INF3510
Information Security

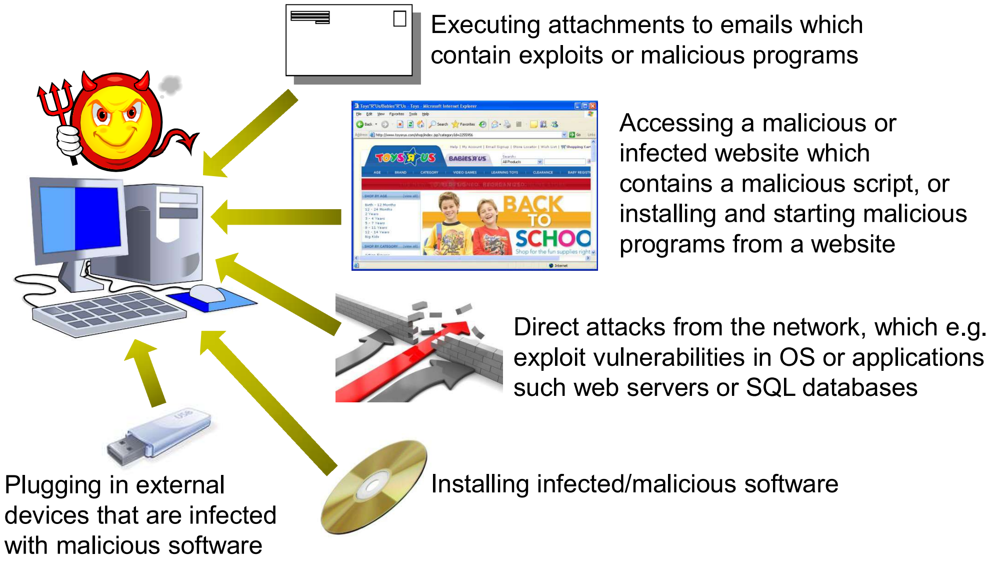
L12: Application Security

Audun Jøsang
University of Oslo
Spring 2018

Outline

1. Application Security (Audun Jøsang)
   – Malicious Software
   – Attacks on applications
   – Secure system development
2. Security by Design (Dagfinn Bergsager, USIT)
3. Industrial Approach (Espen Johansen, VISMA)

How do computers get infected ?



Executing attachments to emails which contain exploits or malicious programs

Accessing a malicious or infected website which contains a malicious script, or installing and starting malicious programs from a website

Direct attacks from the network, which e.g. exploit vulnerabilities in OS or applications such web servers or SQL databases

Installing infected/malicious software

Plugging in external devices that are infected with malicious software

Malware types

- Backdoor or trapdoor
- Logic bomb,
- Trojan horse
- Worm
- Virus
  - Stealth virus
    - Uses techniques to hide itself, e.g. encryption
  - Polymorphic virus
    - Different for every system
  - Metamorphic virus
    - Different after every activation on same system
- Exploit
  - An tool to infect systems by using malicious program or input data (e.g. document) that triggers and exploits a software bug in the systems

Exploits

- A piece of software, data, or a sequence of commands that exploits a software/hardware vulnerability
- Can be carried in common data formats such as pdf documents, office documents or media files.

PDF Office MP3 JPEG

- Often contains carefully designed corrupt datatypes
- Causes unintended or unanticipated behavior to occur on computer software or hardware
- Exploit functionality typically is to
  - Download a program/backdoor which allows the attacker to control the platform
  - Directly take control of a computer system, allowing privilege escalation, or a denial-of-service or other sabotage.

Backdoor or Trapdoor

Installed by exploit:

- Provides remote control capabilities by attackers
- Can reside on system for long periods before being used
- Can be removed after use

