljeva, razmatranje mogućnosti za uvećanje sigurnosti uz što manje ometanje plana i rasporeda razvoja.
- **Projektovanje.** Definisanje globalne strukture softvera posmatrane sa stanovišta sigurnosti. Modeliranje pretnji. Identifikacija i dokumentovanje komponenti i elemenata čije je funkcionisanje ključno za sigurnost. Ovim komponentama se dodeljuje najmanji mogući skup privilegija neophodnih za izvršavanje.
- **Razvoj.** Sprovođenje standarda za kodiranje i testiranje. Prgrameri ne smeju da koriste funkcije i procedure koje mogu dovesti do sigurnosnih propusta. Na primer, umesto funkcija za rad sa nizovima koje ne obraćaju pažnju na veličinu niza koriste se sigurnije funkcije koje ne mogu dovesti do ranjivosti tipa prekoračenja bafera. Kôd se testira na više načina – upotrebom takozvanih „*fuzzing*“ alata (alati koji proizvode strukturiran, ali neispravan ulaz za API i mrežne funkcije i na taj način pomažu u otkrivanju grešaka koje mogu da dovedu do ranjivosti), upotrebom alata za statičku analizu kôda (kao što su PREfix i PREfast) i sprovođenjem revizije kôd (razvojni tim „ručno“ proverava izvorni kôd i traži moguće sigurnosne propuste).
- **Testiranje.** Softver koji je funkcionalno kompletan ulazi u takozvano preliminarno testiranje (engl. *beta testing*). Tokom ove faze razvojni tim testira mogućnost proboja koda, tj. reviziju koda koja je znatno detaljnija od one koja se obavlja u fazi razvoja i prevashodno je usmerena na sigurnost. Microsoft ovo testiranje opisuje terminom „*security push*“.

