DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Secure Coding Phase 2

Team 3: Patrick Sattler, Aurel Roci, Stefan Kohler

Executive Summary

Secode21

We found several vulnerabilities, which could cause severe damage to the *Secode21*. 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 vulnerabilites then the *Secode21* 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

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

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

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

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 Brute Force Password

- Likelihood: *high*
- Implication: *medium*
- Risk: *medium*
- Reference: OWASP OTG-AUTHN-003

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

2.1.4 Directory Traversal

- Likelihood: *high*
- Implication: *medium*
- Risk: *medium*
- Reference: OWASP OTG-AUTHN-001

It is possible to access *SQL* directory through the *url*.

2.2 Team3 Online Banking

2.2.1 Static Session ID

- Likelihood: *high*
- Implication: *high*
- Risk: *high*
- Reference: OWASP OTG-SESS-003

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.

2.2.2 Stored XSS in Registration

- Likelihood: *medium*
- Implication: *high*
- Risk: *high*
- Reference: OWASP OTG-INPVAL-002

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.

2.2.3 Brute Force Password

- Likelihood: *high*
- Implication: *medium*
- Risk: *medium*
- Reference: OWASP OTG-AUTHN-003

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

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	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.
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.

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 *netcat* using the following command: *nc IP_ADDRESS 80* , which did not work.
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	The same applies for our web application.

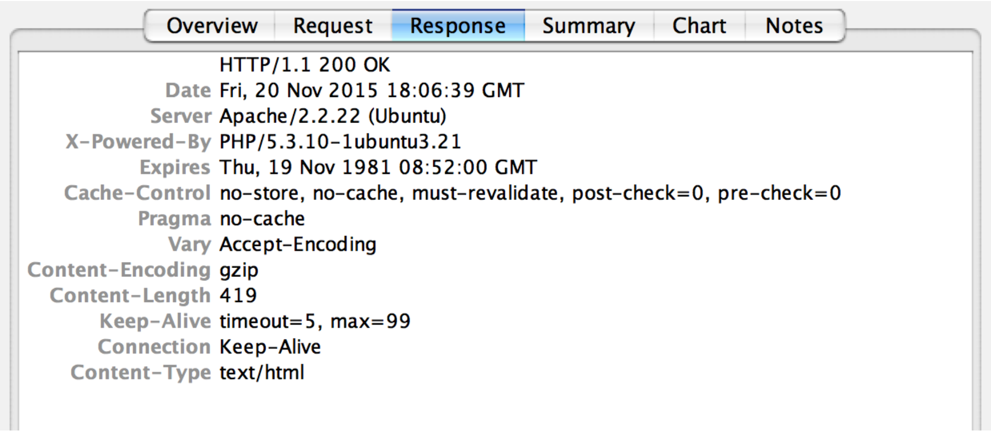
3.1.3 Test HTTP Strict Transport Security (OTG-CONFIG-007)