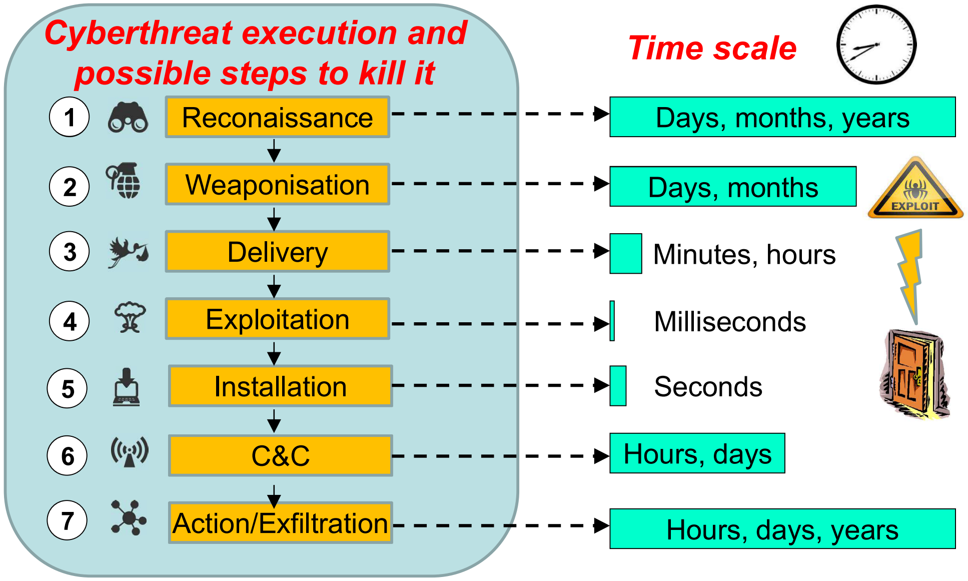
Installed by user:

- User can be tricked to install malicious program (see Trojan horse)

Installed during design:

- is a hidden/secret entry point into a program,
- allows those who know access bypassing usual security procedures
- is commonly used by developers for testing
- is a threat when left in production software allowing, exploit by attackers
- is very hard to block in O/S
- can be prevented with secure development lifecycle

The Cyber Kill Chain (Hutchins et al. 2011)



Logic Bomb

- one of oldest types of malicious software
- code embedded in legitimate program
- activated when specified conditions met
  - eg presence/absence of some file
  - particular date/time
  - particular user
- causes damage when triggered
  - modify/delete files/disks, halt machine, etc

Trojan Horse

- program with hidden side-effects
  - e.g. a back door
- program is usually superficially attractive
  - eg game, s/w upgrade etc
- performs additional tasks when executed
  - allows attacker to indirectly gain access they do not have directly
- often used to propagate a virus/worm or to install a backdoor
- … or simply to destroy data

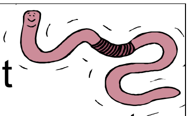
Malicious Mobile Code

- ➢ Program/script/macro that runs unchanged
  - ▪ on homogeneous platforms (e.g. Windows)
    - will only affect specific platforms
  - ▪ on heterogeneous platforms
    - will affect any platform that supports script/macro language
    - e.g. Office macros
- ➢ Transmitted from remote system to local system & then executed on local system
  - ▪ To inject Trojan horse, spyware, virus, worm etc. which can
    - directly perform specific attacks, such as unauthorized data access, root compromise, sabotage
    - indirectly infect other systems and thereby spread

Viruses

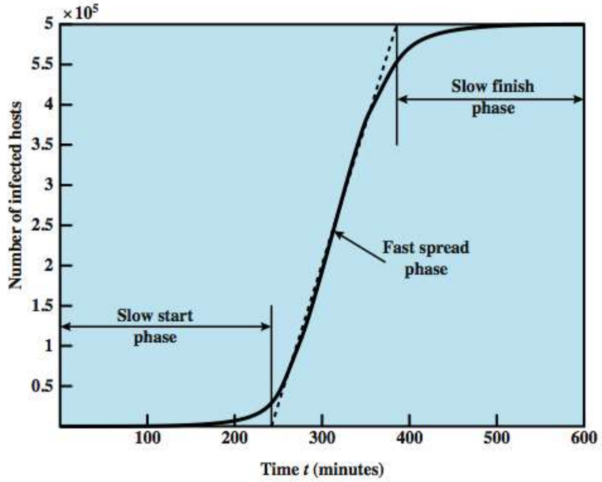
- ➢ piece of software that infects programs
- ➢ specific to operating system and hardware
  - ● taking advantage of their details and weaknesses
- ➢ a typical virus goes through phases of:
  - ● dormant
  - ● propagation
  - ● triggering
  - ● execution

