



DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Report

## **Black Box Testing Report**

Alexis Engelke, Johannes Fischer, Ralph Schaumann,  
Saurabh Nawalgaria





DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Report

# Black Box Testing Report

Author: Alexis Engelke, Johannes Fischer, Ralph Schaumann, Saurabh Nawalgaria  
Team: 9  
Lecture: Secure Coding, Phase 2  
Submission Date: November 24, 2015



# **Executive Summary**

# Contents

<b>Executive Summary</b>	<b>ii</b>
<b>1 Time Tracking</b>	<b>1</b>
<b>2 Vulnerabiliteis Overview</b>	<b>2</b>
2.1 Online Banking . . . . .	2
2.1.1 Stored XSS in Registration and Transaction Description . . . . .	2
2.1.2 Missing check for amount in transactions from batch file . . . . .	2
2.1.3 SQL injection in transaction batch file . . . . .	2
2.1.4 Some critical vulnerability . . . . .	3
2.2 SecureBank . . . . .	3
<b>3 Tools</b>	<b>4</b>
3.1 Zed Attack Proxy (ZAP) . . . . .	4
3.2 SQLmap . . . . .	4
<b>4 Detailed Report</b>	<b>5</b>
4.1 Configuration and Deploy Management Testing . . . . .	6
4.1.1 Test File Extensions Handling for Sensitive Information . . . . .	6
4.1.2 Test HTTP Methods . . . . .	12
4.1.3 Test HTTP Strict Transport Security . . . . .	13
4.1.4 Test RIA cross domain policy . . . . .	14
4.2 Identity Management Testing . . . . .	15
4.2.1 Test Role Definitions . . . . .	15
4.2.2 Test User Registration Process . . . . .	17
4.2.3 Test Account Provisioning Process . . . . .	19
4.2.4 Testing for Account Enumeration and Guessable User Account .	20
4.2.5 Testing for Weak or unenforced username policy . . . . .	21
4.3 Authentcation Testing . . . . .	22
4.3.1 Testing for Credentials Transported over Encrypted Channel . .	22
4.3.2 Testing for default credentials . . . . .	24
4.3.3 Testing for Weak lock out mechanism . . . . .	25

## Contents

---

4.3.4	Testing for bypassing authentication schema . . . . .	27
4.3.5	Testing for Vulnerable Remember Password . . . . .	29
4.3.6	Testing for Browser Cache Weakness . . . . .	30
4.3.7	Testing for Weak password policy . . . . .	31
4.3.8	Testing for Weak security question/answer . . . . .	32
4.3.9	Testing for Weak password change or reset functionalities . . . . .	32
4.3.10	Testing for Weaker authentication in alternative channel . . . . .	32
4.4	Authorization Testing . . . . .	33
4.4.1	Testing Directory traversal/file include . . . . .	33
4.4.2	Testing for bypassing authorization schema . . . . .	33
4.4.3	Testing for Privilege Escalation . . . . .	33
4.4.4	Testing for Insecure Direct Object References . . . . .	33
4.5	Session Management Testing . . . . .	33
4.6	Data Validation Testing . . . . .	33
4.6.1	Testing for Reflected Cross Site Scripting . . . . .	33
4.6.2	Testing for LDAP/ORM/XML/SSI/XPath Injection . . . . .	34
4.6.3	Testing for SQL injection . . . . .	35
4.7	Error Handling . . . . .	38
4.7.1	Analysis of Error Codes . . . . .	38
4.7.2	Testing for Stack Traces . . . . .	40
4.8	Testing for weak Cryptography . . . . .	40
4.9	Business Logic Testing . . . . .	40
4.10	Client Side Testing . . . . .	40
<b>Acronyms</b>		<b>41</b>

# 1 Time Tracking

If a task is prefixed with (o), it refers to the Online Banking web application, if a task is prefixed with (s), the task refers only to the SecureBank web application.

Table 1.1: Time Tracking Table

Name	Task	Time
Alexis Engelke	Setting up LaTeX template	1
Alexis Engelke	(o) Analyzing XSS vulnerabilities using ZAP	2
Alexis Engelke	(o) Analyzing SQL injection vulnerabilities in the web interface using SQLmap	1.5
Alexis Engelke	(o) Analyzing SQL injection vulnerabilities in the file upload	2
Alexis Engelke	(o) Exploiting the TAN verification in the file upload	2
Alexis Engelke	(o) Documenting SQL injection	1
Alexis Engelke	Testing and Documenting Configuration and Deploy Management Testing	2
Alexis Engelke	Testing and Documenting Identity Management Testing	1
Alexis Engelke	Testing and Documenting Authentication Testing	2
Foo	Fixing all issues	10

## 2 Vulnerabiliteis Overview

Through our testing, we identified the following vulnerabilities as the most critical for the Online Banking application and the SecureBank:

### 2.1 Online Banking

#### 2.1.1 Stored XSS in Registration and Transaction Description

- Likelihood: high
- Implication: high
- Risk: high

With stored cross site scripting attacks it is possible to inject JavaScript code, which is run whenever an employee logs in and opens the list of unapproved accounts or transactions. It is also possible to inject script from other sites.

