

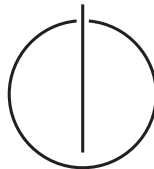


DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Secure Coding Phase 2

Team 3: Patrick Sattler, Aurel Roci, Stefan Kohler



Contents

1	Time Tracking Table	1
2	Vulnerabilities Overview	2
2.1	Secode21	2
2.1.1	Static Session ID	2
2.1.2	Stored XSS in Registration	2
2.1.3	Missing Lock Out Mechanism	3
2.2	Team3 Online Banking	3
2.3	Vulnerability Overview	3
3	Detailed Report	4
3.1	Configuration and Deploy Management Testing	5
3.1.1	Test File Extensions Handling for Sensitive Information (OTG-CONFIG-003)	5
3.1.2	Test HTTP Methods (OTG-CONFIG-006)	7
3.1.3	Test HTTP Strict Transport Security (OTG-CONFIG-007)	8
3.1.4	Test RIA cross domain policy (OTG-CONFIG-008)	9
3.2	Identity Management Testing	10
3.2.1	Test Role Definitions (OTG-IDENT-001)	10
3.2.2	Test User Registration Process (OTG-IDENT-002)	11
3.2.3	Test Account Provisioning Process (OTG-IDENT-003)	12
3.2.4	Testing for Account Enumeration and Guessable User Account (OTG-IDENT-004)	13
3.2.5	Testing for Weak or unenforced username policy (OTG-IDENT-005)	14
3.3	Authentication Testing	15
3.3.1	Testing for Credentials Transported over an Encrypted Channel(OTG-AUTHN-001)	15
3.3.2	Testing for default credentials(OTG-AUTHN-002)	16
3.3.3	Testing for bypassing authentication schema (OTG-AUTHN-004)	17
3.3.4	Testing for Browser cache weakness (OTG-AUTHN-006)	18
3.3.5	Testing for Weak password policy (OTG-AUTHN-007)	19

3.3.6	Testing for Weaker authentication in alternative channel (OTG-AUTHN-010)	20
3.4	Authorization Testing	21
3.4.1	Testing Directory traversal/file include (OTG-AUTHZ-001)	21
3.4.2	Testing for Privilege Escalation (OTG-AUTHZ-003)	22
3.4.3	Testing for Insecure Direct Object References (OTG-AUTHZ-004)	23
3.5	Session Management Testing	24
3.5.1	Testing for Bypassing Session Management Schema(OTG-SESS-001)	24
3.5.2	Testing for Cookies attributes(OTG-SESS-002)	25
3.5.3	Testing for Session Fixation(OTG-SESS-003)	26
3.5.4	Testing for Exposed Session Variables (OTG-SESS-004)	27
3.5.5	Testing for Cross Site Request Forgery (OTG-SESS-005)	28
3.5.6	Testing for logout functionality(OTG-SESS-006)	29
3.5.7	Test Session Timeout(OTG-SESS-007)	30
3.5.8	Testing for Session puzzling(OTG-SESS-008)	31
3.6	Data Validation Testing	32
3.6.1	Testing for Reflected Cross Site Scripting(OTG-INPVAL-001)	32
3.6.2	Testing for Stored Cross Site Scripting(OTG-INPVAL-002)	33
3.6.3	Testing for HTTP Verb Tampering(OTG-INPVAL-003)	34
3.6.4	Testing for HTTP Parameter pollution(OTG-INPVAL-004)	36
3.6.5	Testing for SQL Injection (OTG-INPVAL-005)	37
3.6.6	IMAP/SMTP Injection(OTG-INPVAL-011)	38
3.6.7	Testing for Code Injection, Testing for Local File Inclusion, Testing for Remote File Inclusion(OTG-INPVAL-012)	39
3.6.8	Testing for Command Injection(OTG-INPVAL-013)	40
3.6.9	Testing for Buffer overflow, Testing for Heap overflow, Testing for Stack overflow, Testing for Format string (OTG-INPVAL-014)	41
3.6.10	Testing for incubated vulnerabilities(OTG-INPVAL-015)	42
3.6.11	Testing for HTTP Splitting/Smuggling(OTG-INPVAL-016)	43
3.7	Error Handling	43
3.8	Cryptography	43
3.9	Business Logic Testing	44
3.9.1	Test Business Logic Data Validation(OTG-BUSLOGIC-001)	44
3.9.2	Test Ability to Forge Requests(OTG-BUSLOGIC-002)	45
3.9.3	Test Integrity Checks(OTG-BUSLOGIC-003)	46
3.9.4	Test for Process Timing(OTG-BUSLOGIC-004)	47
3.9.5	Test Number of Times a Function Can be Used Limits(OTG-BUSLOGIC-005)	48
3.9.6	Testing for the Circumvention of Work Flows(OTG-BUSLOGIC-006)	49

