IN2001
Software Engineering og prosjektarbeid
Vår 2018

Sikker systemutvikling

Audun Jøsang

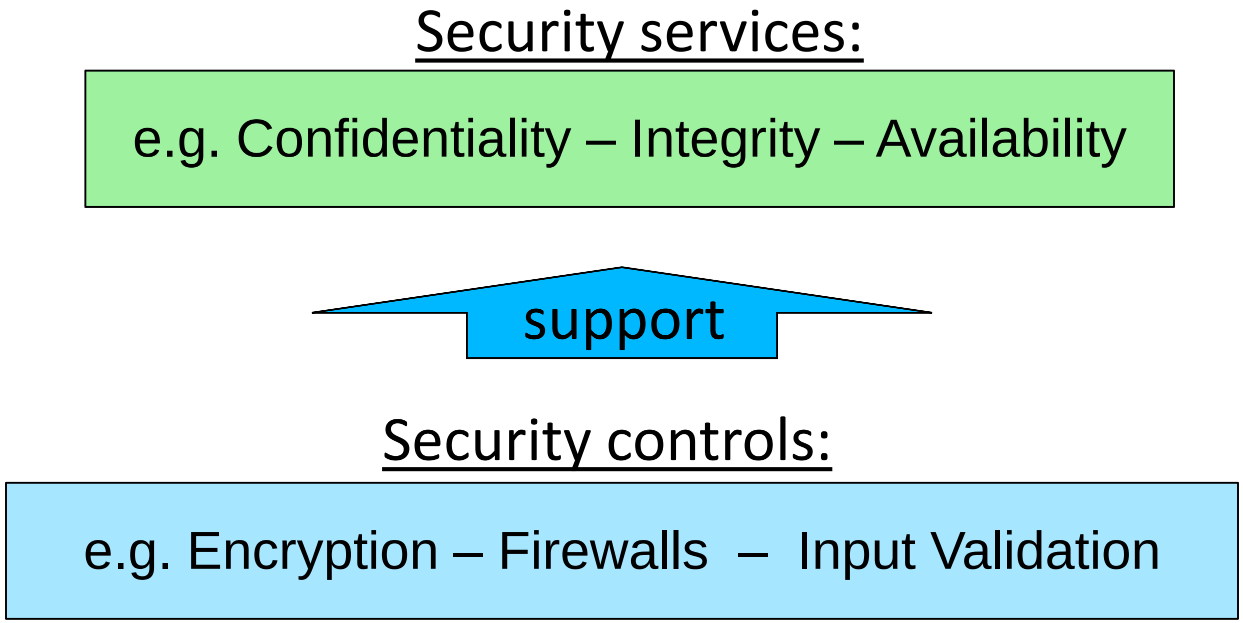
University of Oslo

Hva er informasjonssikkerhet ?

- *Informasjonssikkerhet* betyr å beskytte *informasjonsressurser* mot skade.
- Hvilke informasjonsressurser skal beskyttes?
  - Eksempel: data, programvare, konfigureringer, utstyr og infrastruktur
- Dekker både tilsiktet og utilsiktet skade
  - Trusselagenter kan være mennesker eller naturlige hendelser
  - Mennesker kan gjøre skade både tilsiktet og utilsiktet
- Definisjon av informasjonssikkerhet:
  - *Beskyttelse av informasjonens konfidensialitet, integritet og tilgjengelighet. I tillegg kan andre egenskaper, f.eks. autentisitet, sporbarhet, uavviselighet og pålitelighet omfattes.* (NS 27002:2005)
  - *The preservation of confidentiality, integrity and availability of information; in addition, other properties such as authenticity, accountability, non-repudiation and reliability can also be involved*. (ISO27000:2016)

Security services and controls

- Security services (aka. goals or properties)
  - implementation independent
  - supported by specific controls
- Security controls (aka. mechanisms)
  - Practical mechanisms, actions, tools or procedures that are used to provide security services



Security services:

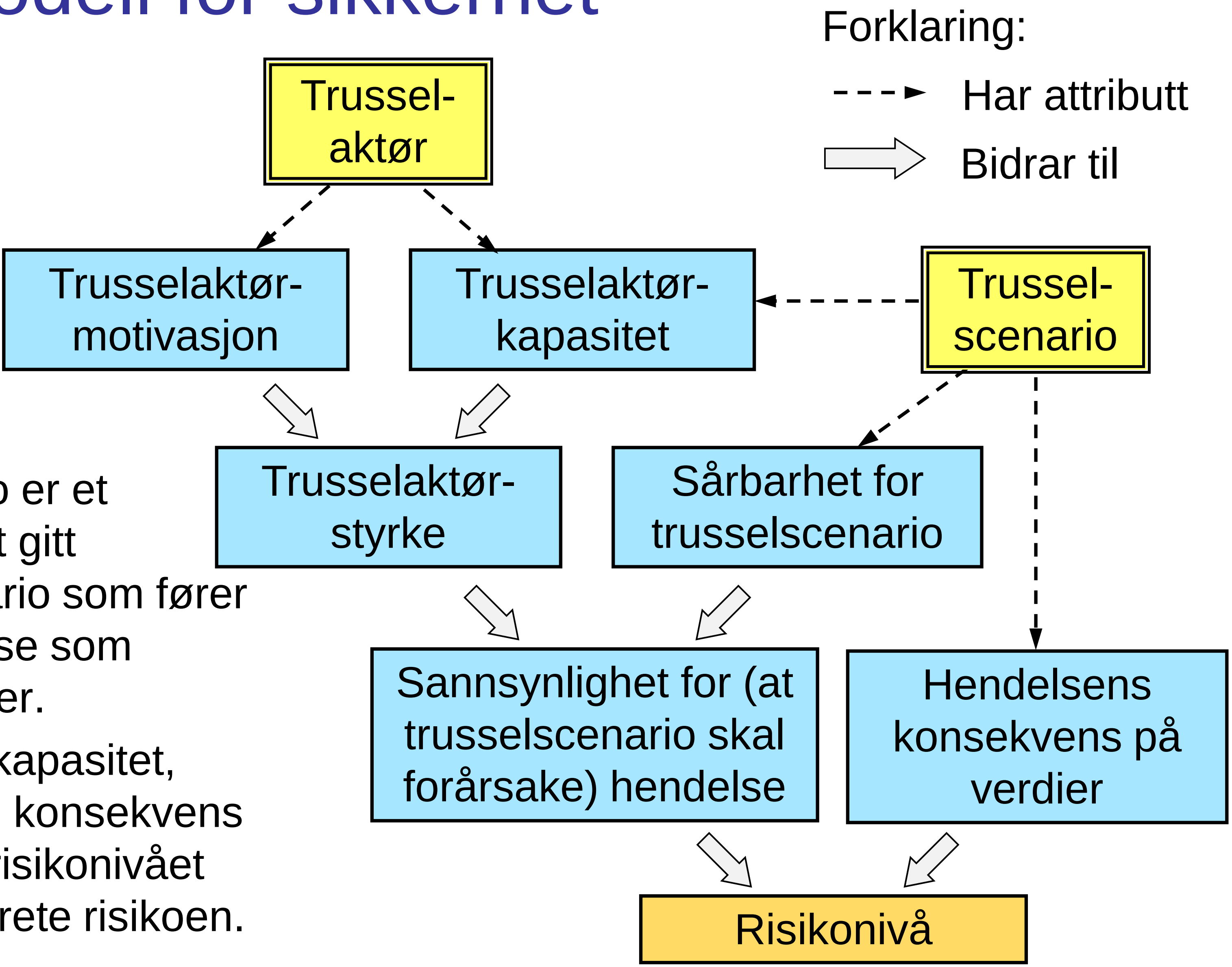
e.g. Confidentiality – Integrity – Availability

support

Security controls:

e.g. Encryption – Firewalls – Input Validation

Risikomodell for sikkerhet

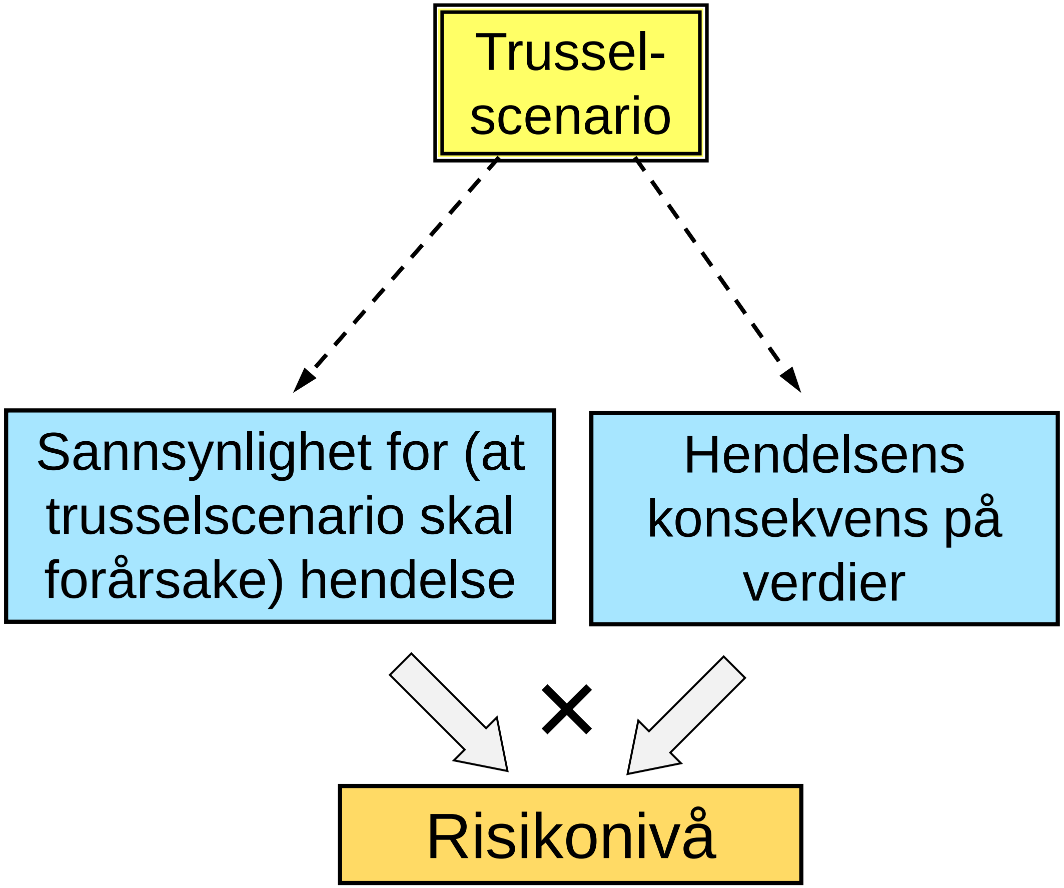


- Enhver risiko er et resultat av et gitt trusselscenario som fører til en hendelse som skader verdier.
- Motivasjon, kapasitet, sårbarhet og konsekvens bestemmer risikonivået for den konkrete risikoen.