#### 2.1.2 Missing check for amount in transactions from batch file

- Likelihood: medium
- Implication: high
- Risk: high

It is possible to get money from another client of the bank by filling in a negative number in the amount field of a transaction batch file. Therefore, one client can generate an infinite amount of money, while reducing the amount of money of other clients.

#### 2.1.3 SQL injection in transaction batch file

- Likelihood: medium
- Implication: high

- Risk: high

The application is vulnerable to SQL injections in the transaction batch files. Therefore, it is possible to perform transactions while using any unused TAN in the system, which is not known to the attacker and might come from another client.

### 2.1.4 Some critical vulnerability

- Likelihood: high
- Implication: high
- Risk: high

The web application is vulnerable.

## 2.2 SecureBank



## 3 Tools

### 3.1 Zed Attack Proxy (ZAP)

Using the Zed Attack Proxy (ZAP), we were able to reveal significant parts of the directory structure in both web applications. In the *Online Banking* web application, we found a stored XSS vulnerability in the registration and the transaction description as well as a SQL injection vulnerability in the login form using the fuzzer. We were also be able to find a buffer overflow vulnerability for the transaction description in the transaction batch files.

### 3.2 SQLmap

Using SQLmap, we found the SQL injection vulnerability in the login form, which we found using ZAP earlier. SQLmap did not reveal further SQL injection possibilities.



## 4 Detailed Report

### 4.1 Configuration and Deploy Management Testing

#### 4.1.1 Test File Extensions Handling for Sensitive Information

##### Online Banking

<b>Observation</b>	We found various files which are served as plain text but are PHP source files. One of these files contains the credentials of the mail server. We were also able to download the compiled executable as well as the source code of the batch file parser.
<b>Discovery</b>	Using the OWASP ZAP tool, we used the forced browse functionality on /InternetBanking/. We received a list of files which were found using this tool, see below.
<b>Likelihood</b>	This can be tested by anyone who enters specific strings into the address bar of a browser. However, the likelihood of this vulnerability is much higher if the attacker uses specific tools which test specific paths systematically.
<b>Impact</b>	The attacker can get sensitive information, e.g. credentials to the mail server or the database. He can analyze the source of the parser and find vulnerabilities there.
<b>Access Vector</b>	Network
<b>Access Complexity</b>	Low
<b>Privileges Required</b>	None
<b>User Interaction</b>	None
<b>Scope</b>	Unchanged
<b>Confidentiality</b>	High
<b>Integrity</b>	No Impact
<b>Availability</b>	No Impact

**Forced Browsing Results:**

<http://vm/InternetBanking/>  
<http://vm/InternetBanking/DataAccess/>  
<http://vm/InternetBanking/auth/>  
<http://vm/InternetBanking/client/>  
<http://vm/InternetBanking/controller/>  
<http://vm/InternetBanking/controller/clientController.php>  
<http://vm/InternetBanking/controller/clientFunctions.inc>  
<http://vm/InternetBanking/controller/employeeController.php>  
<http://vm/InternetBanking/controller/employeeFunctions.inc>  
<http://vm/InternetBanking/controller/loginController.php>  
<http://vm/InternetBanking/controller/logoutController.php>  
<http://vm/InternetBanking/controller/registrationController.php>  
<http://vm/InternetBanking/css/>  
<http://vm/InternetBanking/css/bootstrap-theme.css>  
<http://vm/InternetBanking/css/bootstrap-theme.css.map>  
<http://vm/InternetBanking/css/bootstrap-theme.min.css>  
<http://vm/InternetBanking/css/bootstrap.css>  
<http://vm/InternetBanking/css/bootstrap.css.map>  
<http://vm/InternetBanking/css/bootstrap.min.css>  
<http://vm/InternetBanking/employee/>  
<http://vm/InternetBanking/fonts/>  
<http://vm/InternetBanking/fonts/glyphicons-halflings-regular.eot>  
<http://vm/InternetBanking/fonts/glyphicons-halflings-regular.svg>  
<http://vm/InternetBanking/fonts/glyphicons-halflings-regular.ttf>  
<http://vm/InternetBanking/fonts/glyphicons-halflings-regular.woff>  
<http://vm/InternetBanking/fonts/glyphicons-halflings-regular.woff2>  
<http://vm/InternetBanking/index/>  
<http://vm/InternetBanking/js/>  
<http://vm/InternetBanking/js/bootstrap.js>  
<http://vm/InternetBanking/js/bootstrap.min.js>  
<http://vm/InternetBanking/js/npm.js>  
<http://vm/InternetBanking/login/>  
<http://vm/InternetBanking/logout/>  
<http://vm/InternetBanking/model/>  
<http://vm/InternetBanking/model/Payment.class>  
<http://vm/InternetBanking/model/PaymentRequest.class>  
<http://vm/InternetBanking/model/User.class>

