

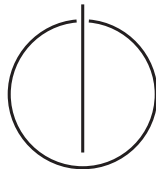


DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Secure Coding Phase 2

Team 3: Patrick Sattler, Aurel Roci, Stefan Kohler



# Executive Summary

## **Team2**

We found several vulnerabilities, which could cause severe damage to the *Team2*. It is possible to get access to the admin page via stealing the session. Thus an attacker can register an arbitrary employee or customer and unlock the registered user. An attacker can also execute a brute force attack on known user ids as there is no lock mechanism to prevent this. Besides the security issues there is also a severe problem with regard to the business logic. In the current state this web application should not be used productively!

## **Team3 Online Banking**

We found some issues, which potentially could cause damage to the *Team3 Online Banking*. However the detected issues are quite easy to fix. If an experienced attacker performs a man in the middle attack he'll be able to track session ids. The implications are severe, as the attacker can take over the role of the customer, but this attack requires advanced knowledge. With regard to the business logic there was only one issue with low risk detected.

## **Comparison**

In summary we were able to clearly state out that the *Team3 Online Banking* web application has less and also less severe vulnerabilities than the *Team2* web application. Furthermore it has to be said that the detected issues of the *Team3 Online Banking* are easier to fix and will cost less money to implement.

# Contents

<b>Executive Summary</b>	<b>i</b>
<b>1 Time Tracking Table</b>	<b>1</b>
<b>2 Vulnerabilities Overview</b>	<b>3</b>
2.1 Team2 . . . . .	3
2.1.1 Static Session ID . . . . .	3
2.1.2 File Upload . . . . .	3
2.1.3 Mail . . . . .	3
2.2 Team3 . . . . .	3
<b>3 Tools used</b>	<b>4</b>
<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 (OTG-CONFIG-003) . . . . .	6
4.1.2 Test HTTP Methods (OTG-CONFIG-006) . . . . .	8
4.1.3 Test HTTP Strict Transport Security (OTG-CONFIG-007) . . . . .	9
4.1.4 Test RIA cross domain policy (OTG-CONFIG-008) . . . . .	10
4.2 Identity Management Testing . . . . .	11
4.2.1 Test Role Definitions (OTG-IDENT-001) . . . . .	11
4.2.2 Test User Registration Process (OTG-IDENT-002) . . . . .	12
4.2.3 Test Account Provisioning Process (OTG-IDENT-003) . . . . .	14
4.2.4 Testing for Account Enumeration and Guessable User Account (OTG-IDENT-004) . . . . .	15
4.2.5 Testing for Weak or unenforced username policy (OTG-IDENT-005) . . . . .	16
4.3 Authentication Testing . . . . .	17
4.3.1 Testing for Credentials Transported over an Encrypted Channel(OTG-AUTHN-001) . . . . .	17

4.3.2	Testing for default credentials(OTG-AUTHN-002) . . . . .	18
4.3.3	Testing for Weak lock out mechanism(OTG-AUTHN-003) . . . . .	19
4.3.4	Testing for bypassing authentication schema (OTG-AUTHN-004) . . . . .	20
4.3.5	Testing for Browser cache weakness (OTG-AUTHN-006) . . . . .	21
4.3.6	Testing for Weak password policy (OTG-AUTHN-007) . . . . .	22
4.3.7	Testing for weak password change or reset functionalities (OTG-AUTHN-009) . . . . .	23
4.4	Authorization Testing . . . . .	24
4.4.1	Testing Directory traversal/file include (OTG-AUTHZ-001) . . . . .	24
4.4.2	Testing for Privilege Escalation (OTG-AUTHZ-003) . . . . .	25
4.4.3	Testing for Insecure Direct Object References (OTG-AUTHZ-004) . . . . .	26
4.5	Session Management Testing . . . . .	27
4.5.1	Testing for Bypassing Session Management Schema(OTG-SESS-001) . . . . .	27
4.5.2	Testing for Cookies attributes(OTG-SESS-002) . . . . .	28
4.5.3	Testing for Session Fixation(OTG-SESS-003) . . . . .	29
4.5.4	Testing for Exposed Session Variables (OTG-SESS-004) . . . . .	30
4.5.5	Testing for Cross Site Request Forgery(OTG-SESS-005) . . . . .	31
4.5.6	Testing for logout functionality(OTG-SESS-006) . . . . .	32
4.5.7	Test Session Timeout(OTG-SESS-007) . . . . .	33
4.5.8	Testing for Session puzzling(OTG-SESS-008) . . . . .	34
4.6	Data Validation Testing . . . . .	35
4.6.1	Testing for Reflected Cross Site Scripting(OTG-INPVAL-001) . . . . .	35
4.6.2	Testing for Stored Cross Site Scripting(OTG-INPVAL-002) . . . . .	36
4.6.3	Testing for HTTP Verb Tampering(OTG-INPVAL-003) . . . . .	37
4.6.4	Testing for SQL Injection (OTG-INPVAL-005) and Mysql testing (OTG-INPVAL-005) . . . . .	39
4.6.5	Testing for Code Injection, Testing for Local File Inclusion, Testing for Remote File Inclusion(OTG-INPVAL-012) . . . . .	42
4.6.6	Testing for Command Injection(OTG-INPVAL-013) . . . . .	43
4.6.7	Testing for Buffer overflow, Testing for Heap overflow, Testing for Stack overflow, Testing for Format string (OTG-INPVAL-014) . . . . .	45
4.6.8	Testing for incubated vulnerabilities(OTG-INPVAL-015) . . . . .	46
4.7	Error Handling . . . . .	47
4.8	Business Logic Testing . . . . .	48
4.8.1	Test Business Logic Data Validation(OTG-BUSLOGIC-001) . . . . .	48