Worms



- Replicating programs that propagate over net
  - Access remote systems via network protocols to open ports
  - Attack vulnerable processes in remote systems
  - Can also use email, remote exec, remote login
- Can have characteristics like a virus:
  - Dormant, triggering, execution, propagation & replication
  - Propagation phase: searches for other systems to infect
  - May disguise itself as a system process when executing
- Morris Worm, the first and the best know worm, 1988
  - released by Robert Morris Jr., paralyzed the Internet (of 1988)
  - exploited vulnerabilities in UNIX systems
- WannaCry Worm, epidemic infection in May 2017
  - exploits known, but unpatched, vulnerability in Windows XP

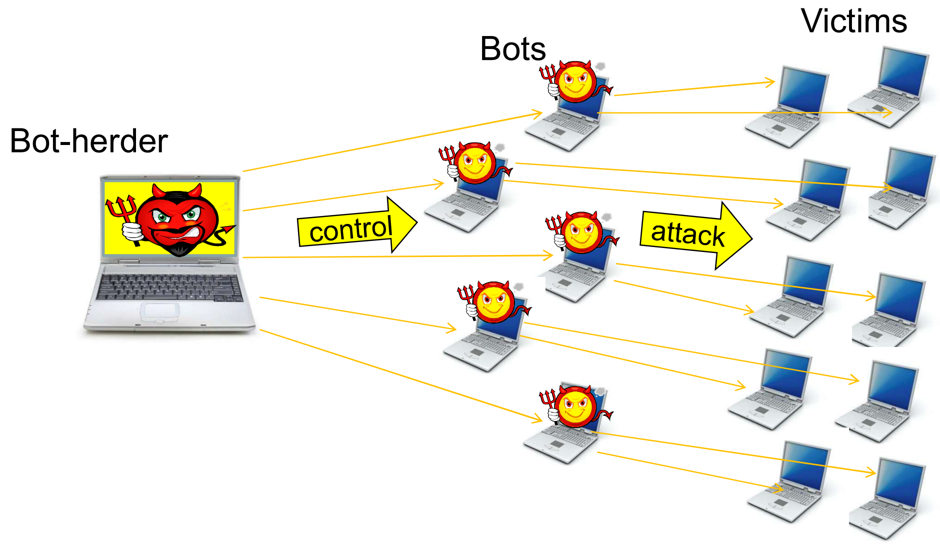
Worm Propagation Speed



What is a botnet ?

- **A botnet** is a collection of computers infected with malicious software agents (robots) that can be controlled remotely by an attacker.
- Owners of bot computers are typically unaware of infection.
- Botnet controller is called a "bot herder" or "bot master"
- Botnets execute malicious functions in a coordinated way:
  - Send spam email
  - Collect identity information
  - Denial of service attacks
  - Create more bots
  - Bitcoin mining
- A botnet is typically named after the malware used to infect
- Multiple botnets can use the same malware, but can still be operated by different criminal groups