<http://vm/InternetBanking/model/UserRequest.class>  
<http://vm/InternetBanking/parser/>  
<http://vm/InternetBanking/parser/Makefile>  
<http://vm/InternetBanking/parser/exec>  
<http://vm/InternetBanking/parser/main.c>  
[http://vm/InternetBanking/parser/mysql\\_query\\_function.c](http://vm/InternetBanking/parser/mysql_query_function.c)  
[http://vm/InternetBanking/parser/mysql\\_query\\_function.h](http://vm/InternetBanking/parser/mysql_query_function.h)  
<http://vm/InternetBanking/registration/>  
<http://vm/InternetBanking/view/>  
<http://vm/InternetBanking/view/account.inc>  
<http://vm/InternetBanking/view/accounts.inc>  
<http://vm/InternetBanking/view/approvepayments.inc>  
<http://vm/InternetBanking/view/approveregistrations.inc>  
<http://vm/InternetBanking/view/client.inc>  
<http://vm/InternetBanking/view/employee.inc>  
<http://vm/InternetBanking/view/file.inc>  
<http://vm/InternetBanking/view/history.inc>  
<http://vm/InternetBanking/view/historypdf.inc>  
<http://vm/InternetBanking/view/login.inc>  
<http://vm/InternetBanking/view/online.inc>  
<http://vm/InternetBanking/view/registration.inc>

## SecureBank

<b>Observation</b>	We found some HTML snippets, which do not contain any sensitive information, and the compiled executable of the transaction file parser. We also found that the library TCPDF appears to be used.
<b>Discovery</b>	Using the OWASP ZAP tool, we used the forced browse functionality on /seccoding-2015/. We received a list of files which were found using this tool, see below.
<b>Likelihood</b>	This can be tested by anyone who enters specific strings into the address bar of a browser. However, the likelihood of this vulnerability is much higher if the attacker uses specific tools which test specific paths systematically.
<b>Impact</b>	The attacker only has access to the parser executable, which might contain information about the database connection. He can analyze the parser and find vulnerabilities there.
<b>Access Vector</b>	Network
<b>Access Complexity</b>	Low
<b>Privileges Required</b>	None
<b>User Interaction</b>	None
<b>Scope</b>	Unchanged
<b>Confidentiality</b>	Low
<b>Integrity</b>	No Impact
<b>Availability</b>	No Impact

### Forced Browsing Results:

`http://vm/seccoding-2015/`  
`http://vm/seccoding-2015/database/`  
`http://vm/seccoding-2015/html/`  
`http://vm/seccoding-2015/html/auth.html`  
`http://vm/seccoding-2015/html/dberror.html`  
`http://vm/seccoding-2015/html/default.html`  
`http://vm/seccoding-2015/html/display_userstate.html`