Praktisk risikovurdering

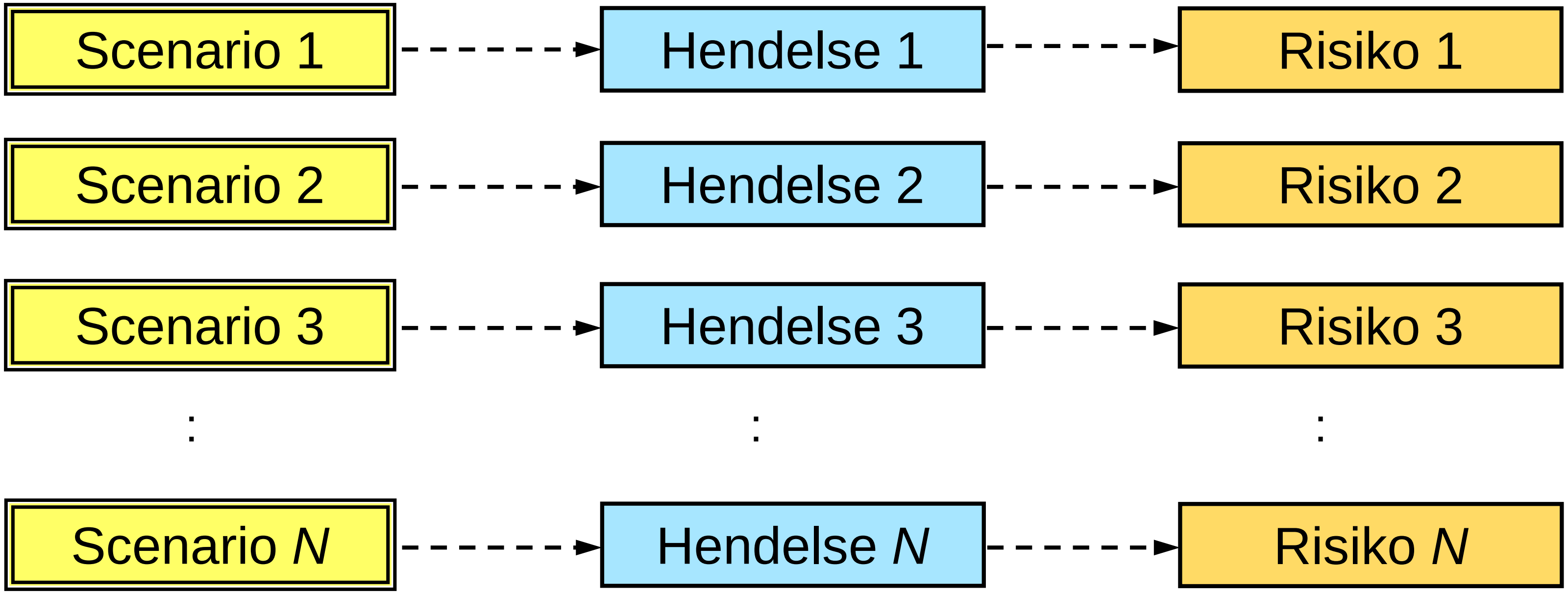
- Estimering av nivå for hver risiko er typisk basert på to faktorer:

1. Sannsynlighet / frekvens for en trusselhendelse
2. Konsekvens av trusselhendelsen

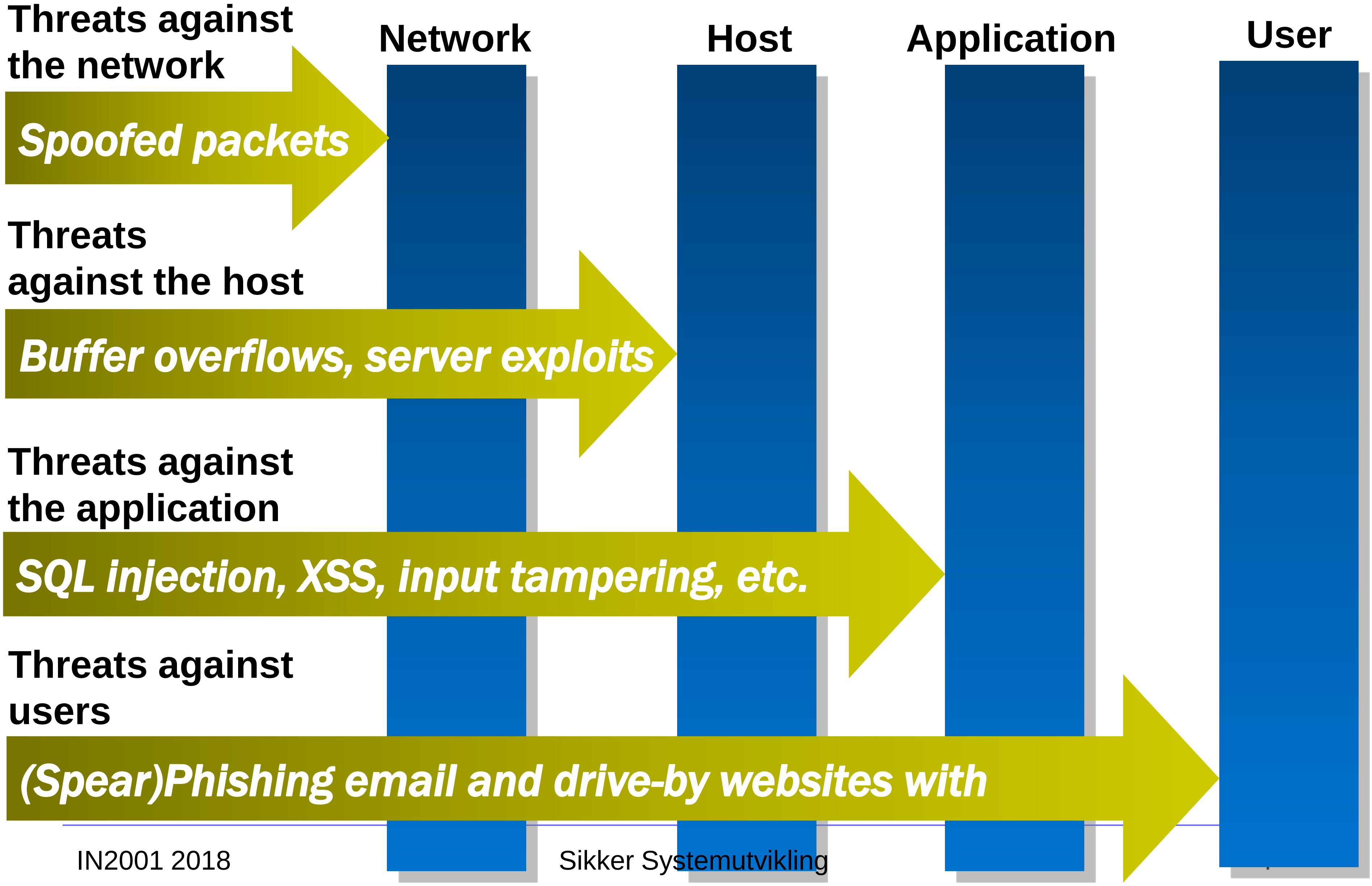

Trussel-scenario
Sannsynlighet for (at trusselscenario skal forårsake) hendelse
Hendelsens konsekvens på verdier
×
Risikonivå

Mange risikoer

- Mange forskjellige trusselsenarioer kan identifiseres
- Hvert trusselscenario kan potensielt føre til en hendelse
- Hver potensielle hendelse har en konsekvens
- Hver kombinasjon av scenario og konsekvens gir en risiko



Types of Threats



Network

Host

Application

User

Threats against the network

Spoofed packets

Threats against the host

Buffer overflows, server exploits

Threats against the application

SQL injection, XSS, input tampering, etc.

Threats against users

(Spear)Phishing email and drive-by websites with

Threats Against the Network

Threat	Examples
Information gathering	Port scanning
	Using trace routing to detect network topologies
	Using broadcast requests to enumerate subnet hosts
Eavesdropping	Using packet sniffers to steal passwords
Denial of service (DoS)	SYN floods
	ICMP echo request floods
	Malformed packets
Spoofing	Packets with spoofed source addresses

Threats Against the Host

Threat	Examples
Arbitrary code execution	Buffer overflows in ISAPI DLLs (e.g., MS01-033)
	Directory traversal attacks (MS00-078)
File disclosure	Malformed HTR requests (MS01-031)
	Virtualized UNC share vulnerability (MS00-019)
Denial of service (DoS)	Malformed SMTP requests (MS02-012)
	Malformed WebDAV requests (MS01-016)
	Malformed URLs (MS01-012)
	Brute-force file uploads
Unauthorized access	Resources with insufficiently restrictive ACLs
	Spoofing with stolen login credentials
Exploitation of open ports and protocols	Using NetBIOS and SMB to enumerate hosts
	Connecting remotely to SQL Server

