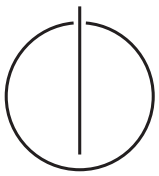
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Secure Coding Phase 5

Team 3: Aurel Roci, Stefan Kohler



Executive Summary

We managed to fix the most of the crucial vulnerabilities of the web application.

Contents

Executive Summary **i**

1 Time Tracking Table **1**

2 Application Architecture **2**

3 Security Measures **3**

4 Fixes **4**

4.0.1 Testing for Weak lock out mechanism(OTG-AUTHN-003) 4
4.0.2 Testing Directory traversal/file include (OTG-AUTHZ-001) . . . 4
4.0.3 Test Session Timeout(OTG-SESS-007) 5
4.0.4 Testing for Reflected Cross Site Scripting(OTG-INPVAL-001) . . 5
4.0.5 Testing for Stored Cross Site Scripting(OTG-INPVAL-002) 6
4.0.6 Testing for SQL Injection (OTG-INPVAL-005) and Mysql testing (OTG-INPVAL-005) 6
4.0.7 Testing for Code Injection, Testing for Local File Inclusion, Testing for Remote File Inclusion(OTG-INPVAL-012) 7
4.0.8 Testing for Command Injection(OTG-INPVAL-013) 7
4.0.9 Testing for incubated vulnerabilities(OTG-INPVAL-015) 7
4.0.10 Analysis of Error Codes (OTG-ERR-001) 7
4.0.11 Test Business Logic Data Validation(OTG-BUSLOGIC-001) 8
4.0.12 Test Integrity Checks(OTG-BUSLOGIC-003) 8
4.0.13 Testing for JavaScript Execution(OTG-CLIENT-002) 8
4.0.14 Testing for HTML Injection(OTG-CLIENT-003) 8
4.0.15 Testing for CSS Injection(OTG-CLIENT-005) 8

1 Time Tracking Table

Name	Task	Time
Aurel Roci	Test File Extensions Handling for Sensitive Information and documenting	1
	Test HTTP Methods and documenting	1
	Test HTTP Strict Transport Security and documenting	1
	Test RIA cross domain policy	1
	Error Handling	0.5
	Testing for default credentials	0.5
	Testing for Reflected Cross Site Scripting and documenting	0.5
	Testing for Stored Cross Site Scripting and documenting	0.5
	Testing for HTTP Verb Tampering and documenting	0.5
	Testing for SQL Injection and documenting	2
	Test Number of Times a Function Can be Used Limits	0.5
	Test Business Logic Data Validation and documenting	1
	Testing for Bypassing Session Management Schema and documenting	1
	Testing for Cookies attributes and documenting	1
	Testing for Session Fixation and documenting	1
	Testing for Exposed Session Variables and documenting	1
	Testing for Cross Site Request Forgery and documenting	1
	Testing for logout functionality and documenting	1
	Test Session Timeout and documenting	1
	Testing for Session puzzling and documenting	1
	Testing Report	2
	Testing for Cross Site Request Forgery	0.5
	Testing for Privilege Escalation	1
	Presentation	0.25
Stefan Ch. Kofler	Reverse-Engineer the binary file	10
	Binary-equivalent	9
	Decompile jar file	1

2 Application Architecture

3 Security Measures

4 Fixes

4.0.1 Testing for Weak lock out mechanism(OTG-AUTHN-003)

After 3 wrong attempts to login the account now is blocked and an employee has now to approve it, for the user to be able to login.
online_banking/models/user.php lines 177-178:

```
$query = "UPDATE users SET approved = 0 WHERE
               username='".mysql_real_escape_string($username)."';";
$result = mysql_query($query);
```

After the user tries to log 3 times and fails the account will be blocked until an employee approves it again. So if the attacker deletes the session and tries again the account is blocked.

4.0.2 Testing Directory traversal/file include (OTG-AUTHZ-001)

Fixed by fixing Command Injection and Local File Inclusion

4.0.3 Test Session Timeout(OTG-SESS-007)

Added a functionality in *online_banking/init.sec.php* in lines 17-21 :

```
(isset($_SESSION['LAST_ACTIVITY']) && (time() - $_SESSION['LAST_ACTIVITY'] > 900)) {
session_unset();
session_destroy();
}
```

This checks the last activity done in the website. If there is no activity for 15 minutes then the session is destroyed.

4.0.4 Testing for Reflected Cross Site Scripting(OTG-INPVAL-001)

Insert a code in line 77 of the *online_banking/employee.php* :

```
$search = htmlspecialchars($search);
```

and line 54 of *online_banking/passwordReset.php*:

```
htmlspecialchars($_GET['id'])
```

to sanitize the input, so no code can be executed.

4.0.5 Testing for Stored Cross Site Scripting(OTG-INPVAL-002)

Insert code in lines 17 of *online_banking/customer.inc.php* to sanitized the input from XSS with the following code:

```
$description = htmlspecialchars($description);
```

Changed the code in line 22 of *online_banking/register.inc.php* from :

```
if(preg_match("/^$passwordRegex/", $password))
to:
if(preg_match("/^$emailRegex/", $email))
```

There was an if statement that checked for the password written twice, and we changed one to check for the email.

4.0.6 Testing for SQL Injection (OTG-INPVAL-005) and Mysql testing (OTG-INPVAL-005)

Modified the query in line 131 of the *online_banking/models/user.php* file to sanitize the input:

```
$query = "UPDATE users SET approved = 1 WHERE
                    id='".mysql_real_escape_string($user_id)."';";
```

Also we sanitized the input in lines 137-140:

```
$user_id = mysql_real_escape_string($user_id);
$balance = mysql_real_escape_string($user_id);
$user_id = htmlspecialchars($user_id);
$balance = htmlspecialchars($balance);
```

This should prevent *SQL Injections*.

4.0.7 Testing for Code Injection, Testing for Local File Inclusion, Testing for Remote File Inclusion(OTG-INPVAL-012)

Inserted a function when we read the variable in *online_banking/applicationDownload.php* line 2 :

```
$user_id = intval($_GET['user_id']);
```

to receive only integer values. This will prevent any code injections.

4.0.8 Testing for Command Injection(OTG-INPVAL-013)

Inserted code in line 3 of the *online_banking/applicationDownload.php* :

```
$user_id = escapeshellarg($user_id);
```

This way the shell commands are removed from the variable

4.0.9 Testing for incubated vulnerabilities(OTG-INPVAL-015)

This vulnerability was fixed by fixing XSS, SQL Injection, Code Injection, Command Injection.

4.0.10 Analysis of Error Codes (OTG-ERR-001)

Changed the PHP *error_reporting()* such that it does not display any errors: *online_banking/init.sec.php*; *online_banking/customer.sec.php*

4.0.11 Test Business Logic Data Validation(OTG-BUSLOGIC-001)

Added a server side check using regular expression in *online_banking/customer.sec.php* line 21:

```
if($amount<0 || preg_match("/^$amountRegex/", $amount))
```

line 8:

```
$amountRegex = "^\d*\.?\d*$";
```

This way the check for the amount of money to transfer is server-side too. This will prevent any input of incorrect amount, even if the client-side check is bypassed.

4.0.12 Test Integrity Checks(OTG-BUSLOGIC-003)

Fixed by fixing Injection vulnerabilities.

4.0.13 Testing for JavaScript Execution(OTG-CLIENT-002)

Fixed by fixing Injection vulnerabilities.

4.0.14 Testing for HTML Injection(OTG-CLIENT-003)

Fixed by fixing Injection vulnerabilities.

4.0.15 Testing for CSS Injection(OTG-CLIENT-005)

Fixed by fixing Injection vulnerabilities.