## *Contents*

---

4.8.2	Test Number of Times a Function Can be Used Limits(OTG-BUSLOGIC-005)	49
4.8.3	Test Upload of Unexpected File Types (OTG-BUSLOGIC-008)	50
4.9	Client Side Testing	51

# 1 Time Tracking Table

## 1 Time Tracking Table

Name	Task	Time
Aurel Roci	Test HTTP Methods	0.25
	Error Handling	1
	Testing for default credentials	0.25
	Testing for Reflected Cross Site Scripting	0.5
	Testing for Stored Cross Site Scripting	2
	Testing for HTTP Verb Tampering	0.5
	Testing for SQL Injection	2
	Test Number of Times a Function Can be Used Limits	0.75
	Test Business Logic Data Validation	1.5
	Executive Summary	0.5
	Testing Report	2
	Testing for Cross Site Request Forgery	0.5
	Testing for Privilege Escalation	1.5
	Presentation	0.25
Stefan Ch. Kofler	Test File Extensions Handling for Sensitive Information	2.0
	Test HTTP Strict Transport Security	0.5
	Test RIA cross domain policy	0.5
	Test Role Definitions	1.25
	Test User Registration Process	0.75
	Test Account Provisioning Process	0.5
	Testing for Account Enumeration and Guessable User Account	0.5
	Testing for Weak or unenforced username policy	0.5
	Testing for DOM based Cross Site Scripting	0.75
	Testing for JavaScript Execution	2.0
	Testing for HTML Injection	0.5
	Testing for Client Side URL Redirect	0.75
	Testing for CSS Injection	0.5
	Testing for Client Side Resource Manipulation	0.5
	Test Cross Origin Resource Sharing	1.25
	Testing for Cross Site Flashing	0.5
	Testing for Clickjacking	1.0
	Testing WebSockets	0.75
	Test Web Messaging	0.5
	Test Local Storage	0.5
Patrick Sattler	understanding w3af and configuration	2
	w3af testing overflow and format string	1.5
	Test bruteforce for password	0.5
	Test bruteforce for TAN	1
	Test Registration process	1
	Test password policy	0.5
	Test credentials transport	1
	Test bypassing authentication schema	1
	Test browser cache weakness	1
	Test weak password policy	0.5
	Testing Directory traversal/file include	1.75
	Testing bypassing session management	1
	Testing cookie attributes	0.75
	Testing session fixation	0.75
	Testing for Exposed Session Variables	0.5
	Testing for Cross Site Request Forgery	1
	Testing for logout functionality	0.5
	Test Session Timeout	0.75
	Testing for Session puzzling	0.25
	Testing for XML/SSI/XPATH Injection	2
	Testing for Code Injection, local or remote file inclusion	1
	Testing for Command Injection	0.5
	Testing for Format string	0.25
	Testing for incubated vulnerabilities	0.5

## 2 Vulnerabilities Overview

Based on our testing, we identified the following vulnerabilities for the Team2 Bank and the OnlineBanking Bank:

### 2.1 Team2

#### 2.1.1 Static Session ID

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

The session id is saved in form of the (static) user id in a cookie. This cookie can be used on any machine to take over the account of a user. The lifetime of this cookie is only limited by the cookie lifetime field.

#### 2.1.2 File Upload

There was no instructions on the structure of the information for the file upload functionality. There was a sample file, but after uploading it, and trying it with several accounts, it did not work.

#### 2.1.3 Mail

The e-mail functionality to send the tans upon sign up was not working.

### 2.2 Team3

After testing the application there were no vulnerabilities found!



### 3 Tools used

*sqlmap* - We used *sqlmap* to test for SQL Injections, and we did not find any.

*RIPS* - We used *RIPS* to do a static testing of the PHP code. It showed warnings for SQL Injections which were *false-positives*. There were also some *false-positive* with command execution.

*FindBugs* - We used this tool to do a static testing of the Java code, and we did not find any vulnerabilities in the code of *Team2*, and we found two *false-positive* warnings in the code of *Team3*.

*netcat* - We used this tool to check the HTTP methods but it did not give any output.

*nmap* We used this tool to check HTTP method, and it showed which methods were allowed and which ones were not allowed.

*w3af* - We used this tool to find if the website had vulnerabilities by running an automatic test on the website. And found CSRF vulnerability.

*Zed Attack Proxy (ZAP)* - We used this for a spider attack towards the website, and did not find any CSRF vulnerability.

*Burp* - We used this tool to intercept the message from the web browser to the server and alter it.

## 4 Detailed Report

The following pages describe for each test how both applications Team2 and Team3. The test is divided in different sections following the OWASP Testing Guide v4.

## 4.1 Configuration and Deploy Management Testing

### 4.1.1 Test File Extensions Handling for Sensitive Information (OTG-CONFIG-003)

<b>Team2</b>		Likelihood: 8 Impact: 9 Risk: 3
<b>Team2</b>		
Observation	File extensions are handled correctly but while testing we found folders which we could access.	
Discovery	TODO	
Likelihood	The likelihood is quite high that someone tries a tool to find these kind of vulnerabilities. There is no need for special knowledge because the tools work quite automatically without much configuration.	
Impact	There is no impact the folders do not contain sensitive information.	
Recommendations	Block the access to files and to those folders, or remove them from the directory since they are not needed there.	