Secode21

Likelihood: 0
Impact: 0
Risk: 0

	Secode21
Observation	The *HTTP Strict Transport Security* protocol is never used.
Discovery	We used *Charles Web Proxy* to check the HTTP response headers and the *Strict-Transport-Security* 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



```
                 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 *wget* we tried to find a *crossdomain.xml* or *clientaccesspolicy.xml* 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 themself 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 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 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.

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 *Chrome* 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 *Charles Web Proxy* 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: 8
Impact: 8
Risk: 8

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. The credetials are sent unencrypted over HTTP to the server and everyone in the network can read them.
Discovery	We used *Zed Attack Proxy (ZED)* in order to capture packet headers and to inspect them. We saw that the request addressed to the web application is using the HTTP protocol and that the credentials were simple POST parameters
Likelihood	Everyone in the adjacent network or who can read the packages could also get the credentials
Implication	Authentication as user
Recommendations	Use https to encrypt this information
Comparison	The same applies for our web application.

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

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 *admin:admin*
Discovery	We were already given these credentials and additionally we tested the webapp with w3af where this credentials were discovered.
Likelihood	It is very likely that people find this information.
Implication	The attacker gain employee 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 *user:password*.

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

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

Secode21

Likelihood: NA
Impact: NA
Risk: NA

Secode21	
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

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

Secode21

Likelihood: 0
Impact: NA
Risk: NA

Secode21	
Observation	The web app set the cache-control to no-cache an no-store and Pragma to no-cache
Discovery	By reviewing the response header with the chrome developer tools we could analyze the parameters
Likelihood	For every registration process
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	NA
Attack Complexity	NA
Privileges Required	NA
User Interaction	NA
Scope	NA
Confidentiality Impact	NA
Integrity Impact	NA
Availability Impact	NA

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

Secode21

Likelihood: 6
Impact: 5
Risk: 5

Secode21	
Observation	The registration process does not have a restriction for weak passwords and the user can't change the password. 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

3.4 Authorization Testing

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

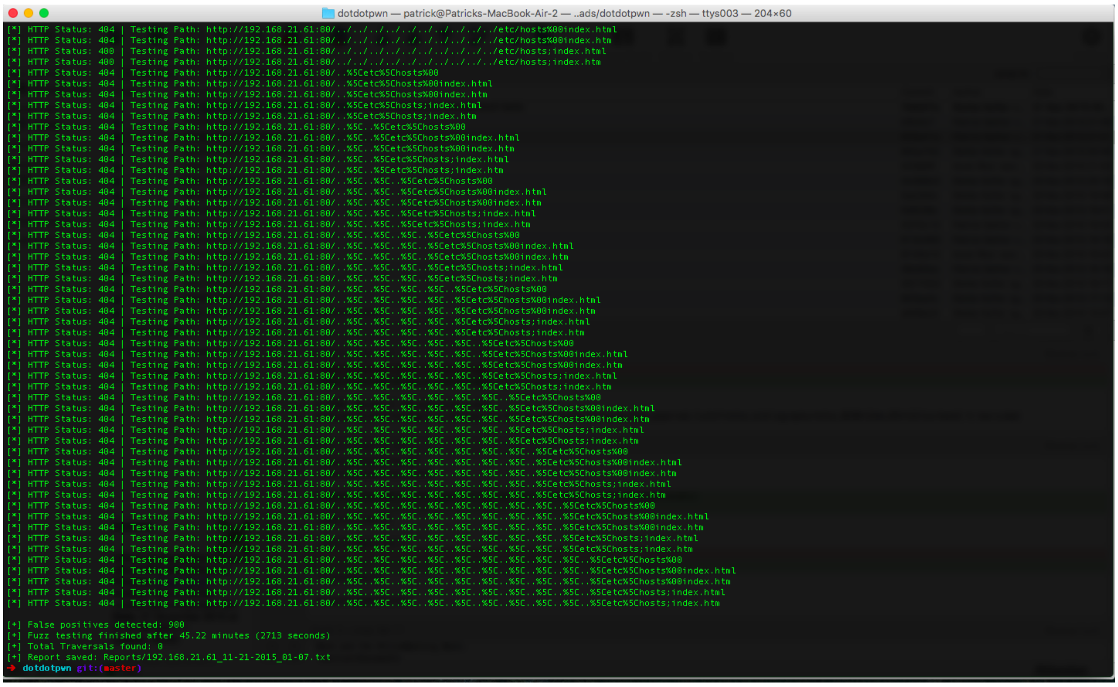
Secode21

Likelihood: 4
Impact: 5
Risk: 5

Secode21	
Observation	We could not find any path traversals with dotdotpwn but w3af found some. The phpinfo.php can be accessed and it contains the paths for several config files. So an attacker directly knows where to search for this files. Also since the attacker can see the phpinfo.php and see the installed software and its versions he can easily search for vulnerabilities for that version

Discovery	We used the dotdotpwn tool to find such traversals with the following command:
	`sudo ./dotdotpwn.pl -O -m http -h 192.168.21.39`
	`-f /etc/hosts -k "localhost" -d 10 -s -E`
	-O is to get the operating system; -d 10 dotdotpwn will search until a deepness of 10; -m is to indicate that the protocol is http and -h for the server ip; -f /etc/hosts searches after the hosts file; -k defines that a file without "localhost" in it is a false positive. Afterwards we tried w3af and it found the phpinfo.php file

Likelihood	This is more an additional information for other attacks but it is a good help for attackers to find vulnerabilities faster
Implication	The attacker knows the position of the config files, how php is configured and what version are used
Recommendations	make the phpinfo page not accessible or delete it
Comparison	Our webapp does not have a phpinfo page.


dotdotpwn — patrick@Patricks-MacBook-Air-2 — ..ads/dotdotpwn — -zsh — ttys003 — 204×60