Contents

3.9.7	Test Defenses Against Application Misuse(OTG-BUSLOGIC-007)	50
3.9.8	Test Upload of Unexpected File Types(OTG-BUSLOGIC-008) . .	51
3.9.9	Test Upload of Malicious Files(OTG-BUSLOGIC-009)	52
3.10	Client Side Testing	53
3.10.1	Testing for DOM based Cross Site Scripting (OTG-CLIENT-001)	53
3.10.2	Testing for JavaScript Execution (OTG-CLIENT-002)	54
3.10.3	Testing for HTML Injection (OTG-CLIENT-003)	55
3.10.4	Testing for Client Side URL Redirect (OTG-CLIENT-004)	56
3.10.5	Testing for CSS Injection (OTG-CLIENT-005)	57
3.10.6	Testing for Client Side Resource Manipulation (OTG-CLIENT-006)	58
3.10.7	Test Cross Origin Resource Sharing (OTG-CLIENT-007)	59
3.10.8	Testing for Cross Site Flashing (OTG-CLIENT-008)	60
3.10.9	Testing for Clickjacking (OTG-CLIENT-009)	61
3.10.10	Testing WebSockets (OTG-CLIENT-010)	62
3.10.11	Test Web Messaging (OTG-CLIENT-011)	63
3.10.12	Test Local Storage (OTG-CLIENT-012)	64
	Glossary	66
	Acronyms	67

1 Time Tracking Table

2 Vulnerabilities Overview

Based on our testing, we identified the following vulnerabilities for the Secode21 Bank and the OnlineBanking Bank:

2.1 Secode21

2.1.1 Static Session ID

- Likelihood: *high*
- Implication: *high*
- Risk: *high*
- Reference: OWASP OTG-SESS-003 (see section ??)

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 Stored XSS in Registration

- Likelihood: *medium*
- Implication: *high*
- Risk: *high*
- Reference: OWASP OTG-INPVAL-002 (see section ??)

Using stored cross-site-scripting attacks, one can inject JavaScript code, that is run, when the Administrator/Employee logs in. Arbitrary code can be loaded from a third party page.

2.1.3 Missing Lock Out Mechanism

- Likelihood: *high*
- Implication: *medium*
- Risk: *medium*
- Reference: OWASP OTG-AUTHN-003 (see section ??)

The application has no lock out mechanism, which allows brute force attacks on known usernames and testing for a valid password

2.2 Team3 Online Banking

2.3 Vulnerability Overview

3 Detailed Report

The following pages describe for each test how both applications Secode21 and Online Banking Bank performed. The test is divided in different sections following the OWASP Testing Guide v4.