[http://vm/seccoding-2015/html/display\\_userstate\\_table.html](http://vm/seccoding-2015/html/display_userstate_table.html)  
[http://vm/seccoding-2015/html/dologin\\_fail.html](http://vm/seccoding-2015/html/dologin_fail.html)  
[http://vm/seccoding-2015/html/dologin\\_success.html](http://vm/seccoding-2015/html/dologin_success.html)  
[http://vm/seccoding-2015/html/doregister\\_fail.html](http://vm/seccoding-2015/html/doregister_fail.html)  
[http://vm/seccoding-2015/html/doregister\\_success.html](http://vm/seccoding-2015/html/doregister_success.html)  
[http://vm/seccoding-2015/html/edoverify\\_fail.html](http://vm/seccoding-2015/html/edoverify_fail.html)  
<http://vm/seccoding-2015/html/ehome.html>  
[http://vm/seccoding-2015/html/ehome\\_transaction.html](http://vm/seccoding-2015/html/ehome_transaction.html)  
[http://vm/seccoding-2015/html/ehome\\_user.html](http://vm/seccoding-2015/html/ehome_user.html)  
[http://vm/seccoding-2015/html/entry\\_page.html](http://vm/seccoding-2015/html/entry_page.html)  
<http://vm/seccoding-2015/html/etakeover.html>  
[http://vm/seccoding-2015/html/etransactionpdf\\_fail.html](http://vm/seccoding-2015/html/etransactionpdf_fail.html)  
<http://vm/seccoding-2015/html/home.html>  
<http://vm/seccoding-2015/html/login.html>  
<http://vm/seccoding-2015/html/register.html>  
<http://vm/seccoding-2015/html/root.html>  
<http://vm/seccoding-2015/html/simpleRoot.html>  
<http://vm/seccoding-2015/html/transaction.html>  
[http://vm/seccoding-2015/html/udotransactionupload\\_fail.html](http://vm/seccoding-2015/html/udotransactionupload_fail.html)  
<http://vm/seccoding-2015/html/utransaction.html>  
<http://vm/seccoding-2015/html/utransactionupload.html>  
<http://vm/seccoding-2015/index/>  
<http://vm/seccoding-2015/login/>  
<http://vm/seccoding-2015/parser/>  
<http://vm/seccoding-2015/parser/parser>  
<http://vm/seccoding-2015/tcpdf/>  
<http://vm/seccoding-2015/tcpdf/CHANGELOG.TXT>  
<http://vm/seccoding-2015/tcpdf/LICENSE.TXT>  
<http://vm/seccoding-2015/tcpdf/README.TXT>  
<http://vm/seccoding-2015/tcpdf/composer.json>  
<http://vm/seccoding-2015/tcpdf/config/>  
[http://vm/seccoding-2015/tcpdf/config/tcpdf\\_config.php](http://vm/seccoding-2015/tcpdf/config/tcpdf_config.php)  
<http://vm/seccoding-2015/tcpdf/fonts/>  
<http://vm/seccoding-2015/tcpdf/include/>  
<http://vm/seccoding-2015/tcpdf/include/barcodes/>  
<http://vm/seccoding-2015/tcpdf/include/barcodes/datamatrix.php>  
<http://vm/seccoding-2015/tcpdf/include/barcodes/pdf417.php>  
<http://vm/seccoding-2015/tcpdf/include/barcodes/qrcode.php>  
<http://vm/seccoding-2015/tcpdf/include/sRGB.icc>

[http://vm/seccoding-2015/tcpdf/include/tcpdf\\_colors.php](http://vm/seccoding-2015/tcpdf/include/tcpdf_colors.php)  
[http://vm/seccoding-2015/tcpdf/include/tcpdf\\_filters.php](http://vm/seccoding-2015/tcpdf/include/tcpdf_filters.php)  
[http://vm/seccoding-2015/tcpdf/include/tcpdf\\_font\\_data.php](http://vm/seccoding-2015/tcpdf/include/tcpdf_font_data.php)  
[http://vm/seccoding-2015/tcpdf/include/tcpdf\\_fonts.php](http://vm/seccoding-2015/tcpdf/include/tcpdf_fonts.php)  
[http://vm/seccoding-2015/tcpdf/include/tcpdf\\_images.php](http://vm/seccoding-2015/tcpdf/include/tcpdf_images.php)  
[http://vm/seccoding-2015/tcpdf/include/tcpdf\\_static.php](http://vm/seccoding-2015/tcpdf/include/tcpdf_static.php)  
<http://vm/seccoding-2015/tcpdf/tcpdf.php>  
[http://vm/seccoding-2015/tcpdf/tcpdf\\_autoconfig.php](http://vm/seccoding-2015/tcpdf/tcpdf_autoconfig.php)  
[http://vm/seccoding-2015/tcpdf/tcpdf\\_barcode\\_1d.php](http://vm/seccoding-2015/tcpdf/tcpdf_barcode_1d.php)  
[http://vm/seccoding-2015/tcpdf/tcpdf\\_barcode\\_2d.php](http://vm/seccoding-2015/tcpdf/tcpdf_barcode_2d.php)  
[http://vm/seccoding-2015/tcpdf/tcpdf\\_import.php](http://vm/seccoding-2015/tcpdf/tcpdf_import.php)  
[http://vm/seccoding-2015/tcpdf/tcpdf\\_parser.php](http://vm/seccoding-2015/tcpdf/tcpdf_parser.php)  
<http://vm/seccoding-2015/tcpdf/tools/>  
[http://vm/seccoding-2015/tcpdf/tools/convert\\_fonts\\_examples.txt](http://vm/seccoding-2015/tcpdf/tools/convert_fonts_examples.txt)  
[http://vm/seccoding-2015/tcpdf/tools/tcpdf\\_addfont.php](http://vm/seccoding-2015/tcpdf/tools/tcpdf_addfont.php)

### Comparison

The web application of the SecureBank discloses less sensitive information. However, both applications disclose information which should not be available to unauthorized persons.



#### 4.1.2 Test HTTP Methods

##### Online Banking

<b>Observation</b>	The server responded that the method POST, GET, OPTIONS and HEAD are supported.
<b>Discovery</b>	We submitted the request OPTIONS / HTTP/1.1 to the server via NetCat on port 80.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

##### SecureBank

<b>Observation</b>	The server responded that the method POST, GET, OPTIONS and HEAD are supported.
<b>Discovery</b>	We submitted the request OPTIONS / HTTP/1.1 to the server via NetCat on port 80.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

##### Comparison

There are no significant differences between both applications.

