



Zaštita i sigurnost informacijskih sustava

Sigurnost programske podrške

prof. dr. sc. Krešimir Fertilj

Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Zaštićeno licencijom <http://creativecommons.org/licenses/by-nc-sa/2.5/hr/>



Creative Commons



- **slobodno smijete:**

- **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
- **remiksirati** — prerađivati djelo

- **pod sljedećim uvjetima:**

- **imenovanje.** Morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
- **nekomercijalno.** Ovo djelo ne smijete koristiti u komercijalne svrhe.
- **dijeli pod istim uvjetima.** Ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, prerađu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela. Najbolji način da to učinite je linkom na ovu internetsku stranicu.

Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.

Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

Tekst licencije preuzet je s <http://creativecommons.org/>.

Osnovni pojmovi

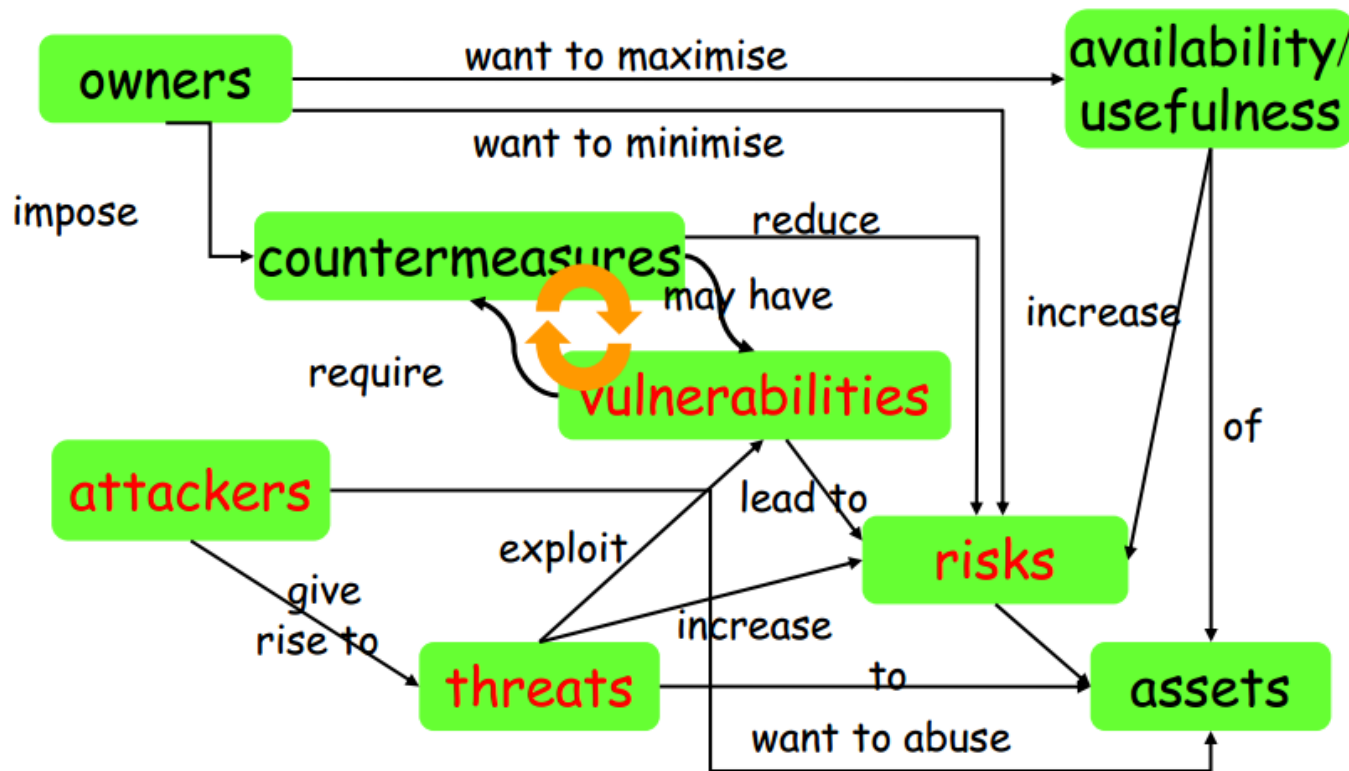
- ◆ Sigurnost programske podrške (software security)
 - Inženjerstvo softvera koji će pri napadu nastaviti ispravno raditi
 - *the science and study of protecting software (including data in software) against unauthorized access, modification, analysis or exploitation*
 - Software security = risk management
 - Management = administrative policies + patch security holes + testing + auditing
- ◆ Sigurnosna programska podrška (security software)
 - Računalni programi i knjižnice za potporu sigurnosti računala ili mreže
 - Antivirusni sw, kriptografski sw, vatrozid, sw za detekciju upada, sigurnosni dijelovi OS, ...
- ◆ software security ≠ security software
- ◆ Osiguranje softvera (software assurance)
 - Razina pouzdanosti da softver nema ranjivosti (bilo namjerne ili slučajne)

