Table of Contents

- SDLC
- The Design Process
- Threat Modeling
- Security Design Patterns
- Misuse Cases
SDLC

- Requirements
- Design
- Implementation
- Testing
- Maintenance
Security during Requirements

- Functional Requirements
- Security Requirements
- Compliance Requirements
- Privacy requirements
- Open Source frameworks to use
  - Log4j, log4net
- 3rd part software
  - CMS
  - Portal software
- Database or NoSql databases
- Java, .NET, ruby on rails, php,
The Design Process

- Set of blueprints for the system
- Class diagrams and ORM
- UML Models and Data Flow Diagrams
- Deployment Diagrams
- Application Layers and Tiers
- ...
■ Misuse Cases
■ Threat Modeling
■ Security Design Patterns
Misuse Cases
Use Case vs Misuse Case

- Use Case is a sequence of steps by which a **actor** can obtain **value** from a system
- Misuse case is a sequence of steps by which an **actor (attacker)** can **abuse/attack** a system
Value of misuse cases

- Most people are familiar with use cases already
- Can use the same tools used to create misuse cases
- The output can be used by designers/developers
- Can be used to communicate potential risks to stakeholders
- Can go from high level misuse cases to detailed misuse cases
- Defense mechanisms can be enumerated and documented
Misuse Case – Online Shopping

Misuse Case Diagram

- Register
- Login
- Purchase goods
- Pay online
- Check order information
- Steal credentials
- Price tampering
- Bypass payment
- Steal credit card information
- View other customers order information
- DOS

Customer

Hacker
Threat Modeling
Threat Modeling

- Technique to help identify threats, attacks, vulnerabilities, and countermeasures in the context of an application scenario.

The threat modeling activity helps to:

- Identify your security objectives.
- Identify relevant threats.
- Identify relevant vulnerabilities and countermeasures.
Threat Modeling

1. IDENTIFY SECURITY OBJECTIVES
2. APPLICATION OVERVIEW
3. DECOMPOSE APPLICATION
4. IDENTIFY THREATS
5. IDENTIFY VULNERABILITIES
Step 1: Identify Security Objectives

- What can we prevent?
- What do we care most about?
- What is the worst thing that can happen?
- What regulations do we need to be aware of?
Step 2: Identify Trust Boundaries

- Any place where the level of trust changes
- Where are the entry points?
  - Search page
  - Registration page
  - Login
  - Shopping Cart
- Can you trust the data?
- Can you trust the caller?
- Where are the exit points where data is being written back?
- Trust boundary between
  - Application and database
    - Give the user accessing the database minimal privileges
  - Application and web services
    - Validate
  - 3rd party systems
    - More validation
Step 3: Identify Threats

- Brute force attacks against the dictionary store
- Network eavesdropping between browser and Web server to capture client credentials
- Attacker captures authentication cookie to spoof identity
- SQL injection
- Cross-site scripting (XSS) where an attacker injects script code
- Cookie replay or capture, enabling an attacker to spoof identity and access the application as another user
- Information disclosure with sensitive exception details propagating to the client
- Unauthorized access to the database if an attacker manages to take control of the Web server and run commands against the database
- Discovery of encryption keys used to encrypt sensitive data (including client credit card numbers) in the database
- Unauthorized access to Web server resources and static files
Step 4: Identify and Document Vulnerabilities and Counter-Measures

- Armed with a list of threats consider how the application handles these threats.
- Rate the threats
- Sample questions to consider:
  - How, specifically, will input validation be performed in this application?
  - Are we validating all input? How are cookie values validated?
  - What level of logging will be in place? How will this be handled?
  - How will we protect user sessions?
Step 4 contd - Vulnerabilities in components

- Top 10 2013-A9-Using Components with Known Vulnerabilities
  - Remote code vulnerability in Spring Framework for Java
  - .NET padding oracle (now fixed)
  - Apache Struts 2 vulnerability
    - https://cwiki.apache.org/confluence/display/WW/S2-015
Step 5 : Rate the threat

- Risk = Probability * Damage Potential
- 1-10 rating
- High, Medium, Low
- CVSS – Common Vulnerability Scoring System
DREAD

- **Damage potential**: How great is the damage if the vulnerability is exploited?
- **Reproducibility**: How easy is it to reproduce the attack?
- **Exploitability**: How easy is it to launch an attack?
- **Affected users**: As a rough percentage, how many users are affected?
- **Discoverability**: How easy is it to find the vulnerability?
The STRIDE threat system:

- Spoofing
- Tampering
- Repudiation
- Information Disclosure
- Denial of Service
- Elevation of Privilege
Security Design Patterns
Design Patterns

A pattern can be characterized as “a solution to a problem that arises within a specific context”.

A proven solution to a problem.

Idea comes from architecture of buildings (C. Alexander)

Security Design Patterns are a subset
Value of Patterns

- Reusable solutions, but maybe not directly, usually require tailoring
- Encapsulate experience and knowledge of designers (best practices)
- Free of errors after a while
- Need to be catalogued to be useful
- Used as guidelines for design
- Good to evaluate systems and standards
Value of Security Patterns

- Can guide the design and implementation of the security mechanism itself
- Can guide the use of security mechanisms in an application (stop specific threats)
- Extensive catalogues of security patterns have been developed
- Care must be taken in their use
Security Design Patterns examples

- Secure Logger
  - Remote logging for decentralized systems
- Input Validator
  - Validate input against acceptable criteria
- Clear Sensitive Information
- Exception Manager
  - Wrap and sanitize exceptions
Questions?
Microsoft Threat Modeling:

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


- https://www2.opengroup.org/ogsys/catalog/g03
- https://www.owasp.org/index.php/Detail_misuse_cases
- https://www.owasp.org/index.php/OWASP_Secure_Application_Design_Project