3.1 Configuration and Deploy Management Testing

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

Secode21

Likelihood: 8

Impact: 5

Risk: 5

Secode21	
Observation	File extensions are handled correctly but while testing we found a folder called SQL with sql files and pdf files describing the database structure and the sql commands used by the web application.
Discovery	Thanks to the tool <i>dotdotpwn</i> , that tries automatically different URLs, we found the SQL folder. We passed the following parameters:
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.
Implication	These vulnerabilities could help attackers to perform sql injection attacks because you know the database structure and the sql commands used in the implementation of the web application.
Recommendations	Block the access to sql files and to those folders that describe the web applications architecture.
Comparison	Our web application handles file extensions correctly, but it is possible to access the compiled c program that handles the batch files. This is a problem because you can reverse engineer the code and use the vulnerabilities found. This scenario is possible but is very complex.

3 Detailed Report

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

3.1.2 Test HTTP Methods (OTG-CONFIG-006)

Secode21

Likelihood: 0

Impact: 0

Risk: 0

Secode21	
Observation	he application is not accessable 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.
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	The same applies for our web application.

3.1.3 Test HTTP Strict Transport Security (OTG-CONFIG-007)

Secode21

Likelihood: 0

Impact: 0

Risk: 0

Secode21	
Observation	The <i>HTTP Strict Transport Security</i> protocol is never used.
Discovery	We used <i>Charles Web Proxy</i> to check the HTTP response headers and the <i>Strict-Transport-Security</i> header was not found.
Likelihood	N/A
Implication	N/A
Recommendations	It would be better so transport some data via https and use the HSTS protocol.
Comparison	The same results apply for our web application.

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

Overview	Request	Response	Summary	Chart	Notes
HTTP/1.1 200 OK Date Fri, 20 Nov 2015 18:06:39 GMT Server Apache/2.2.22 (Ubuntu) X-Powered-By PHP/5.3.10-1ubuntu3.21 Expires Thu, 19 Nov 1981 08:52:00 GMT Cache-Control no-store, no-cache, must-revalidate, post-check=0, pre-check=0 Pragma no-cache Vary Accept-Encoding Content-Encoding gzip Content-Length 419 Keep-Alive timeout=5, max=99 Connection Keep-Alive Content-Type text/html					

3.1.4 Test RIA cross domain policy (OTG-CONFIG-008)

Secode21

Likelihood: 0

Impact: 0

Risk: 0

Secode21	
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

3.2 Identity Management Testing

3.2.1 Test Role Definitions (OTG-IDENT-001)

Secode21

Likelihood: 10

Impact: 4

Risk: 4

Secode21	
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 employee. 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 are described.
Likelihood	It is very likely that people find this information.
Implication	There is no direct implication, but knowing the roles and their functionality helps with other attacks.
Recommendations	Don't describe the roles on the web page.
Comparison	Our web application provides the same roles, but the roles are not described on the web page.

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

3.2.2 Test User Registration Process (OTG-IDENT-002)

Secode21

Likelihood: 5

Impact: 5

Risk: 5

Secode21	
Observation	Any person can register themselves as a user and this registration then 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 name, surname, phone number, 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, especially the email address and phone number can be verified very easily. The name can be validated by hand if a customer would go to the bank and the employee would then accept his registration.
Comparison	Our web application doesn't require a phone number for the registration as 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.

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

3.2.3 Test Account Provisioning Process (OTG-IDENT-003)

Secode21

Likelihood: N/A

Impact: N/A

Risk: N/A

Secode21	
Observation	Our observation showed us that employees can accept customer registrations and can make customer accounts to employee accounts.
Discovery	All the observations were made with the <i>Chrome</i> web browser.
Implication	If an employee account gets hacked you can make even other accounts to employees and accept new registrations.
Recommendations	N/A
Comparison	In our web application the employee doesn't make customer accounts to employee accounts but rather accepts special employee registrations. It makes no difference in the security.

Metric	Value
Access Vector	N
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

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

Secode21

Likelihood: 0

Impact: 0

Risk: 0

Secode21	
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.