Botnet Architecture


Victims

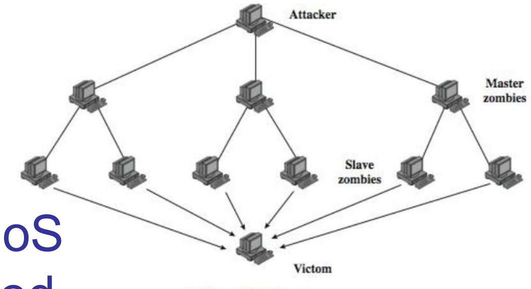
Bots

Bot-herder

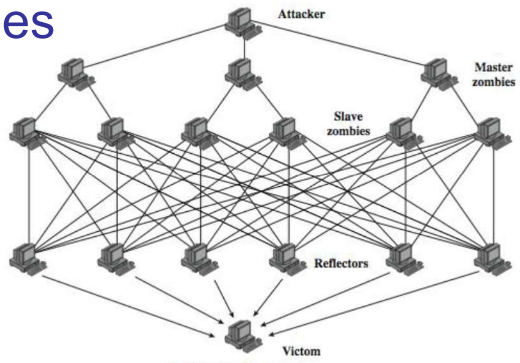
control

attack

DDoS Flood Types



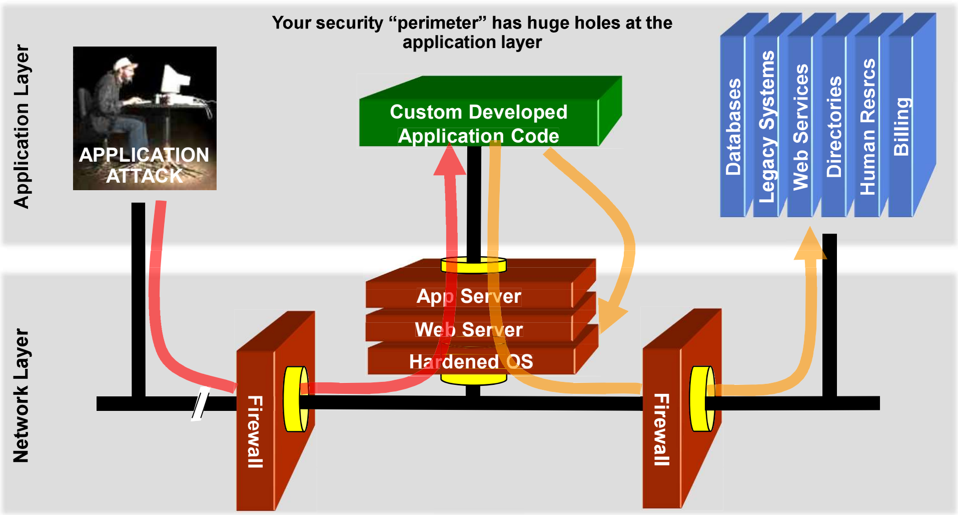
(a) Direct DDoS Attack



(b) Reflector DDoS Attack

- Direct attack
  - Bots send traffic with own or spoofed sender address to victim
- Reflected attack
  - Bots send traffic to innocent hosts with victim address as sender address. Innocent hosts become part of attack by replying to victim.

The web application security challenge


Your security "perimeter" has huge holes at the application layer

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

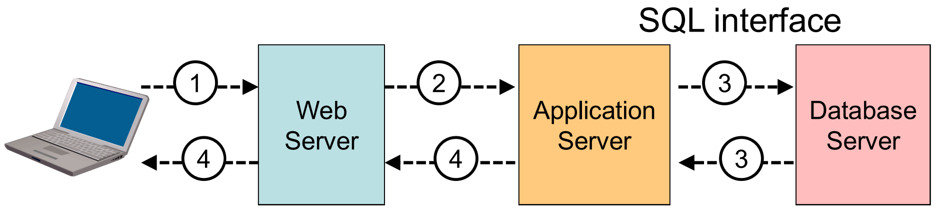
What is SQL?

- Structured Query Language: interface to relational database systems.
- Allows for insert, update, delete, and retrieval of data in a database.
- ANSI, ISO Standard, used extensively in web applications.
- Example:

```
select ProductName from products where
ProductID = 40;
```

SQL at back-end of websites

1. Take input from a web-form via HTTP methods such as POST or GET, and pass it to a server-side application.
2. Application process opens connection to SQL database.
3. Query database with SQL and retrieve reply.
4. Process SQL reply and send results back to user.



What is SQL Injection?

- Database system misinterpretation of input data
  - Attacker disguises SQL commands as data-input
  - Disguised SQL commands = ‘injected’ SQL commands
- With SQL injection, an attacker can get complete control of database
  - no matter how well the system is patched,
  - no matter how well the firewall is configured,
- Vulnerability exists when web application fails to sanitize data input before sending to it database
- Flaw is in web application, not in SQL database.

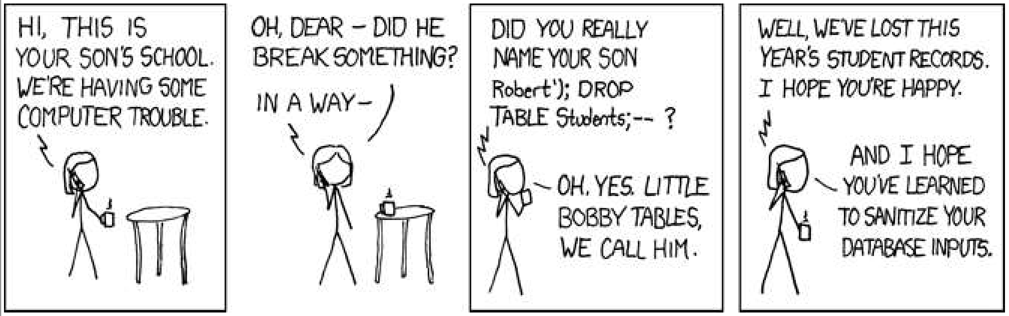
What is SQL Injection?