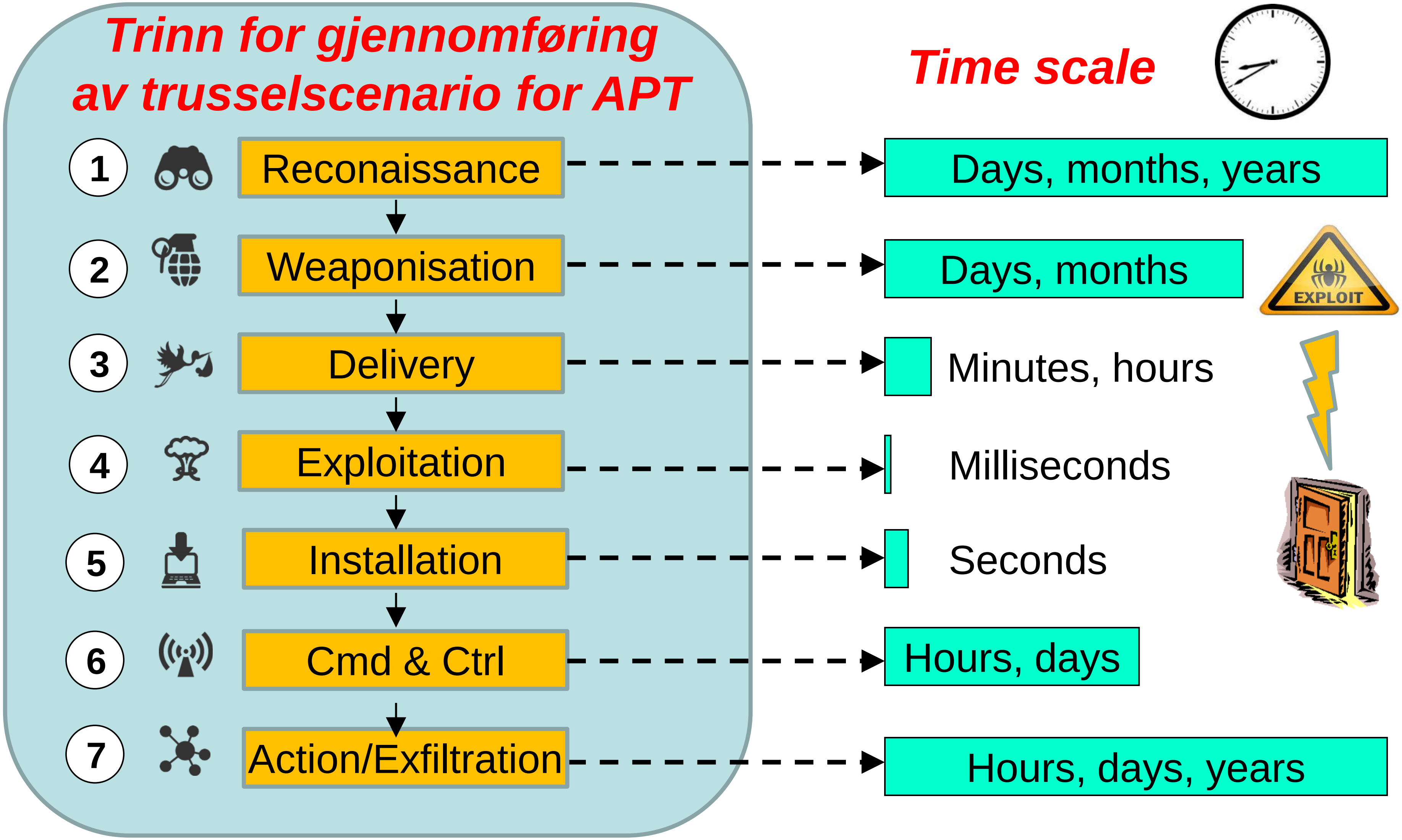
Threats Against the Application

Threat	Examples
SQL injection	Including a DROP TABLE command in text typed into an input field
Cross-site scripting	Using malicious client-side script to steal cookies
Hidden-field tampering	Maliciously changing the value of a hidden field
Eavesdropping	Using a packet sniffer to steal passwords and cookies from traffic on unencrypted connections
Session hijacking	Using a stolen session ID cookie to access someone else's session state
Identity spoofing	Using a stolen forms authentication cookie to pose as another user
Information disclosure	Allowing client to see a stack trace when an unhandled exception occurs

Threats Against the User

Threat	Examples
Phishing email	Pointer to false website, request to login, stolen credentials
Phishing email	Pointer to compromised website with drive-by malicious content, e.g. malware, exploits, XSS,
Phishing email	Attachment with malware and exploits, camouflaged as legitimate program or document
Malicious website	Containing malicious content e.g. malware XSS, drive-by infection
Compromised website	Containing malicious content e.g. malware XSS, drive-by infection
Plug-in media	USB-device with malware
Malicious wifi router	Interception of data & credentials, possibly combined with TLS stripping or fake certificates

APT Scenario (Advanced Persistent Threats)

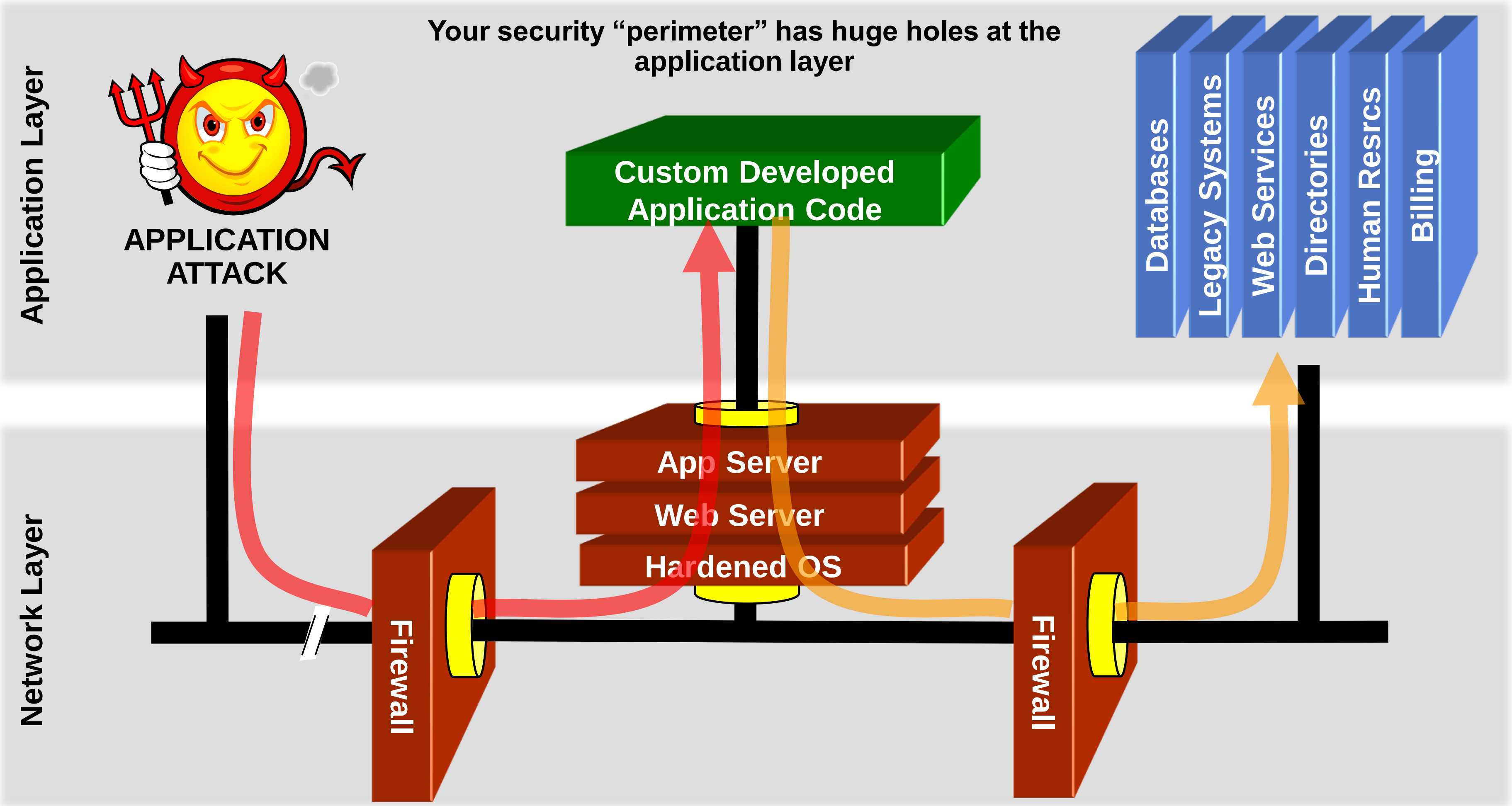


Trinn for gjennomføring av trusselscenario for APT

Time scale

1. Reconaissance → Days, months, years
2. Weaponisation → Days, months
3. Delivery → Minutes, hours
4. Exploitation → Milliseconds
5. Installation → Seconds
6. Cmd & Ctrl → Hours, days
7. Action/Exfiltration → Hours, days, years

The web application security challenge



Network security (firewall, SSL, IDS, hardening) does not stop application attacks

OWASP
The Open Web Application Security Project

- Non-profit organisation
  - Local chapters in most countries, also in Norway
- OWASP promotes security awareness and security solutions for Web application development.
- OWASP Top-10 security risks identify the most critical security risks of providing online services
  - The Top 10 list also recommends relevant security solutions.
- OWASP ASVS (Application Security Verification Standard) specifies requirements for application-level security.
- OWASP website provides and maintains many free tools for scanning and security vulnerability fixing

Top-10 2017 Web Application Risks

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

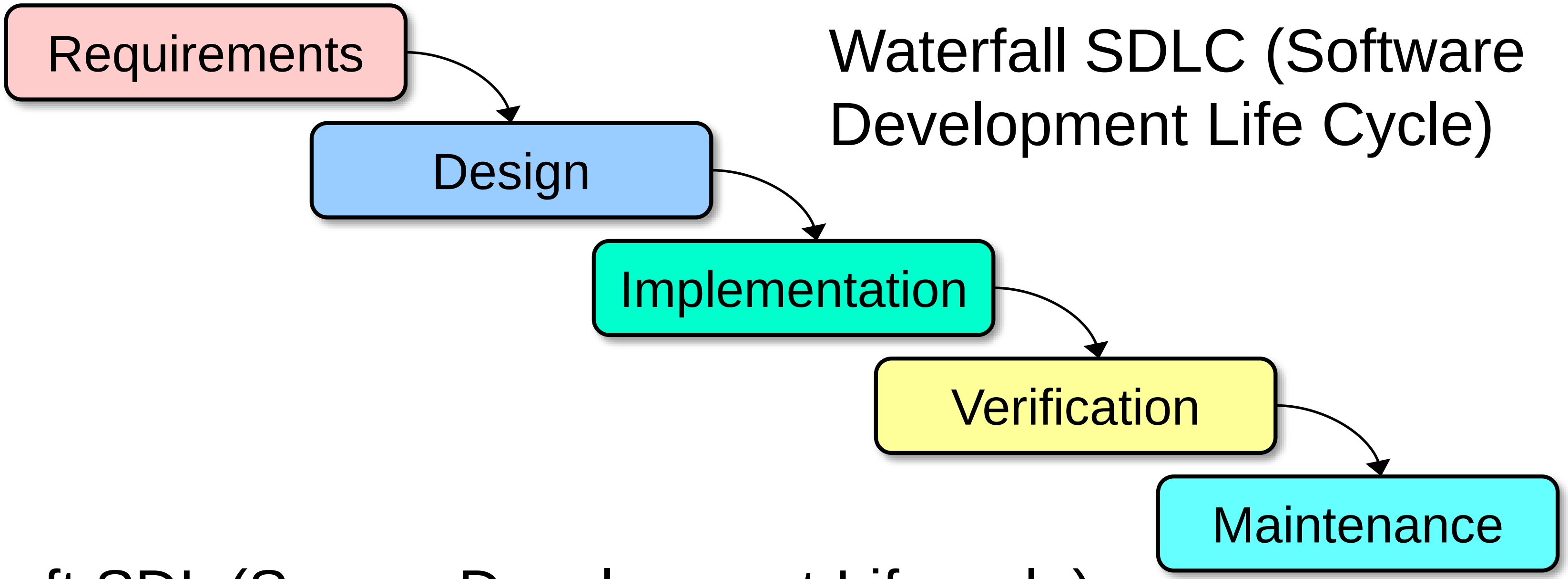
ASVS
Application Security Verification Standard

ASVS Edition 2016 contains 16 sets of verification requirements

https://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project

- V1: Architecture
- V2: Authentication
- V3: Session Management
- V4: Access Control
- V5: Input validation and output encoding
- V7: Cryptography
- V8: Error Handling
- V9: Data Protection
- V10: Communications
- V13: Malicious Code
- V15: Business Logic Flaws
- V16: Files and Resources
- V17: Mobile
- V18: API
- V19: Configuration
- V20: Internet of Things