Metric	Value : 9.8
Access Vector	N
Attack Complexity	L
Privileges Required	N
User Interaction	N
Scope	U
Confidentiality Impact	H
Integrity Impact	H
Availability Impact	H

**Team3**

Likelihood: 8

Impact: 0

Risk: 1

---

Team3	
Observation	File extensions are handled correctly but while testing we found folders which we could access smart-card simulator folder.
Discovery	TODO
Likelihood	The likelihood is quite high that someone tries a tool to find these kind of vulnerabilities. There is no need for special knowledge because the tools work quite automatically without much configuration.
Impact	There is no impact since there is nothing important hard-coded in the java file.
Recommendations	Block the access to files and to those folders.

---

Metric	Value : 0
Access Vector	N
Attack Complexity	L
Privileges Required	N
User Interaction	N
Scope	U
Confidentiality Impact	N
Integrity Impact	N
Availability Impact	N

#### 4.1.2 Test HTTP Methods (OTG-CONFIG-006)

Team2

Likelihood: 0

Impact: 0

Risk: 0

Team2	
Observation	The application is not accessible over HTTP. HTTPS is enforced.
Discovery	We also tried to connect via <i>netcat</i> using the following command: <i>nc IP_ADDRESS 80</i> , which did not work. We also used <i>nmap</i> for testing which returned the methods used by the webapp.
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	The same applies for our web application.

```
samurai@samurai-wtf:~$ nmap -p 80 127.0.0.1 --script http-methods
```

```
Starting Nmap 6.25 ( http://nmap.org ) at 2016-01-01 11:04 CET
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000036s latency).
PORT      STATE SERVICE
80/tcp    open  http
|_http-methods: GET HEAD POST OPTIONS
```

```
Nmap done: 1 IP address (1 host up) scanned in 0.33 seconds
```

Figure 4.1: RIPS

### 4.1.3 Test HTTP Strict Transport Security (OTG-CONFIG-007)

**Team2**

Likelihood: 0

Impact: 0

Risk: 0

Team2	
Observation	The webapp is not using Strict Transport Security.
Discovery	We found this using <i>Curl</i> .
Likelihood	It is not complicated to perform a MitM attack to exploit this vulnerability.
Impact	A man-in-the-middle attacker attempts to intercept traffic from a victim user using an invalid certificate and hopes the user will accept the bad certificate.
Recommendations	Enable Strict Transport Security.
Comparison	Our webapp has Strict Transport Security enabled.

```
samurai@samurai-wtf:~$ curl -s -D- 127.0.0.1
HTTP/1.1 200 OK
Date: Fri, 01 Jan 2016 10:06:28 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Tue, 26 Jun 2012 09:23:08 GMT
ETag: "a1940-4f-4c35ca5889f57"
Accept-Ranges: bytes
Content-Length: 79
Vary: Accept-Encoding
Content-Type: text/html
```

Figure 4.2: RIPS

#### 4.1.4 Test RIA cross domain policy (OTG-CONFIG-008)

**Team2**

Likelihood: 0

Impact: 0

Risk: 0

Team2	
Observation	There are no RIA applications on the system and therefore is no crossdomain.xml file provided.
Discovery	Using <i>wget</i> we tried to find a <i>crossdomain.xml</i> or <i>clientaccesspolicy.xml</i> file and couldn't find it.
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	The same results applies for our web application.

Metric	Value
Access Vector	N/A
Attack Complexity	N/A
Privileges Required	N/A
User Interaction	N/A
Scope	N/A
Confidentiality Impact	N/A
Integrity Impact	N/A
Availability Impact	N/A

## 4.2 Identity Management Testing

### 4.2.1 Test Role Definitions (OTG-IDENT-001)

**Team2**

Likelihood: 10

Impact: 0

Risk: 0

---

<b>Team2</b>	
Observation	We found out that there exist two different roles in the system. There is the role of a normal customer and the role of an banker. Employees have the additional functionality to view account and transaction details of all the customers. Transactions over 10000 euro and new user registrations can be accepted by the employee.
Discovery	No special tools except a browser were needed because all the roles and their available functions can be seen on the webapp. All the roles have the functionalities they are supposed to have.
Likelihood	It is very likely that people find this information.
Impact	There is no direct implication.
Recommendations	N/A
Comparison	Our web application provides the same roles, but the roles are not described on the web page.

---



#### 4.2.2 Test User Registration Process (OTG-IDENT-002)

**Team2**

Likelihood: 5

Impact: 5

Risk: 5

---

<b>Team2</b>	
Observation	Any person can register themselves as an user and this registration than gets validated by an employee. One person can register multiple times and with different roles. There is no proof of the identity of a user possible. The identification requirements include the email address and username, but only two of these can be verified.
Discovery	No special tools are needed to get this information. A browser and multiple registration tests provided the necessary results.
Likelihood	It is quite likely that this information can be retrieved by any user with minimal experience.
Implication	User could try to register multiple times and with wrong information to get access to user accounts with more permissions or to create multiple bank accounts.
Recommendations	The information passed in the registration form should be validated. The name can be validated by hand if a customer would go to the bank and the employee would than accept his registration.
Comparison	Our web application doesn't require a phone number for the registration an the role of the user can be selected in the registration form. It doesn't make our application less secure, because the registration has still to be accepted by an employee.