- For example, if input field ask for a product number, but the malicius user inputs “**40 or 1 = 1**”
- The result SQL command becomes:

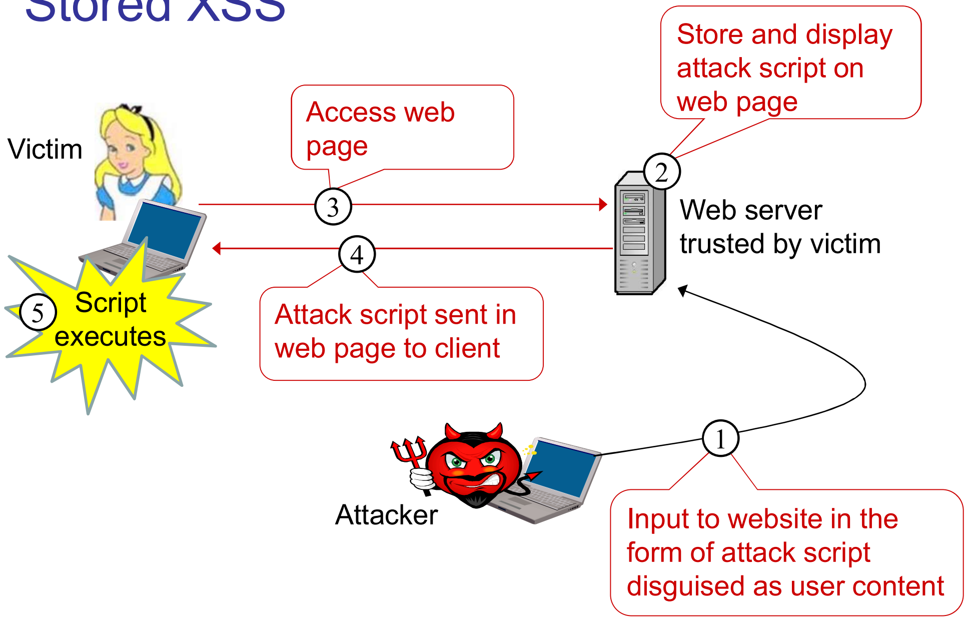
```
select ProductName from products where
  ProductID = 40 or 1 = 1
```

- 1=1 is always TRUE so the “where” clause will always be satisfied, even if ProductID ≠ 40.
- All product records will be returned.
- Data leak.

XKCD – Little Bobby tables


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH, YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Stored XSS