Sigurnost aplikacija

- ◆ Sigurnost aplikacije (application security)
 - **Mjere poduzete tijekom životnog ciklusa aplikacije** radi prevencije iznimki u odnosu na politiku sigurnosti aplikacije ili sustava uslijed pogrešaka u projektiranju, razvoju, ugradnji, nadogradnji ili održavanju aplikacije.
- ◆ Ključni pojmovi
 - Imovina, **sredstvo** (asset) – resurs
 - npr. podatci u bazi podataka/datoteci ili sistemski resursi
 - **Prijetnja** (threat) – opasnost, negativan učinak
 - **Povredivost, ranjivost** (vulnerability) – slabost koja omogućuje prijetnju
 - tj. koja napadaču dozvoljava smanjenje sigurnosti
 - **Napad** (attack, exploit) – akcija povrede sredstva
 - **Protumjera** (countermeasure) – mjera zaštite i ublažavanja rizika

Koncepti sigurnosti

- ◆ Svako razmatranje sigurnosti treba započeti
 - Inventurom dionika, resursa i prijetnji ...
 - od strane zaposlenika, klijenata, ... kriminalaca



Sigurnost kao softverski problem

- ◆ Kada je sigurnost softverski problem ?
 - ovisi o zahtijevanim promjenama
 - mrežni problem – zahtijeva promjenu mrežnih mehanizama, pr. mrežni protokoli
 - problem OS – zahtijeva promjenu mehanizama OS, pr. politika upravljanja resursima (resource management policy)
 - **softverski problem** – zahtijeva promjenu implementacije ili dizajna (softvera)

- ◆ Povećanje nesigurnosti
 - Povećanjem mrežne povezanosti sve više softvera može biti napadnuto !
 - Web aplikacije i preglednici – najslabija karika i predmet napada
 - Smanjenje razlike između OS, mreže i aplikacija
 - OS-like funkcionalnosti platformi Java i .NET
 - preglednik kao "OS" budućnosti ?

Uzroci problema softverske sigurnosti

- ◆ Glavni uzroci
 - nedostatak svijesti, značaja (awareness)
 - nedostatak znanja
- ◆ Sigurnost kao sekundarna briga
 - primarna je funkcionalnost, servis, udobnost
 - (truli) kompromis u kojem sigurnost gubi ...
- ◆ **Funkcionalnost** – ono što aplikacija radi
- ◆ **Sigurnost** – bavi se onim što aplikacija ne bi smjela raditi

Sigurnosni ciljevi : CIA

- ◆ **C**onfidentiality (povjerljivost, tajnost)
 - uskraćivanje “čitanja” neautoriziranim korisnicima
- ◆ **I**ntegrity (integritet, cjelovitost)
 - uskraćivanje promjena neautoriziranim korisnicima
- ◆ **A**vailability (dostupnost)
 - omogućavanje pristupa autoriziranim korisnicima, uskraćivanje ostalima
- ◆ Neporecivost odgovornosti (Non-repudiation for accountability)
 - autorizirani korisnici ne mogu odbiti, negirati, zaobići ugrađene postupke

Realizacija ciljeva: AAAA

♦ Autentifikacija (**authentication**)

- ovjera, utvrđivanje vjerodostojnosti, **provjera autentičnosti**
- proces identificiranja pojedinca, obično temeljen na korisničkim imenima i lozinkama, zasnovan na ideji da svaki pojedini korisnik ima nešto čime se razlikuje od ostalih korisnika
- provjera je li korisnik doista onaj kojim se predstavlja

♦ Autorizacija (**authorization**)

- **provjera ovlaštenosti**
- proces davanja ili odbijanja pristupa (access) resursima

♦ Nadzor, praćenje (**auditing**)

- provjera je li nešto pošlo krivo

♦ Djelovanje (**action**)

- ukoliko jest, poduzeti mjere

Gdje je tu zaštita ?

- ◆ Zaštita protiv napada?
 - *Anti-virus, intrusion detection, firewalls, etc.*
- ◆ Zaštita protiv prijetnji?
 - *Use forensics to find & eliminate*
 - *Mitigate by punishment, if possible*
- ◆ Zaštita protiv ranjivosti?
 - *Engineer secure software!*

