IIG University of Freiburg

Web Security, Summer Term 2012

Web Application: Testing Security

Dr. E. Benoist

Sommer Semester

Table of Contents

Introduction and objectives

- **Methodology**
  - First overview of the system
  - List all the entry points
- **Present a check list of controls**
  - Testers should test systematically all controls
- **Write a report**
  - Describe the scope of the testing
  - Write the list of tests done and their result
  - For each of the problems, state its *“risk”*

Testing Techniques

- **Manual Inspections and reviews**
  - human-driven reviews
  - they can be among the most powerful and effective techniques available.
  - Can be time consuming
  - Requires significant human thought and skill to be effective!
- **Threat Modeling**
  - Decomposing the application
  - Defining and classifying the assets
  - Exploring potential vulnerabilities
  - Exploring potential threats

Testing Techniques(Cont.)

- **Code Review**
  - Completeness and effectiveness,Accuracy,Fast (for competent reviewers)
  - Requires highly skilled security developers,
  - Can miss issues in compiled libraries
- **Penetration Testing**
  - *"If you fail a penetration test you know you have a very bad problem indeed. If you pass a penetration test you do not know that you don't have a very bad problem"*

Methodology for Testing

- **Make a map of the site**
  - Technics seen in the first course
- **Find "Entry Points"**
  - DB Listeners
  - SSL /TLS
  - HTTP GET, POST and Cookies parameters
- **Systematically check input validation**
  - Metacharacters for SQL- , XML-, LDAP-, Shell-injection or XSS
  - Do not forget error messages
- **Attack authentication/authorisation mecanism**
- **Test Business logic**

Collect as much information as possible

- **Information Gathering**

Information Gathering

Collect as much information as possible about a target application

- **Spiders, robots and Crowlers**
  - Browse and capture resources related to the application
  - For instance `robots.txt` the crowler policy file (`http://www.anysite.com/robots.txt`)
- **Search engine discovery/ Reconnaissance**
  - *"google"* the web site you are testing, to search for error pages or application structure,
  - Use the "cache" version in google to access information already removed.
- **Identify application entry points**
  - Enumerate the application and its attack surface
  - identify what should be investigated further
  - Pages, Forms (POST and/or GET), links, cookies, redirects (HTTP code 300)

Information Gathering (Cont.)

- **Testing for Web Application Fingerprint**
  - Find which server, which version, which default configuration : Hence, which known vulnerability
  - Use the “server” http head information (can easily be obfuscated or manipulated)
  - Use the fingerprint of an application : the order or http headers, the capitalisation of some http fields, different response to malformed requests.
  - You can use a tool for this, for instance `httprint` or `netcraft`

Information Gathering (Cont.)

- **Application Discovery**
  - identify web applications on a given infrastructure.
  - different base URL: Try "standard" names, like `webmail.anysite.com`, `www.anysite.com/admin`, `mail.anysite.com`, `www.anysite.com/typo3`, ...
  - non standard ports (other than 80 for http and 443 for https),
  - virtual hosts (one IP address, many web sites)
- **Analysis of Error Codes**
  - 404 error page may indicate the OS or Web Server
  - Can indicate which DB server is used (or the SQL query or part of it).

Find Entry Points in the system

- **DB Listeners**
- **SSL/ TLS**
- **HTTP GET, POST and Cookies**

Configuration Managment Testing

- **SSL/TLS Testing**
  - Verify the usage of strong cipher algorithm
  - Use `nmap` for testing the SSL service

    ```
    [root@test]# nmap -F -sV localhost
    ```
  - Identify weak cipher using `Nessus` scanner (reading the certificate)
  - Badly configured clients may use the weak algorithms to connect.
- **DB Listener Testing**
  - Can reveal configuration settings of Oracle server
  - Serves as input to more impacting follow-on tests.
  - Tests if the listener is open on port 1521 for Oracle (or port 2483 for TNS or 2484 for TNS with SSL) or any other port
  - Can be subject to brute force attacks for accessing the DB

Configuration Managment Testing (Cont.)

- **Infrastructure configuration management testing**
  - Need to find all elements in the infrasctructure and their interactions
  - All elements need to be reviewed in order to make sure they don't hold any known vulnerability
  - Review: Firewall, OS, server, Database, LDAP (or any authentication mecanism). . . .
  - List all possible administrative interfaces,
  - Available from internal / external network
  - How access control is determined?
  - Change default user and password

Configuration Managment Testing (Cont.)

- **Application configuration management testing**
  - Test for default installation and sample applications or files
  - Review comments
  - Test that only needed server modules are installed
  - Server errors point to custom-made pages (even if information is usefull while develping)
  - Verify that the server runs with minimised privileges in the OS.
  - Make sure that server logs all access and errors
  - Ensure that the server handles overloads and prevent Denial of Service attacks.

Configuration Managment Testing (Cont.)