Store and display attack script on web page

Access web page

Attack script sent in web page to client

Input to website in the form of attack script disguised as user content

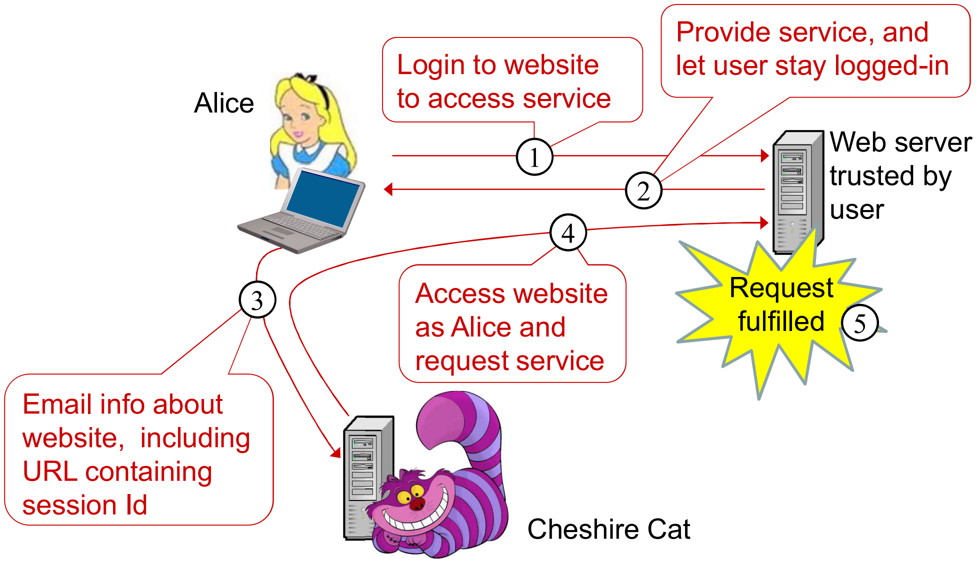
Stored XSS

- Data provided by users to a web application is stored persistently on server (in database, file system, …) and later displayed to users in a web page.
- Typical example: online message boards.
- Attacker uploads data containing malicious script to server.
- Every time the vulnerable web page is visited, the malicious script gets executed in client browser.
- Attacker needs to inject script just once.

Preventing SQL Injection and XSS

- Validate all user entered parameters
  - CHECK data types and lengths
  - DISALLOW unwanted data (HTML tags, JavaScript, SQL commands)
  - ESCAPE questionable characters (ticks, --,semi-colon, brackets, etc.)
- Hide information about Error handling
  - Error messages divulge information that can be used by hacker
  - Error messages must not reveal potentially sensitive information

Broken Authentication and Session Mgmt



Login to website to access service

Provide service, and let user stay logged-in

Email info about website, including URL containing session Id

Access website as Alice and request service

Request fulfilled

Broken Authentication and Session Mgmnt Problem and Fix

- User authentication does not necessarily provide continuous authentication assurance
  - User authentication is only at one point in time
- Insecure implementation of session control with a static session Id which is passed in the URL
  - Unfortunately this can be misused
- Recommendations for session Id must be followed
  - E.g friom OWASP
- Examples of controls for session Id:
  - Link session Id to e.g. IP address, TLS session Id
- .

OWASP

The Open Web Application Security Project

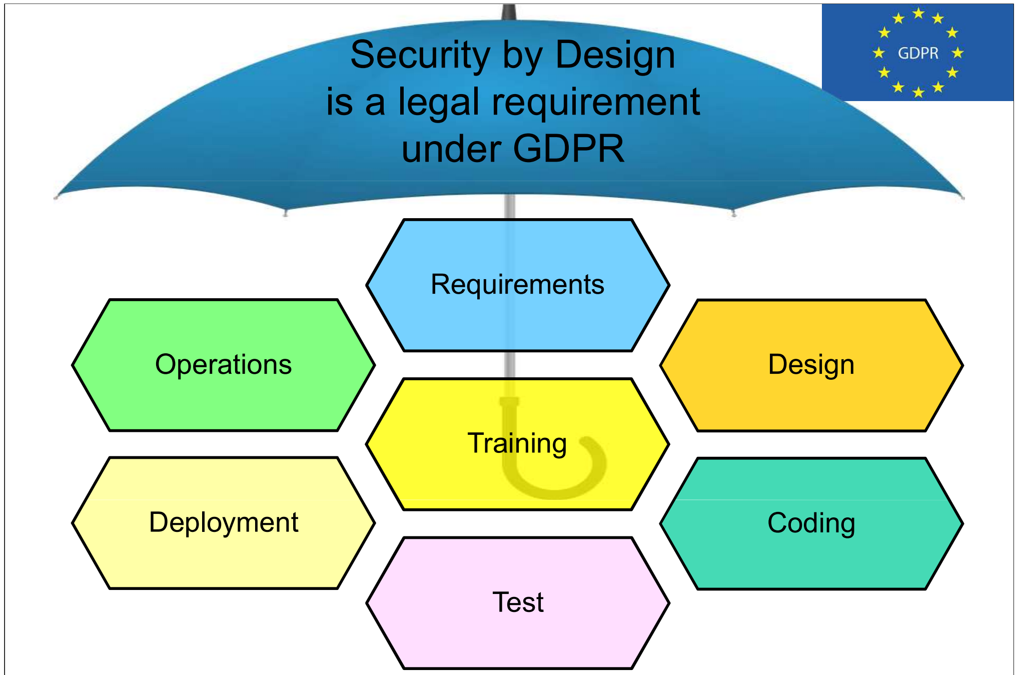
- Non-profit organisation
  - Local chapters in most countries, also in Norway
- OWASP promotes security awareness and security solutions for Web application development.
- OWASP Top-10 security risks identify the most critical security risks of providing online services
  - The Top 10 list also recommends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