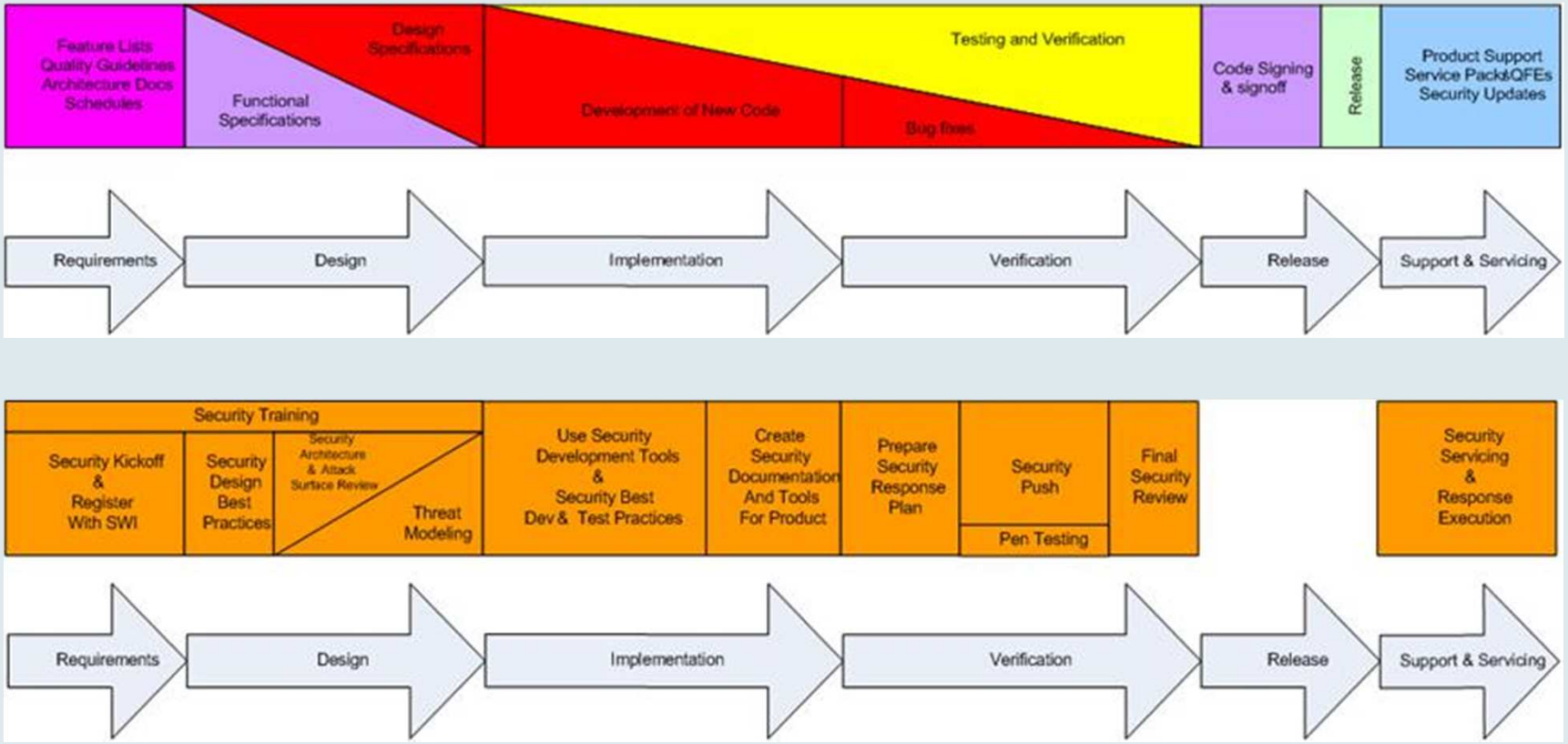
# SDL unapređenja - nastavak



- **Objavljivanje.** U fazi objavljivanja, softver je izložen konačnoj sigurnosnoj proceni (engl. *Final Security Review*, *FSR*). Cilj FSR je davanje odgovora na pitanje „Da li je, posmatrano sa stanovišta sigurnosti, softver spreman za isporuku krajnjim korisnicima?“ Softver mora da bude u stabilnom stanju pre obavljanja FSR, što znači da se u softveru mogu obaviti samo manje kozmetičke izmene, ali nikako funkcionalne ili one koje se tiču sigurnosti. FSR najčešće obavlja tim stučnjaka razvojne kompanije zadužen za sigurnost. FSR nije situacija tipa „prošao/pao“, niti za cilj ima otkrivanje svih sigurnosnih propusta, već treba rukovodstvu kompanije da stvori sliku o implementaciji sigurnosti u proces razvoja softvera i da proceni kako će softver isporučen kupcima reagovati na napade. Ukoliko se u ovoj fazi pronađe neka ranjivost, onda je potrebno da se, osim ispravljanja, otkrije uzrok u ranijim fazama razvojnog procesa koji je doveo do tog sigurnosnog propusta. Na osnovu može da se odredi da li je potrebno, na primer, da se poja obuku razvojnog tima vezana za sigurnost ili da se promene alati za testiranje ranjivosti.
- **Održavanje i servisiranje.** Sigurnost je proces, a ne završno stanje ili proizvod. Čak i ako su u prethodnim fazama razvojnog procesa uklonjeni svi sigurnosni propusti, moguće je da će nakon objavljivanja i isporučivanja softvera biti otkrivene nove vrste napada, odnosno da će softver koji je se u fazi objavljivanja pokazao kao „siguran“ vremenom postati ranjiv. Razvojni tim mora biti spreman da odgovori na novootkrivene ranjivosti i da potrošačima, tj. krajnjim korisnicima na vreme isporuči sve potrebne sigurnosne zakrpe koje će ukloniti te ranjivosti. U ovoj fazi razvojni tim i tim stručnjaka zaduženih za sigurnost uočavaju elemente koji su omogućili da softver postane ranjiv – na osnovu toga, razvojni proces se modifikuje tako da se spreči pojava ovakvih ranjivosti u softveru koji će se razvijati u budućnosti.

