DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Secure Coding Phase 2

Team 3: Patrick Sattler, Aurel Roci, Stefan Kohler

Contents

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 Information Gathering Testing

3.1.1 Conduct Search Engine Discovery and Reconnaissance for Information Leakage (OWASP OTG-INFO-001)

Team 21

Observation	Searches performed on several search engines didn't provide useful information with regard to the provided web application, because the web application is not available on the internet but is rather running locally in a virtual machine.
Discovery	Search was not possible.
Likelihood	N/A
Implication	N/A
Recommendations	N/A

Table 3.1: OWASP OTG-INFO-001 results for team 21

Team 3

Observation	Searches performed on several search engines didn't provide useful information with regard to the provided web application, because the web application is not available on the internet but is rather running locally in a virtual machine.
Discovery	Search was not possible.
Likelihood	N/A
Implication	N/A
Recommendations	N/A

Table 3.2: OWASP OTG-INFO-001 results for team 3

3.1.2 Fingerprint Web Server (OWASP OTG-INFO-002)

Team 21

Observation	The fingerprint method works on this server. The server runs an Apache web server version 2.2.22 and runs on the Ubuntu operating system. We used the application ‚Charles Web Proxy' to analyze the http header responses.
Discovery	A lookup for already known vulnerabilities of this Apache version showed us that there exist 10 known vulnerabilities with one more or less critical vulnerability. Source: `http://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-66/version_id-142323/Apache-Http-Server-2.2.22.html`
Likelihood	The likelihood is on upper medium level because the one vulnerability is quite critical and of lower complexity.
Implication	The vulnerabilities of this Apache version belong mostly to the categories of Denial of Service and XSS. That means they can affect the availability of the banking system and partially also the integrity of the shown data, which should never happen in an online banking system.
Recommendations	Update the web server to the newest version of Apache and hide the version information sharing in the apache settings.

Table 3.3: OWASP OTG-INFO-002 results for team 21

Team 3

Observation	The fingerprint method works on this server. The server runs an Apache web server version 2.2.22 and runs on the Ubuntu operating system. We used the application ‚Charles Web Proxy' to analyze the http header responses.
Discovery	A lookup for already known vulnerabilities of this Apache version showed us that there exist 10 known vulnerabilities with one more or less critical vulnerability. Source: `http://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-66/version_id-142323/Apache-Http-Server-2.2.22.html`
Likelihood	The likelihood is on upper medium level because the one vulnerability is quite critical and of lower complexity.
Implication	The vulnerabilities of this Apache version belong mostly to the categories of Denial of Service and XSS. That means they can affect the availability of the banking system and partially also the integrity of the shown data, which should never happen in an online banking system.
Recommendations	Update the web server to the newest version of Apache and hide the version information sharing in the apache settings.

Table 3.4: OWASP OTG-INFO-002 results for team 3

3.1.3 Review Webserver Metafiles for Information Leakage (OWASP OTG-INFO-003)

Team 21

Observation	The web application doesn't provide a robots.txt file and no meta tags in the source code. With no robots.txt file a search engine would search for everything available on a website and that would lead to information leakage if the website would be online.
Discovery	We have searched for a robots.txt file with a browser and checked the source code for meta tags referring to a robots.txt file or including other sensitive information. We used the ‚Chrome Developer Tools' for viewing the html source code of the web pages.
Likelihood	N/A (because the website is not online)
Implication	N/A
Recommendations	If the website would go online they should add a robots.txt file that defines the paths that shouldn't be crawled by the search engines.

Table 3.5: OWASP OTG-INFO-003 results for team 21

Team 3

Observation	The web application doesn't provide a robots.txt file and no meta tags in the source code. With no robots.txt file a search engine would search for everything available on a website and that would lead to information leakage if the website would be online.
Discovery	We have searched for a robots.txt file with a browser and checked the source code for meta tags referring to a robots.txt file or including other sensitive information. We used the ‚Chrome Developer Tools' for viewing the html source code of the web pages.
Likelihood	N/A (because the website is not online)
Implication	N/A
Recommendations	If the website would go online they should add a robots.txt file that defines the paths that shouldn't be crawled by the search engines.

Table 3.6: OWASP OTG-INFO-003 results for team 3

3.1.4 Enumerate Applications on Webserver (OWASP OTG-INFO-004)

Team 21

Observation	With the use of the ‚nmap security scanner' we found out which ports are open on the ip address of the web server. We found several open ports and the services with their version running on the system.
Discovery	We found out that there are two web server ports open, one the standard port 80 and one the port 443 for https connections. A test using a browser showed us, that the https server is not running. Lastly we found the SSH port open and can even see which version of OpenSSH is installed.
Likelihood	Medium
Implication	If there is a bug in one of the applications running on the web server we could use them to hack the system.
Recommendations	Keep the software on the system up to date and check for possibilities to hide some information (e.g. the versions of the applications).

Table 3.7: OWASP OTG-INFO-004 results for team 21

Team 3

Observation	With the use of the ‚nmap security scanner' we found out which ports are open on the ip address of the web server. We found several open ports and the services with their version running on the system.
Discovery	We found out that there are two web server ports open, one the standard port 80 and one the port 443 for https connections. A test using a browser showed us, that the https server is not running. Lastly we found the SSH port open and can even see which version of OpenSSH is installed.
Likelihood	Medium
Implication	If there is a bug in one of the applications running on the web server we could use them to hack the system.
Recommendations	Keep the software on the system up to date and check for possibilities to hide some information (e.g. the versions of the applications).

