
# Protection and security of information systems

# Software support security

**prof. Ph.D. Krešimir Fertalj**

**University of Zagreb**
**Faculty of Electrical Engineering and Computing**




# Zaštita i sigurnost informacijskih sustava

## Sigurnost programske podrške

**prof. dr. sc. Krešimir Fertalj**

**Sveučilište u Zagrebu**
**Fakultet elektrotehnike i računarstva**



# Creative Commons

- **you are free to:**

  ÿ **share** — duplicate, distribute and communicate the work to the public ÿ **remix** — rework the work **under the following**

- **conditions:** ÿ **naming.** You must acknowledge and mark the authorship of the work as it is specified by the author or licensor (but not in a way that suggests that you or your use of his work has his direct endorsement). ÿ **non-commercial.** You may not use this work for commercial purposes. ÿ **shares under the same conditions.** If you modify, reshape, or you create using it, you may distribute the adaptation only under a license that is the same or similar to this one.

In the case of further use or distribution, you must make clear to others the license terms of this work. The best way to do this is to link to this website.

Any of the above conditions may be waived with the permission of the copyright holder.

Nothing in this license infringes or limits the author's moral rights.

The text of the license was taken from http://creativecommons.org/.

# 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, preradu 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/.

# basic terms

- Software security
  - Software engineering that will continue to work properly in the event of an attack *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
- Security software
  - Computer programs and libraries to support computer or network security
    - Antivirus sw, cryptographic sw, firewall, intrusion detection sw, OS security parts, …
- software security security
- software Software assurance
  - Level of confidence that the software has no vulnerabilities (whether intentional or accidental)

# 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)

# Application security

ÿ Application security

ÿ Measures taken during the application life cycle to prevent exceptions to the application or system security policy due to errors in the design, development, installation, upgrade or maintenance of the application.

ÿ Key terms ÿ

Property, **asset** - resource ÿ
e.g. data in a database/file or system resources

ÿ **Threat** – danger, negative effect ÿ **Vulnerability**

– a weakness that enables a threat ÿ i.e. that allows an attacker to reduce security ÿ **Attack** (attack, exploit) – an action to violate a resource ÿ

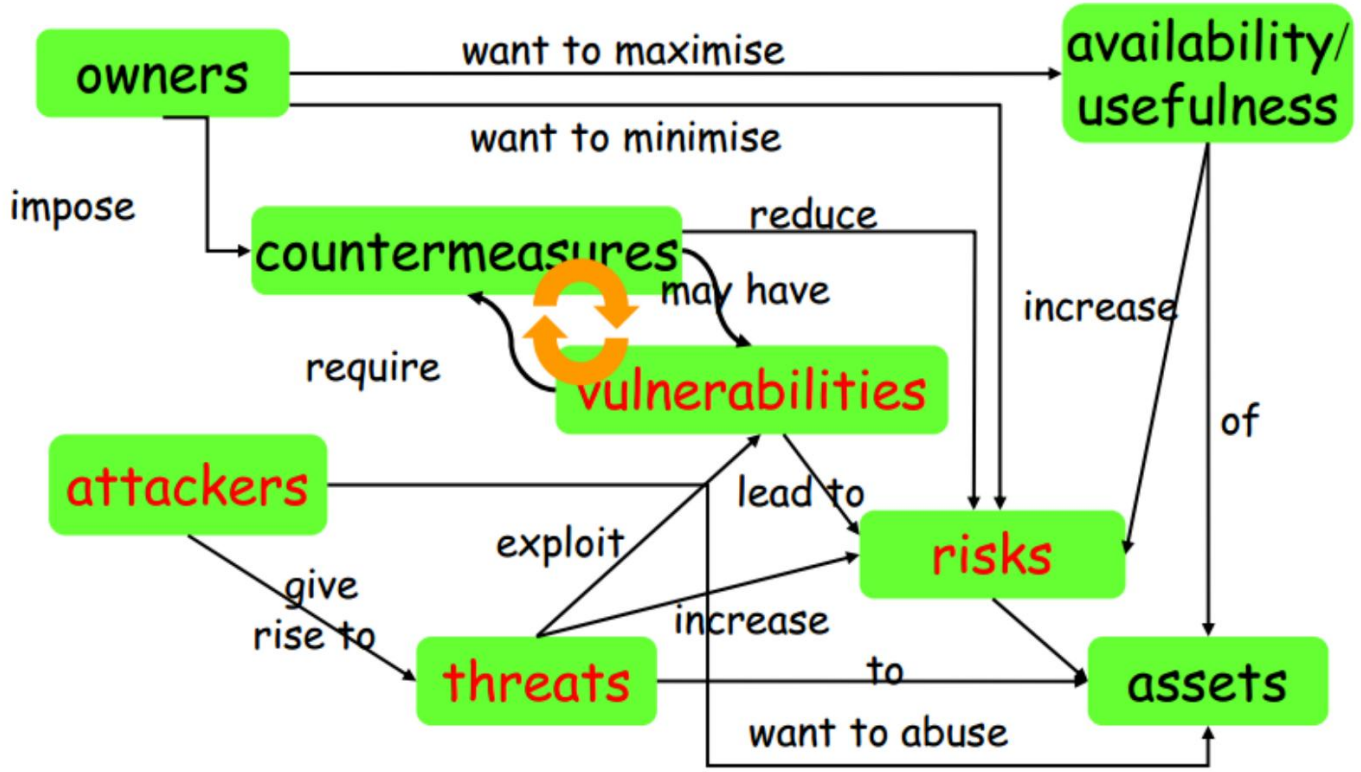
**Countermeasure** – a measure of protection and risk mitigation