# Security Development Lifecycle

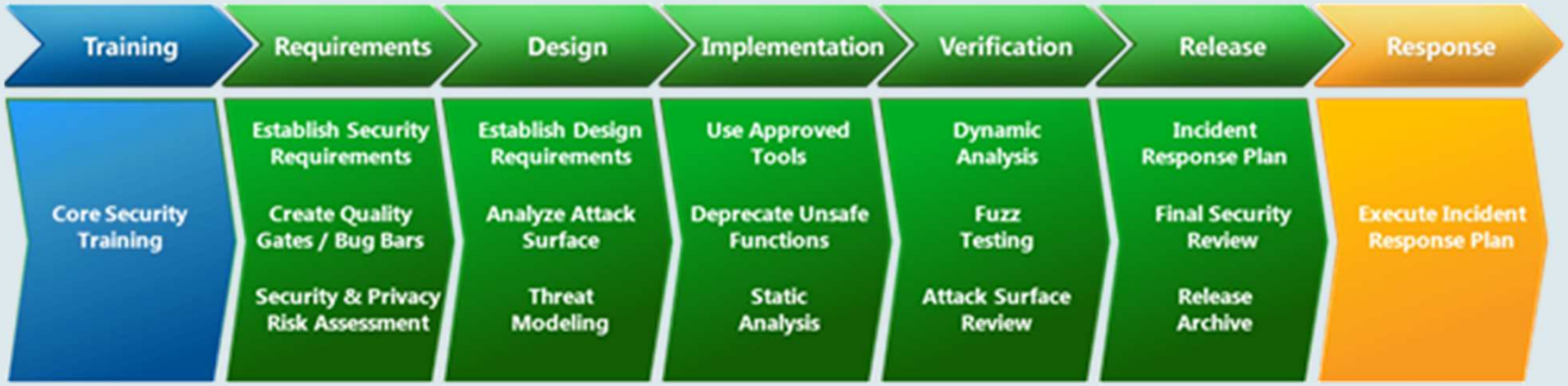





*Source: Microsoft Web*

# Security Development Lifecycle






*Source: Microsoft Web*

# OWASP



- The Open Web Application Security Project (OWASP)
  - www.owasp.org:
    - "not-for-profit worldwide charitable organization focused on improving the security of application software. Our mission is to make application security visible, so that people and organizations can make informed decisions about true application security risks. Everyone is free to participate in OWASP and all of our materials are available under a free and open software license."

# OWASP Top 10 for 2010



- The OWASP Top 10 Web Application Security Risks for 2010 are:
  - A1: Injection
  - A2: Cross-Site Scripting (XSS)
  - A3: Broken Authentication and Session Management
  - A4: Insecure Direct Object References
  - A5: Cross-Site Request Forgery (CSRF)
  - A6: Security Misconfiguration
  - A7: Insecure Cryptographic Storage
  - A8: Failure to Restrict URL Access
  - A9: Insufficient Transport Layer Protection
  - A10: Unvalidated Redirects and Forwards

# Alati za analizu koda



- Softverski alati koji vrše statičku i dinamičku analizu programskog koda otkrivajuću nedostatke i ranjivosti u pogledu sigurnosti
- Neki od ovih alata mogu uraditi analizu arhitekture i builda
- Pronalaze i označavaju nedostatke i ranjivosti kako u izvornom, tako i u objektnom i izvršnom kodu u određenoj meri
- Imaju dodatke (engl. *plug-in*) za različita razvojna okruženja i programske jezike, a mogu raditi i analizu nad kodom u *source control* sistemima kao što su VSS, TFS, CVS…

# Alati za analizu koda



- Fortify www.fortify.com
- Ounce Labs www.ouncelabs.com
- Coverity www.coverity.com