- **Testing for File extensions handling**
  - Determine the way the server handles requests corresponding to files having different extensions may help to understand server behaviour.
  - Determine what is returned as text and what is executed on the server
  - For example `connection.inc` is returns:

```
<?php
  mysql_connect("127.0.0.1", "root", "")
  or die("Could not connect");
?>
```

  - Examples of sensitive extensions:
    `.asa`,`.inc`,`.zip`,`.tar`,`.tgz`,(or other compressed files extensions) `.java`,`.txt`,`.pdf`,`.doc`,`.rtf`, ...

Configuration Managment Testing (Cont.)

- **Old, backup and unreferenced files**
  - some files may still be present `index.php.back` or `login.asp.old`
  - Archives used for deployment may be accessible: `myapplication.zip`
  -
- **Infrastructure and Application Admin interfaces**
  - Test for known directories ((`admin` or `/administrator`)
  - Test default admin platform for the CMS
  - Parameter Tampering or Cookie for enabling admin fonctionalities (`?admin=1`).
- **Testing for HTTP Methods and XST**
  1. Which methods are available
  2. GET / POST but also HEAD, PUT, DELETE, TRACE, OPTIONS, CONNECT

Check Input Validation

- **Data Validation Testing**
- **Try different Metacharacters and Check their validation resp. escaping**

Data Validation Testing

- **Testing for Reflected Cross Site Scripting**
  - Test all the fields entered by the users where they are displayed directly
- **Testing for Stored Cross Site Scripting**
  - Test if the user can enter XSS in the DB, and how it is displayed
- **Testing for DOM based Cross Site Scripting**
  - Testing what can be accessed by JavaScript
- **Testing for Cross Site Flashing**
  - Similar to DOM based XSS but for Flash
- **SQL Injection (depending on Data Base)**
  - Test escaping chars, comments, etc.
  - Use a tool like `sqlmap`
- **LDAP Injection**
  - If the authentication tool use LDAP,
  - Trys to fool LDAP, to grant more access
  - Or access informations about other users

Data Validation Testing (Cont.)

- ORM Injection
- XML Injection
- SSI Injection
- XPath Injection
- IMAP/SMTP Injection
- Code Injection
- OS Commanding
- Buffer overflow Testing
- Testing for HTTP Splitting/Smuggling

Authentication and Authorisation Testing

- **Authentication Testing**
- **Session Management Testing**
- **Authorization Testing**

Authentication Testing

- **Credentials transport over an encrypted channel**
  - data that users put into the web form, in order to log into a web site, are transmitted using secure protocols that protect them from an attacker or not.
- **Testing for user enumeration**
  - Is it possible to collect a set of valid users by interacting with the authentication mechanism of the application? (for brute force attack).
- **Default of guessable (dictionary) user account**
- **Testing Brute Force**
  - Brute force testing is not easy to accomplish for testers because of the time required and the possible lockout of the tester.
- **Testing by Bypassing authentication schema**
  - Not all resources are enought protected,
  - First page may require login, but not the second,
  - Admin pages may be available without the right login.

Authentication Testing (Cont.)

- **Testing for Vulnerable remember password and pwd reset**
  - Password reset in the form
  - Remember me functionality activated
- **Testing for Logout and Browser Cache Management**
  - the logout and caching functions are properly implemented.
  - So that someone can not access pages using the back function of the browser
- **Testing for Captcha**
  - The value of the captcha may be sent by the client (in a hidden field)
  - Possibility of replay attacks

Authentication Testing (Cont.)

- **Testing for Multiple factors Authentication**
  - One-time password (OTP) generator tokens
  - Crypto devices like USB stiks or smart cards with certificates
  - Random OTP sent via SMS
  - Personal information
- **Testing for Race Conditions**
  - Flaw occurring when multiple users access the service at the same time (difficult to test for)

Session Management Testing

- **Testing for Session Management Schema**
  - Understand how the Session Management mechanism has been developed and if it is possible to break it to bypass the user session.
- **Testing for Cookies attributes**
  - Test the attributes of cookies,
    secure : it is only sent back using HTTPS
    httponly: can not be used in JavaScript
    domain or path : cookie only sent to the legitimate issuer
- **Testing for Session Fixation**
  - When an application does not renew the cookie after a successful user authentication, it could be possible to find a session fixation vulnerability and force a user to utilize a cookie known to the attacker.

Session Management Testing (Cont.)

- **Testing for Exposed Session Variables**
  - Session Tokens represent confidential information because they tie the user identity with his own session.
- **Testing for CSRF**
  - Test if actions use a specific tocken
  - Test if sensitive actions (change password, money transfer) need a second authentication (or any form of TAN)

Authorization testing

- **Testing for path traversal**
  - First, we test if it is possible to find a way to execute a path traversal attack and access reserved information
  - Aggressor could exploit the system in order to read/write files that are not intended to be accessible.
- **Testing for bypassing authorization schema**
  - Verifying how the authorization schema has been implemented for each role/privilege to get access to reserved functions/resources.
- **Testing for Privilege Escalation**
  - horizontal escalation : Access resources granted to similarly configured account
  - vertical escalation : access resources granted to more privileged accounts