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

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

Secode21

Likelihood: 0

Impact: 0

Risk: 0

Secode21	
Observation	The usernames are not auto-generated and therefore there is no special structure in the usernames.
Discovery	No tool is used here. The username field in the registration form gives us all the information we need.
Implication	N/A
Recommendations	N/A
Comparison	The same 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

3.3 Authentication Testing

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

Secode21

Likelihood: 0

Impact: 0

Risk: 0

Secode21	
Observation	This ensures that our credentials are sent using an encrypted channel and that the credentials are not readable by a malicious user using a sniffer.
Discovery	We used <i>Zed Attack Proxy (ZED)</i> in order to capture packet headers and to inspect them. We saw that the request addressed to the web application is using the HTTPS protocol.
Likelihood	N/A
Implication	N/A
Recommendations	N/A.
Comparison	The same applies for our web application.

3.3.2 Testing for default credentials(OTG-AUTHN-002)

Secode21

Likelihood: 10

Impact: 4

Risk: 6

Secode21	
Observation	We found out that there exists the default credentials <i>admin:admin</i>
Discovery	We were already given these credentials.
Likelihood	It is very likely that people find this information.
Implication	The attacker gain administrator access in the web application.
Recommendations	Use other credentials for testing, or delete the default ones after you launch the application.
Comparison	Our web application has a different combination of <i>user:password</i> .

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

3.3.3 Testing for bypassing authentication schema (OTG-AUTHN-004)

3.3.4 Testing for Browser cache weakness (OTG-AUTHN-006)

3.3.5 Testing for Weak password policy (OTG-AUTHN-007)

Secode21

Likelihood: 6

Impact: 6

Risk: 6

Secode21	
Observation	For the password policy the following applies: <ul style="list-style-type: none"> • There is no password policy enforced • The password cannot be changed by the user • The password can contain parts of the username or even be the username itself
Discovery	The password can be of any length so we tried entering a single character for example: @ and after the registration we logged in as an employee and we did not see any new registration.
Likelihood	As there is no password policy enforced, any password can be chosen when registering a new account.
Implication	If weak passwords are chosen the time of brute force attacks is low, those passwords cannot provide proper security.
Recommendations	A password policy should enforce strong passwords when new users register a new account. Users should not be able to use (parts of) their usernames as passwords.
Comparison	Our web application has a stated password policy of larger than 4 characters. But it will accept registrations with password length less than 4.

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

3.3.6 Testing for Weaker authentication in alternative channel (OTG-AUTHN-010)

3.4 Authorization Testing

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

Secode21

Likelihood: 8

Impact: 5

Risk: 5

Secode21	
Observation	Directory traversal is possible for two folders that don't contain an <i>index</i> file. That are the two folders <i>SQL</i> and <i>fpdf17</i> . Especially the possibility to access the SQL folder that contains information about the database structure is a huge vulnerability. On the other hand it wasn't possible to access any system files.
Discovery	We used <i>Charles Web Proxy</i> to analyze the web traffic to find entry points that could possibly be used to access files. We didn't find any vulnerability there, but by testing manually with <i>wget</i> we found the two folders where directory traversal is possible. Running the tool <i>dotdotpwn</i> didn't provide us any system resources that are accessible.
Likelihood	It is quite likely to find the two folders.
Implication	There is no direct impact, but the SQL database structure could help with SQL injection attacks.
Comparison	Make these folders not accessible trough web requests.

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

3.4.2 Testing for Privilege Escalation (OTG-AUTHZ-003)

Secode21

Likelihood: 0

Impact: 0

Risk: 0

Secode21	
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.

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

3.5 Session Management Testing

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

Secode21

Likelihood: 0

Impact: 0

Risk: 0

Secode21	
Observation	PHP session ids are used and such session ids normally can't be bypassed that means calculated easily
Discovery	We used the Chrome extension "Advanced Rest Client" to analyze the Request and the Cookies
Likelihood	NA
Implication	NA
Recommendations	NA
Comparison	Our web application also uses PHP session ids

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

3.5.2 Testing for Cookies attributes(OTG-SESS-002)

3.5.3 Testing for Session Fixation(OTG-SESS-003)

Secode21

Likelihood: 8

Impact: 5

Risk: 3

Secode21	
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 the Chrome extension "Advanced Rest Client" to analyze the Request and the Cookies
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 application has exact the same vulnerability

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

3.5.4 Testing for Exposed Session Variables (OTG-SESS-004)

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

3.5.6 Testing for logout functionality(OTG-SESS-006)

3.5.7 Test Session Timeout(OTG-SESS-007)

3.5.8 Testing for Session puzzling(OTG-SESS-008)

3.6 Data Validation Testing

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

Secode21		Likelihood: 8 Impact: 5 Risk:5
Secode21		
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.	