---

#### 4 Detailed Report

---

Metric	Value
Access Vector	N
Attack Complexity	L
Privileges Required	N
User Interaction	N
Scope	U
Confidentiality Impact	N
Integrity Impact	N
Availability Impact	N

#### 4.2.3 Test Account Provisioning Process (OTG-IDENT-003)

**Team2**

Likelihood: N/A

Impact: N/A

Risk: N/A

---

Team2	
Observation	Our observation showed us that employees can accept customer registrations.
Discovery	All the observations were made with the <i>Chrome</i> web browser.
Impact	If an employee account gets hacked you can accept new registrations.
Recommendations	N/A
Comparison	In our web application the employee accepts new registration. It makes no difference in the security.

---

Metric	Value
Access Vector	N/A
Attack Complexity	N/A
Privileges Required	N/A
User Interaction	N/A
Scope	N/A
Confidentiality Impact	N/A
Integrity Impact	N/A
Availability Impact	N/A

#### 4.2.4 Testing for Account Enumeration and Guessable User Account (OTG-IDENT-004)

Team2

Likelihood: 0

Impact: 0

Risk: 0

---

Team2	
Observation	We found out that the web application makes no difference between existing usernames and non existing usernames when trying to login with wrong credentials. The same html response and the same response headers are provided by the system.
Discovery	We used the <i>Charles Web Proxy</i> to analyze the web application responses.
Implication	N/A
Recommendations	N/A
Comparison	Our web application makes no difference between login tries with existing usernames and non existing ones. Both web applications aren't vulnerable here.

---

#### 4.2.5 Testing for Weak or unenforced username policy (OTG-IDENT-005)

**Team2**

Likelihood: 0

Impact: 0

Risk: 0

---

<b>Team2</b>	
Observation	The usernames should be more than 2 characters.
Discovery	No tool is used here. Trying to sign up with short usernames gave us the warning.
Impact	N/A
Recommendations	N/A
Comparison	Our application has no username restriction.

---

## 4.3 Authentication Testing

### 4.3.1 Testing for Credentials Transported over an Encrypted Channel(OTG-AUTHN-001)

Team2		Likelihood: 0
		Impact: 0
		Risk: 0
Team2		
Observation	The webapp uses HTTPS protocol, so all the information is sent encrypted through the channel therefore we cannot sniff the data.	
Discovery	The website s using an SSL certificate	
Likelihood	NA	
Impact	NA	
Recommendations		
Comparison	Our webapp is using HTTPS, so the data is transported over an encrypted channel.	

#### 4.3.2 Testing for default credentials(OTG-AUTHN-002)

**Team2**

Likelihood: 10

Impact: 4

Risk: 6

---

<b>Team2</b>	
Observation	There are no default credentials for the webapp.
Discovery	We saw the credentials text file that was given by the other team.
Likelihood	N/A
Impact	N/A
Recommendations	N/A
Comparison	The same goes for our webapp.

---

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

**Team2**

Likelihood: 0

Impact: 0

Risk: 0

---

Team2	
Observation	The user is shown redirected from the login page after a 3 failed tries.
Discovery	We entered the wrong credentials until we had to click the login button to be able to enter the credentials again.
Likelihood	N/A
Impact	N/A
Recommendations	N/A
Comparison	The same goes for our webapp.

---



#### 4.3.4 Testing for bypassing authentication schema (OTG-AUTHN-004)

**Team2**

Likelihood: NA

Impact: NA

Risk: NA

---

Team2	
Observation	We did not find any possibility to bypass the authentication schema.
Discovery	N/A
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	Neither we found a possibility in our web app

---

Metric	Value
Access Vector	NA
Attack Complexity	NA
Privileges Required	NA
User Interaction	NA
Scope	NA
Confidentiality Impact	NA
Integrity Impact	NA
Availability Impact	NA

#### 4.3.5 Testing for Browser cache weakness (OTG-AUTHN-006)

Team2

Likelihood: 0

Impact: NA

Risk: NA

---

Team2	
Observation	The web app set the cache-control to no-cache an no-store and Pragma to no-cache
Discovery	By checking the <i>about:cache</i> in Firefox we saw the Headers
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	The same goes for our webapp

---

Metric	Value
Access Vector	NA
Attack Complexity	NA
Privileges Required	NA
User Interaction	NA
Scope	NA
Confidentiality Impact	NA
Integrity Impact	NA
Availability Impact	NA

#### 4.3.6 Testing for Weak password policy (OTG-AUTHN-007)

**Team2**

Likelihood: 6

Impact: 5

Risk: 5

---

Team2	
Observation	The registration process does not have a restriction for weak passwords. Furthermore the password can contain the username
Discovery	Tested manually the registration process with a one character password and the username as password
Likelihood	For every registration process the user has the possibility to choose a weak password
Implication	Brute Force is to easy for simple passwords
Recommendations	Introduce password restrictions
Comparison	The same problem we encountered in our webapp

---

Metric	Value
Access Vector	N
Attack Complexity	L
Privileges Required	N
User Interaction	N
Scope	U
Confidentiality Impact	L
Integrity Impact	L
Availability Impact	N

#### 4.3.7 Testing for weak password change or reset functionalities (OTG-AUTHN-009)

Team2

Likelihood: 10

Impact: 8

Risk: 5

Team2	
Observation	The password reset functionality can cause unwanted password change or the user to loose access to their account. Since the functionality changes the current password to a random one and sends an email to the user with the new password.
Discovery	Tested manually to reset the password. An error occurred and the email was not send but the password was changed.
Likelihood	It is very high that someone can enter an email address of another user and cause change their password without the users request.
Impact	The impact can be quite high especially in case when the email with the new password fails to send, the user can loose access to his account.
Recommendations	Instead of setting a new random password, sending an email with a link where the user can set a new password himself will be the best solution.
Comparison	Our webapp sends an email with link to change the password. The random number generated for the link is long, so trying to brute force it will take a very long time.