Waterfall and Secure Waterfall



Requirements

Design

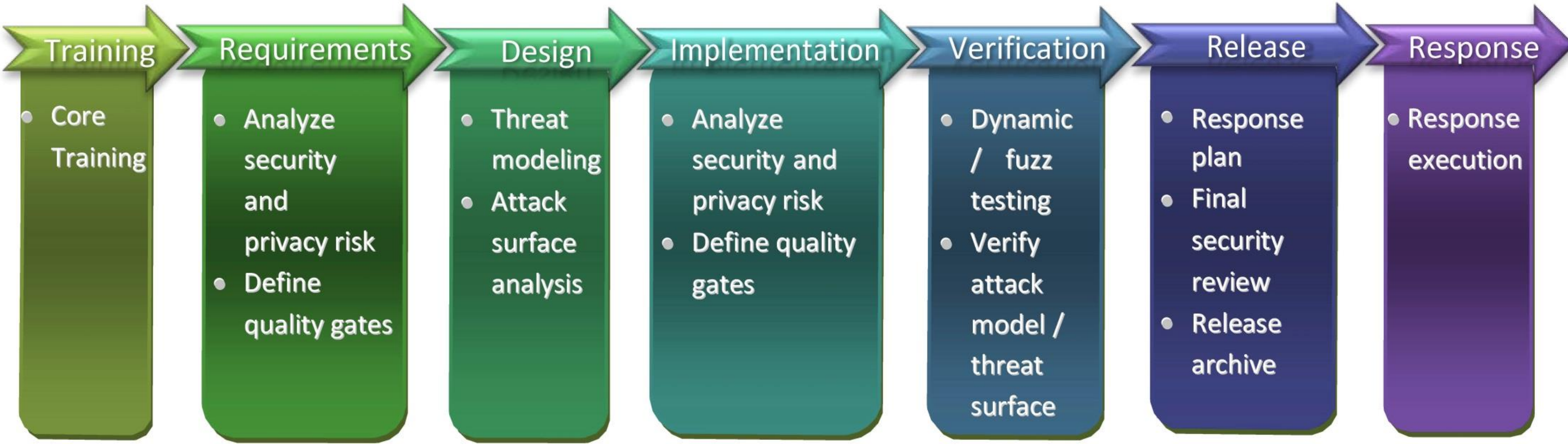
Implementation

Verification

Maintenance

Waterfall SDLC (Software Development Life Cycle)

Microsoft SDL (Secure Development Lifecycle)



Training
- Core Training

Requirements
- Analyze security and privacy risk
- Define quality gates

Design
- Threat modeling
- Attack surface analysis

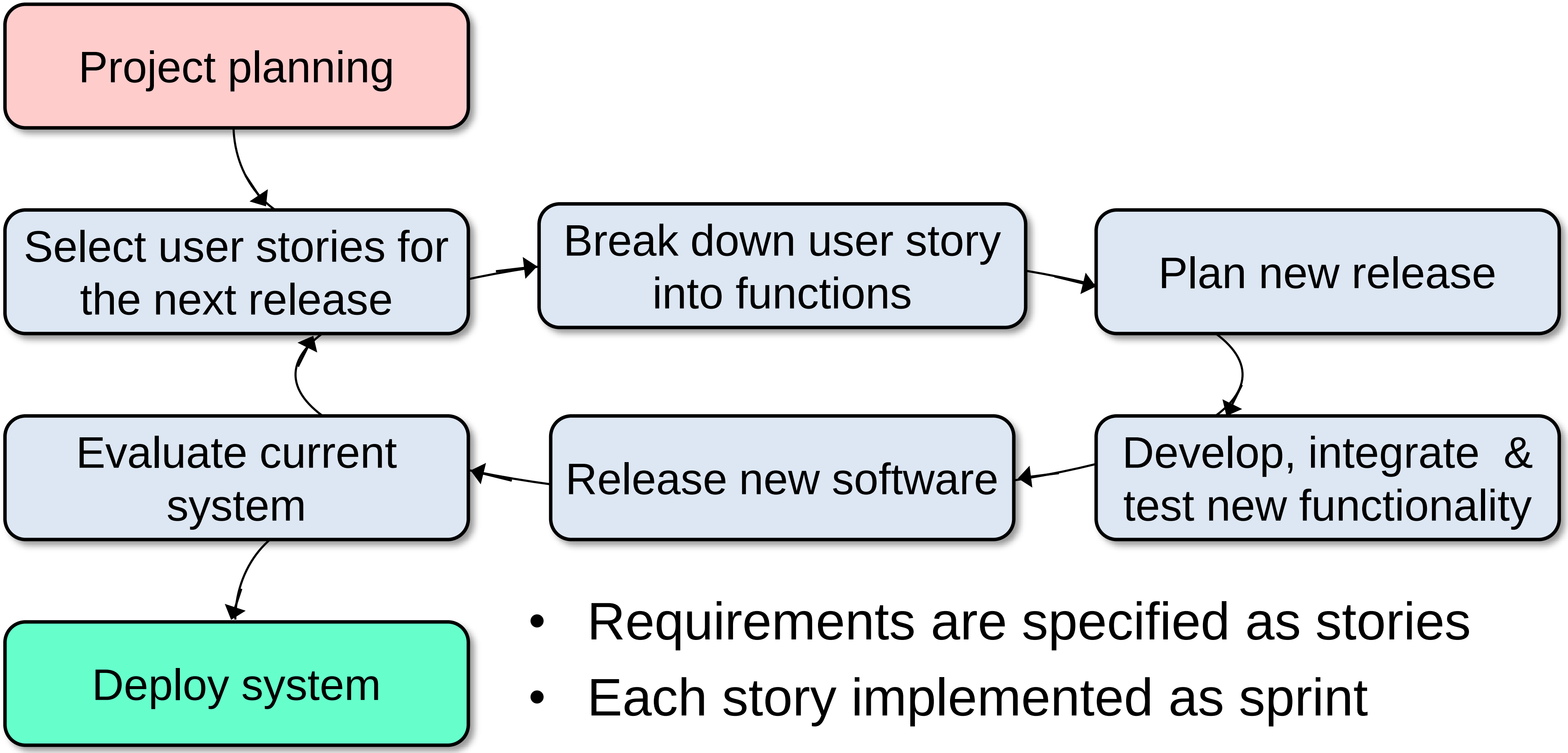
Implementation
- Analyze security and privacy risk
- Define quality gates

Verification
- Dynamic / fuzz testing
- Verify attack model / threat surface

Release
- Response plan
- Final security review
- Release archive

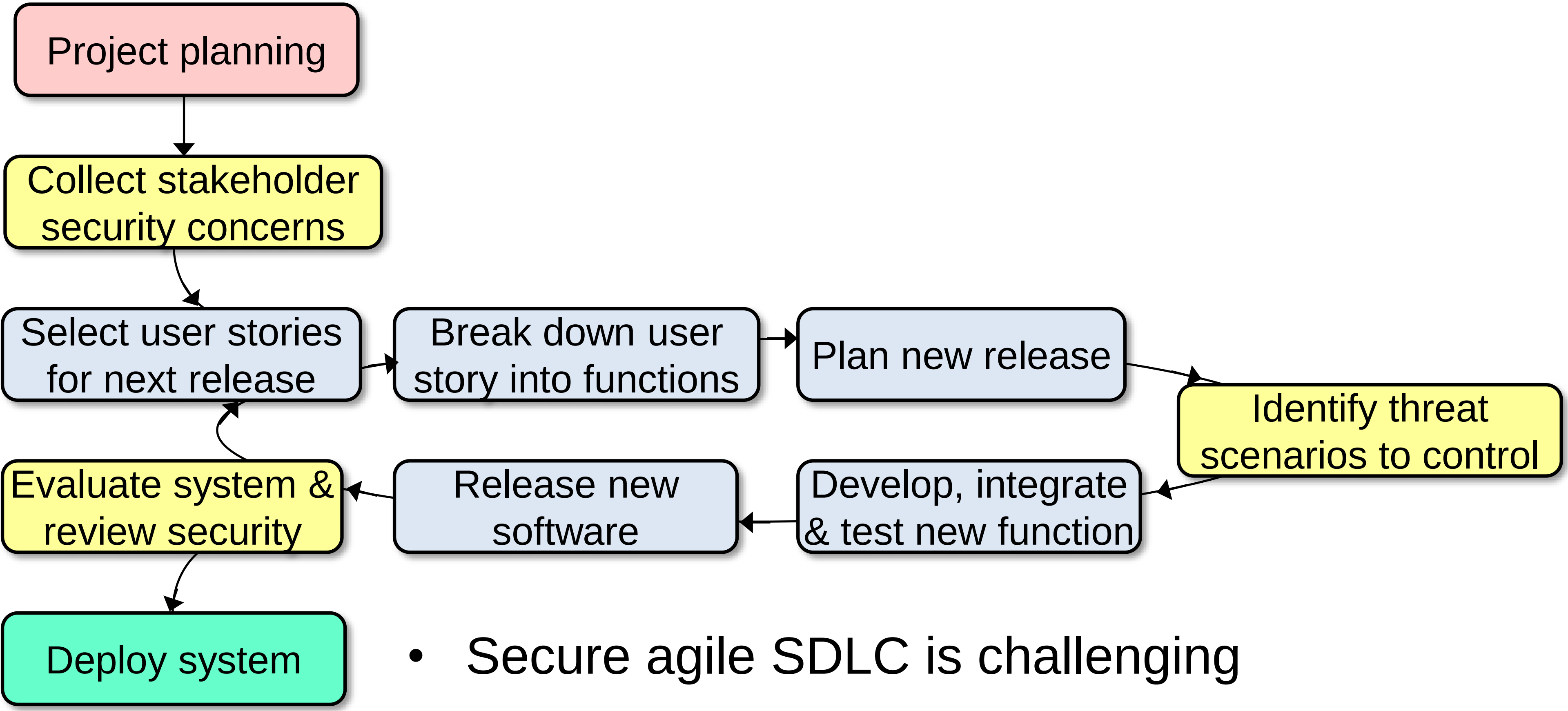
Response
- Response execution

Agile Software Development


Project planning
Select user stories for the next release
Break down user story into functions
Plan new release
Develop, integrate & test new functionality
Release new software
Evaluate current system
Deploy system