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

Secode21		Likelihood: 8 Impact: 5 Risk:5
Secode21		
Observation	We observed several possibilities to execute a stored XSS attack. But not all of them could be exploited as the length of the corresponding database fields was often very restricted. We manually tried to inject JavaScript code in every input field. Therefore we used the following code, which just alerts a message.	
Discovery	We inserted Javascript code in the name field on the register page. When we logged in as an employee the script was executed. There were cases when the script caused for new registered users after the script was entered to not appear.	
Likelihood	This vulnerability can be easily detected, but require some JavaScript knowledge to exploit it. Therefore we estimated the likelihood to be medium.	
Implication	The implications are severe as we proofed that it is possible to steal the session. As we injected the code on the admin landing-page, which implies that we were able to act as an admin and register an abitrary account.	
Recommendations	Implement a input sanitation on all input fields on the backend side! Try to use whitelisting for the different datatypes and do not rely on the frontend input validation.	
Comparison		

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

3.6.3 Testing for HTTP Verb Tampering(OTG-INPVAL-003)

Secode21

Likelihood: 0

Impact: 0

Risk: 0

Secode21	
Observation Discovery	<p>We did not observe any notable behavior.</p> <p>We used the Zed Attack Proxy (ZAP) to change the HTTP requests method to the ones listed below. The requests that were allowed responded with the index page or an empty body. The rejected requests responded with an error message in the body.</p> <p>Methods that were allowed</p> <ul style="list-style-type: none"> • HEAD • OPTIONS • GET • POST <p>Methods that were rejected</p> <ul style="list-style-type: none"> • TRACE • PUT • PROPFIND
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	Both webapps have the same allowed methods.

```
samurai@samurai-wtf:Desktop$ ./tamper 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
samurai@samurai-wtf:Desktop$ █
```

Figure 3.1: 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 3.2: Script used for testing

3.6.4 Testing for HTTP Parameter pollution(OTG-INPVAL-004)

3.6.5 Testing for SQL Injection (OTG-INPVAL-005)

Secode21		Likelihood: 8 Impact: 5 Risk:5
Secode21		
Observation	We observed that no SQL Injection was possible.	
Discovery	We tried inserting various SQL statements in the fields of using <i>SQL Inject Me</i> tool and failed.	
Likelihood	N/A	
Implication	N/A	
Recommendations	N/A	
Comparison	Our web application is also immune to SQL Injections	

3.6.6 IMAP/SMTP Injection(OTG-INPVAL-011)

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

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

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

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

3.6.11 Testing for HTTP Splitting/Smuggling(OTG-INPVAL-016)

3.7 Error Handling

Team21

Team21 does not provide a lot of error messages for incorrect inputs (e.g. incorrect TAN length, wrong TAN, TAN used).

Based on the client side input validation, there are also no messages for manipulated input via proxy or by removing the validation patterns, which can lead to problems. Examples would be a malformed email which results in a not working account or a longer input than expected, which cuts off the end of the input. There are some cases when the page returns the path of the file where the error occurred.

Team3

3.8 Cryptography

3.9 Business Logic Testing

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

Secode21		Likelihood: 0 Impact: 0 Risk:0
Secode21		
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.	