```
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/../../../../../../../../../../etc/hosts%00index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/../../../../../../../../../../etc/hosts%00index.htm
[*] HTTP Status: 400 | Testing Path: http://192.168.21.61:80/../../../../../../../../../../etc/hosts;index.html
[*] HTTP Status: 400 | Testing Path: http://192.168.21.61:80/../../../../../../../../../../etc/hosts;index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5Cetc%5Chosts%00
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5Cetc%5Chosts%00index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5Cetc%5Chosts%00index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5Cetc%5Chosts;index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5Cetc%5Chosts;index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5Cetc%5Chosts%00
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5Cetc%5Chosts%00index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5Cetc%5Chosts%00index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5Cetc%5Chosts;index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5Cetc%5Chosts;index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5Cetc%5Chosts%00
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5Cetc%5Chosts%00index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5Cetc%5Chosts%00index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5Cetc%5Chosts;index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5Cetc%5Chosts;index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5Cetc%5Chosts%00
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5Cetc%5Chosts%00index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5Cetc%5Chosts%00index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5Cetc%5Chosts;index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5Cetc%5Chosts;index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5Cetc%5Chosts;index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5Cetc%5Chosts;index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts;index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts;index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts;index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts;index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts;index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts;index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts;index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts;index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts%00index.htm
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts;index.html
[*] HTTP Status: 404 | Testing Path: http://192.168.21.61:80/..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5C..%5Cetc%5Chosts;index.htm