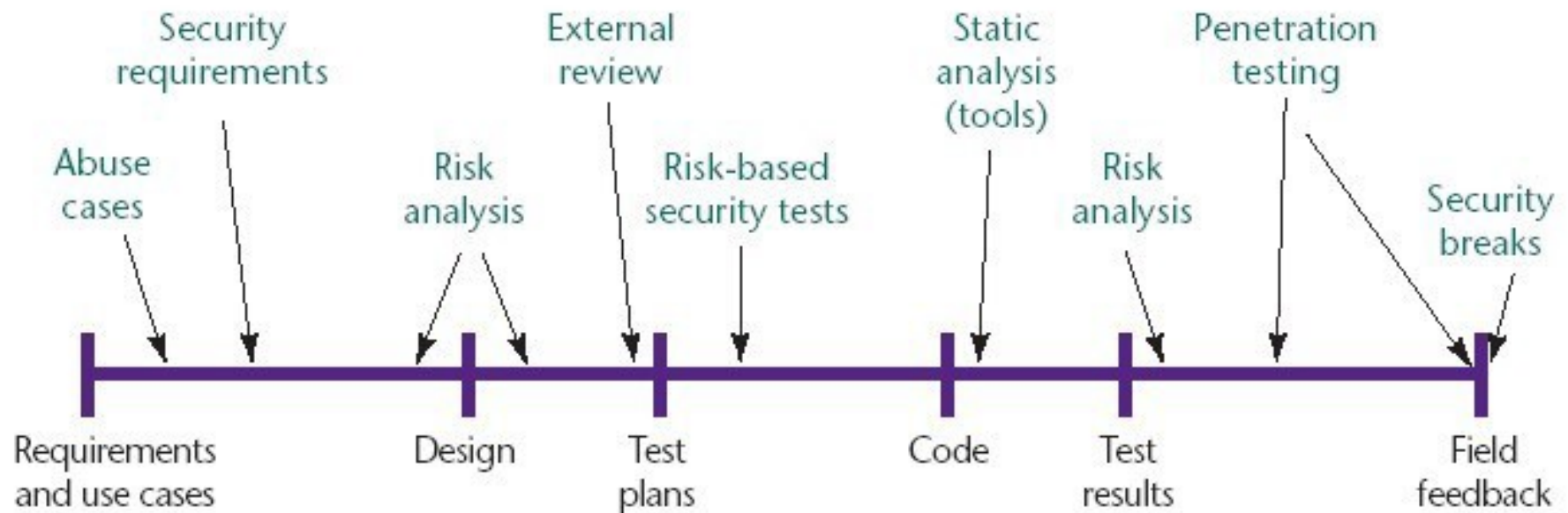
Životni ciklus sigurnog softvera

Secure Software Development Life Cycle

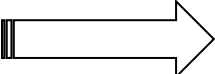
Životni ciklus sigurnog softvera

- ◆ Postupci, tehnike i metodologije
 - Sigurnost u životnom ciklusu
 - Inženjerski i projektantski principi
 - Sigurnosne tehnologije
- ◆ Pojednostavljeno:

[Source: Gary McGraw, Software security, Security & Privacy Magazine, IEEE, Vol 2, No. 2, pp. 80-83, 2004.]