Metric	Value : 7.5
Access Vector	N
Attack Complexity	L
Privileges Required	N
User Interaction	N
Scope	U
Confidentiality Impact	N
Integrity Impact	N
Availability Impact	H

## 4.4 Authorization Testing

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

Team2

Likelihood: 0

Impact: 0

Risk: 0

---

Team2	
Observation	We could not find any path traversals .
Discovery	We used the <i>dotdotpwn</i> tool to find such traversals with the following command:  <pre>sudo ./dotdotpwn.pl -O -m http -h IP_ADDRESS</pre> -O is to get the operating system; -m is to indicate that the protocol is http and -h for the server ip; which gave no results. Afterwards we tried <i>w3af</i> and it did not find anything.
Likelihood	This is more an additional information for other attacks but it is a good help for attackers to find vulnerabilities faster
Impact	N/A
Recommendations	N/A
Comparison	The results were the same for our webapp.

---

#### 4.4.2 Testing for Privilege Escalation (OTG-AUTHZ-003)

**Team2**

Likelihood: 0

Impact: 0

Risk: 0

---

Team2	
Observation	It is not possible to escalate privileges of the user.
Discovery	We tried to change the user privilege by changing the user id after we saw that they are generated by incrementing from the first user ID, using <i>Burp</i> .
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	The same results apply for our web application.

---

#### 4.4.3 Testing for Insecure Direct Object References (OTG-AUTHZ-004)

**Team2**

Likelihood: 0

Impact: 0

Risk: 0

---

Team2	
Observation	It is not possible to retrieve objects belonging to other users or otherwise bypass authorization.
Discovery	We tried to change the user privilege by changing the user id after we saw that they are generated by incrementing from the first user ID, using <i>Burp</i> .
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	The same results apply for our web application.

---

## 4.5 Session Management Testing

### 4.5.1 Testing for Bypassing Session Management Schema(OTG-SESS-001)

**Team2**

Likelihood: 10

Impact: 7

Risk: 7

---

Team2	
Observation	There is no Random Session Token generated, nor any Session Time-out functionality. Also there is no cookie configuration.
Discovery	We observed the php code which we were given.
Likelihood	It is very likely that an attacker notices this, since the SessionID does not change.
Impact	This can cause a Session Hijack.
Recommendations	Generate random SessionIDs and create a Session Time-out functionality.
Comparison	Our web app generates random SessionIDs creates a session time-out, also set the cookie parameters.

---

Metric	Value : 6.7
Access Vector	L
Attack Complexity	H
Privileges Required	N
User Interaction	N
Scope	U
Confidentiality Impact	H
Integrity Impact	H
Availability Impact	N



#### 4.5.2 Testing for Cookies attributes(OTG-SESS-002)

<b>Team2</b>		Likelihood: 10 Impact: 6 Risk: 3
<b>Team2</b>		
Observation	The cookie for the PHP session id has a to general path ("/"). So the application is vulnerable to other web application on the same server. They will also get the cookie from the user. The HttpOnly is not set also there is no expiration time for the cookie.	
Discovery	We used the Firefox Web developer toolbar to analyze the cookie attributes.	
Likelihood	It is vary easy and straight forward to use this tool so it is highly likely for someone to find that out.	
Impact	The cookies can be read and used by other web applications that match the path value.	
Recommendations	Set the path as tight as possible.	
Comparison	Our web app has the all the cookie parameters set, and the timeout of the cookie is set to 10 minutes.	

Metric	Value : 6.1
Access Vector	L
Attack Complexity	L
Privileges Required	N
User Interaction	R
Scope	U
Confidentiality Impact	H
Integrity Impact	L
Availability Impact	N

### 4.5.3 Testing for Session Fixation(OTG-SESS-003)

**Team2**

Likelihood: 9

Impact: 5

Risk: 5

Team2	
Observation	The session id is not invalidated and therefore does not change after the user is authenticated. This means an attacker can force a known session id on a user. Once the user is authenticated the attacker can access also as authenticated user
Discovery	We used ZAP proxy to intercept the traffic.
Likelihood	This attack is pretty easy and can also be performed by low skilled people
Implication	The attacker can do everything the user can
Recommendations	Change the session id after logging in
Comparison	Our web app generates a new random SessionID after the user logs in.

Metric	Value : 8.1
Access Vector	N
Attack Complexity	L
Privileges Required	N
User Interaction	R
Scope	U
Confidentiality Impact	H
Integrity Impact	H
Availability Impact	N

#### 4.5.4 Testing for Exposed Session Variables (OTG-SESS-004)

Team2

Likelihood: 10

Impact: 6

Risk: 6

Team2	
Observation	The application from Team 2 use SSL certificate and therefore the session variables are not accessible during transport. But the Session variable is not destroyed after the user logs out.
Discovery	We used the <i>Firefox web developer tools</i> to analyze the cookies. Also the <i>Firebug</i> tool and we got the same outcome.
Likelihood	This attack is pretty easy and can also be performed by low skilled people
Impact	The attacker can read the session variables and depending on the information in them
Recommendations	Remove the cookie after logout.
Comparison	Our application destroys the session after the user logs out.