# Microsoft alati za analizu koda



**Application Verifier**

http://www.microsoft.com/downloads/details.aspx?familyid=c4a25ab9-649d-4a1b-b4a7-c9d8b095df18&displaylang=en

**Fx COP**

http://www.microsoft.com/downloads/details.aspx?familyid=9AEAA970-F281-4FB0-ABA1-D59D7ED09772&displaylang=en

**Microsoft Code Analysis Tool .NET (CAT.NET)**

http://www.microsoft.com/downloads/details.aspx?familyid=0178E2EF-9DA8-445E-9348-C93F24CC9F9D&displaylang=en

**Microsoft Anti-Cross Site Scripting Library V3.1**

http://www.microsoft.com/downloads/details.aspx?familyid=051EE83C-5CCF-48ED-8463-02F56A6BFC09&displaylang=en

**BinScope Binary Analyzer**

http://www.microsoft.com/downloads/details.aspx?familyid=90E6181C-5905-4799-826A-772EAFD4440A&displaylang=en

# 12.5 Zaštita softvera



- Metode zaštite softvera
  - Zaštita CD-a od kopiranja
  - Ključ za registraciju
  - Autorizacija
  - Zaštita programa pomoću registracione datoteke
  - Hardverski "ključ" (engl. *dongle*)
  - On-line registracija i provera

# Komercijalna rešenja



- FLEXlm (Flex License Manager)
- HASP (*Hardware Against Software Piracy*)
- PC Guard
- SyncroSoft MCFACT

# Metode za ukidanje zaštite



- *Debugging*
- Disasembleri i dekompajleri

# Literatura




- D. Pleskonjić, N. Maček, B. Đorđević, M. Carić: **"Sigurnost računarskih sistema i mreža"**, Mikro knjiga, Beograd, 2007., ISBN: 978-86-7555-305-2, knjiga – udžbenik
- www.conwex.info/draganp/books_SRSiM.html
- www.mikroknjiga.rs/store/prikaz.php?ref=978-86-7555-305-2

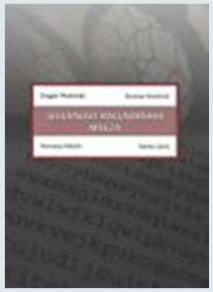
- Za predavanje 12:
  - Poglavlje 12: Sigurnosni aspekti programiranja

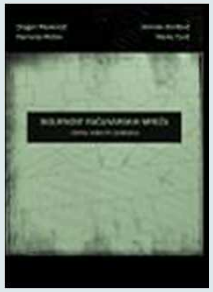
# Literatura - nastavak



- D. Pleskonjić, B. Đorđević, N. Maček, Marko Carić: **"Sigurnost računarskih mreža"**, Viša elektrotehnička škola, Beograd, 2006., ISBN 86-85081-16-5, knjiga - udžbenik
- D. Pleskonjić, B. Đorđević, N. Maček, Marko Carić: **"Sigurnost računarskih mreža - priručnik za laboratorijske vežbe"**, Viša elektrotehnička škola, Beograd, 2006., ISBN 86-85081-49-1
- D. Pleskonjić, B. Đorđević, N. Maček, Marko Carić: **"Sigurnost računarskih mreža - zbirka rešenih zadataka"**, Viša elektrotehnička škola, Beograd, 2006., ISBN 86-85081-55-6

www.conwex.info/draganp/books.html





# Dodatna literatura



- **Applied Cryptography**
  Bruce Schneier
  John Wiley & Sons, 1995

- **Cryptography and Network Security**
  William Stallings
  Prentice Hall, 1998

- **The CISSP Prep Guide – Mastering the Ten Domains of Computer Security**
  Ronald L. Krutz, Russell Dean Vines
  John Wiley & Sons, 2001

  Druge knjige i razni *online* resursi

- ***Napomena:*** tokom predavanja će biti naglašena dodatna literatura, po potrebi.

# Pitanja



?