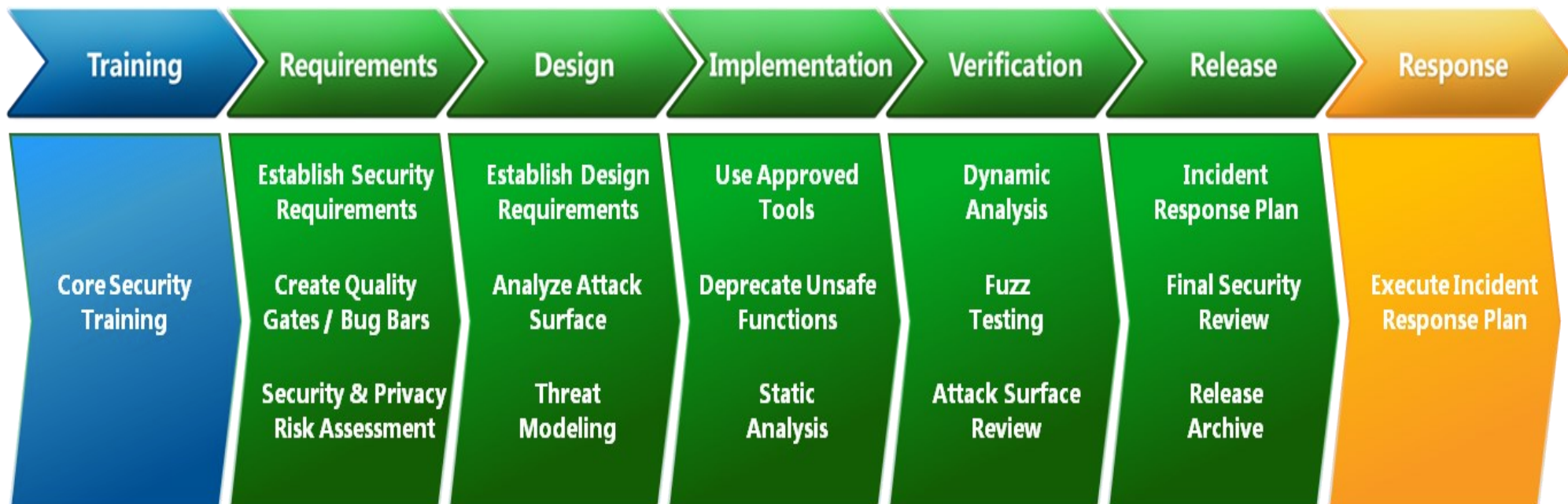


Modeli procesa životnog ciklusa sigurnog softvera

- ◆ Capability Maturity Models
 - CMMI for Development
- ◆ Team Software Process
 - TSP for Secure Software Development
- ◆ Correctness by Construction
- ◆ Common Criteria
- ◆ Software Assurance Maturity Model
- ◆ Building Security In – Maturity Model
 - Software Security Framework (SSF)
- ◆ **Microsoft's Trustworthy Computing Security Development LC**
 - **Životni ciklus razvoja sigurnosti (skraćeno SDL)** 

SDL aktivnosti - prakse

- ♦ aktivnosti prikazane prema tradicionalnom ciklusu razvoja softvera



- Analiza: sigurnosni zahtjevi, procjena rizika, ...
- Dizajn: modeliranje prijetnji, analiza površine napada, ...
- Implementacija: statička analiza, ...
- Verifikacija: dinamička analiza, *fuzz* testiranje, ...
- Isporuka: plan odgovora na incidente, finalni pregled

Pre-SDL Requirements: Security Training

◆ SDL Practice 1: Training Requirements

- poduka svih članova da bi znali osnove i ostali u trendu
- tehničari (razvojnici, tester, ...) – **barem jedan tečaj godišnje**

◆ Osnovni tečajevi, s temama (skraćeno)

- Sigurni dizajn (Secure design)
 - Attack surface reduction, Principle of least privilege, Secure defaults
- Modeliranje prijetnji (Threat modeling)
 - Overview, Design implications, Coding constraints
- Sigurno kodiranje (Secure coding)
 - Buffer overruns, Cross-site scripting, SQL injection, Weak cryptography
- Testiranje sigurnosti (Security testing)
 - Security and functional testing, Risk assessment, Security testing methods
- Privatnost (Privacy)
 - Types of privacy-sensitive data, design/development/testing best practices

◆ Napredni tečajevi - napredni dizajn, arhitektura, trusted GUI, ...

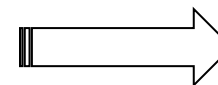
Phase One: Requirements

◆ SDL Practice 2: Security Requirements

- rano postavljanje pouzdanih (trustworthiness) zahtjeva
 - pri početnom planiranju
- identifikacija ključnih prekretnica (milestones) i isporuka
- specifikacija minimalnih zahtjeva na sigurnost aplikacija
- uspostava sustava za praćenje (vulnerability/work item tracking system)

◆ SDL Practice 3: Quality Gates/Bug Bars

- uspostava minimalno prihvatljivih razina kvalitete sigurnosti i privatnosti
- brana kvalitete (quality gate) – za svaku fazu
 - npr. ukloniti upozorenja kompilatora prije nego se napravi check-in
- prepreka za bugove (bug bar) – primjenjuje se na čitav projekt
 - npr. "bez poznatih kritičnih/važnih ranjivosti u trenutku isporuke"
- tim dokazuje sukladnost kroz Final Security Review (FSR)



Phase One: Requirements (nastavak)

- ◆ SDL Practice 4: Security and Privacy Risk Assessment
 - Security risk assessments (SRAs) and privacy risk assessments (PRAs)
- ◆ Procjene
 1. Dijelovi projekta koji zahtijevaju modeliranje prijetnji
 2. Dijelovi projekta koji zahtijevaju pregled dizajna
 3. Dijelovi projekta koji zahtijevaju penetracijsko testiranje
 4. Dodatno testiranje ili zahtjevi radi procjene rizika
 5. Doseg zahtjeva za *fuzz* testiranjem (pogledati praksu 12)
 6. Rangiranje utjecaja na privatnost (Privacy Impact Rating)
- ◆ Rang utjecaja (rizika) na privatnost
 - **P1** : visok – *feature*/proizvod/servis sprema ili prenosi osobne podatke, mijenja postavke ili instalira softver
 - **P2** : srednji – ponašanje koje se odnosi na privatnost je jednokratni, korisnički pokrenut prijenos podataka (npr. klik za odlazak na web)
 - **P3** : nizak – nema instalacije, promjena, prijenosa (kako je prethodno navedeno)