Metric	Value : 5.9
Access Vector	P
Attack Complexity	L
Privileges Required	N
User Interaction	R
Scope	U
Confidentiality Impact	H
Integrity Impact	H
Availability Impact	N

#### 4.5.5 Testing for Cross Site Request Forgery(OTG-SESS-005)

**Team2**

Likelihood: 10

Impact: 6

Risk: 6

---

Team2	
Observation	The application is vulnerable to CRSF.
Discovery	WE used <i>w3af</i> tool to analyze the website and it showed that there is CRSF vulnerability.
Likelihood	N/A
Impact	N/A
Recommendations	N/A
Comparison	N/A

---

#### 4.5.6 Testing for logout functionality(OTG-SESS-006)

Team2

Likelihood: 4

Impact: 6

Risk: 5

Team2	
Observation	The logout functionality has been tested and works without any problems. The user gets correctly logged out and pages where authentication is needed can't be accessed afterwards. Also reusing the session id does not work. But the application seems to have no automatic logout after a certain amount of time.
Discovery	We tested the functionality manually and used the Chrome extension "Advanced Rest Client" to reuse the session cookie
Likelihood	The only problem is that user sometimes only close the browser tab and then the session continues to exist.
Impact	An attacker would be logged in if he can access afterwards the computer. Possible scenario Internet cafe or something similar
Recommendations	implement an automatic server side logout
Comparison	Our webapp has the same problem that it does not log out after the web browser is closed, but it does log out after 10 minutes of inactivity.

Metric	Value : 4.6
Access Vector	P
Attack Complexity	L
Privileges Required	N
User Interaction	N
Scope	U
Confidentiality Impact	N
Integrity Impact	H
Availability Impact	N

#### 4.5.7 Test Session Timeout(OTG-SESS-007)

**Team2**

Likelihood: 4

Impact: 6

Risk: 7

<b>Team2</b>	
Observation	The application has the timeout of the session set to the browsers session lifetime.
Discovery	We tested the functionality manually and used the Chrome developertools to check the cookies.
Likelihood	The same as for OTG-SESS-006. Public computers are here the biggest problem
Impact	An attacker is directly authenticated if the session is not ended
Recommendations	Implement a server side session invalidation and delete the stored data on the client
Comparison	Our webapp has a session timeout of 10 minutes, which invalidates the session after 10 minutes of inactivity.

Metric	Value : 4.6
Access Vector	P
Attack Complexity	L
Privileges Required	N
User Interaction	N
Scope	U
Confidentiality Impact	N
Integrity Impact	H
Availability Impact	N

#### 4.5.8 Testing for Session puzzling(OTG-SESS-008)

**Team2**

Likelihood: 0

Impact: 0

Risk: 0

---

Team2	
Observation	The application has only one authorization method so a session puzzling is not applicable.
Discovery	Manually searched
Likelihood	NA
Implication	NA
Recommendations	NA
Comparison	We provide also only one possibility to login so session puzzling is not possible

---

## 4.6 Data Validation Testing

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

<b>Team2</b>		Likelihood: 8 Impact: 5 Risk:5
<b>Team2</b>		
Observation	We observed no reflected cross site scripting vulnerability.	
Discovery	It seems that all parameters are stored in the database before inserting the values in the HTML.	
Likelihood	N/A	
Implication	N/A	
Recommendations	N/A	
Comparison	The same results apply for our web application.	



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

<b>Team2</b>		Likelihood: 8 Impact: 5 Risk:5
<hr/>		
<b>Team2</b>		
Observation	We cannot insert javascript code in the webapp.	
Discovery	We tried to insert the following javascript code in the username of the sign up page but there is a restriction in the length of the username to 23 characters therefore it failed.  <script>alert("1");</script>	
Likelihood	N/A	
Impact	N/A	
Recommendations	N/A	
Comparison	Our web app does not allow XSS but because we sanitize the input of the user.	

---

#### 4.6.3 Testing for HTTP Verb Tampering(OTG-INPVAL-003)

**Team2**

Likelihood: 0

Impact: 0

Risk: 0

---

Team2	
Observation	We did not observe any notable behavior.
Discovery	We used a script to automatically test the HTTP methods. Methods that were allowed <ul style="list-style-type: none"><li>• OPTIONS</li><li>• GET</li><li>• POST</li></ul> Methods that were rejected <ul style="list-style-type: none"><li>• TRACE</li><li>• PUT</li><li>• PROPFIND</li></ul>
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	Both webapps have the same allowed methods.

---

```
samurai@samurai-wtf:Desktop$ ./tet 127.0.0.1
GET HTTP/1.1 200 OK
POST HTTP/1.1 200 OK
PUT HTTP/1.1 405 Method Not Allowed
TRACE HTTP/1.1 405 Method Not Allowed
CONNECT HTTP/1.1 400 Bad Request
OPTIONS HTTP/1.1 200 OK
PROPFIND HTTP/1.1 405 Method Not Allowed
```

Figure 4.3: Results

```
#!/bin/bash

for webservmethod in GET POST PUT TRACE CONNECT OPTIONS PROPFIND;
do
    printf "$webservmethod " ;
    printf "$webservmethod / HTTP/1.1\nHost: $1\n\n" | nc -q 1 $1 80 | grep "HTTP/1.1"
done
```

Figure 4.4: Script used for testing

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

Team2

Likelihood: 0

Impact: 0

Risk: 0

Team2	
Observation	We observed that no SQL Injection was possible. Since we knew that the other team had to use Mysql we tested also specifically for Mysql
Discovery	We tried inserting various SQL statements in the fields of using <i>sqlmap</i> tool and failed, also we did static test of the PHP code using <i>RIPS</i> which did not find any.
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	Our web application is also immune to SQL Injections

Using *RIPS* with verbosity level 1 and vulnerability all server-side we get the above results. The 3 cases of *SQL Injection* are *false-positives*. The input in the query is already set and is not affected by the user input.