### 4.1.3 Test HTTP Strict Transport Security

#### Online Banking

<b>Observation</b>	The server did not send any Strict-Transport-Security header.
<b>Discovery</b>	Executing the command <code>curl -s -D-http://vm/InternetBanking/   grep Strict</code> resulted in no results.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

#### SecureBank

<b>Observation</b>	The server did not send any Strict-Transport-Security header.
<b>Discovery</b>	Executing the command <code>curl -s -D-http://vm/InternetBanking/   grep Strict</code> resulted in no results.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

#### Comparison

There are no significant differences between both applications.

#### 4.1.4 Test RIA cross domain policy

##### Online Banking

<b>Observation</b>	No cross domain policy files were found.
<b>Discovery</b>	We scanned the traffic using ZAP.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

##### SecureBank

<b>Observation</b>	No cross domain policy files were found.
<b>Discovery</b>	We scanned the traffic using ZAP.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

##### Comparison

There are no significant differences between both applications.

## 4.2 Identity Management Testing

### 4.2.1 Test Role Definitions

#### Online Banking

<b>Observation</b>	We found the following functionality for the different roles:		
		<b>Client</b>	<b>Employee</b>
	View own account	×	×
	View own transaction history	×	—
	Create new transactions	×	—
	View account and transaction history of clients and employees	—	×
	Change account details and balance of clients and employees	—	×
	Approve transactions	—	×
	Approve registrations of clients and employees	—	×
<b>Discovery</b>	We noticed that there are links to view the transaction history and change the account balance of employees, too.		
	We gathered the information by exploring the web application interface manually.		
<b>Impact</b>	n/a		
<b>Likelihood</b>	n/a		
<b>CVSS</b>	n/a		

## SecureBank

Observation	We found the following functionality for the different roles:		
		Client	Employee
	View own account	×	–
	View own transaction history	×	–
	Create new transactions	×	–
	View account and transaction history of clients	–	×
	Approve transactions	–	×
	Approve registrations of clients and employees	–	×
Discovery	We gathered the information by exploring the web application interface manually.		
Impact	n/a		
Likelihood	n/a		
CVSS	n/a		

## Comparison

The SecureBank web application does not offer a possibility for an employee to change the account balance of a client. However, the Online Banking application allows to view the transaction history and change the account balance also for employees, which have no account. This behaviour might be confusing.

## 4.2.2 Test User Registration Process

### Online Banking

<b>Observation</b>	For registration, a username, an e-mail address, a password and whether the registrant is a client or an employee are needed. Anyone can register for access. The registration has to be approved by an employee before the registrant can use the account. A person can register only one time with the same e-mail address. However, a person can register many times with the same username. (The activation of such an account fails with a database error.) We could not find out, whether the registrants are verified personally before the approval.
<b>Discovery</b>	We tried to register several accounts with the same e-mail address and/or username using the web application.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

### SecureBank

<b>Observation</b>	For registration, the full name, an e-mail address, a password and whether the registrant is a client or an employee are needed. Anyone can register for access. The registration has to be approved by an employee before the registrant can use the account. A person can register only one time with the same e-mail address. We could not find out, whether the registrants are verified personally before the approval.
<b>Discovery</b>	We tried to register several accounts with the same e-mail address and/or names using the web application.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

### **Comparison**

The Online Banking web application allows the double-registration of the same username at first, it only fails at the activation. This behaviour is confusing. Also, the application should ask for the full name be able to verify the name. Otherwise, there are no significant differences between both applications.

### 4.2.3 Test Account Provisioning Process

#### Online Banking

<b>Observation</b>	There is no way to change the role of a user. Account requests (both, client and employee) must be approved by an employee.
<b>Discovery</b>	We followed the links in the user interface and tried to login as a non-verified user.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

#### SecureBank

<b>Observation</b>	There is no way to change the role of a user. Account requests (both, client and employee) must be approved by an employee.
<b>Discovery</b>	We followed the links in the user interface and tried to login as a non-verified user.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

#### Comparison

There are no significant differences between both applications.



#### 4.2.4 Testing for Account Enumeration and Guessable User Account

##### Online Banking

<b>Observation</b>	There are no differences in the servers response for not activated accounts, valid usernames and invalid usernames.
<b>Discovery</b>	We tested the login for activated and non-activated accounts, existing and not-existing usernames and valid or invalid passwords.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

##### SecureBank

<b>Observation</b>	There are no differences in the servers response for not activated accounts, valid usernames and invalid usernames.
<b>Discovery</b>	We tested the login for activated and non-activated accounts, existing and not-existing usernames and valid or invalid passwords.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

##### Comparison

There are no significant differences between both applications.