[+] False positives detected: 900
[+] Fuzz testing finished after 45.22 minutes (2713 seconds)
[+] Total Traversals found: 0
[+] Report saved: Reports/192.168.21.61_11-21-2015_01-07.txt
➔ dotdotpwn git:(master)
```

Figure 3.1: dotdotpwn screenshot

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 *Burp*.
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	The same results apply for our web application.

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)

Secode21 Likelihood: 10
Impact: 3
Risk: 3

Secode21	
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.
Discovery	We used the Chrome developer tools to analyze the cookies
Likelihood	N/A
Implication	The cookies can be read and used by other web applications that match the path value.
Recommendations	Set the path as thight as possible. For Team21 for example "//secode//"
Comparison	Our web application has exact the same vulnerability.

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

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

Secode21

Likelihood: 8
Impact: 5
Risk: 5

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)

Secode21

Likelihood: 10
Impact: 7
Risk: 7

	Secode21
Observation	The application from Team 21 does not use HTTPS and therefore the session variables are accessible during transport. An attacker could hijack the session simply read the session id by eavesdropping and reusing it
Discovery	We used the Chrome developer tools to analyze the requests
Likelihood	This attack is pretty easy and can also be performed by low skilled people
Implication	The attacker can read the session variables and depending on the information in them
Recommendations	Use HTTP with TLS encryption and avoid GET request including the session id
Comparison	Our web application has exact the same vulnerability

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

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

Secode21

Likelihood: 5
Impact: 6
Risk: 5

Secode21	
Observation	The logout functionallity has been tested an 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 functionallity 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 than the session continues to exist
Implication	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 web application works also as expected but also does not have an automatic logout on the server side

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

3.5.6 Test Session Timeout(OTG-SESS-007)

Secode21

Likelihood: 4
Impact: 6
Risk: 7

Secode21	
Observation	The application has the timeout of the session set to the browsers session lifetime.
Discovery	We tested the functionallity manually and used the Chrome developertools to check the cookies
Likelihood	same as for OTG-SESS-006. Public computers are here the biggest problem
Implication	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	Also our webapp has this vulnerability

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

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

Secode21

Likelihood: 0
Impact: 0
Risk: 0

Secode21	
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

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. `<script type="javascript">alert("XSS");</script>`
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	

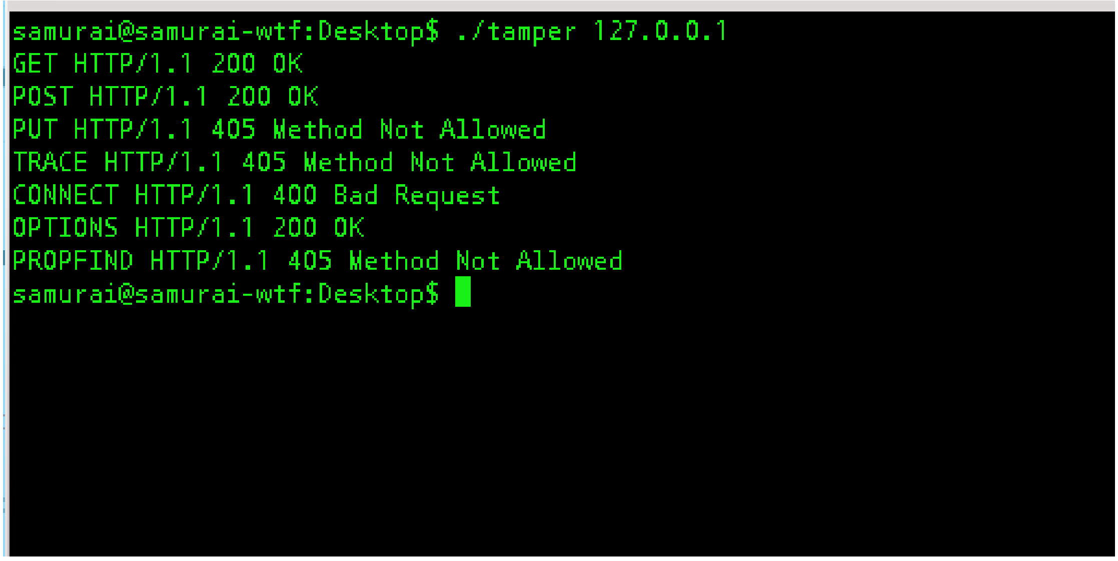
Metric	Value
Access Vector	N
Attack Complexity	M
Privileges Required	N
User Interaction	Y
Scope	U
Confidentiality Impact	M
Integrity Impact	M
Availability Impact	L

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

Secode21

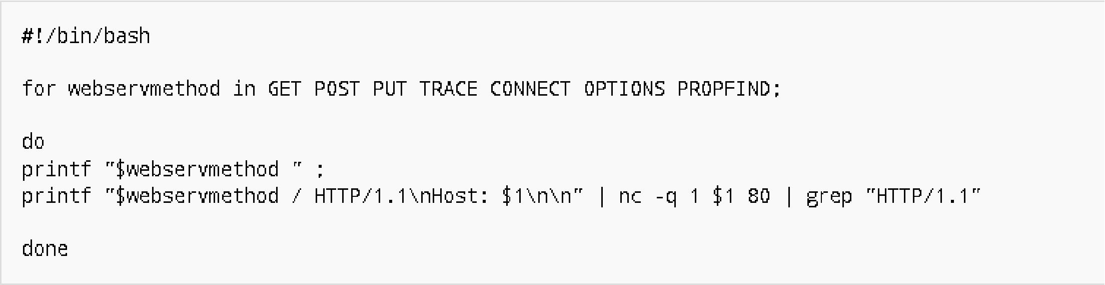
Likelihood: 0
Impact: 0
Risk: 0

Secode21	
Observation	We did not observe any notable behavior.
Discovery	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. Methods that were allowed • HEAD • OPTIONS • GET • POST Methods that were rejected • 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 3.2: 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 3.3: Script used for testing

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

Secode21

Likelihood: 0
Impact: 0
Risk: 0

Secode21	
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 *SQL Inject Me* tool and failed. Also w3af was used and it could not find any vulnerabilities
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	Our web application is also immune to SQL Injections

3.6.5 Testing for XML Injection(OTG-INPVAL-008)

Secode21

Likelihood: 0
Impact: 0
Risk: 0

Secode21	
Observation	We did not find any vulnerability regarding XML Injection
Discovery	We used w3af to find XML injection vulerabilities and it could not find any.
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	Our web application is also immune to XML Injections

3.6.6 Testing for SSI Injection(OTG-INPVAL-009)

Secode21

Likelihood: 0
Impact: 0
Risk: 0

Secode21	
Observation	We did not find any vulnerability regarding SSI Injection
Discovery	We used w3af to find SSI injection possibilities and it could not find any vulnerabilities.
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	Our web application is also immune to XML Injections

3.6.7 Testing for XPATH Injection(OTG-INPVAL-010)

Secode21

Likelihood: 0
Impact: 0
Risk: 0

Secode21	
Observation	We did not find any vulnerability regarding XPATH Injection
Discovery	We used w3af to find XPATH injections and it could not find any vulnerabilities
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	Our web application is also immune to XPATH Injections

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

Secode21

Likelihood: 0
Impact: 0
Risk: 0

Secode21	
Observation	We did not find any vulnerability regarding code injection and local or remote file inclusion in our web app. Team 21 did not implemented that feature
Discovery	Tryed to perform a command execution via the backticks (') and also the semicolon (;) in the filename but our webapp correctly handled the files without injections