The case of file manipulation is *false-positive* since the variables passed as parameters here have nothing to do with user input.

```
Userinput reaches sensitive sink. (Blind exploitation)
• 30: move_uploaded_file move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $target_file))
    7: $target_file = $target_dir . $_SESSION['uid'] . basename($_FILES['fileToUpload']['name']);
    6: $target_dir = "uploads/";
```

Figure 4.7: RIPS

Also using the following *sqlmap* command we did not find any SQL Injection possible.

```
sqlmap -u https://localhost/ex1/securecoding/
```

## 4 Detailed Report

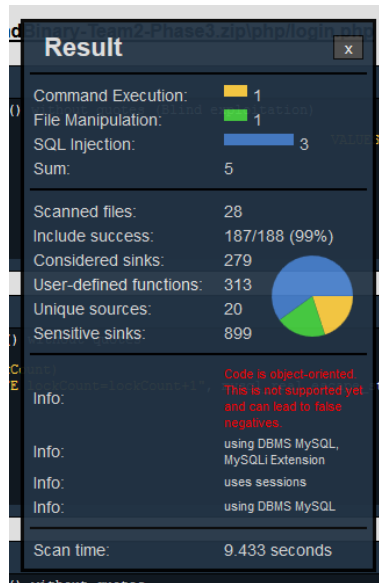


Figure 4.5: RIPS

```
Userinput reaches sensitive sink due to insecure usage of mysql_real_escape_string() without quotes
* 46: mysql_query $query = mysql_query(sprintf("INSERT INTO 'logins' (ipAddr, lockCount)
VALUES ('%s', 1) ON DUPLICATE KEY UPDATE lockCount=lockCount+1", mysql_real_escape_string($_SERVER['REMOTE_ADDR']))); or die (mysql_error());
```

Figure 4.6: RIPS

```
try to explicitly set it using option --dbms
[10:53:26] [WARNING] URI parameter '#1*' is not injectable
[10:53:26] [CRITICAL] all tested parameters appear to be not injectable. Try to increase '--level'/'--risk' value
s to perform more tests. Also, you can try to rerun by providing either a valid value for option '--string' (or '--regexp')
[10:53:26] [WARNING] HTTP error codes detected during run:
```

Figure 4.8: RIPS

### Team3

Using *RIPS* with verbosity level 1 and vulnerability all server-side we get the above results.

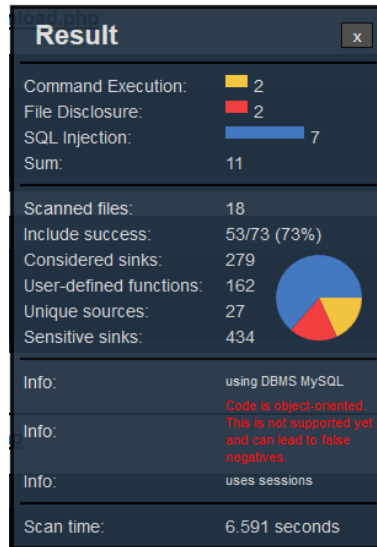


Figure 4.9: RIPS

For the *SQL Injection* cases the input data is sanitized before being executed into the queries. So the warnings given by *RIPS* are *false-positive*.

Also using the following *sqlmap* command we did not find any SQL Injection possible.

```
sqlmap -u https://localhost/SecureCoding-Group3/online_banking/
```

```
try to explicitly set it using option '--dbms'
[10:53:26] [WARNING] URI parameter '#1*' is not injectable
[10:53:26] [CRITICAL] all tested parameters appear to be not injectable. Try to increase '--level'/'--risk' value
s to perform more tests. Also, you can try to rerun by providing either a valid value for option '--string' (or '
--regexp')
[10:53:26] [WARNING] HTTP error codes detected during run:
```

Figure 4.10: RIPS

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

**Team2**

Likelihood: 0

Impact: 0

Risk: 0

---

<b>Team2</b>	
Observation	We did not find any vulnerability regarding code injection and local or remote file inclusion in our web app. Team 2 we were able to upload a file with code but there was no abnormal behavior of the web app.
Discovery	Tried to perform a command execution via the backticks (') and also the semicolon (;) in the filename but our webapp correctly handled the files without injections. We upload files with code written in them but nothing happened.
Likelihood	N/A
Implication	N/A
Recommendations	N/A

---

#### 4.6.6 Testing for Command Injection(OTG-INPVAL-013)

##### Team2

The case of command execution is yet again a *false-positive* since the variable entered into the *exec* function is a file. If the file contains malicious user input is another problem, and we have to do the check for that elsewhere.

```
Userinput reaches sensitive sink.  
32: exec exec("./c-fileparser/fileparser " . $target_file, $output);  
    • 7: $target_file = $target_dir . $_SESSION['uid'] . basename($_FILES['fileToUpload']['name']);  
      6: $target_dir = "uploads/";
```

Figure 4.11: RIPS



### Team3

RIPS finds two vulnerabilities for file disclosure and command execution in *application-Download.php*, which are *false-positive* since the variable is taken from the database, the user does not input it directly.