Phase Two: Design

◆ SDL Practice 5: Design Requirements

- što ranije uklanjanje problema sigurnosti i privatnosti
- izbjegavati "šarafljenje" ("bolting on") sigurnosti na kraju razvoja
- ◆ **razlikovati "secure features" i "security features" !**
 - sigurne mogućnosti – opća funkcionalnost koju treba osigurati (npr. unos, robusnost)
 - sigurnosne mogućnosti – f-nost koja se odnosi na sigurnost (npr. autentifikacija)
- ◆ Specifikacija dizajna treba
 - opisati mogućnosti softvera izravno izložene korisniku
 - opisati kako sigurno ugraditi funkcionalnost
- ◆ provjerava se naspram funkcionalne specifikacije koja
 - točno i potpuno opisuje korištenje mogućnosti
 - opisuje kako sigurno postaviti (deploy) *feature* ili funkciju

Phase Two: Design (nastavak)

◆ SDL Practice 6: Attack Surface Reduction

- redukcija rizika smanjenjem prostora za napad
- isključenjem ili restrikcijom pristupa na sistemske resurse
- primjenom principa najmanjeg prava (least privilege)
- uslojavanjem, gdje je moguće

◆ SDL Practice 7: Threat Modeling

- gdje postoji rizik sigurnosti
- razmatranje i dokumentiranje posljedica u planiranom operativnom okruženju
- razmatranje sigurnosti pojedinih komponenti ili aplikacije
- **glavna aktivnost dizajna u kojoj sudjeluju**
 - program/projekt menadžeri, razvojnici, testeri

Phase Three: Implementation

◆ SDL Practice 8: Use Approved Tools

- tim određuje alate - npr. kompilator/linker opcije, upozorenja
- savjetnik (advisor) odobrava
- tim treba ustrajati na zadnjim verzijama dokazanih alata (oprez !)

◆ SDL Practice 9: Deprecate Unsafe Functions

- analiza korištenih funkcija i API-ja s obzirom na sigurnost
- stvaranje liste "zabranjenih" (banned list)
- označavanje (npr. banned.h, strsafe.h)
- korištenje odgovarajućih opcija prevoditelja za provjeru ili posebnih alata
 - npr. opcija kompilatora /GS (Buffer Security Check), zaseban alat *StackGuard*

◆ SDL Practice 10: Static Analysis

- osigurava inspekciju programskog koda, ali ju ne može zamijeniti !
- npr. alati *StyleCop*, *CodeSmart*, *Ndepend/JDepend*

Phase Four: Verification

◆ SDL Practice 11: Dynamic Program Analysis

- pogonska (run-time) verifikacija koja utvrđuje da program radi kako je projektiran
- provjera korupcije memorije, korištenje privilegija, ...
- npr. *AppVerifier*, *ANTS profiler*, *Rational* ...

◆ SDL Practice 12: Fuzz Testing

- varijanta dinamičke analize kojom se
- nastoji izazvati zastoj unosom neispravnih ili pseudoslučajnih podataka

◆ SDL Practice 13: Threat Model and Attack Surface Review

- tokom razvoja dolazi do odstupanja od specifikacija
- ponovni pregled modela prijetnji i mjerenje površine napada
- verifikacija promjena u odnosu na specifikacije

Phase Five: Release

◆ SDL Practice 14: Incident Response Plan

- plan odziva na incidente definira
 - sustained engineering (SE) tim, ili emergency response plan (ERP) ako nema resursa
 - *on-call* kontakt koji ima autoritet odlučivanja, 24x7
 - plan servisiranja sigurnosti za izvana nabavljene komponente

◆ SDL Practice 15: Final Security Review (FSR)

- promišljena provjera svih sigurnosnih aktivnosti, prije objave
 - nije "penetrate and test" ili aktivnost "ugradimo zanemareno i zaboravljeno" !
- ishodi:
 - **passed FSR** – svi problemi su uočeni, te uklonjeni ili ublaženi
 - **passed FSR with exceptions** – nerazriješeni se evidentiraju i ispravljaju u narednoj objavi
 - **FSR with escalation** – projekt ne može biti objavljen, radi se plan razrješenja prije objave ili ide menadžmentu na daljnje odlučivanje

Phase Five: Release (*nastavak*)

◆ SDL Practice 16: Release/Archive

- *security advisor* potvrđuje (temeljem FSR i šire) da su zahtjevi zadovoljeni
- zasebno se potvrđuju komponente utjecaja na privatnost **P1 (praksa 4)**
- arhiviranje
 - specifikacija,
 - izvornog koda,
 - kompilata,
 - modela prijetnji,
 - dokumentacije,
 - planova odziva na incidente,
 - uvjeta licenciranja za nabavljene komponente,
 - ...