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	NA

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

Secode21

Likelihood: 0
Impact: 0
Risk: 0

Secode21	
Observation	Could not find any possibilities too for such a injection
Discovery	Searched manually and used the OWASPTOP10 profile for w3af and did not found a possibility
Likelihood	N/A
Implication	N/A
Recommendations	N/A
Comparison	Neither our app showed such a vulnerability

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

Secode21

Likelihood: 0
Impact: 0
Risk: 0

Secode21	
Observation	We did not find any vulnerability regarding buffer overflow, heap overflow, stack overflow or string formatting
Discovery	We used w3af to locate such vilnerabilities.
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

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

Secode21

Likelihood: 7
Impact: 7
Risk: 5

Secode21	
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.

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 malformated email which results in a not working account or a longer input then 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 *Burp* 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 *Chrome* 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 *w3af* and *ZED Attack Proxy* 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 *w3af* and *ZED Attack Proxy*.
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 *Chrome* 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)

Secode21

Likelihood: 0
Impact: 0
Risk: 0

	Secode21
Observation	We couldn't find any vulnerability here, but we could only check if such a vulnerability exists in the javascript code and not in the php code, because we had no access to the php source code.
Discovery	We used *Chrome* and its developer tools to inspect the html/-javascript code.
Likelihood	N/A
Implication	N/A
Comparison	Our web application uses javascript more often, but user controlled input which specifies the path of a resource was not found.

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

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

Secode21

Likelihood: 0
Impact: 0
Risk: 0

Secode21	
Observation	The inspected web application doesn't make use of XMLHttpRequests and therefor no cross origin resource sharing vulnerabilities exist.
Discovery	We used *Chrome* and its developer tools to inspect the html/-javascript code and *Charles Web Proxy* to make sure that no request is executed.
Likelihood	N/A
Implication	N/A
Comparison	Our web application uses XMLHttpRequests but sends the requests to the same origin and therefor there exist no cross origin resource sharing 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

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

Secode21

Likelihood: 0
Impact: 0
Risk: 0

	Secode21
Observation	*ActionScript* and *Flash* are never used in this web application.
Discovery	We tried to use the web application on a pc with no *Adobe Flash* installed and got no request to install it. Additionally the html code was inspected with *Chrome* and no reference to *Adobe Flash* was found.
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

3.10.9 Testing for Clickjacking (OTG-CLIENT-009)

Secode21

Likelihood: 8
Impact: 9
Risk: 8

	Secode21
Observation	We found a vulnerability in the web application that allows attackers to make clickjacking attacks by bundling the website inside an iframe to give the user the feeling of interacting with the target website but being instead on a malicious web page.
Discovery	The tool *w3af* found out that the web application does not make use of protection techniques to prevent click jacking attacks. The use of *X-Frame-Options* header would help on the server side to prevent against this type of attacks, but is never used by this web application.
Likelihood	It is quite likely that someone would use this kind of exploits on an online banking website, because the people trust these websites. It is not very difficult to use this vulnerability to attack the users.
Implication	The user would think he would interact with the secure online banking system, but in reality he is on a malicious website that can record his interaction and filter out sensitive information.
Comparison	The same results apply for our web application.

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

3.10.10 Testing WebSockets (OTG-CLIENT-010)

Secode21

Likelihood: 0
Impact: 0
Risk: 0

	Secode21
Observation	We inspected the html/javascript source code to find an use of WebSockets but could't find any of them. That means also, that there are no WebSockets vulnerabilities applicable.
Discovery	*Chrome* and its developer tools can show the source code of the web page and can show you if WebSockets are used to communicate with other resources.
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

3.10.11 Test Web Messaging (OTG-CLIENT-011)

Secode21

Likelihood: 0
Impact: 0
Risk: 0

	Secode21
Observation	The web application makes doesn't use the Web Messaging technology (aka Cross Document Messaging) and therefor we couldn't find any vulnerability.
Discovery	We used *Charles Web Proxy* and *Chrome* and its developer tools to see if any other requests are executed from the web application.
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

3.10.12 Test Local Storage (OTG-CLIENT-012)

Secode21

Likelihood: 0
Impact: 0
Risk: 0

	Secode21
Observation	The web application make no use of the local storage functionality of the browsers.
Discovery	We tested the web application with a browser and tested all the functionality and *Chromes* web inspector didn't show any use of the local storage functionality of the browser.
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