Business Logic Testing

- **Logic in the application**
- **Ajax functions**

Business Logic testing

- **Attacks against the logic of the business**
- **Example 1**
  - Setting the quantity of a product on an e-commerce site as a negative number may result in funds being credited to the attacker.
- **Example 2**
  - Accessing directly to the validation of an order without having payed for it
- **Example 3**
  - Being able to enter data with another identity

AJAX Testing

- **AJAX Vulnerabilities**
  - Increased attack surface with many more inputs to secure
  - Exposed internal functions of the application
  - Client access to third-party resources with no built-in security and encoding mechanismes
  - Failure to protect authentication information and sessions
  - Blurred line between client-side and server-side code, resulting in security mistakes
- **Testing for AJAX**
  - Application Discovery : more complicated (larger), more easy (code available)
  - Discover framework used
  - Find Endpoints: entry points

Denial of Service Testing

- Testing for SQL Wildcard Attacks
- Locking Customer Accounts
- Buffer Overflows
- User Specified Object Allocation
- User Input as a Loop Counter
- Writing User Provided Data to Disk
- Failure to Release Resources
- Storing too Much Data in Session

Writing Reports
How to Value the real risk

- **OWASP Risk Rating methodology**
  - Standard risk model : **risk = likelihood * Impact**
- **Steps:**
  1. Identifying a risk
  2. Factors for estimating likelihood
     Threat agent factors
     Vulnerability factors
  3. Factors for estimating impact
     Technical impact
     business impact
  4. Determining the severity of the risk
     likelihood * impact
  5. Deciding what to Fix
  6. Customizing your risk rating model

How to Value the real risk (Cont.)

- **Threat agent factors**
  Skill level (No technical skills (1), some technical skills (3), advanced computer user (4), network and programming skills (6), security penetration skills (9))
  Motive (Low or no reward (1), possible reward (4), high reward (9))
  Opportunity (No known access (0), limited access (4), full access (9))
  Size (How large is this group of attackers? Developers (2), system administrators (2), intranet users (4), partners (5), authenticated users (6), anonymous Internet users (9))

How to Value the real risk (Cont.)

- **Vulnerability Factors**
  Ease of discovery (Practically impossible (1), difficult (3), easy (7), automated tools available (9))
  Ease of exploit (Theoretical (1), difficult (3), easy (5), automated tools available (9))
  Awareness (Unknown (1), hidden (4), obvious (6), public knowledge (9))
  Intrusion detection (Active detection in application (1), logged and reviewed (3), logged without review (8), not logged (9))

How to Value the real risk (Cont.)

- **Technical impact**
  Loss of confidentiality (Minimal non-sensitive data disclosed (2), minimal critical data disclosed (6), extensive non-sensitive data disclosed (6), extensive critical data disclosed, all data disclosed (9))
  Loss of integrity (Minimal slightly corrupt data (1), minimal seriously corrupt data (3), extensive slightly corrupt data (5), extensive seriously corrupt data, (7) all data totally corrupt (9))
  Loss of availability (Minimal secondary services interrupted (1), minimal primary services interrupted (5), extensive secondary services interrupted (5), extensive primary services interrupted (7), all services completely lost (9))
  Loss of accountability (Fully traceable (1), possibly traceable (7), completely anonymous (9))

How to Value the real risk (Cont.)

- **Business Impact**
  Financial damage (Less than the cost to fix the vulnerability (1), minor effect on annual profit (3), significant effect on annual profit (7), bankruptcy (9))
  Reputation damage (Minimal damage (1), Loss of major accounts (4), loss of goodwill (5), brand damage (9))
  Non-Compliance (Minor violation (2), clear violation (5), high profile violation (7))
  Privacy Violation (One individual (3), hundreds of people (5), thousands of people (7), millions of people (9))

How to Value the real risk (Cont.)

- **Risk = (Thread agent factors + Vulnerability Factors)/2 * (technical impact + business impact)/2**

How to Write the report of the testing

- **Executive Summary**
- **Technical Management Overview**
  1. Contains more details
  2. Describe the scope of the assessment
  3. The targets included and any caveats, such as system availability
  4. Introduction on the risk rating used
  5. Technical summary of the findings

How to Write the report of the testing

- **Assessment Findings**
  Aims at the technical level, should include all the necessary information
  1. For each of the previous controls
  2. Write the Finding, a technical description of the issue
  3. A section resolving the issue
  4. The risk rating and impact value

  Write if a control has been done, and its result.
- **Toolbox**
  1. Describes the methodology used
  2. Describes the tools technological tools used for testing
  3. Gives an idea of the thoroughness of the assessment and an idea what areas were included.

Conclusion

- **Testing**
  - Lot of formalisme: Don't forget an issue
  - Lot of technics : Combine all possible new findings
  - Bigest issue: Social engineering!!!
    Send an email to the wive of an executive containing a trojan horse
    Listen to the Wifi network of a manager at home
    ...

Bibliography

- **OWASP Testing Guide V3**[1]
  http://www.owasp.org/index.php/Category:OWASP_Testing_Project
- HTTprint a tool for reading server fingerprint.
  http://net-square.com/httprint/index.shtml
- Netcraft Web site summery
  http://www.netcraft.com

[1] Is the main source for this course