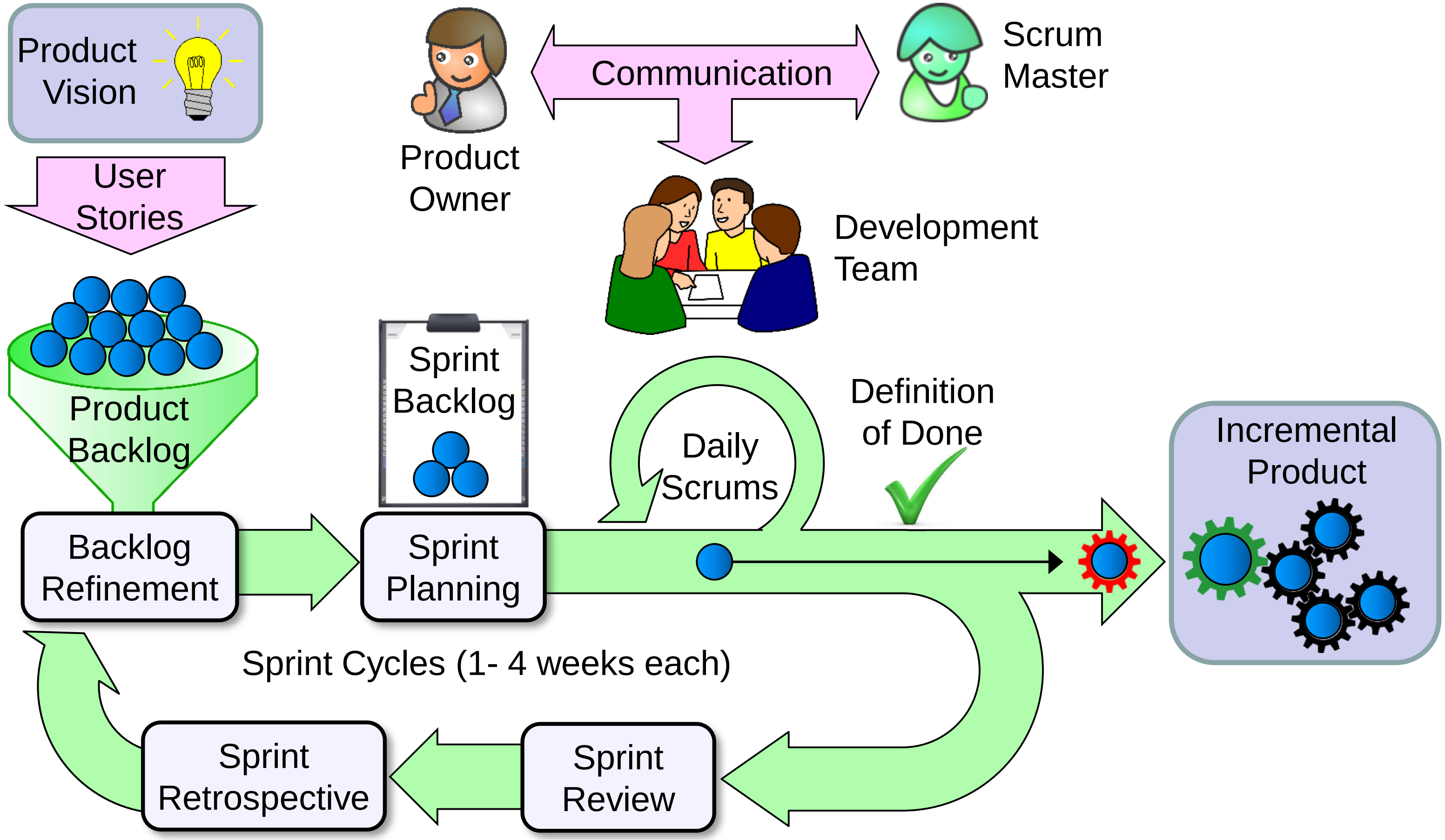
- Requirements are specified as stories
- Each story implemented as sprint
- Repeated sprint cycles until all stories are implemented

Secure Agile Software Development


Project planning
Collect stakeholder security concerns
Select user stories for next release
Break down user story into functions
Plan new release
Identify threat scenarios to control
Develop, integrate & test new function
Release new software
Evaluate system & review security
Deploy system

- Secure agile SDLC is challenging
- Add security related development tasks
  - (yellow boxes)
- Security necessarily maks SDLC less agile

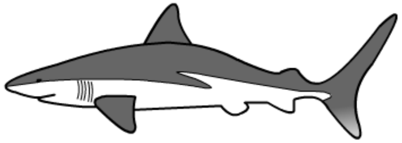
Scrum Model for Agile Software Development



Threat Modelling Objectives

By performing Threat Modeling you can:

- Identify relevant threats to your particular application scenario.



- Identify key vulnerabilities in your application design.

- Improve your security design

Threat Modelling

- Attacker-centric (STRIDE)

  (**S**poofing with identity, **T**ampering with data, **R**epudiation, **I**nformation disclosure, **D**enial of service, **E**levation of privilege)

  - Starts from attackers, evaluates their goals, and how they might achieve them through attack tree. Usually starts from entry points or attacker action.
- System-centric (ASF – Application Security Frame)
  - Starts from model of system, and attempts to follow model dynamics and logic, looking for types of attacks against each element of the model. This approach is e.g. used for threat modeling in Microsoft's Security Development Lifecycle.
- Asset-centric (CORAS)
  - Starts from information assets, such as a collection of sensitive personal information, and attempts to identify how security breaches of CIA properties can happen.

STRIDE

S **Spoofing**

Can an attacker gain access using a false identity?

T **Tampering**

Can an attacker modify data as it flows through the application?

R **Repudiation**

If an attacker denies doing something, can we prove he did it?

I **Information disclosure**

Can an attacker gain access to private or potentially injurious data?

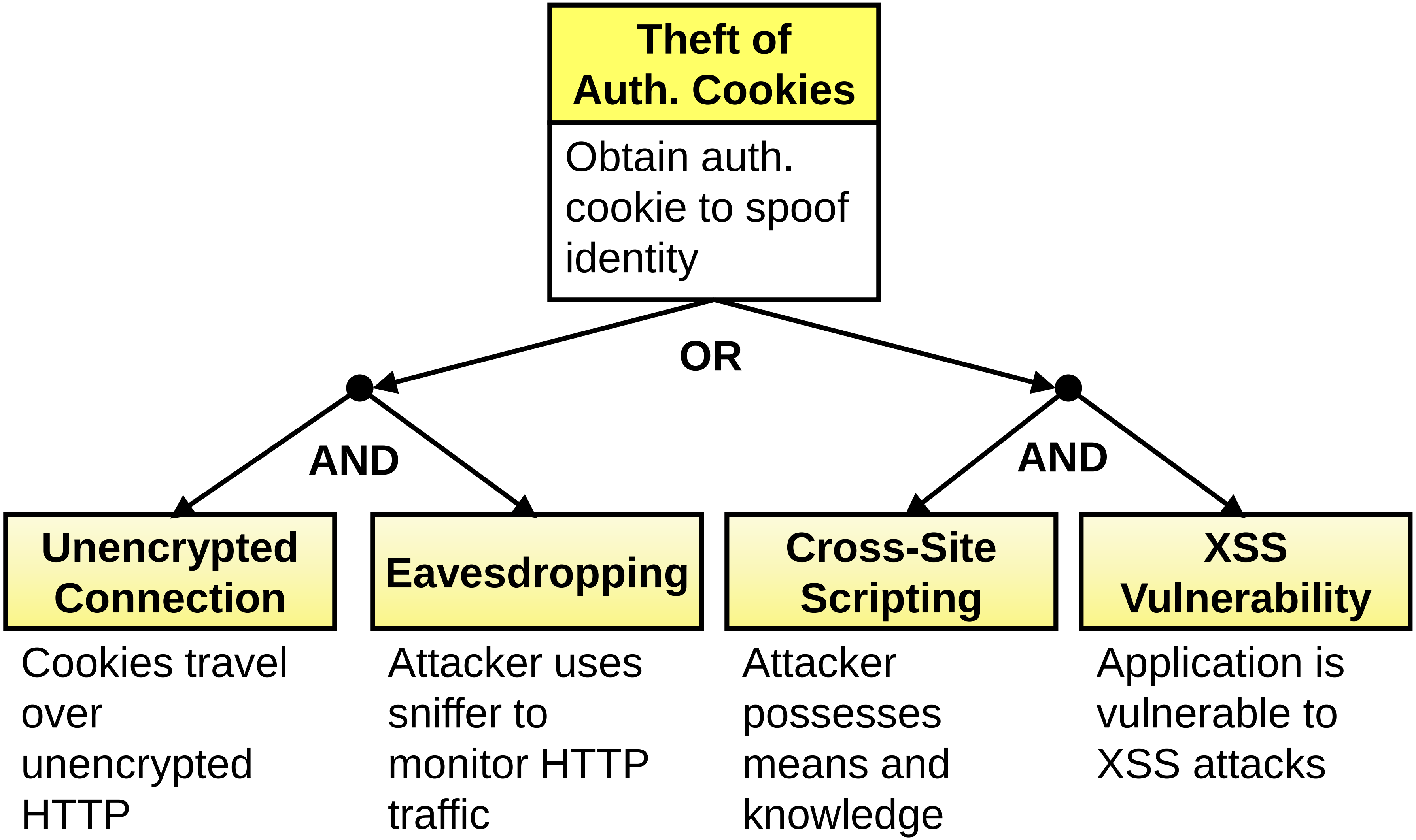
D **Denial of service**

Can an attacker crash or reduce the availability of the system?

E **Elevation of privilege**

Can an attacker assume the identity of a privileged user?