Opcionalne aktivnosti

- ◆ Nadzor, ručna inspekcija koda (code review)
 - vješti pojedinci ili sigurnosni tim ili savjetnik sigurnosti
 - usmjerena na "kritične" komponente
 - najčešće dijelova koji obrađuju ili pohranjuju osobne podatke
 - također dijelova koji se odnose na šifriranje

- ◆ Penetracijsko testiranje
 - *white box* analiza simuliranjem napada hakera
 - otkrivanje potencijalnih povredivosti uslijed pogreški u kodiranju, pogreški konfiguracije ili drugih slabosti u primjeni
 - u kombinaciji s automatiziranom ili ručnom analizom programskog koda

- ◆ Analiza povredivosti sličnih aplikacija
 - analizom dostupnih informacija na Internetu

RACI Chart – uloge (odgovoran, odobravatelj, savjetnik, informiran)

- ◆ RACI – akronim (Responsible, Accountable, Consulted, Informed)

Tasks	Architect	System Administrator	Developer	Tester	Security Professional
Security Policies		R		I	A
Threat Modeling	A		I	I	R
Security Design Principles	A	I	I		C
Security Architecture	A	C			R
Architecture and Design Review	R				A
Code Development			A		R
Technology Specific Threats			A		R
Code Review			R	I	A
Security Testing	C		I	A	C
Network Security	C	R			A
Host Security	C	A	I		R
Application Security	C	I	A		R
Deployment Review	C	R	I	I	A

Sigurnosni zahtjevi

Security Requirements

Sigurnosni zahtjevi

◆ Zahtjevi općenito

- Funkcionalni zahtjevi – opisuju što softver treba moći raditi
- Nefunkcionalni zahtjevi – sistemski, kvaliteta, ugovori, standardi, ograničenja

◆ **Sigurnosni – nefunkcionalni**

- Procjene vrijednosti sustava – vrijednost sustava i podataka
 - Ispad košta 50 kkn/h, gubitak podataka procjenjuje se na 20 Mkn
- Zahtjevi za kontrolu pristupa – ograničenje na pristup podacima
 - Voditelji mogu ..., operateri mogu ... ili anon/regi/admin mogu ...
- Zahtjevi za enkripcijom i autentifikacijom – kako, gdje i kada
- Zahtjevi za kontrolom virusa

◆ Neki mogu zahtijevati funkcionalnost

- Duljina korisničkog unosa, validacija podataka

Primjeri sigurnosnih zahtjeva

Scenarij	Zahtjev
Aplikacija pohranjuje osjetljive informacije koje treba štititi radi HIPAA usklađenosti.	Treba koristiti jaku enkripciju za zaštitu osjetljivih podataka.
Aplikacija prenosi osjetljive podatke o korisniku preko potencijalno nepouzdatih ili nesigurnih mreža.	Komunikacijski kanali moraju uključivati šifriranje kako bi se spriječilo njuškanje a kriptografskom autentifikacijom spriječiti <i>man-in-the-middle</i> napade.
Aplikacija podržava više korisnika s različitim razinama privilegija.	Treba definirati ovlaštenja za akcije na svakoj razini privilegija. Testirati različite razine.
Aplikacija pri unosu podataka koristi SQL	Definirati prevenciju SQL ubrizgavanja
Aplikacija je pisana u C/C++	Kontrolirati veličine međuspremnika, spriječiti modifikaciju formata upisa i preljev cijelih brojeva.
Podaci se prikazuju u HTMLu	Spriječiti XSS napade
Aplikacija zahtijeva praćenje promjena	Definirati funkcije praćenja. Osigurati dnevnik promjena.
Aplikacija koristi kriptografiju.	Treba koristiti sigurni generator pseudoslučajnih brojeva

Izvori zahtjeva

- ◆ Korisnici
- ◆ Sigurnosna implikacija funkcionalnosti
 - Zaštita od SQL ubrizgavanja za aplikacije nad BP
 - Zaštita od XSS ubrizgavanja za web aplikacije
- ◆ Regulatorna sukladnost
 - Zakon o informacijskoj sigurnosti
 - Zakon o zaštiti osobnih podataka
 - Federal Information Security Management Act (FISMA) – resursi vlade SAD
 - Sarbanes-Oxley (Sarbox ili SOX) – javna poduzeća u SAD
 - Health Insurance Portability & Accountability Act (HIPAA) – medicinski podaci

Postupci inženjerstva zahtjeva

- ◆ SQUARE
 - Security **Q**UAlity **R**equirements **E**ngineering Methodology from CMU/SEI
- ◆ TRIAD
 - Trustworth **R**efinement through Intrusion-**A**ware **D**esign from CMU/SEI
- ◆ ...
- ◆ SecureUML (UML, OCL), UMLintr, UMLsec
- ◆ ...
- ◆ Security Use Cases, Misuse Cases, Abuse Cases (MUCs)
 - Slučajevi korištenja, zloporabe (nenamjerno) ili zlostavljanja (namjerno)
 - scenariji u kojima sudionik kompromitira sustav

Slučajevi zloporabe

- ◆ Pogled protivnika/napadača
 - Dohvat podataka korisnika
 - Izmjena cijene, ocjene, ...
 - Uskraćivanje usluge
- ◆ Razvoj slučajevea
 - Brainstorming – pretpostavke, obrasci napada, rizici
- ◆ Sigurnosni zahtjevi – generalizirana forma MUCova
 - Anti-zahtjevi – što o**NE**moгуćiti