#### 4.2.5 Testing for Weak or unenforced username policy

##### Online Banking

<b>Observation</b>	We were not able to find a username policy.
<b>Discovery</b>	We tested various usernames.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

##### SecureBank

<b>Observation</b>	The username has to be a valid e-mail address of the client/employee. There is no policy regarding the e-mail address.
<b>Discovery</b>	We tested valid and invalid e-mail addresses.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

##### Comparison

The only difference between the applications is that the Online Banking application uses usernames, which might have less correlation to the user than the e-mail address.

## 4.3 Authentication Testing

### 4.3.1 Testing for Credentials Transported over Encrypted Channel

#### Online Banking

<b>Observation</b>	We were unable to connect to the application via HTTPS.
<b>Discovery</b>	We tried to access the application via HTTPS on port 443.
<b>Impact</b>	An attacker can retrieve sensitive information by sniffing the network.
<b>Likelihood</b>	High
<b>Access Vector</b>	Adjacent Network
<b>Access Complexity</b>	Medium
<b>Privileges Required</b>	None
<b>User Interaction</b>	None
<b>Scope</b>	Unchanged
<b>Confidentiality</b>	High
<b>Integrity</b>	No Impact
<b>Availability</b>	No Impact

### SecureBank

<b>Observation</b>	We were unable to connect to the application via HTTPS.
<b>Discovery</b>	We tried to access the application via HTTPS on port 443.
<b>Impact</b>	An attacker can retrieve sensitive information by sniffing the network.
<b>Likelihood</b>	High
<b>Access Vector</b>	Adjacent Network
<b>Access Complexity</b>	Medium
<b>Privileges Required</b>	None
<b>User Interaction</b>	None
<b>Scope</b>	Unchanged
<b>Confidentiality</b>	High
<b>Integrity</b>	No Impact
<b>Availability</b>	No Impact

### Comparison

Both applications transport sensitive information about non-encrypted channels.

#### **4.3.2 Testing for default credentials**

We decided to not test for default credentials, because we are working with custom software and therefore assume that all users and administrators choose secure passwords.

### 4.3.3 Testing for Weak lock out mechanism

#### Online Banking

<b>Observation</b>	We were not able to find any lock out mechanism. Therefore, brute force attacks on passwords are possible.
<b>Discovery</b>	We entered a valid username and incorrect passwords 10 times, and always got the error message about an incorrect password. Afterwards, we were able to log in with a correct password.
<b>Impact</b>	An attacker can brute-force the password of any user and therefore take the user over.
<b>Likelihood</b>	High
<b>Access Vector</b>	Network
<b>Access Complexity</b>	Low
<b>Privileges Required</b>	None
<b>User Interaction</b>	None
<b>Scope</b>	Unchanged
<b>Confidentiality</b>	Low
<b>Integrity</b>	Low
<b>Availability</b>	No Impact

### SecureBank

<b>Observation</b>	We were not able to find any lock out mechanism. Therefore, brute force attacks on passwords are possible.
<b>Discovery</b>	We entered a valid username and incorrect passwords 10 times, and always got the error message about the failed login. Afterwards, we were able to log in with a correct password.
<b>Impact</b>	An attacker can brute-force the password of any user and therefore take the user over.
<b>Likelihood</b>	High
<b>Access Vector</b>	Network
<b>Access Complexity</b>	Low
<b>Privileges Required</b>	None
<b>User Interaction</b>	None
<b>Scope</b>	Unchanged
<b>Confidentiality</b>	Low
<b>Integrity</b>	Low
<b>Availability</b>	No Impact

### Comparison

Both applications do not provide any lock out mechanism.

#### 4.3.4 Testing for bypassing authentication schema

##### Online Banking

<b>Observation</b>	We were able to bypass the authentication schema via a SQL injection. This gave us the ability to login as any user without knowing the password.
<b>Discovery</b>	Using the fuzzer jbrofuzz / SQL Injection of ZAP on the username field of the login page, we were able to login as admin or another user without knowing the password. We had no success with direct page requests, modifying the session ID and parameter modification.
<b>Impact</b>	An attacker can take over a user without knowing the valid access credentials.
<b>Likelihood</b>	High
<b>Access Vector</b>	Network
<b>Access Complexity</b>	Low
<b>Privileges Required</b>	None
<b>User Interaction</b>	None
<b>Scope</b>	Unchanged
<b>Confidentiality</b>	Low
<b>Integrity</b>	Low
<b>Availability</b>	No Impact



### SecureBank

<b>Observation</b>	We were not able to bypass the authentication schema.
<b>Discovery</b>	Using the fuzzer jbrofuzz / SQL Injection of ZAP and SQLmap on the username field of the login page, we were not able to find SQL injection vulnerabilities to bypass the authentication schema. We also had no success with direct page requests, modifying the session ID and parameter modification.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

### Comparison

The Online Banking web application provides a way to bypass the authentication schema via SQL injection. The SecureBank application does not offer such vulnerabilities.