3.9.2 Test Ability to Forge Requests(OTG-BUSLOGIC-002)

3.9.3 Test Integrity Checks(OTG-BUSLOGIC-003)

3.9.4 Test for Process Timing(OTG-BUSLOGIC-004)

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

Secode21		Likelihood: 0 Impact: 0 Risk:0
Secode21		
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.	

3.9.6 Testing for the Circumvention of Work Flows(OTG-BUSLOGIC-006)

3.9.7 Test Defenses Against Application Misuse(OTG-BUSLOGIC-007)

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

3.9.9 Test Upload of Malicious Files(OTG-BUSLOGIC-009)

3.10 Client Side Testing

3.10.1 Testing for DOM based Cross Site Scripting (OTG-CLIENT-001)

Secode21

Likelihood: 0

Impact: 0

Risk: 0

Secode21	
Observation	Observing the HTML source code showed us that they don't use javascript and therefore there can't be any DOM XSS vulnerabilities.
Discovery	We used <i>Chrome</i> and its developer tools to take a look at the HTML source code.
Likelihood	N/A
Implication	N/A
Comparison	Our web application uses javascript in many different cases, but we couldn't find any DOM XSS vulnerabilities.

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

3.10.2 Testing for JavaScript Execution (OTG-CLIENT-002)**Secode21**

Likelihood: 9

Impact: 10

Risk: 9

Secode21	
Observation	We found several XSS vulnerabilities allowing the execution of arbitrary javascript code in the clients browser.
Discovery	We used the tools <i>w3af</i> and <i>ZED Attack Proxy</i> to find some XSS vulnerabilities and found enough of them.
Likelihood	It is very likely that these vulnerabilities are found and you don't need much experience to use them.
Implication	The vulnerabilities found can be used to hijack the session of an user, accept user registrations or even making user accounts to employee accounts.
Comparison	Our app is also vulnerable against XSS attacks but the difficulty is higher as in their web application. More experienced people are necessary to exploit them.

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

3.10.3 Testing for HTML Injection (OTG-CLIENT-003)**Secode21**

Likelihood: 8

Impact: 7

Risk: 7

Secode21	
Observation	The HTML injection vulnerability exists
Discovery	The vulnerability was found by the tools <i>w3af</i> and <i>ZED Attack Proxy</i> .
Likelihood	It is quite likely that this vulnerability is found and can be used very easily.
Implication	vulnerability can have many consequences, like disclosure of a user's session cookies that could be used to impersonate the victim, or, more generally, it can allow the attacker to modify the page content seen by the victims.
Comparison	Our web application is vulnerable as well, but javascript validations and text length restrictions of the input fields make it more difficult to exploit these vulnerabilities.

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

3.10.4 Testing for Client Side URL Redirect (OTG-CLIENT-004)

Secode21

Likelihood: 0

Impact: 0

Risk: 0

Secode21	
Observation	We couldn't find any client side redirections in the html source code of the web application and therefore exists no client side url redirect vulnerability.
Discovery	We used <i>Chrome</i> and its web inspector to look at the html code.
Likelihood	N/A
Implication	N/A
Comparison	The same results apply 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

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

Secode21

Likelihood: 0

Impact: 0

Risk: 0

Secode21	
Observation	Our search didn't find any spots in the html source code where there is user generated input used to change some css attributes.
Discovery	Chrome and its web inspector were used to read the html code.
Likelihood	N/A
Implication	N/A
Comparison	The same results apply 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

3.10.6 Testing for Client Side Resource Manipulation (OTG-CLIENT-006)

3.10.7 Test Cross Origin Resource Sharing (OTG-CLIENT-007)

3.10.8 Testing for Cross Site Flashing (OTG-CLIENT-008)

3.10.9 Testing for Clickjacking (OTG-CLIENT-009)

3.10.10 Testing WebSockets (OTG-CLIENT-010)

3.10.11 Test Web Messaging (OTG-CLIENT-011)

3.10.12 Test Local Storage (OTG-CLIENT-012)

Glossary

computer is a machine that. . .

Acronyms

TUM Technische Universität München.