# 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

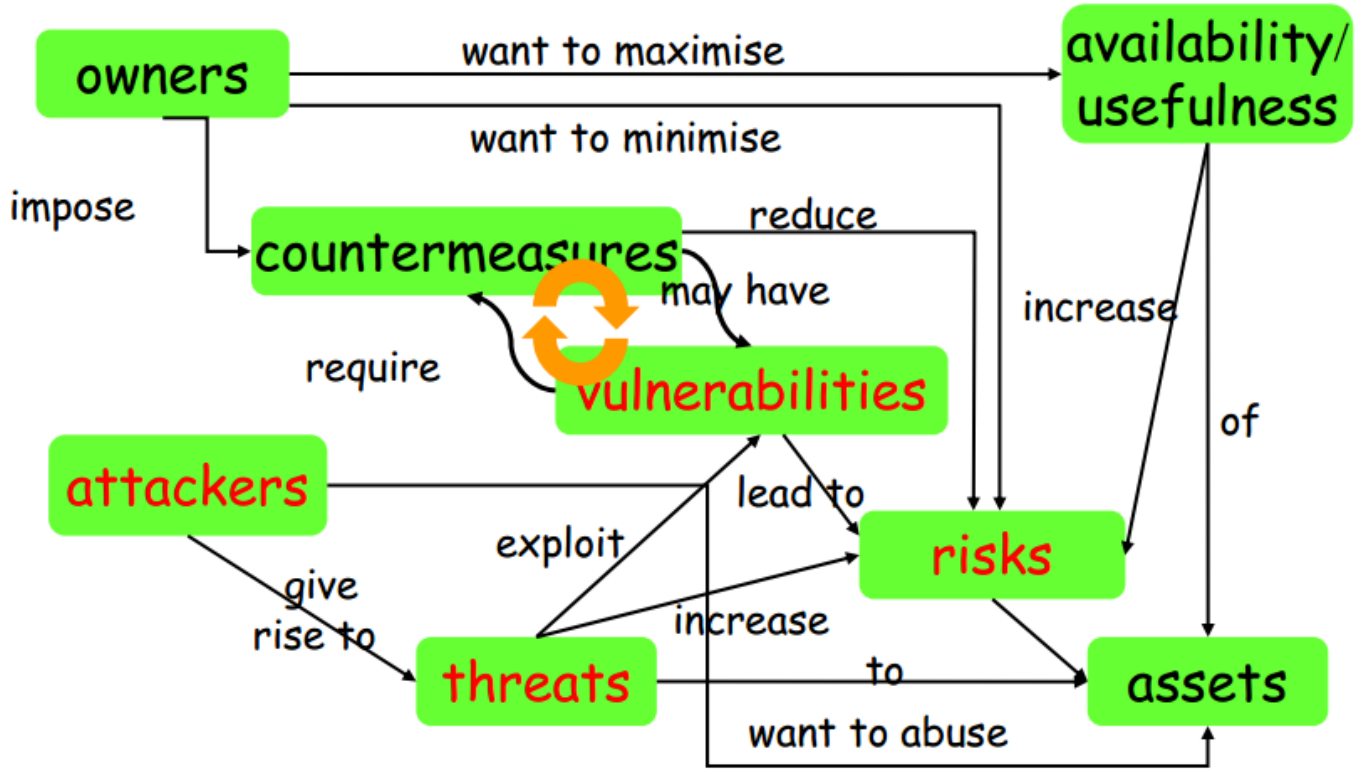
# Security concepts

ÿ Any security consideration should begin ÿ With an inventory of stakeholders, resources and threats … ÿ from employees, customers, … criminals



# Koncepti sigurnosti

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



# Security as a software problem

ÿ When is security a software problem ? ÿ

depends on the required changes ÿ

network problem – requires a change in network mechanisms, e.g. network

protocols ÿ OS problem – requires changing OS mechanisms, e.g. resource management p

(resource management policy)

ÿ software problem – requires a change in implementation or design (software)

ÿ Increasing insecurity ÿ By

increasing network connectivity, more and more software can be attacked! ÿ Web

applications and browsers - the weakest link and subject of attack

ÿ Reducing the difference between OS, network and applications

ÿ OS-like functionality of the Java and .NET platforms ÿ

browser as the "OS" of the future ?

# 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 ?

# Causes of software security problems

ÿ Main causes

ÿ lack of consciousness, significance

(awareness) ÿ lack of knowledge

ÿ Security as a secondary concern ÿ

primary is functionality, service, comfort ÿ

(rotten) compromise in which security loses...

ÿ **Functionality** – what the application does

ÿ **Security** – deals with what the application should not do

# 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

# Security targets: CIA

ÿ Confidentiality

ÿ denying "reading" to unauthorized users

ÿ Integrity (integrity,

completeness) ÿ denial of changes to unauthorized users

ÿ Availability ÿ

enabling access to authorized users, denying it to others

ÿ Non-repudiation for accountability

ÿ authorized users cannot refuse, negate, bypass built-in procedures

# 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

# Realization of goals: AAAA

ÿ Authentication **(authentication)** ÿ

verification, determination of credibility, authentication ÿ

process of identifying an individual, usually based on usernames and passwords,

based on the idea that each individual user has something that differentiates them from other users

ÿ checking whether the user is really who he is

ÿ Authorization ÿ **authorization**

check ÿ process of granting or

denying access to resources

ÿ Supervision, monitoring **(auditing)**

ÿ check if something went wrong

ÿ Action **(action)**

ÿ if it is, take measures

# 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

# Where is the protection?

ÿ Protection against attacks?

- ÿ *Anti-virus, intrusion detection, firewalls, etc.*

ÿ Protection against threats?

- ÿ *Use forensics to find & eliminate*
- ÿ *Mitigate by punishment, if possible*

ÿ Protection against vulnerabilities?

*Engineer secure software!*

# 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!*

# The lifecycle of secure software

Secure Software Development Life Cycle

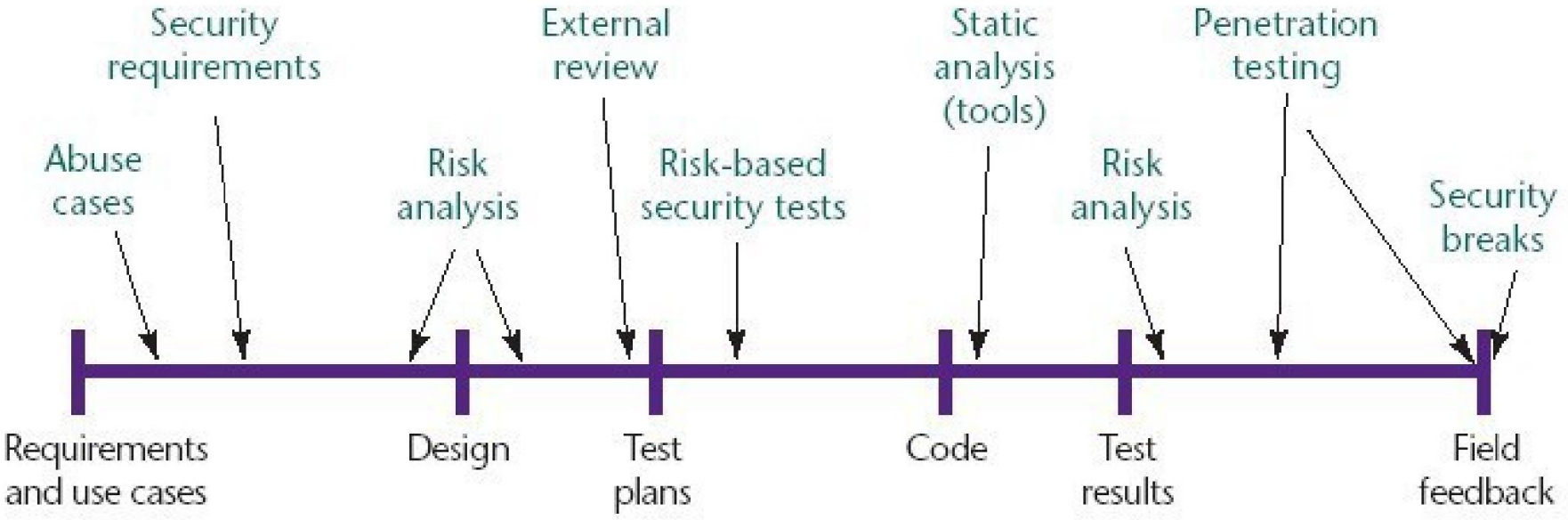
# Životni ciklus sigurnog softvera

Secure Software Development Life Cycle

# The lifecycle of secure software

ÿ Procedures, techniques and methodologies ÿ Safety in the life cycle ÿ Engineering and design principles ÿ Safety technologies ÿ Simplified:

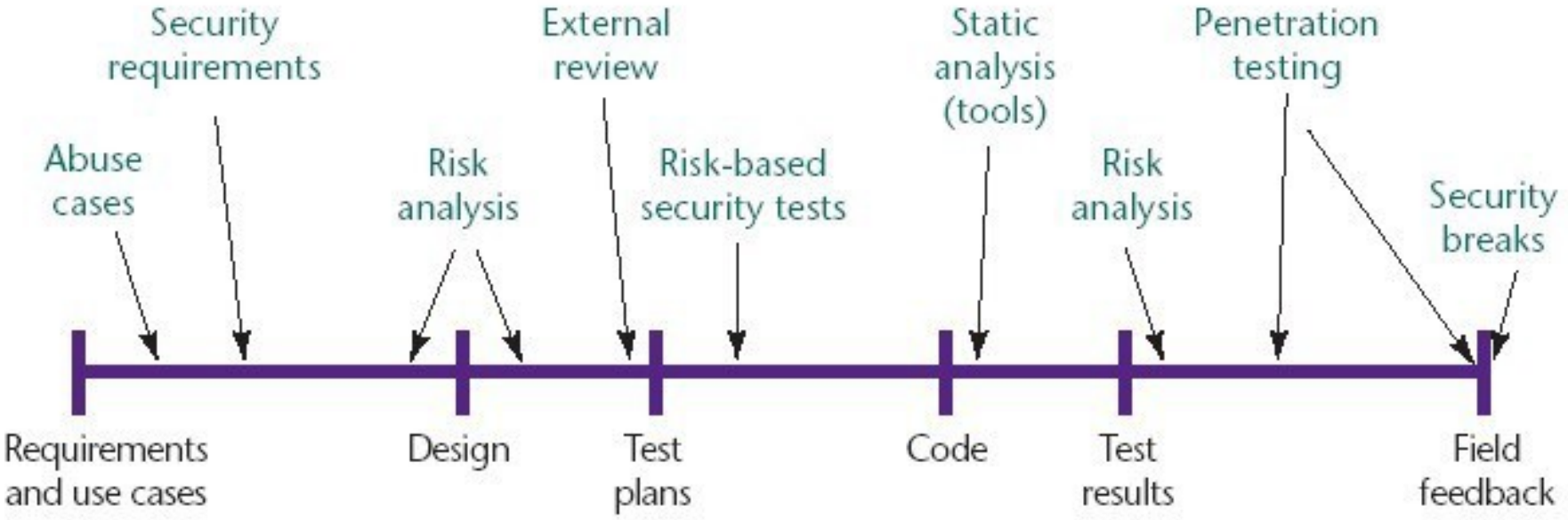
[Source: Gary McGraw, Software security, Security & Privacy Magazine, IEEE, Vol 2, No. 2, pp. 80-83, 2004. ]



# Životni ciklus sigurnog softvera

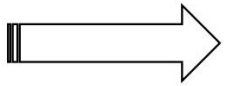
- Postupci, tehnike i metodologije
  - Sigurnost u životnom ciklusu
  - Inženjerski i projektantski principi
  - Sigurnosne tehnologije
- Pojednostavljeno:

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



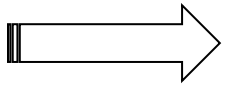
# Process models of the secure software life cycle

ÿ Capability Maturity Models ÿ
CMMI for Development ÿ
Team Software Process ÿ
TSP for Secure Software Development
ÿ Correctness by Construction ÿ Common
Criteria ÿ Software Assurance Maturity
Model ÿ Building Security In – Maturity
Model ÿ Software Security Framework
(SSF)
ÿ **Microsoft's Trustworthy Computing Security Development LC** ÿ
Security Development **Life Cycle (abbreviated SDL)**



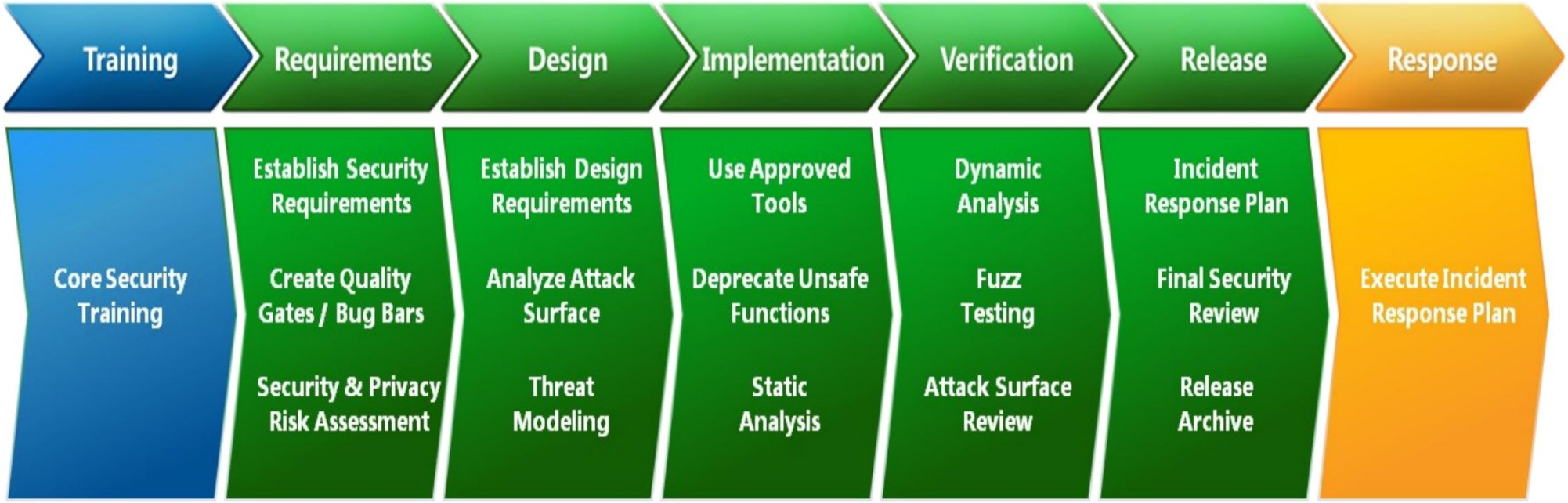
# Modeli procesa životnog ciklusa sigurnog softvera

- 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 activities - practices

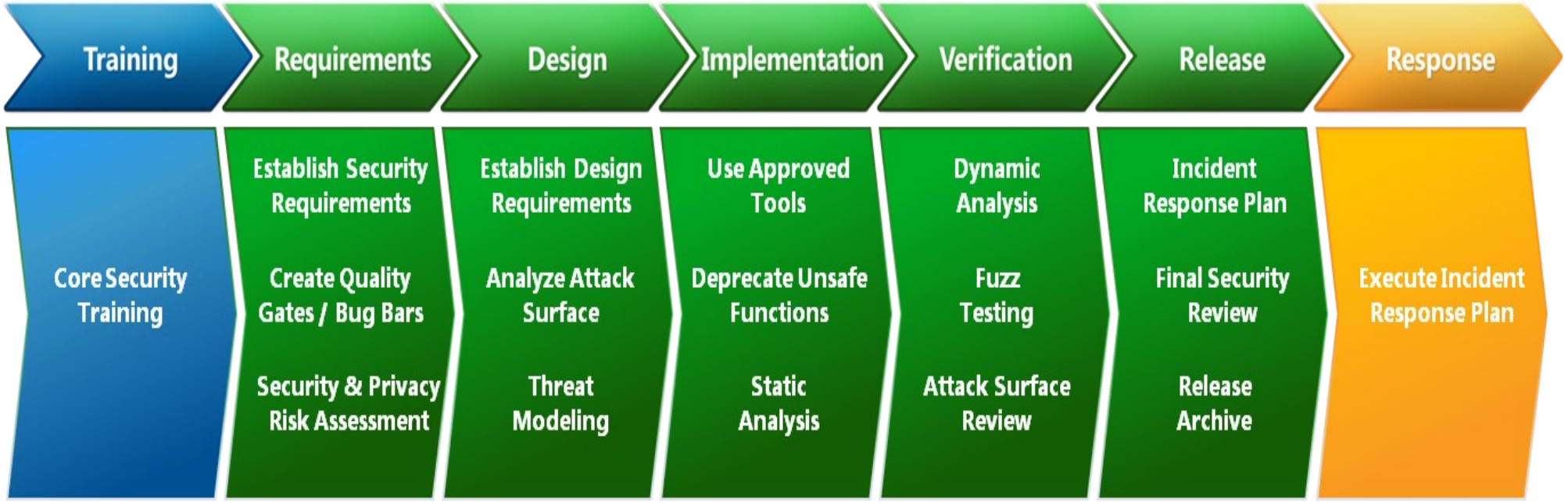
ÿ activities shown according to the traditional software development cycle




ÿ Analysis: security requirements, risk assessment, … ÿ Design: threat modeling, attack surface analysis, … ÿ Implementation: static analysis, … ÿ Verification: dynamic analysis, *fuzz* testing, … ÿ Delivery: incident response plan, final review

# 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

ÿ SDL Practice 1: Training Requirements

- ÿ training of all members to know the basics and stay on trend
- ÿ technicians (developers, testers, ...) - at least one course per year

ÿ Basic courses, with topics (abbreviated)

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

ÿ Advanced courses - advanced design, architecture, trusted GUI, ...