#### **4.3.5 Testing for Vulnerable Remember Password**

We did not found a remember password functionality, so we decided to not further test on this.

### 4.3.6 Testing for Browser Cache Weakness

#### Online Banking

<b>Observation</b>	Clicking the back button in the browser does not cause a re-login. All sites have the header Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0 and the Pragma: no-cache as well as an Expires: <date in the past> header set.
<b>Discovery</b>	Using ZAP, we analyzed the response header for different pages which are only available when a user is logged in.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

#### SecureBank

<b>Observation</b>	Clicking the back button in the browser does not cause a re-login. All sites have the header Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0 and the Pragma: no-cache as well as an Expires: <date in the past> header set.
<b>Discovery</b>	Using ZAP, we analyzed the response header for different pages which are only available when a user is logged in.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

#### Comparison

There is no significant difference between both applications.

### 4.3.7 Testing for Weak password policy

#### Online Banking

<b>Observation</b>	There is a password policy enforced, which states that a password has have a length $\geq 6$ and has to include at least one number, one lowercase character, one uppercase character and one symbol. There is no way to change the password. The password does not expire.
<b>Discovery</b>	We tested various passwords, like 123456. If a password does not match the policy, an error message is shown which informs the user about the policy.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

#### SecureBank

<b>Observation</b>	The password has to have a length $\geq 6$ . There is no way to change the password. The password does not expire.
<b>Discovery</b>	We tested various passwords, like 123456. If a password does not match the policy, an error message is shown which informs the user about the policy.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

#### Comparison

The Online Banking application enforces a more strict password policy than the Secure-Bank application. This reduces the risk of brute force attacks.

#### **4.3.8 Testing for Weak security question/answer**

We could not find such functionality in both application. Therefore, we decidede to not proceed testing on this.

#### **4.3.9 Testing for Weak password change or reset functionalities**

We could not find such functionality in both application. Therefore, we decidede to not proceed testing on this.

#### **4.3.10 Testing for Weaker authentication in alternative channel**

We could not find an alternative channel for authentication. Therefore, we decidede to not proceed testing on this.

## 4.4 Authorization Testing

### 4.4.1 Testing Directory traversal/file include

TODO!

### 4.4.2 Testing for bypassing authorization schema

TODO!

### 4.4.3 Testing for Privilege Escalation

TODO!

### 4.4.4 Testing for Insecure Direct Object References

TODO!

## 4.5 Session Management Testing

TODO!

## 4.6 Data Validation Testing

TODO!

### 4.6.1 Testing for Reflected Cross Site Scripting

#### Online Banking

<b>Observation</b>	We were not able to find any reflected XSS vulnerability in this application.
<b>Discovery</b>	We tried to inject basic scripts into the register, login and transaction form as well as into the administration form.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

### SecureBank

<b>Observation</b>	We were not able to find any reflected XSS vulnerability.
<b>Discovery</b>	We tried to inject basic scripts into the register, login and transaction form as well as into the administration form.
<b>Impact</b>	n/a
<b>Likelihood</b>	n/a
<b>CVSS</b>	n/a

### Comparison

Based on our testing, we found that both applications operate on a similar security level.

#### 4.6.2 Testing for LDAP/ORM/XML/SSI/XPath Injection

We did not test any of these injection types as it appears that these techniques are not used in any of the two applications. Therefore, we decided to not further investigate in these types of injection.

### 4.6.3 Testing for SQL injection

#### Online Banking

<b>Observation</b>	We were able to log into the application as any user without knowing the password and to perform transactions with any unused TAN of any user when uploading a transaction batch file.
<b>Discovery</b>	<p>We used the fuzzer of ZAP on the username field of the login form, with the query parameters <code>username=admin&amp;password=123</code>, where the fuzzing point was at the end of the username. We compared the answers of the server and found that in one case we got redirected to the employee page. We analyzed the login page also using SQLmap, which delivered a similar result. Additionally, we ran SQLmap on the fields of the perform transaction form, it returned that none of the four inputs were exploitable for SQL injection.</p> <p>We tested the fields of the transaction batch form manually by trying some standard SQL injection parts (e.g. ' ; - and " ; -). We noted that the application gives no usual feedback when uploading a transaction file where the username or the comment contains strings that apparently results in a syntactically erroneous SQL query. We continued to determine the table structure using a brute force script which tests for some expectable table and column names, the results can be found below. Finally, we were able to exploit the SQL injection vulnerability in the TAN field to use any unused TAN from any user.</p>
<b>Impact</b>	An attacker can log into any account, of which he knows the username, and perform transactions without knowing valid TANs for that account by uploading a transaction batch file. Furthermore, as an attacker can take over an administrator account, if he knows the username, he has access to all accounts and can change the account balances at will. Also, an attacker can analyze the structure of a database.
<b>Likelihood</b>	High