Attack Centric: Threat Trees



Theft of Auth. Cookies

Obtain auth. cookie to spoof identity

OR

AND

Unencrypted Connection

Cookies travel over unencrypted HTTP

Eavesdropping

Attacker uses sniffer to monitor HTTP traffic

AND

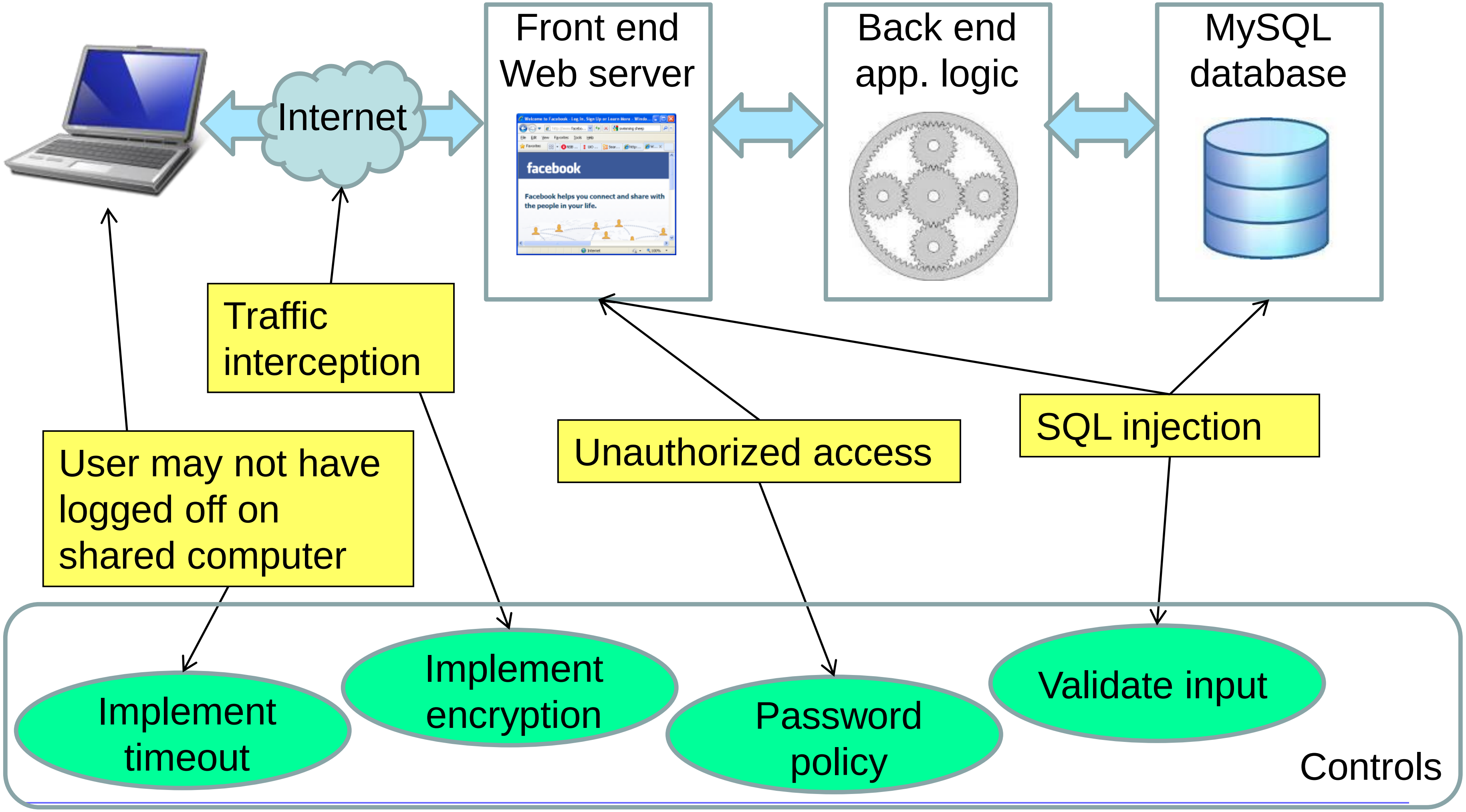
Cross-Site Scripting

Attacker possesses means and knowledge

XSS Vulnerability

Application is vulnerable to XSS attacks

System-centric threat modelling example



Internet

Front end Web server

Back end app. logic

MySQL database

Traffic interception

User may not have logged off on shared computer

Unauthorized access

SQL injection

Implement timeout

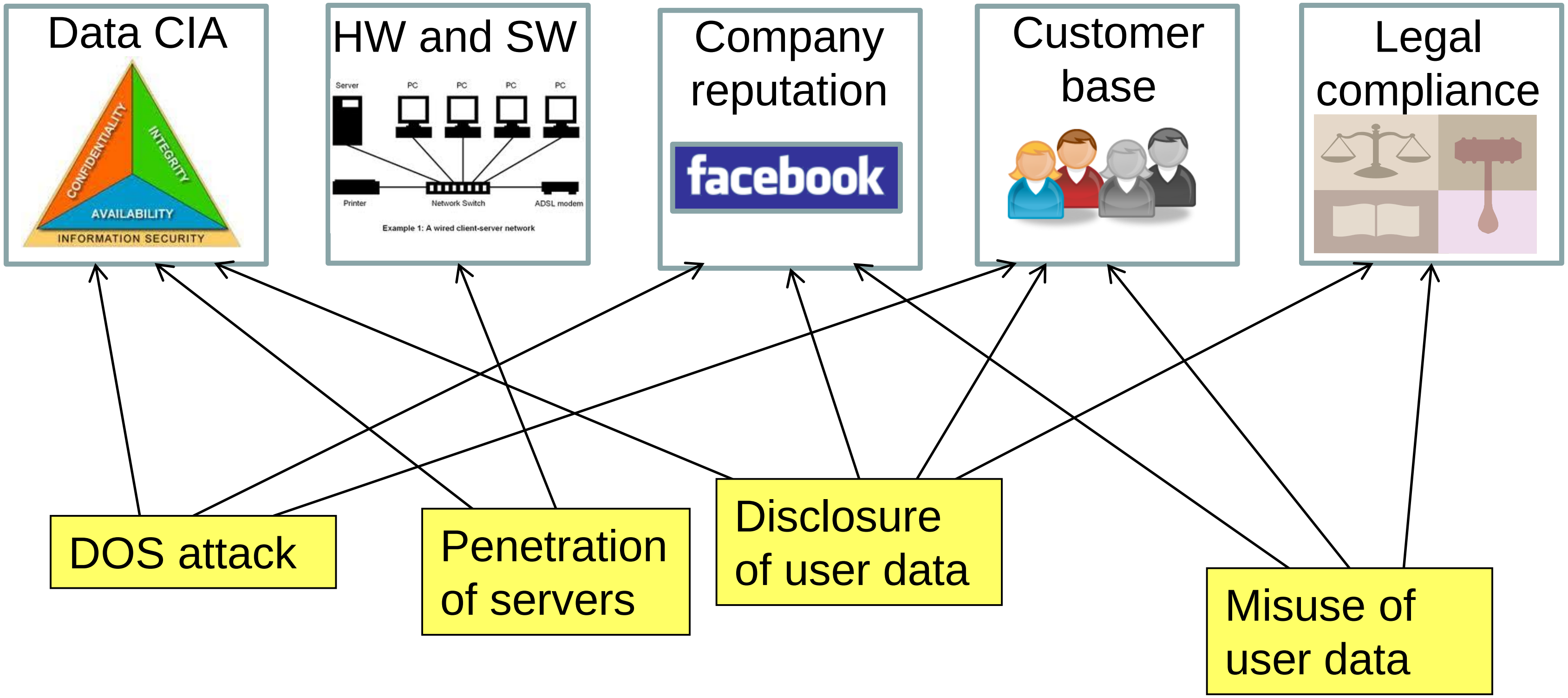
Implement encryption

Password policy

Validate input

Controls

Asset-centric threat modelling



Data CIA

HW and SW

Company reputation

Customer base

Legal compliance

DOS attack

Penetration of servers

Disclosure of user data

Misuse of user data

Risk Levels of Threats with DREAD

D **Damage potential**

What are the consequences of a successful exploit?

R **Reproducibility**

Would an exploit work every time or only under certain circumstances?

E **Exploitability**

How skilled must an attacker be to exploit the vulnerability?

A **Affected users**

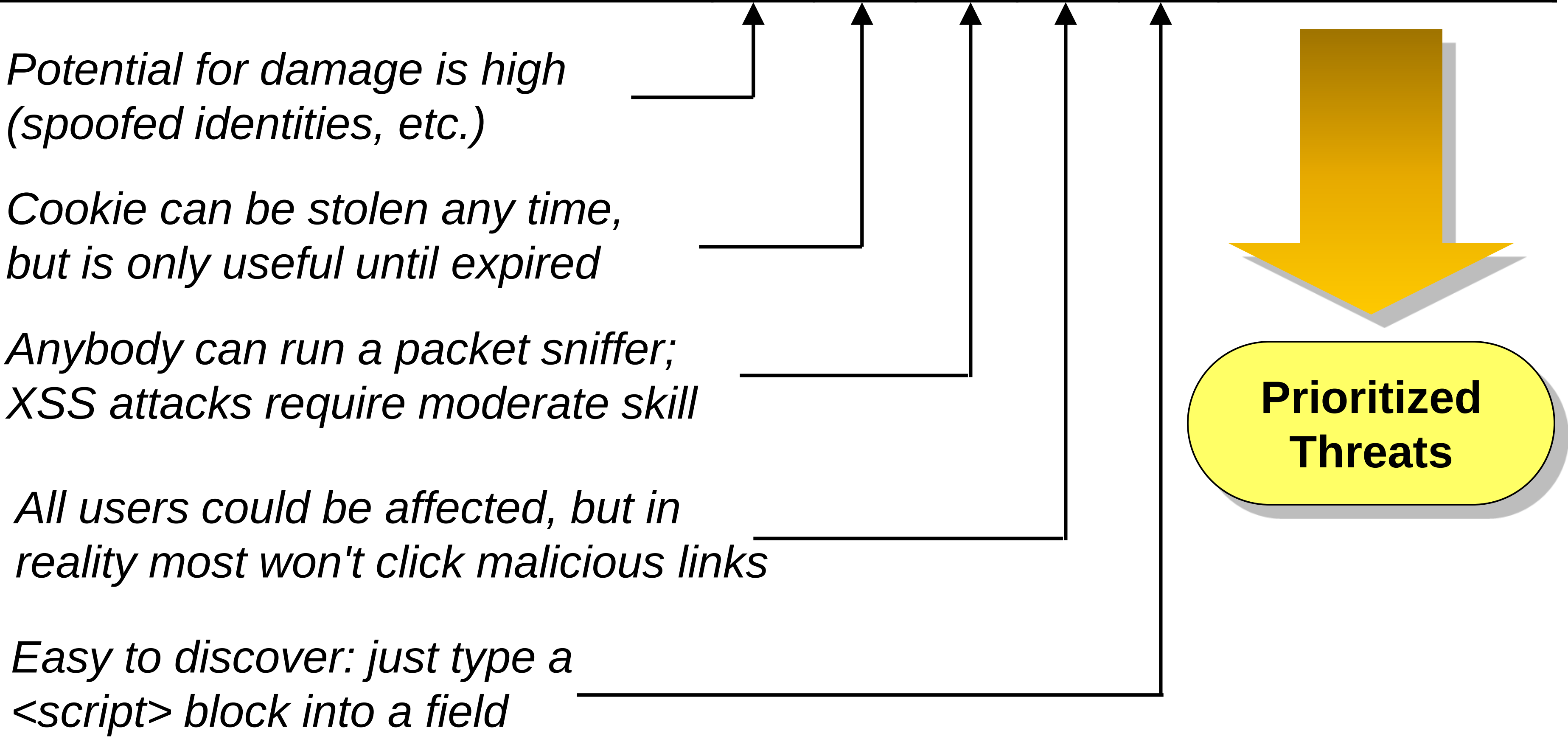
How many users would be affected by a successful exploit?

D **Discoverability**

How likely is it that an attacker will know the vulnerability exists?

Example

Threat	D	R	E	A	D	Risk Lvl
Auth. cookie theft (eavesdropping)	3	2	3	2	3	13
Auth. cookie theft (XSS)	3	2	2	2	3	12



Potential for damage is high (spoofed identities, etc.)