# *Pre-SDL Requirements: Security Training*

- SDL Practice 1: Training Requirements
  - poduka svih članova da bi znali osnove i ostali u trendu
  - tehničari (razvojnici, testeri, …) – 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, …

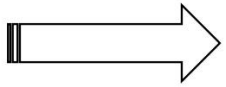
# *Phase One: Requirements*

## ÿ SDL Practice 2: Security Requirements

ÿ early setting of trustworthiness requirements
ÿ during initial planning
ÿ identification of key milestones (milestones) and delivery ÿ
specification of minimum application security requirements ÿ
establishment of a monitoring system (vulnerability/work item tracking system)

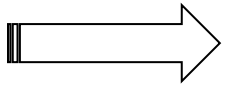
## ÿ SDL Practice 3: Quality Gates/Bug Bars

ÿ establishment of minimum acceptable quality levels of security and privacy
ÿ quality gate (quality gate) – for each phase ÿ e.g. remove compiler warnings
before check-in ÿ barrier for bugs (bug bar) – applies to the entire project
ÿ e.g. "no known critical/important vulnerabilities at time of delivery" ÿ team
proves compliance through Final Security Review (FSR)



# *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 (continued)*

ÿ SDL Practice 4: Security and Privacy Risk Assessment

ÿ Security risk assessments (SRAs) and privacy risk assessments (PRAs) ÿ Assessments 1. Project parts that require threat modeling 2. Project parts that require design review 3. Project parts that require penetration testing 4. Additional testing or risk assessment requirements 5 Scope of *fuzz* testing requirements (see Practice 12)

6. Ranking of impact on privacy (Privacy Impact Rating)

ÿ Rank of impact (risk) on privacy ÿ **P1 :** high – feature/ product/service saves or transfers personal data, changes settings or install software

ÿ **P2 :** medium – privacy-related behavior is a one-time, user- initiated data transfer (eg click to go to the web)

ÿ **P3 :** low – no install, change, transfer (as previously stated)

# *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*

ÿ SDL Practice 5: Design Requirements

ÿ removal of security and privacy problems as early as possible ÿ avoid "bolting on" of security at the end of development

ÿ distinguish between "secure features" and "security features" !

ÿ secure capabilities – general functionality to be ensured (e.g. input, robustness) ÿ security capabilities – security-related functionality (e.g. authentication)

ÿ The design specification should

ÿ describe the software capabilities directly exposed to the user ÿ describe how to safely incorporate functionality ÿ

be checked against a functional specification that ÿ accurately and completely describes the use of capabilities ÿ describe how to safely deploy (deploy) a *feature* or function

# *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 (continued)*

## ÿ SDL Practice 6: Attack Surface Reduction

ÿ risk reduction by reducing the space for attack ÿ by

excluding or restricting access to system resources ÿ by applying the

principle of least privilege ÿ by layering, where possible

## ÿ SDL Practice 7: Threat Modeling

ÿ where there is a security risk ÿ

consideration and documentation of the consequences in the planned operating environment

ÿ consideration of the security of individual components or applications ÿ the main design activity in which they participate

ÿ program/project managers, developers, testers

# *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*

ÿ SDL Practice 8: Use Approved Tools ÿ

team determines tools - eg compiler/linker options, warnings

ÿ advisor approves ÿ team should stick to latest versions of proven tools (caution!)

ÿ SDL Practice 9: Deprecate Unsafe Functions

ÿ analysis of used functions and APIs with regard to security ÿ creation of "banned" list (banned list) ÿ marking (eg banned.h, strsafe.h) ÿ use of appropriate checking compiler options or special tools ÿ eg compiler options /GS (Buffer Security Check), a separate *StackGuard* tool

ÿ SDL Practice 10: Static Analysis

ÿ provides code inspection, but cannot replace it! ÿ eg *StyleCop, CodeSmart, Ndepend/ JDepend tools*

# *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

ÿ SDL Practice 11: Dynamic Program Analysis

ÿ drive (run-time) verification that determines that the program works as designed
ÿ check for memory corruption, use of privileges, ... ÿ e.g. *AppVerifier, ANTS profiler, Rational ...*

ÿ SDL Practice 12: Fuzz Testing

ÿ a variant of dynamic analysis that ÿ tries to
cause a deadlock by entering incorrect or pseudo-random data

ÿ SDL Practice 13: Threat Model and Attack Surface Review

ÿ during development there are deviations from the
specifications ÿ review of the threat model and measurement of the
attack surface ÿ verification of changes in relation to the specifications

# *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*

ÿ SDL Practice 14: Incident Response Plan ÿ

the incident response plan defines

ÿ sustained engineering (SE) team, or emergency response plan (ERP) if there are no resources ÿ *on-call* contact with decision authority, 24x7 ÿ safety service plan for outsourced components

ÿ SDL Practice 15: Final Security Review (FSR)

ÿ thoughtful verification of all security activities, before publication

ÿ it is not "penetrate and test" or "let's incorporate the neglected and forgotten" activity!

ÿ outcomes: ÿ **passed FSR** – all problems were noticed and removed or mitigated ÿ **passed**

**FSR with exceptions** – unresolved issues are recorded and corrected in the next announcement ÿ **FSR with escalation** – the project cannot be announced, a solution plan is made before the announcement or goes to management for further decision

# *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 (continued)*

## ÿ SDL Practice 16: Release/Archive

ÿ *security advisor* confirms (based on FSR and beyond) that the requirements are met

ÿ privacy impact components **P1 are separately confirmed (practice 4)** ÿ archiving

ÿ specifications, ÿ source code, ÿ compilations, ÿ threat models, ÿ documentation, ÿ response plans to incidents, ÿ licensing conditions for purchased components,

ÿ …

# *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,
    - …

# Optional activities

ÿ Supervision, manual code inspection (code review)

ÿ skilled individuals or security team or security consultant ÿ focused on "critical" components ÿ most often parts that process or store personal data ÿ also parts related to encryption

ÿ Penetration testing ÿ *white box*

analysis by simulating hacker attacks ÿ detection of potential vulnerabilities due to coding errors, errors configuration or other weaknesses in the application

ÿ in combination with automated or manual analysis of program code

ÿ Vulnerability analysis of similar applications

ÿ by analyzing available information on the Internet

# 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 – roles (responsible, approver, advisor, informed)

ÿ RACI - acronym (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 |

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

- RACI – akronim (Responsable, 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 |

# Security requirements

Security Requirements

# Sigurnosni zahtjevi

Security Requirements

# Security requirements

ÿ Requirements in general

ÿ Functional requirements - describe what the software should be able to do ÿ Non-functional requirements - system, quality, contracts, standards, restrictions

ÿ **Security - non-functional** ÿ Estimates of system value - system and data value ÿ Outage costs 50 kk/h, data loss is estimated at 20 Mkn

ÿ Access control requirements – restriction on data access ÿ Managers can …, operators can … or anon/regi/admin can … ÿ Encryption and authentication requirements – how, where and when ÿ Virus control requirements ÿ Some may require functionality

ÿ Length of user input, data validation

# 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

# Examples of security requirements

| Scenario | |
|---|---|
| An application stores sensitive information that needs to be protected for HIPAA compliance. | **Requirement** Strong encryption should be used to protect sensitive data. |
| The application transmits sensitive user data over potentially untrusted or insecure networks. | Communication channels must include encryption to prevent snooping and cryptographic authentication to prevent *man-in-the-middle* attacks. |
| The application supports multiple users with different privilege levels. | Action authorizations at each privilege level should be defined. Test different levels. |
| The application uses SQL for data entry. The | Define SQL injection prevention Control buffer |
| application is written in C/C++ | sizes, prevent write format modification and integer overflow. |
| The data is displayed in HTML | Prevent XSS attacks |
| The application requires change tracking | Define tracking functions. Provide a change log. |
| The application uses cryptography. | A secure pseudo-random number generator should be used |