<b>Login vulnerability</b>	<b>Access Vector</b>	Network
	<b>Access Complexity</b>	Low
	<b>Privileges Required</b>	None
	<b>User Interaction</b>	None
	<b>Scope</b>	Unchanged
	<b>Confidentiality</b>	High
	<b>Integrity</b>	High
	<b>Availability</b>	No Impact

<b>TAN vulnerability</b>	<b>Access Vector</b>	Network
	<b>Access Complexity</b>	Low
	<b>Privileges Required</b>	Low
	<b>User Interaction</b>	None
	<b>Scope</b>	Unchanged
	<b>Confidentiality</b>	High
	<b>Integrity</b>	High
	<b>Availability</b>	No Impact

### Results of the brute-force script

Table payment: id, trancode, payer, receipt, amount, purpose  
 Table user: id, balance, email, username, password, isemployee  
 Table userrequest: id, email, username, password, isemployee  
 Table paymentrequest: id, trancode, payer, receipt, amount, purpose  
 Table trancode: id, clientid

## SecureBank

### Observation

#### Discovery

We used the fuzzer of ZAP on the username field of the login form, with the query parameters `username=admin@localhost&password=123`, where the fuzzing point was at the end of the username. We also analyzed the login page also using SQLmap. Using both methods, we were not able to log in without knowing the password. Additionally, we ran SQLmap on the fields of the registration form and the perform transaction form, it returned that none of the four inputs were exploitable for SQL injection.

#### Impact

n/a

#### Likelihood

n/a

#### CVSS

n/a

## Comparison

The Online Banking web application is vulnerable to SQL injection at different places, allowing an attacker to take over user accounts and transfer money without knowing any credentials. In the web application of the SecureBank, we were not able to find SQL injection vulnerabilities.

## 4.7 Error Handling

### 4.7.1 Analysis of Error Codes

#### Online Banking

<b>Observation</b>	We were not able to get reveal any information about the database, but we were able to get usual error codes of an apache server. The error pages also revealed information about the version, which is 2.2.22 on an Ubuntu server.
<b>Discovery</b>	By using the fuzzer jbrofuzz / SQL Injection / MySQL Injection 101 of ZAP on the purpose field of the online transaction form, we got one response which simply stated Database error, but gave no further information. Using a browser, we requested the page /InternetBanking/foobar/, which returned a 404 error, with further information about the server operating system and the Apache version.
<b>Impact</b>	Using the apache version, an attacker can run known exploits for this specific version.
<b>Likelihood</b>	Medium
<b>Access Vector</b>	Network
<b>Access Complexity</b>	Low
<b>Privileges Required</b>	None
<b>User Interaction</b>	None
<b>Scope</b>	Unchanged
<b>Confidentiality</b>	No Impact
<b>Integrity</b>	No Impact
<b>Availability</b>	No Impact

## SecureBank

<b>Observation</b>	We were not able to get reveal any information about the database, but we were able to get usual error codes of an apache server. The error pages also revealed information about the version, which is 2.2.22 on an Ubuntu server. On the login page, we found an error code if the login was unsuccessful. Apparently, the code is always 1, if the credentials are incorrect.
<b>Discovery</b>	By using the fuzzer jbrofuzz / SQL Injection / MySQL Injection 101 of ZAP on the purpose field of the on-line transaction form, we were not able to produce a databse error. Using a browser, we requested the page /seccoding-2015/foobar/, which returned a 404 error, with further information about the server operating system and the Apache version.
<b>Impact</b>	Using the apache version, an attacker can run known exploits for this specific version.
<b>Likelihood</b>	Medium
<b>Access Vector</b>	Network
<b>Access Complexity</b>	Low
<b>Privileges Required</b>	None
<b>User Interaction</b>	None
<b>Scope</b>	Unchanged
<b>Confidentiality</b>	No Impact
<b>Integrity</b>	No Impact
<b>Availability</b>	No Impact

## Comparison

Although the Online Banking application produces an error message about a database error, they provide the similar information in error messages.

#### **4.7.2 Testing for Stack Traces**

We were not able to produce stack traces in both applications using invalid inputs (e.g. negative numbers), or SQL injection.

#### **4.8 Testing for weak Cryptography**

TODO!

#### **4.9 Business Logic Testing**

TODO!

#### **4.10 Client Side Testing**

TODO!

# Acronyms

**TUM** Technische Universität München.