Table 3.8: OWASP OTG-INFO-004 results for team 3

3.1.5 Review webpage comments and metadata for information leakage (OTG-INFO-005)

Team 21

Observation	We found one HTML comment and no metadata within the web application html source code.
Discovery	We found out that a 'remember me' function was planned but afterwards commented out. Maybe the functionality was removed because of an vulnerability and we could still use that one by adding that button back in. We used the ‚Chrome Developer Tools' to take a look at the HTML source code of the pages.
Likelihood	N/A
Implication	N/A
Recommendations	N/A

Table 3.9: OWASP OTG-INFO-005 results for team 21

Team 3

Observation	We didn't find any comments or metadata.
Discovery	We used the ‚Chrome Developer Tools' to take a look at the HTML source code of the pages.
Likelihood	N/A
Implication	N/A
Recommendations	N/A

Table 3.10: OWASP OTG-INFO-005 results for team 3

3.1.6 Identify application entry points (OTG-INFO-006)

Team 21

Observation	
Discovery	
Likelihood	N/A
Implication	N/A
Recommendations	N/A

Table 3.11: OWASP OTG-INFO-006 results for team 21

Team 3

Observation	
Discovery	
Likelihood	N/A
Implication	N/A
Recommendations	N/A

Table 3.12: OWASP OTG-INFO-006 results for team 3

3.1.7 Map execution paths through application (OTG-INFO-007)

Team 21

Observation	
Discovery	
Likelihood	N/A
Implication	N/A
Recommendations	N/A

Table 3.13: OWASP OTG-INFO-007 results for team 21

Team 3

Observation	
Discovery	
Likelihood	N/A
Implication	N/A
Recommendations	N/A

Table 3.14: OWASP OTG-INFO-007 results for team 3

3.1.8 Fingerprint Web Application Framework (OTG-INFO-008)

Team 21

Observation	
Discovery	
Likelihood	N/A
Implication	N/A
Recommendations	N/A

Table 3.15: OWASP OTG-INFO-008 results for team 21

Team 3

Observation	
Discovery	
Likelihood	N/A
Implication	N/A
Recommendations	N/A

Table 3.16: OWASP OTG-INFO-008 results for team 3

3.1.9 Fingerprint Web Application (OTG-INFO-009)

Team 21

Observation	
Discovery	
Likelihood	N/A
Implication	N/A
Recommendations	N/A

Table 3.17: OWASP OTG-INFO-009 results for team 21

Team 3

Observation	
Discovery	
Likelihood	N/A
Implication	N/A
Recommendations	N/A

Table 3.18: OWASP OTG-INFO-009 results for team 3

3.1.10 Map Application Architecture (OTG-INFO-010)

Team 21

Observation	
Discovery	
Likelihood	N/A
Implication	N/A
Recommendations	N/A

Table 3.19: OWASP OTG-INFO-010 results for team 21

Team 3

Observation	
Discovery	
Likelihood	N/A
Implication	N/A
Recommendations	N/A

Table 3.20: OWASP OTG-INFO-010 results for team 3

3.2 Configuration and Deploy Management Testing

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

3.2.2 Test HTTP Methods(OTG-CONFIG-006)

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

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

3.3 Identity Management Testing

3.3.1 Test Role Definitions(OTG-IDENT-001)

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

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

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

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

3.4 Authentication Testing

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

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

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

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

3.4.5 Test remember password functionality(OTG-AUTHN-005)

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

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

3.4.8 Testing for Weak security question/answer(OTG-AUTHN-008)

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

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

3.5 Authorization Testing

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

3.5.2 Testing for bypassing authorization schema(OTG-AUTHZ-002)

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

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

3.6 Session Management Testing

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

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

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

Secode21	Likelihood: 0 Impact: 0 Risk:0

Secode21	
Observation	N/A
Discovery	N/A
Likelihood	N/A
Implication	N/A
Recommendations	N/A

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

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

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

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

3.6.7 Test Session Timeout(OTG-SESS-007)

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

3.7 Data Validation Testing

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

Secode21	Likelihood: 0 Impact: 0 Risk:0

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

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

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

Secode21	Likelihood: 7 Impact: 10 Risk: 10

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. But the BeEF framework allows to quickly test several attacks, 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 landingpage, 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.

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

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

Secode21	Likelihood: 0 Impact: 0 Risk: 0

Secode21	
Observation	We did not observe any notable behaviour.
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 • PUT Methods that were rejected • TRACE • CONNECT
Likelihood	N/A
Implication	N/A
Recommendations	N/A

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

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

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

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

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

3.7.8 Testing for XPath Injection(OTG-INPVAL-010)

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

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

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

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

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

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

3.8 Error Handling

Secode21

Secode21 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 Online Banking

3.9 Cryptography

3.10 Business Logic Testing

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

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

3.10.3 Test Integrity Checks(OTG-BUSLOGIC-003)

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

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

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

3.10.7 Test Defenses Against Application Mis-use(OTG-BUSLOGIC-007)

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

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

3.11 Client Side Testing

Glossary

computer is a machine that....

Acronyms

TUM Technische Universität München.