# 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 nepouzdanih ili nesigurnih mreža. | Komunikacijski kanali moraju uključivati šifriranje kako bi se spriječilo njuškanje a kriptografskom autentifikacijom spriječiti *man-in-the-middle* 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 |

# Request sources

ÿ Users

ÿ Security implication of functionality

- ÿ Protection against SQL injection for applications over BP
- ÿ Protection against XSS injection for web applications

ÿ Regulatory compliance ÿ

Information Security Act ÿ Personal

Data Protection Act ÿ Federal Information

Security Management Act (FISMA) – US government resources ÿ Sarbanes-Oxley

(Sarbox or SOX) – US public companies ÿ Health Insurance Portability & Accountability

Act (HIPAA ) – medical data

# 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

# Requirements engineering procedures

ÿ SQUARE ÿ

Security QUAlity Requirements Engineering Methodology from CMU/SEI

ÿ TRIAD

ÿ Trustworth Refinement through Intrusion-Aware Design from CMU/SEI

ÿ ...

ÿ SecureUML (UML, OCL), UMLintr, UMLsec

ÿ ...

ÿ Security Use Cases, Misuse Cases, Abuse Cases (MUCs)

ÿ Cases of use, abuse (unintentional) or abuse (intentional) ÿ scenarios where a participant compromises the system

A systematic review of security requirements engineering, Mellado et.a., 2010

# Postupci inženjerstva zahtjeva

- SQUARE
  - **S**ecurity **QUA**lity **R**equirements **E**ngineering Methodology from CMU/SEI
- TRIAD
  - **T**rustworth **R**efinement through **I**ntrusion-**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

A systematic review of security requirements engineering, Mellado et.a., 2010

# Cases of abuse

- View of the adversary/attacker
  - Access to user data
  - Change of price, rating, …
  - Denial of service
- Case development
  - Brainstorming – assumptions, attack patterns, risks
- Security requirements – generalized form of MUCs
  - Anti-requests – what NOT to disable

# Slučajevi zloporabe

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

ÿ **UC1:** Login to the web store ÿ Primary participant: User

ÿ Stakeholders and interests: User - wants to buy products ÿ Prerequisites: User has access to the web

ÿ Consequences: User sees his account, can pay and deliver ÿ Summary: User accesses the system via username and password

ÿ **MUC1:** Password sniffing

ÿ Primary participant: Attacker ÿ Stakeholders and interests: Attacker - wants to obtain user credentials ÿ Prerequisites: Attacker has access to the machine or network path to the system ÿ Consequences: Attacker has obtained one or more valid usernames / passwords ÿ Summary: Attacker obtains and later abuses unauthorized system access

# 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 dobaviti korisničke vjerodajnice
  - Preduvjeti: Napadač ima pristup stroju ili mrežnom putu do sustava
  - Posljedice: Napadač je dobavio jedan ili više ispravnih imena / lozinki
  - Sažetak: Napadač dobavi i kasnije zlorabi neautorizirani pristup sustavu

## An example of a MUC scenario

### ÿ Basic flow:

1. The attacker installs a network sniffer 2.

The sniffer saves packets containing "Logon", "Username", "Password"

