What I want to share

• Secure software
  - Development process and security development lifecycle
  - Threat modeling
  - Security requirements
  - Design vs. implementation
  - Microsoft SDL

• Mycompany’s demo
  - Dynamic analysis
  - Client-side and server-side penetration tests
  - Network Sniffing
  - Denial of service attacks

• Wrap up
  - More attacks
  - Cyber security resources
Secure software

• Flawed software -> Ignoring security during the design phase
  Add security when functional requirements are satisfied

• Better approach -> Incorporate security into all phases of the development process
Development process

Four common phases:

1. Requirements + security
2. Design + security
3. Implementation + security
4. Testing + security

Where is security?
Development process

Four common phases:

1. Requirements + security
   - Security requirements
   - Abuse cases
2. Design + security
   - Architectural risk analysis
   - Security-oriented design
3. Implementation + security
   - Code review (manual + tools)
4. Testing + security
   - Risk-based security tests
   - Penetration testing

Where is security?
Secure software vs. hardware

• Software
  • Easily changeable
  • Can be weak at security

• Hardware
  • Hard to change
  • Exploiting hardware is not easy
  • Intel AES-NI (crypto instructions)
  • Intel SGX. (encrypted computation)
  • Intel MPX. (memory safety)
Threat modeling

• The threat model makes explicit the adversary’s assumed powers

• The threat model is critically important

• This is part of architectural risk analysis

• STRIDE is a model of threats (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of service, Elevation of privilege)
Example: Network user

As a user who can connect to a network and work with services:

1. Measure size of packets and timing of requests and responses
2. Execute parallel sessions
3. Feed malformed inputs
4. Drop or modify packets

... SQL injection, XSS, Authentication bypass, CSRF, Buffer overflow, ROP, ....
Example: Snooping user

As a malicious user who is connected to a network where others are also working:

1. Read others’ messages
2. Intercept, modify, and duplicate messages
3. Flood the network with invalid packets

...
Example: Co-located user

As a malicious user who has installed a malware on a user’s machine:

1. Read/write users’ files
2. Read/write users’ memory
3. Record user’s keystroke

... Confidential information theft, Encrypting users’ data (Ransomware)
Threat-driven design

• Different threat models can provide different aspects of your software security

• E.g. In the snooping attack scenario encrypting user traffic is important (IPsec, SSL, WPA3, …)

• E.g. In the co-located user scenario providing users with additional means of authentication is necessary
Weak Model = weak security

• Each assumption is a security threat that an adversary can exploit

• #1 encrypted traffic carries no information
  User can gain information by analyzing the size and quantity of the messages

• #2 Timing channels are not threatening to software security
  Measurement of time can reveal crypto-related keys

1. Compare against other systems
2. Understand attack patterns
3. Challenge your assumptions in your design
Security requirements

• Software requirements: what the software should do?

• Security requirements
  • Security goals or policies (secrecy of user’s bank balance)
  • Security mechanisms (passwords)
Kinds of requirements

• Policies
  - Confidentiality: sensitive information should not be leaked
  - Integrity: sensitive information should not be damaged
  - Availability: a system should stay responsive

• Mechanisms
  - Authentication: what is the subject of security policies?
  - Authorization: what action does a subject want to perform?
  - Auditability: determine the circumstances of a breach or misbehavior

<table>
<thead>
<tr>
<th>Title</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confidentiality</td>
<td>Hacking the system directly to display Bob’s account or based on the time the system needs to respond in the authentication phase</td>
</tr>
<tr>
<td>Integrity</td>
<td>Changing the content of a network packet</td>
</tr>
<tr>
<td>Availability</td>
<td>Denial of service attempts to weaken availability</td>
</tr>
<tr>
<td>Authentication</td>
<td>Password, biometrics, multi-factor authentication</td>
</tr>
<tr>
<td>Authorization</td>
<td>Access controls, role-based and user-based permissions</td>
</tr>
<tr>
<td>Auditability</td>
<td>Logging every event in the system, backups</td>
</tr>
</tbody>
</table>
Design vs. implementation

• Flaws are problems in the design phase

• Bugs are the problems in the implementation phase

• It is important to avoid flaws!

• 50% of security problems are design flaws
Design vs. implementation

Different levels of system design decisions:

1. Main actors: processes, subjects, interactions...

2. Identifying the core functionalities and how they need to work together

3. Data structures, functions, libraries, etc.
Security principles

- **Prevention**
  - Eliminate software defects completely
  - Heartbleed bug would have been prevented by a type safe language

- **Mitigation**
  - Reduce the damage from unknown exploitation possibilities
  - Run each browser’s tab in a separate process

- **Detection**
  - Identify the attack and undo the damage
  - Monitoring and taking snapshots periodically
Microsoft SDL – Security Development Lifecycle

• The Microsoft SDL incorporate security and privacy considerations into all phases of the development process, supporting developers build highly secure software, address security compliance requirements, and reduce development costs.

Mycompany – A very bad example!

Client side – C++ application

Server side – PHP application

Password + User’s name

Checks the password + Stores the name

Security through obscurity
Terminology – DoS attacks

• A Denial-of-Service (DoS) or Distributed DoS attacks meant to shut down or slow down a machine or a network

• DoS attacks accomplish this by flooding the target with traffic, which is commonly useless. Sometimes the traffic triggers a crash in the remote program

• The attack is easy to perform for attackers

• ICMP and SYN flood
Terminology – SQL injection

• SQL injection makes it possible to execute malicious SQL statements

• An attacker can insert, update, or delete a record

• The problem is rooted in unchecked inputs
Mycompany - notes

• Obfuscate the code

• Validate the inputs

• Limit the number of request per machine + firewalls

• Encryption + SSL

• Not made-up approaches such as concatenation of digits
What else?

• Footprinting and reconnaissance

• Network scanning

• System hacking

• Malware threats

• Hacking mobile platforms
What else?

• Social engineering
• Session hijacking
• Evading IDS or firewalls
• Hacking web servers
• Cryptography
Certifications in cyber security

• Security+

• Certified Information Systems Security Professional (CISSP)

• Certified Ethical Hacker (CEH)

• Offensive Security Certified Professional (OSCP)
Now you should know

• What is a secure software?

• What is Microsoft SDL?

• How a vulnerability can be exploited?

• What security aspects must be taken into account when writing software?