Cookie can be stolen any time, but is only useful until expired

Anybody can run a packet sniffer; XSS attacks require moderate skill

All users could be affected, but in reality most won't click malicious links

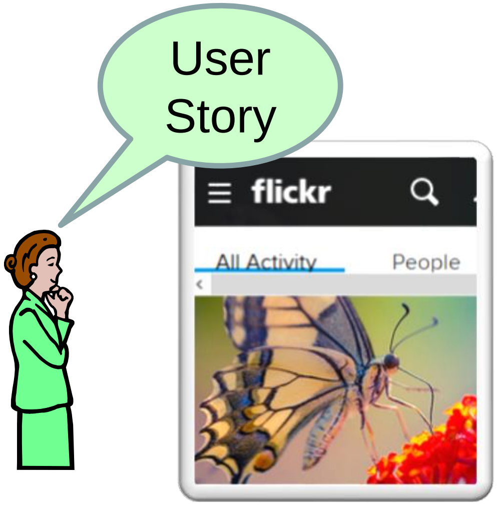
Easy to discover: just type a <script> block into a field

Prioritized Threats

User Stories vs. Usecases

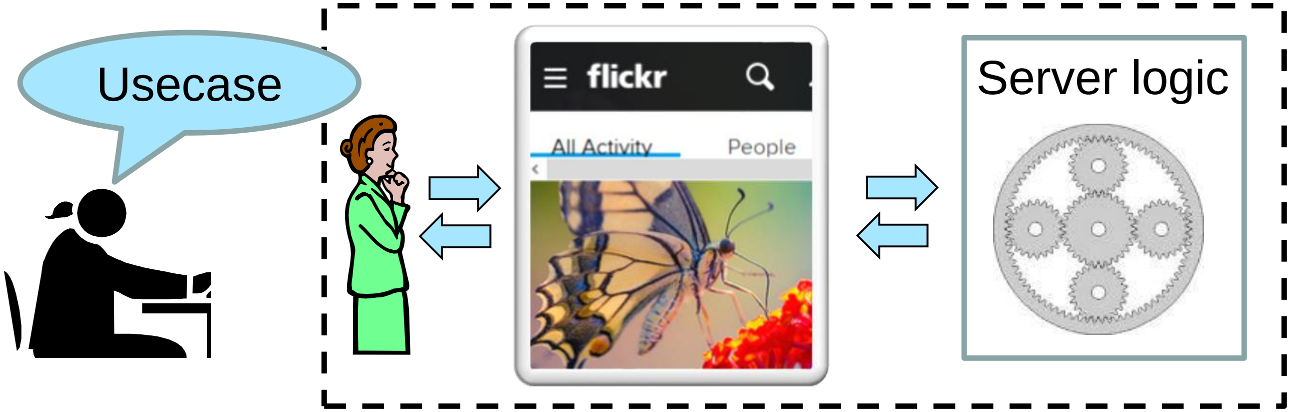
User Story – Seen from the user perspective:

As an [actor] I want [action] so that [achievement].
For example: *As a Flickr member, I want to set different privacy levels on my photos, so I can control who sees which of my photos.*



Usecase – Seen from the design perspective:

Description of a set of interactions between a system and one or more actors (where an ‘actor’ can be a user or another system).



Attacker Story and Misuse Case (Attacker Goal and Threat Scenario)

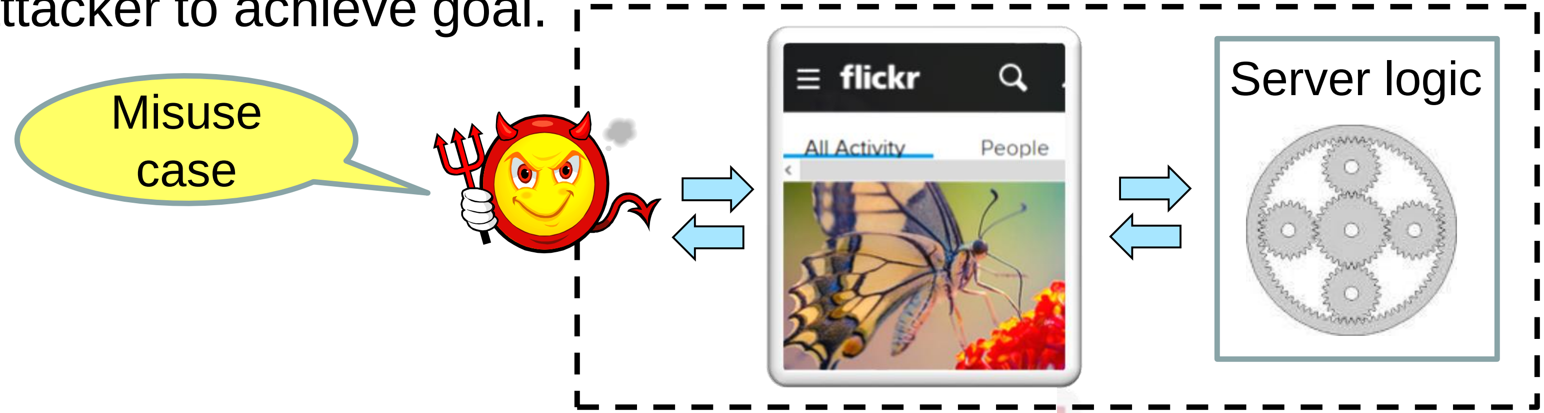
Attacker Story – The goal of the attacker:

As an [attacker] I want [action] so that [achievement]. So, for example: *As an attacker, I want to hack into Flickr accounts to steal photos and personal info.*

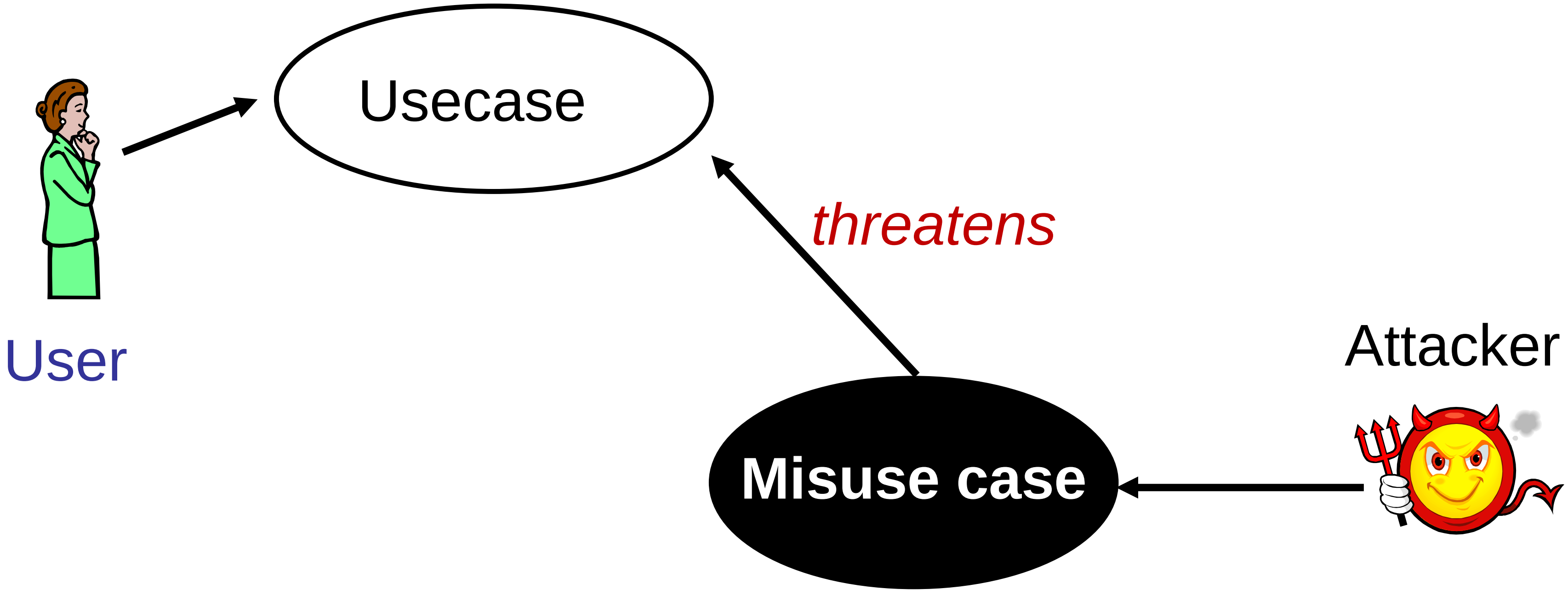
Misuse Case (Threat Scenario)

Seen from the threat scenario perspective:

Description of a set of steps and interactions to be executed by attacker to achieve goal.