3. The attacker reads the sniffer logs

4. The attacker finds the correct *login / password*

5. The attacker uses the found *login / password* to access the system

### ÿ Alternative streams:

1a: The attacker is not in the path between the user and

the system 1a1. An attacker uses *ARP poisoning* or similar to redirect packets

1b: The attacker uses a wireless connection

1b1. The attacker goes to the user's location

1b2. The attacker uses a *wifi sniffer* to intercept traffic

# 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

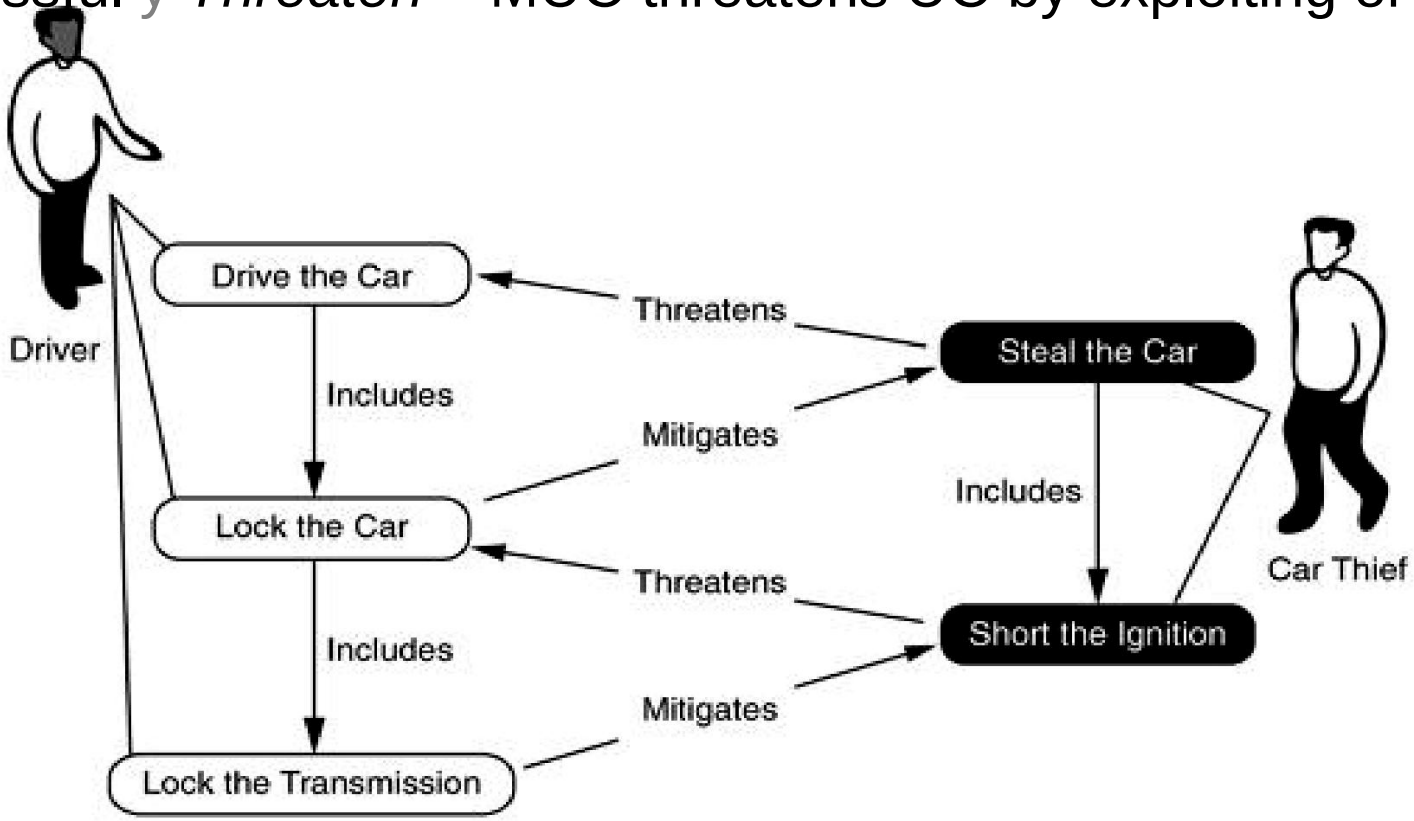
1b: Napadač koristi bežičnu konekciju

1b1. Napadač odlazi na lokaciju korisnika

1b2. Napadač koristi *wifi sniffer* za presretanje prometa

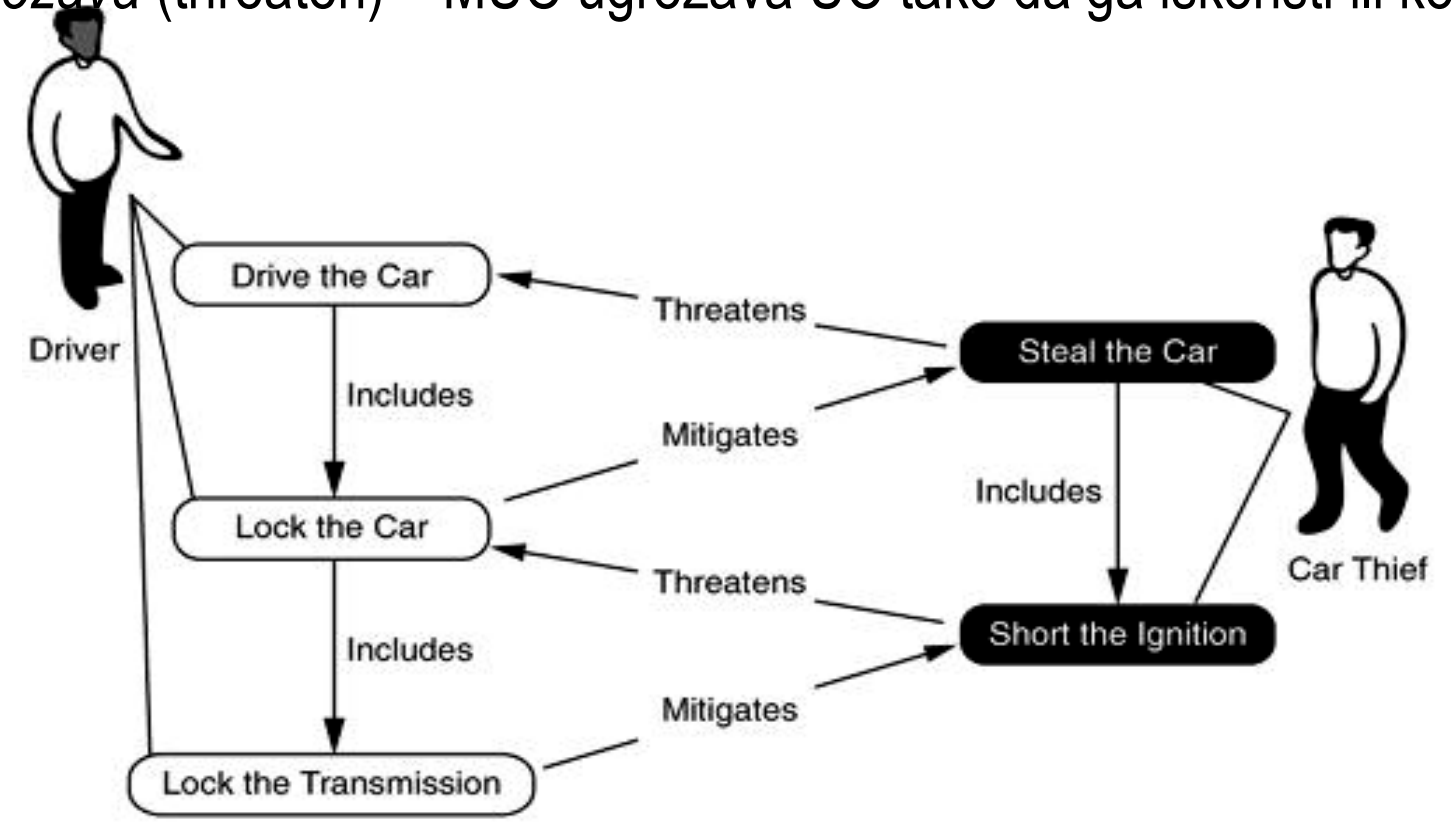
# Linking cases of abuse

- Extending the use case diagram
  - *Mitigate* – UC reduces the chance for MUC to be successful
  - *Threaten* – MUC threatens UC by exploiting or inhibiting it