Top-10 Web Application Risks

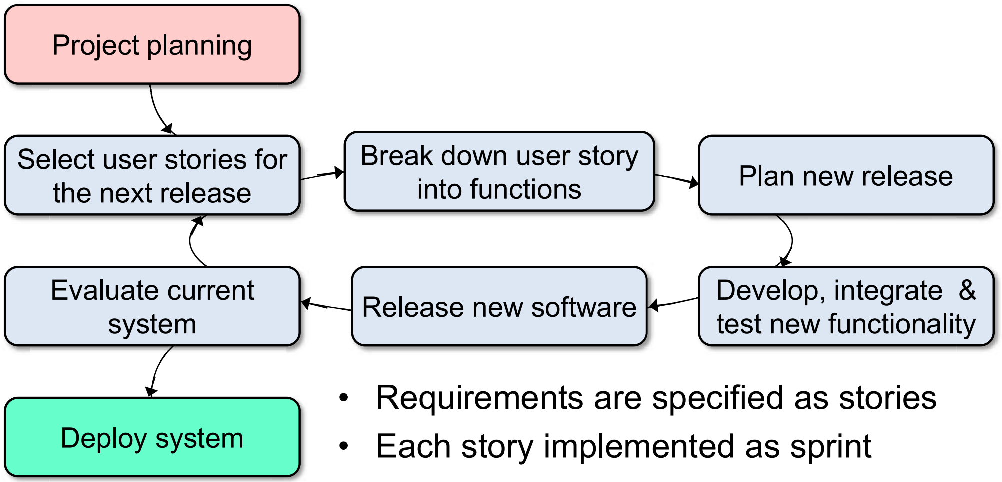
1. Injection
2. Broken Authentication and Session Management
3. Cross-Site Scripting (XSS)
4. Insecure Direct Object References
5. Security Misconfiguration
6. Sensitive Data Exposure
7. Missing Function Level Access Control
8. Cross-Site Request Forgery (CSRF)
9. Using Components with Known Vulnerabilities
10. Unvalidated Redirects and Forwards

General Data Protection Regulation

- EU legislation is translated unchanged
- Paragraphs §25 and §32 are particularly relevant to security
- Requirement of privacy by design and security by design
- IT designers must know the principles of privacy by design and security by design,
  - otherwise, the system are illegal to use by definition
- Fines up to € 20,000,000 or 4% of revenue
- The Norwegian Parliament Stortinget decided on 10 April 2018 that IT education programs must have mandatory courses in information security
- ttps://www.tekna.no/aktuelt/tekna-gjennomslag-om-ikt-sikkerhet-i-utdanningen/

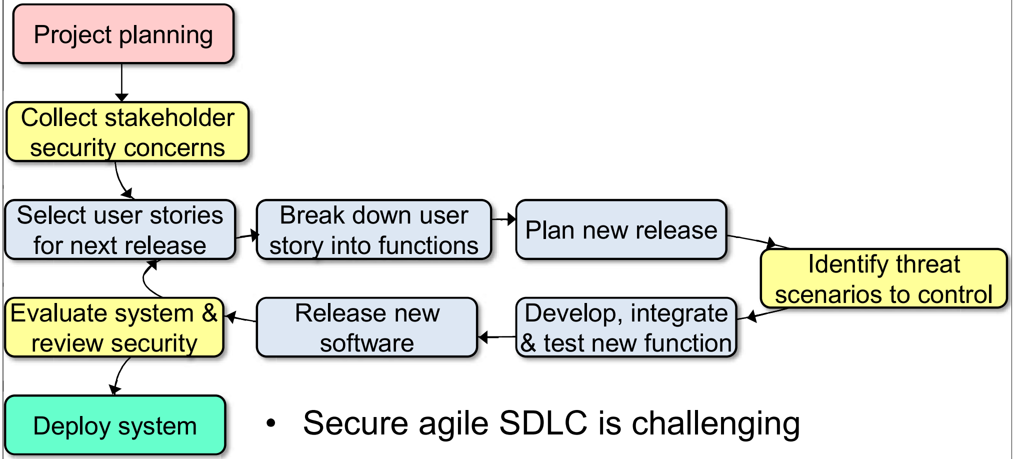


Agile Software Development


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

End of Lecture