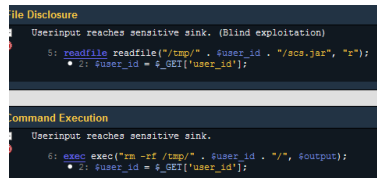


Figure 4.12: RIPS

RIPS finds two vulnerabilities for file disclosure and command execution in *customer.inc.php*, the same case applies here, they are *false-positive* the variables passed on the sensitive functions are not user input.

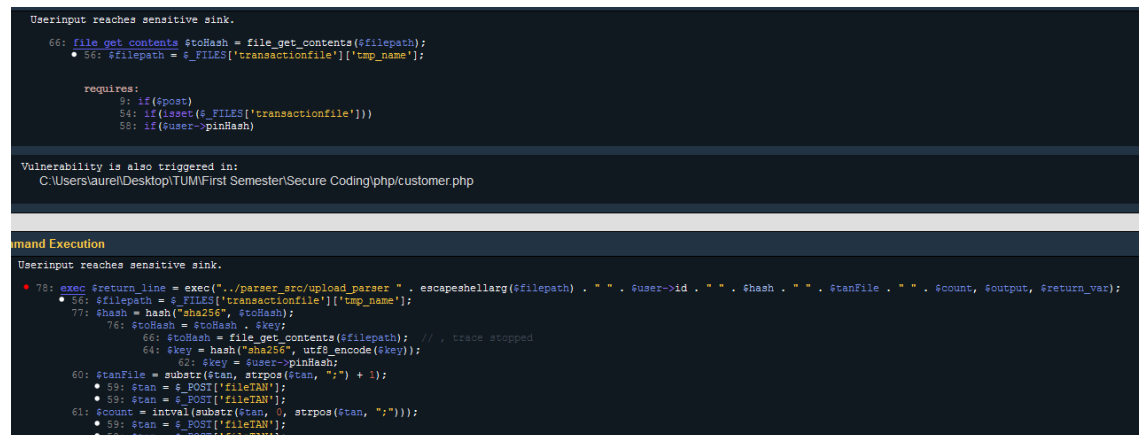


Figure 4.13: RIPS

#### 4.6.7 Testing for Buffer overflow, Testing for Heap overflow, Testing for Stack overflow, Testing for Format string (OTG-INPVAL-014)

**Team2**

Likelihood: 0

Impact: 0

Risk: 0

---

<b>Team2</b>	
Observation	We did not find any vulnerability regarding buffer overflow, heap overflow, stack overflow or string formatting
Discovery	We used w3af to locate such vulnerabilities.
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	Our web application is also immune to buffer overflow, heap overflow, stack overflow and string formatting

---

#### 4.6.8 Testing for incubated vulnerabilities(OTG-INPVAL-015)

<b>Team2</b>		Likelihood: 7 Impact: 7 Risk: 5
<hr/>		
<b>Team2</b>		
Observation	A part of the XSS injection counts also to this vulnerability thats possible on the web app of team 21. Code like the example on the owasp page for OTG-INPVAL-015 could exploit the web app	
Discovery	We knew that stored XSS is possible so also this attack works and someone could hijack an admins account simply by creating an user	
Likelihood	The attack is pretty easy and the employee only has to view the accounts page and if the attacker is a bit skilled the employee does not even discover that something was wrong	
Implication	The attacker can hijack the session and do all the other things possible with XSS	
Recommendations	Validate and escape the user input	
Comparison	Our web application has the same vulnerability but there it is a lot more restricted. Only really short injection code can be used so the possibilities are limited.	

---

## 4.7 Error Handling

### Team2

Team2 provides an error for the transactions using file upload only when the file format is not correct. Every other time when the correct file format is uploaded with incorrect data, the file is accepted even though no transaction is executed.

## 4.8 Business Logic Testing

### 4.8.1 Test Business Logic Data Validation(OTG-BUSLOGIC-001)

Team2		Likelihood: 0 Impact: 0 Risk:0
Team2		
Observation	Tests show that data validation is both: client side and server side.	
Discovery	We intercepted the input before it gets send to the server using <i>Burp</i> and manipulated the data, and we received an error message.	
Likelihood	N/A	
Implication	N/A	
Recommendations	N/A	
Comparison	We got the same result with our application.	

#### 4.8.2 Test Number of Times a Function Can be Used Limits(OTG-BUSLOGIC-005)

<b>Team2</b>		Likelihood: 0 Impact: 0 Risk:0
<hr/>		
<b>Team2</b>		
Observation	We tried inserting the same tan multiple times.	
Discovery	The web application did not accept requests with a TAN that was already used.	
Likelihood	N/A	
Implication	N/A	
Recommendations	N/A	
Comparison	We got the same result with our application.	

### 4.8.3 Test Upload of Unexpected File Types (OTG-BUSLOGIC-008)

Team2	
Observation	The is no possibility to upload any unexpected file type.
Discovery	We tried to upload different type of files from the one presented on the website. There was an error every time the file had different extension from the one specified on the website.
Likelihood	N/A
Implication	N/A
Comparison	Our application reacts the same way, it does not allow uploading file types different from the one specified.

## **4.9 Client Side Testing**

We did not do any client-side testing on this website.