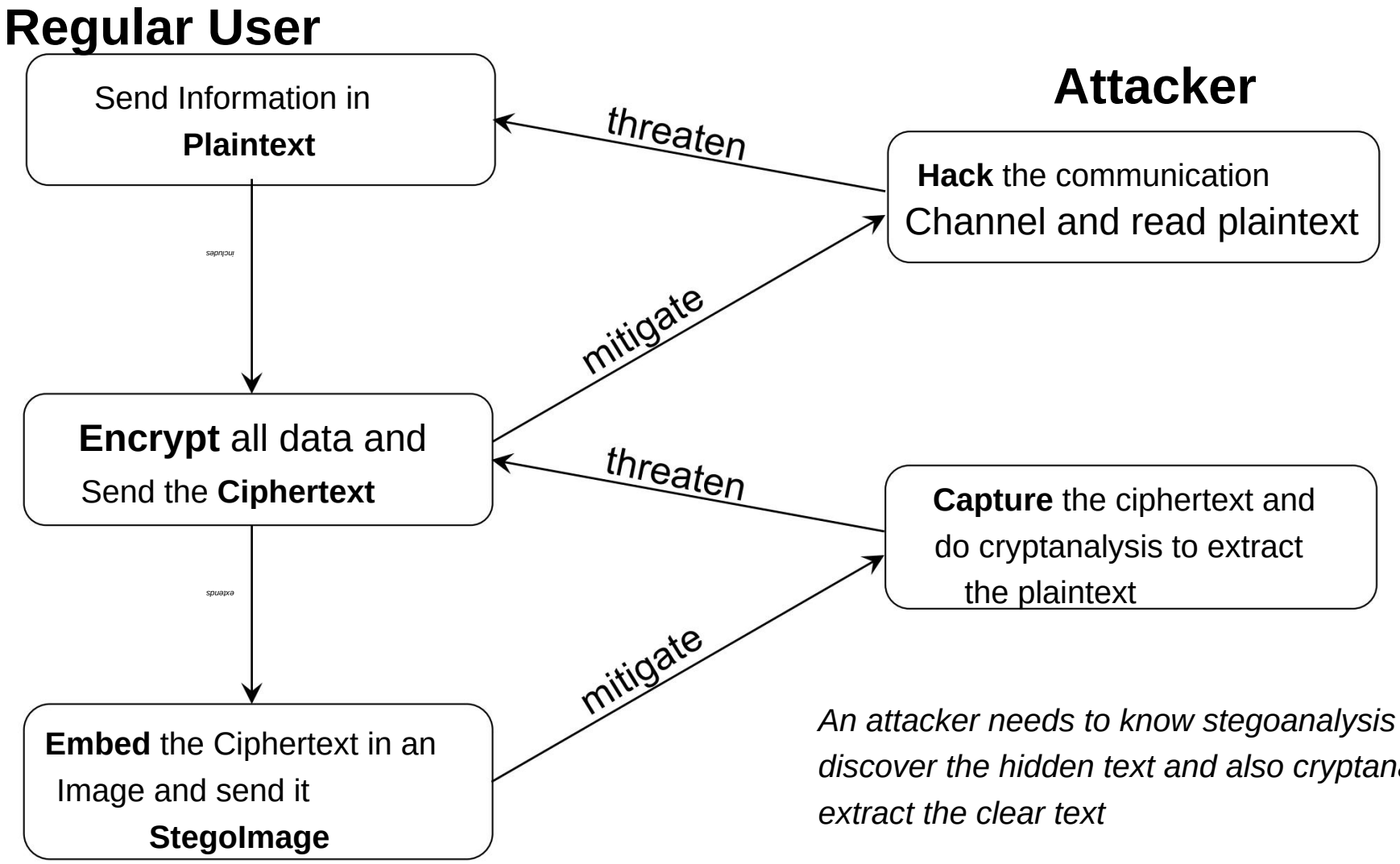


# Povezivanje slučajeva zloporabe

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

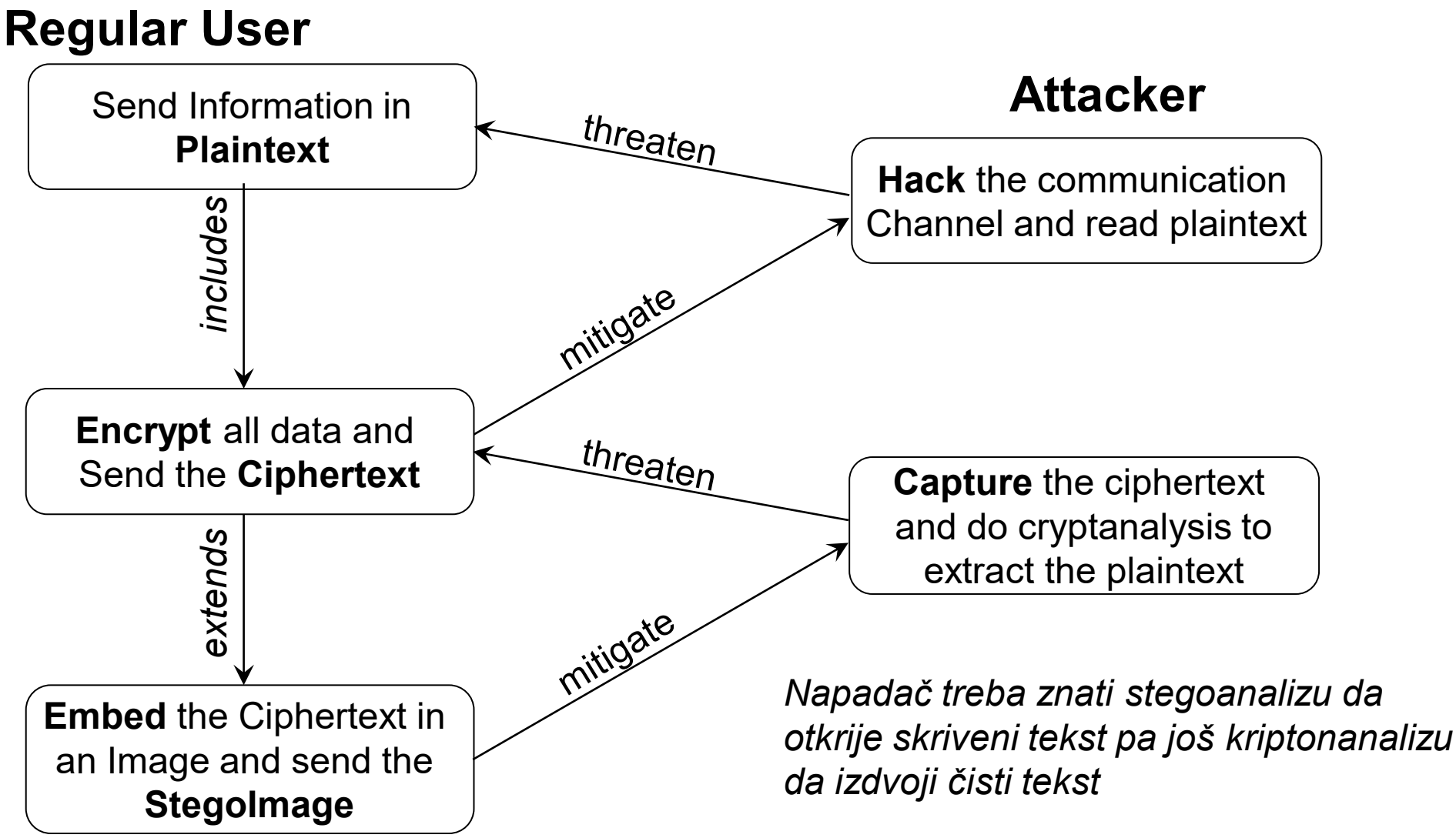


# Example: UC-MUC secure communication diagram



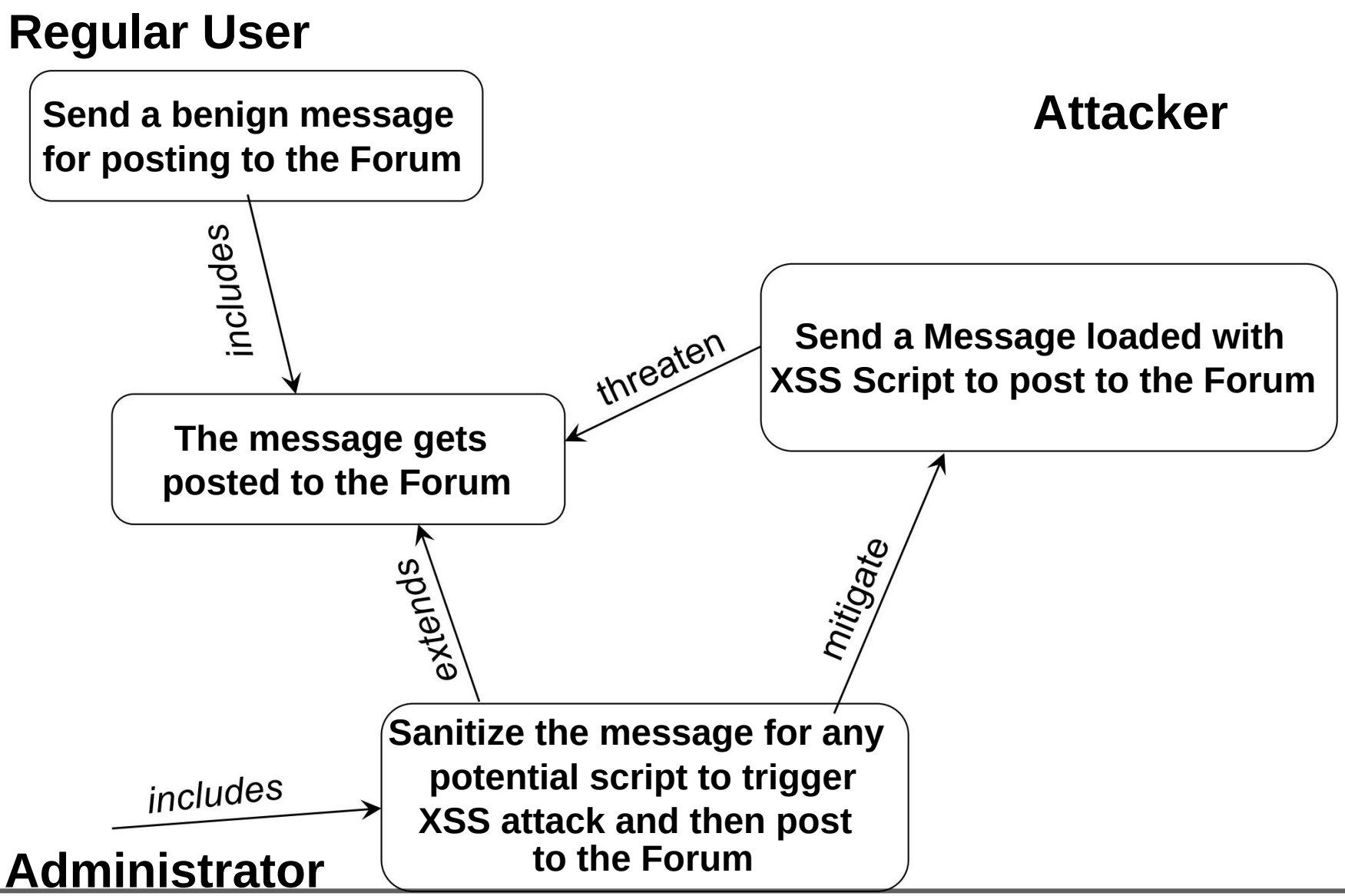

*An attacker needs to know stegoanalysis to discover the hidden text and also cryptanalysis to extract the clear text*