Primjer SZ

◆ **UC1:** Prijava u web trgovinu

- Primarni sudionik: Korisnik
- Dionici i interesi: Korisnik – želi kupiti proizvode
- Preduvjeti: Korisnik ima pristup webu
- Posljedice: Korisnik vidi svoj račun, ima mogućnost plaćanja i isporuke
- Sažetak: Korisnik pristupi sustavu putem korisničkog imena i lozinke

◆ **MUC1:** Njuškanje lozinke

- Primarni sudionik: Napadač
- Dionici i interesi: Napadač – želi dobiti korisničke vjerodajnice
- Preduvjeti: Napadač ima pristup stroju ili mrežnom putu do sustava
- Posljedice: Napadač je dobio jedan ili više ispravnih imena / lozinki
- Sažetak: Napadač dobiva i kasnije zlorabi neautorizirani pristup sustavu

Primjer MUC scenarija

◆ Osnovni tok:

1. Napadač instalira mrežno njuškalo
2. Njuškalo sprema pakete koji sadrže "Logon", "Username", "Password"
3. Napadač čita dnevnike njuškala
4. Napadač nalazi ispravan *login / password*
5. Napadač koristi nađeni *login / password* za pristup sustavu

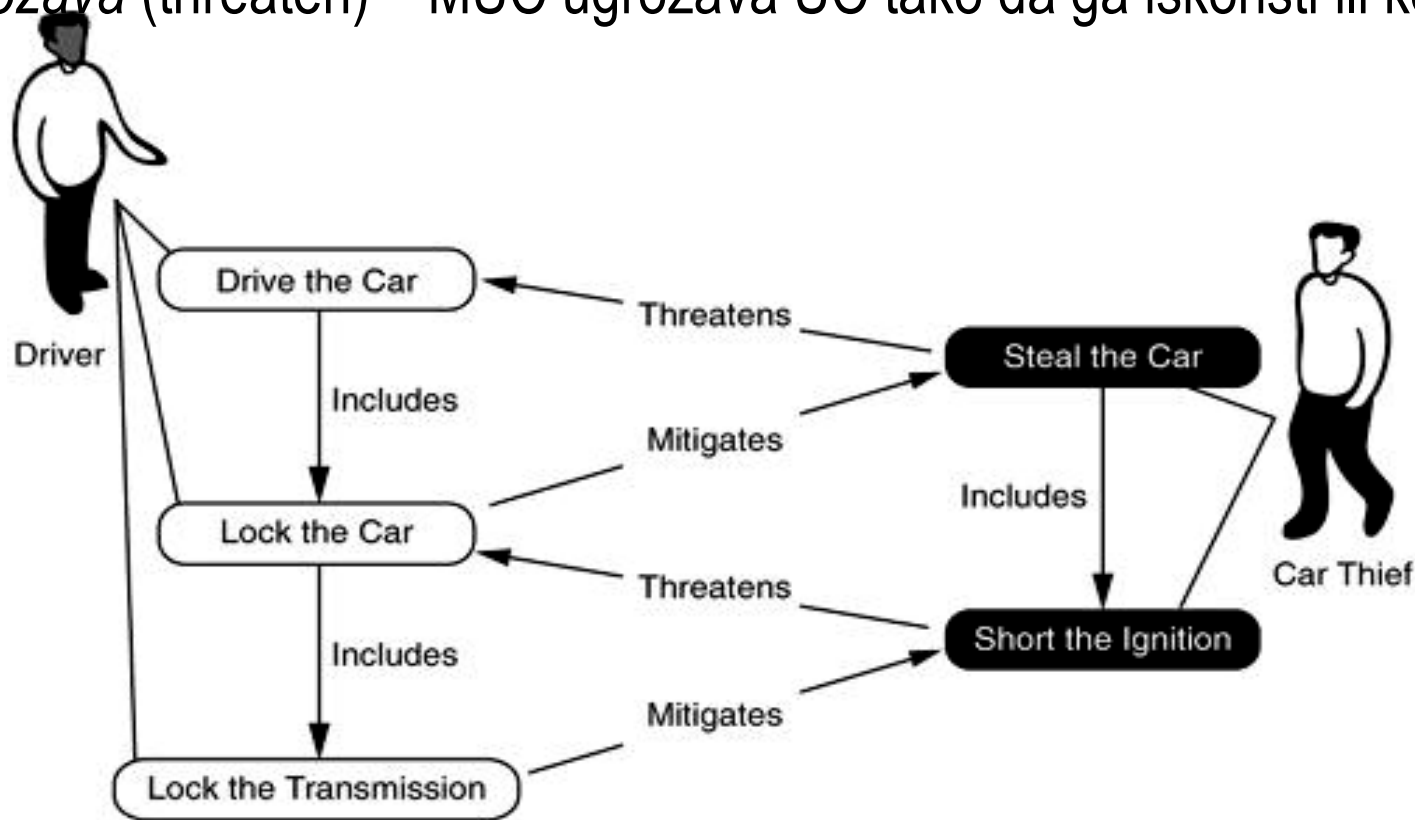
◆ Alternativni tokovi:

- 1a: Napadač nije na putu između korisnika i sustava
 - 1a1. Napadač koristi *ARP poisoning* ili slično da bi preusmjerio pakete
- 1b: Napadač koristi bežičnu konekciju
 - 1b1. Napadač odlazi na lokaciju korisnika
 - 1b2. Napadač koristi *wifi sniffer* za presretanje prometa

Povezivanje slučajeva zloporabe

◆ Proširenje dijagrama slučaja korištenja

- *Ublažava* (mitigate) – UC smanjuje priliku da MUC bude uspješan
- *Ugrožava* (threaten) – MUC ugrožava UC tako da ga iskoristi ili koči



Primjer: UC-MUC dijagram sigurne komunikacije

Regular User

Send Information in
Plaintext

includes

Encrypt all data and
Send the **Ciphertext**

extends

Embed the Ciphertext in
an Image and send the
StegoImage

Attacker

Hack the communication
Channel and read plaintext

Capture the ciphertext
and do cryptanalysis to
extract the plaintext

threaten

mitigate

threaten

mitigate

*Napadač treba znati stegoanalizu da
otkrije skriveni tekst pa još kriptonanalizu
da izdvoji čisti tekst*

Primjer: UC-MUC diagram za web forum

Regular User

Send a benign message
for posting to the Forum

includes

The message gets
posted to the Forum

extends

Sanitize the message for any
potential script to trigger
XSS attack and then post
to the Forum

includes

Administrator

Attacker

Send a Message loaded with
XSS Script to post to the Forum

threaten

mitigate



Alati za softversku sigurnost (ne mrežnu)

◆ Microsoft SDL i derivati



- [Attack Surface Analyzer](#) – smanjenje površine napada
- [Microsoft Threat Modeling Tool](#) – modeliranje prijetnji
- [MiniFuzz basic file fuzzing tool](#) – *fuzz* testiranje
- [Regular expression file fuzzing tool](#) – testiranje potencijalnih DoS ranjivosti

◆ Statička analiza

- StyleCop <https://stylecop.codeplex.com/> # slično, FxCop
- CodeSmart <http://www.axtools.com/>
- NDepend <http://www.ndepend.com/>
- PMD Java, Checkstyle, FindBugs+Find Security Bugs

Resursi

- ◆ Open Web Application Security Project (OWASP)
 - <http://www.owasp.org>, OWASP Top Ten vulnerabilities in web applications.
- ◆ Building Security In
 - <https://buildsecurityin.us-cert.gov/bsi/home.html>
- ◆ SANS Institute
 - <http://www.sans.org/> , CWE/SANS Top 25 Most Dangerous Prog. Errors
- ◆ CERT (Computer Security Incident Response Team)
 - <http://www.cert.org/> , <http://www.cert.org/secure-coding/>, <https://www.cert.hr>
- ◆ Cloud Security Alliance
 - <https://cloudsecurityalliance.org/>
- ◆ Ostalo
 - CVE (Common Vulnerabilities and Exposures), <http://cve.mitre.org/>
 - Security Tracker , <http://securitytracker.com/>
 - US-CERT Cyber Security Bulletins <http://www.us-cert.gov/cas/bulletins/>
 - Web Application Security Consortium (WASC), <http://www.webappsec.org/>
 - MSDN, <http://msdn.microsoft.com/security>

Reference

- ◆ Noopur Davis: Secure Software Development Life Cycle Processes, Software Engineering Institute, Carnegie Mellon University, 2013
 - http://resources.sei.cmu.edu/asset_files/whitepaper/2013_019_001_297287.pdf
- ◆ Microsoft SDL , <http://www.microsoft.com/security/sdl>
 - STRIDE, <http://msdn.microsoft.com/en-us/magazine/cc163519.aspx>
 - DREAD, <http://msdn.microsoft.com/en-us/library/ff648644.aspx>
 - SDL Quick Security References, <http://www.microsoft.com/en-us/download/details.aspx?id=13759>
- ◆ BSIMM Building Security In – Maturity Model, <http://bsimm.com>
- ◆ OpenSAMM Software Assurance Maturity Model, <http://opensamm.org>
- ◆ OWASP Application Threat Modeling
 - https://www.owasp.org/index.php/Application_Threat_Modeling

- ◆ Bruce Schneier, <https://www.schneier.com/>
 - If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology.
 - Unless you think like an attacker, you will be unaware of any potential threats!
 - You can't defend. You can't prevent. The only thing you can do is detect and respond.