Threat Considerations for Usecases



Usecase

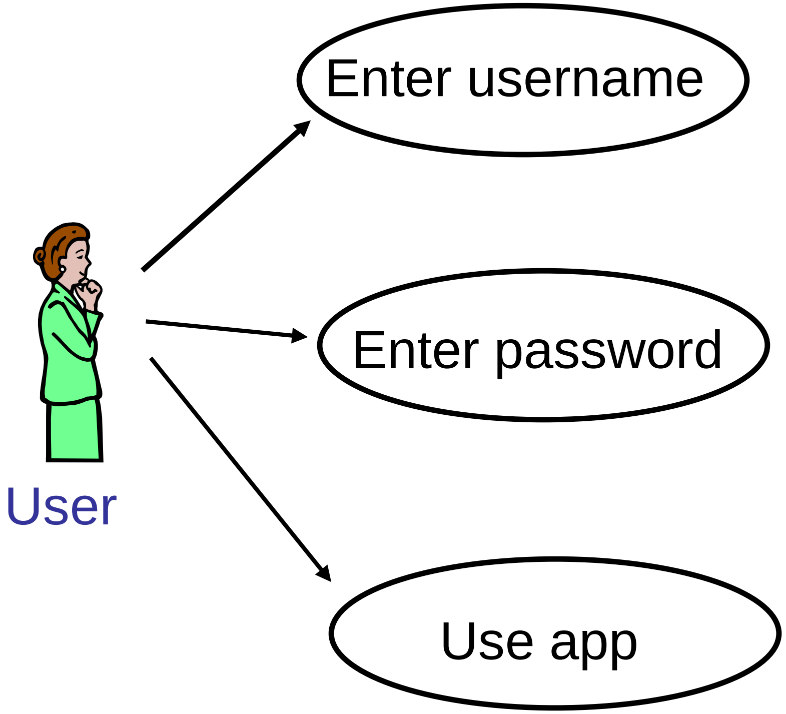
User

threatens

Attacker

Misuse case

Example part 1



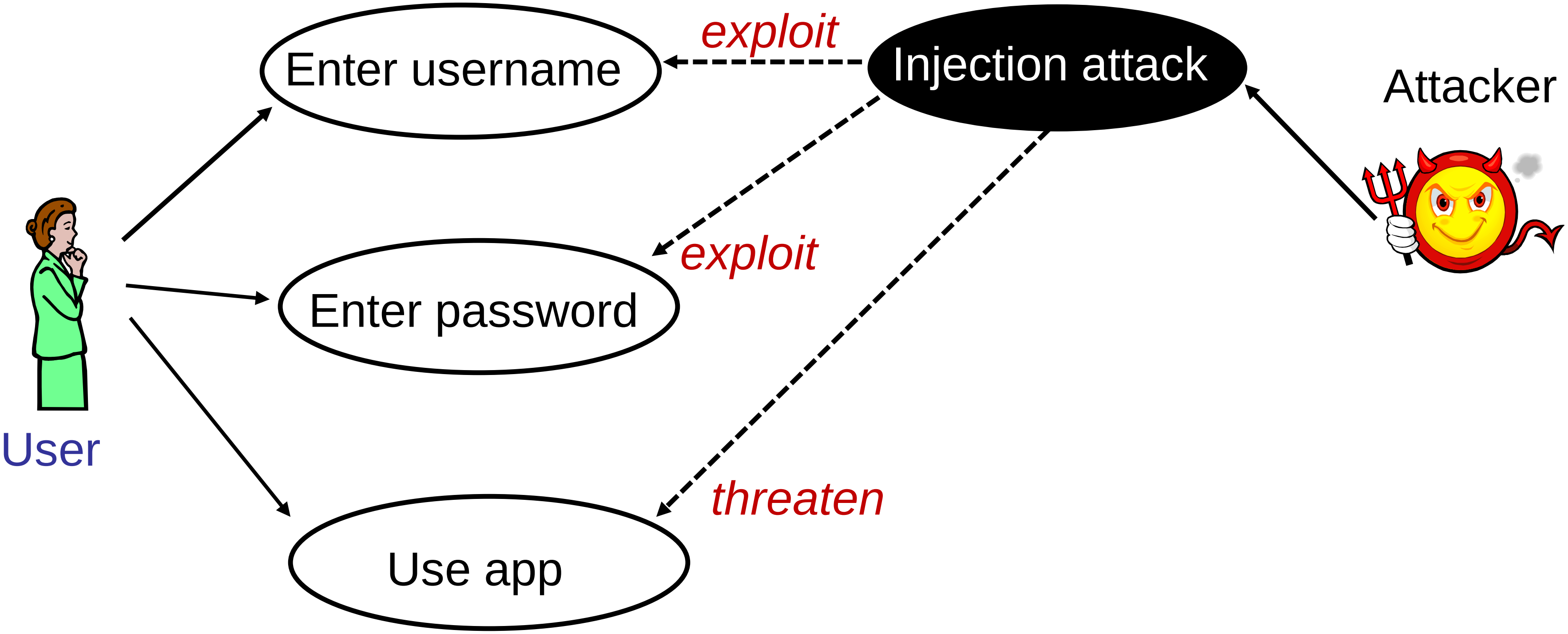
Enter username

Enter password

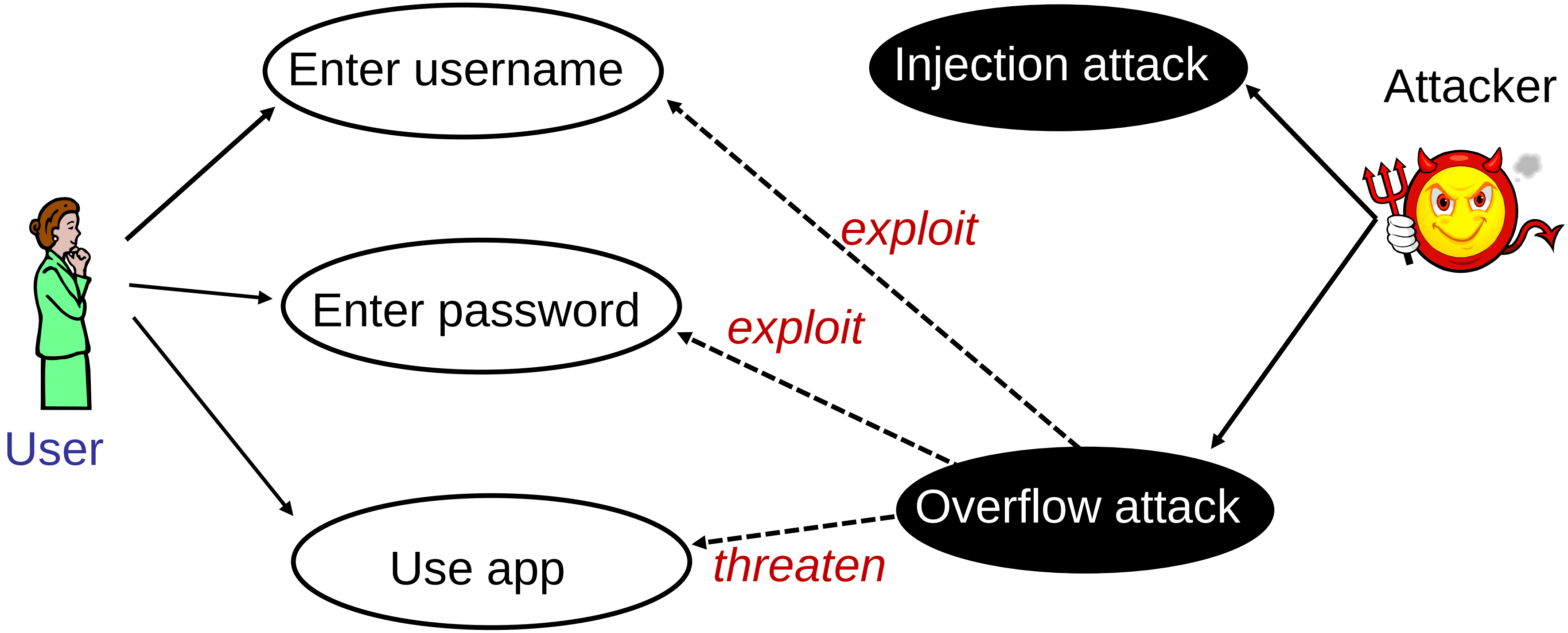
User

Use app

Example part 2



Example part 3



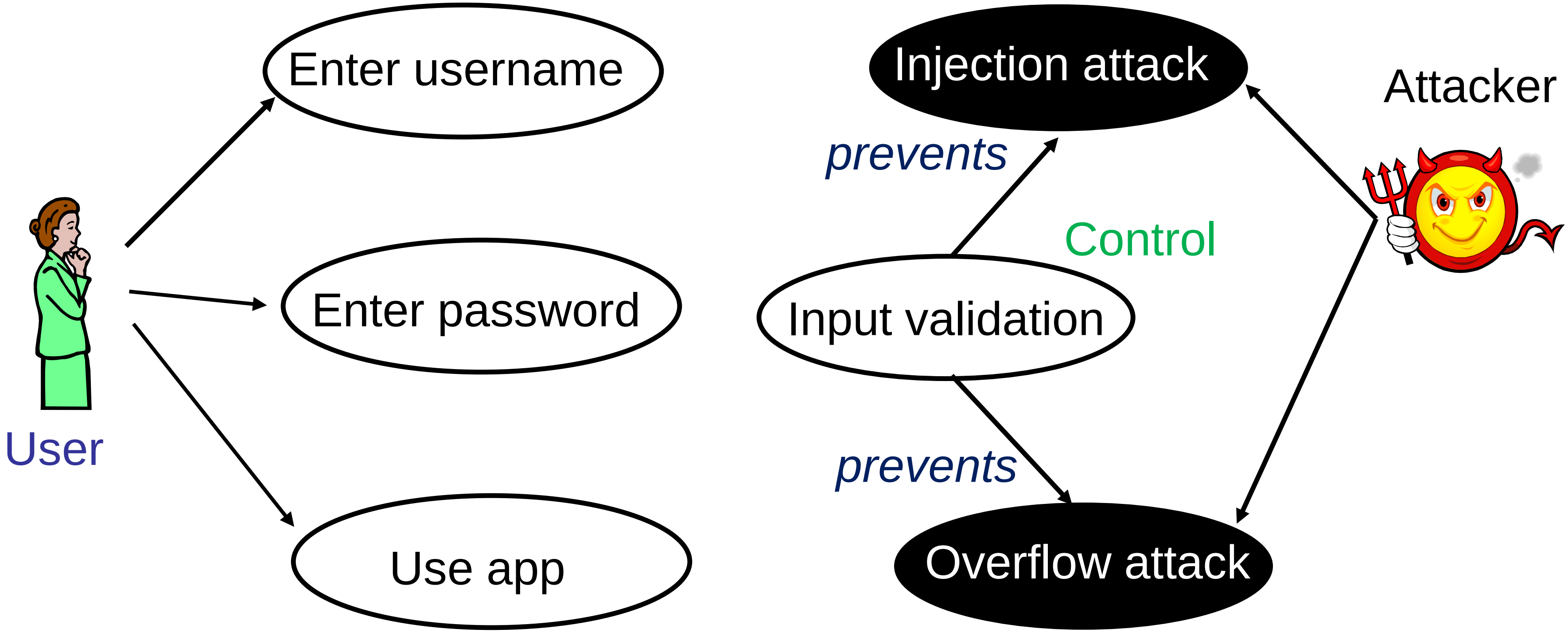
Threat Description (Misuse Case Description)

- Name of threat scenario
- Attacker profile
- Description
  - Basic scenario
  - Alternative scenarios
- Vulnerabilities exploited by threat scenario
- Assumptions
- D.R.E.A.D. risk level
- Potential security control(s) to eliminate or mitigate threat
  - Expected **simplicity** of implementing security control(s)

Security Controls

- Propose security controls that will:
  - remove vulnerabilities exploited by the threat
  - prevent, detect or correct damage from the threat scenario
- Security controls examples:
  - Using non-vulnerable libraries
  - Filtering input
  - Encryption
  - Logging of events
  - Strong user authentication
  - Adequate access control
  - Configuration and patch management
  - Backup
  - etc.

Example part 4



Enter username

Enter password

Use app

User

Injection attack

Overflow attack

Attacker

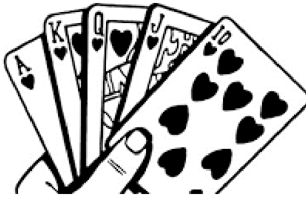
Input validation

Control

prevents

prevents

Threat Poker for Each Threat Scenario



- A little bit similar to planning poker.
  - Multiple rounds for everyone to express their opinions
  - Stimulates discussion in the team
  - Avoids bias induced by strong/loud team members

- Redback Cards: Risk-level of threat (e.g. from DREAD)
  - 2-value: Min risk,
  - Ace-value (14): Max risk

- Blueback Cards: How **simple/easy** is it to implement control(s) that would prevent/detect/correct threat scenario
  - 2-value: Very difficult/expensive/long time to stop threat,
  - Ace-value (14): Very simple/easy/cheap/quick to stop threat

- Priority for removing threat is the sum: +

Summary

- Without threat modelling, developed software will be insecure
- Fundamental for developers to understand common attacks
- Expertise is needed to understand specialised attacks
- Developers must learn and use secure coding practices
- The amount of threat modelling is a business decision

End of Lecture