# Primjer: UC-MUC dijagram sigurne komunikacije

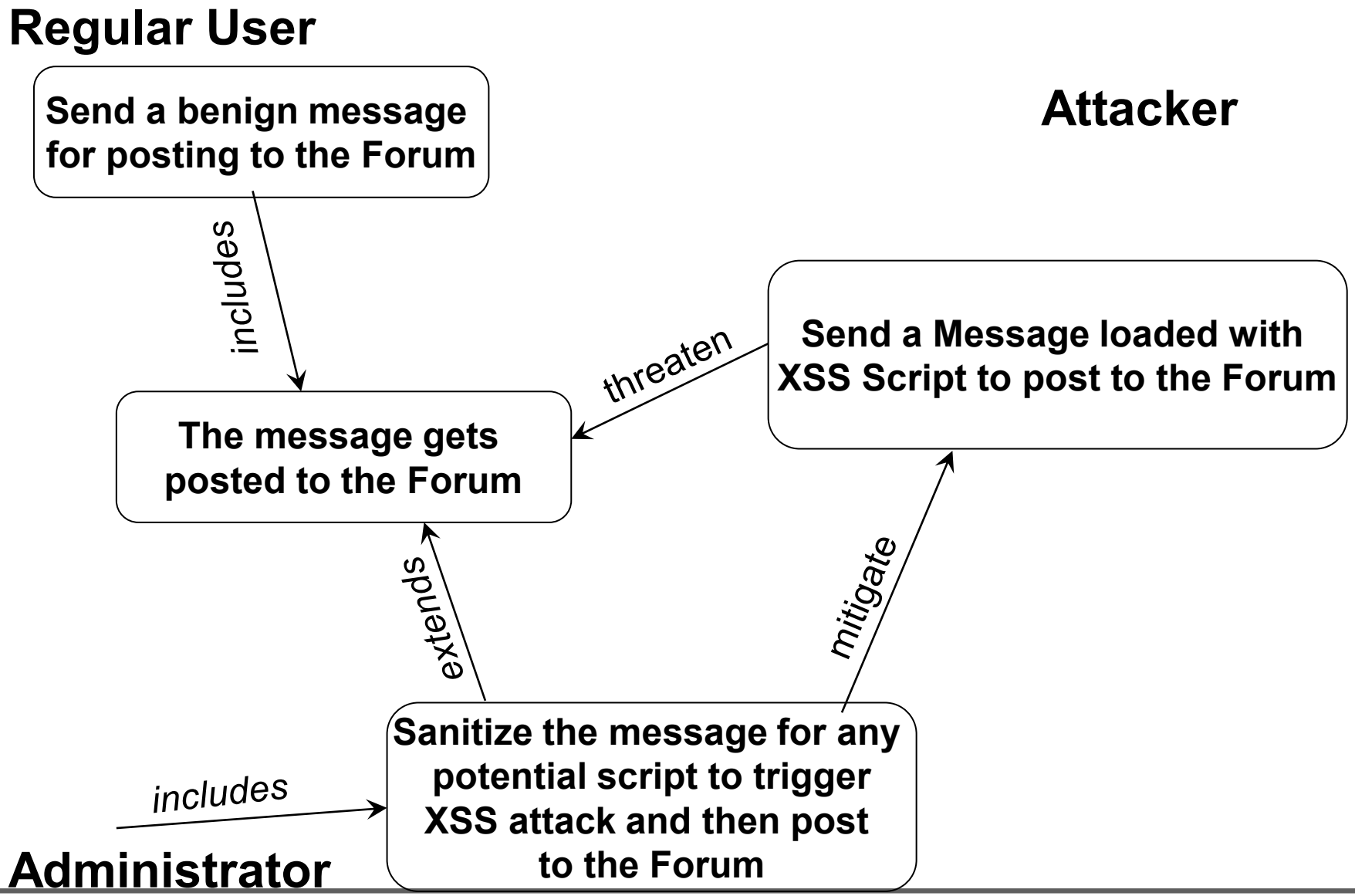


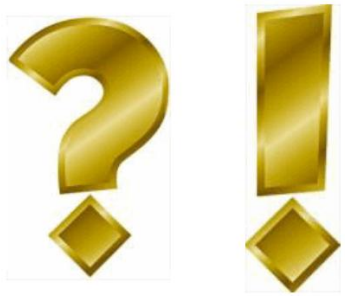

*Napadač treba znati stegoanalizu da otkrije skriveni tekst pa još kriptonanalizu da izdvoji čisti tekst*

# Example: UC-MUC diagram for a web forum



# Primjer: UC-MUC dijagram za web forum





# Tools for software security (not network)

ÿ Microsoft SDL and derivatives

1

ÿ Attack Surface Analyzer – reducing the attack surface

ÿ Microsoft Threat Modeling Tool – threat modeling ÿ

MiniFuzz basic file fuzzing tool – *fuzz* testing ÿ Regular

expression file fuzzing tool – testing of potential DoS vulnerabilities

ÿ Static analysis

ÿ StyleCop https://stylecop.codeplex.com/ # similar, FxCop

ÿ CodeSmart http://www.axtools.com/ ÿ NDepend http://

www.ndepend.com/ ÿ PMD Java, Checkstyle, FindBugs+Find

Security Bugs

# 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

# Resources

ÿ Open Web Application Security Project (OWASP) ÿ http://www.owasp.org, OWASP Top Ten vulnerabilities in web applications.

ÿ Building Security In ÿ https://buildsecurityin.us-cert.gov/bsi/home.html ÿ SANS Institute ÿ http://www.sans.org/ , CWE/SANS Top 25 Most Dangerous Prog. Errors ÿ CERT (Computer Security Incident Response Team)

ÿ http://www.cert.org/ , http://www.cert.org/secure-coding/, https://www.cert.hr

ÿ Cloud Security Alliance ÿ https://cloudsecurityalliance.org/ ÿ Other ÿ CVE (Common Vulnerabilities and Exposures), http://cve.mitre.org/ ÿ Security Tracker, http://securitytracker.com/ ÿ US-CERT Cyber Security Bulletins http://www.us-cert.gov/cas/bulletins/ ÿ Web Application Security Consortium (WASC), http://www.webappsec.org/ ÿ MSDN, http://msdn.microsoft.com/

# 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

# References


ÿ Noopur Davis: Secure Software Development Life Cycle Processes, Software Engineering Institute, Carnegie Mellon University, 2013 ÿ http://resources.sei.cmu.edu/asset_files/whitepaper/2013_019_001_297287.pdf

ÿ Microsoft SDL, http://www.microsoft.com/security/sdl ÿ STRIDE, http://msdn.microsoft.com/en-us/magazine/cc163519.aspx ÿ DREAD, http://msdn.microsoft.com/en-us/library/ff648644.aspx ÿ SDL Quick Security References, http://www.microsoft.com/en-us/download/details.aspx?id=13759

ÿ BSIMM Building Security In – Maturity Model, http://bsimm.com

ÿ OpenSAMM Software Assurance Maturity Model, http://opensamm.org

ÿ OWASP Application Threat Modeling

ÿ https://www.owasp.org/index.php/Application_Threat_Modeling

# 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

## For the end

ÿ Bruce Schneier, https://www.schneier.com/

ÿ If you think technology can solve your security problems, then you don't understand the problems and you don't understand the technology.

ÿ Unless you think like an attacker, you will be unaware of any potential threats!

ÿ You can't defend. You can't prevent it. The only thing you can do is detect and respond.

# Za kraj

- 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.