DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Secure Coding - Phase 2

Blackbox Testing Report

Team 12
Alexander Lill
Lorenzo Donini
Florian Mauracher
Mahmoud Naser

Executive Summary

1 DogeBank- Team 27

During our intensive testing of the DogeBank application, we found many severe vulnerabilities, as well as bugs, that could easily be exploited to compromise the entire application. Under no circumstances should this web application be used productively!

For starters, directory listing is enabled and allows to browse the structure of the web application, providing knowledge about the application flow and some functionalities. Several checks are not performed, therefore it is possible to access some pages as an unauthenticated user and perform operations that should only be allowed to employees (e.g. approving registrations/transactions). This also holds for users without sufficient role privileges. Even without all these vulnerabilities, session hijacking would still be a viable option to obtain employee privileges. SQL injection and XSS is also possible on almost all pages, giving an attacker plenty of possibilities to cause damage to both customers and employees.

Furthermore any registered user may potentially upload arbitrary malicious code to the server by exploiting the batch transaction functionality. Since no checks are performed whatsoever on the extension of the uploaded file, it is perfectly feasible to upload scripts that can later be executed from just anyone. Although this issue is easy to fix, it may give an attacker full control over the web application as well as over part of the system it runs on. All database and email passwords are also accessible this way. Additionally, the file upload functionality is vulnerable to direct code injection, which will result in allowing an attacker to execute arbitrary code.

The business logic also seems to be somewhat broken, as a customer can increase his/her balance arbitrarily, by exploiting multiple flaws in the transaction functionalities. Moreover TANs are not entirely random and will be generated only once per user, allowing endless transactions after all the codes have been used up.

2 Goliath National Bank- Team 12

We found several vulnerabilities and bugs, most of them having minor impact on the web application as a whole, but still allowing an attacker to gain access to other user accounts (both clients and employees).

Even though the folder listing is disabled on the server, an astute attacker may either brute-force or figure out the names of some PHP files, by analyzing the logical names inside the client code. Since not all of these files seem to perform checks (and automatically redirect a user to a safe path), some Among the major issues found, SQL injection and XSS is possible from the registration page.

3 Comparison

In summary

Contents

1 Timetracking

User	Task	Time
Alexander Lill	Task1	X h
	Sum	XX h
Lorenzo Donini	Task1	X h
	Sum	XX h
Florian Mauracher	Set up pentesting environment	2 h
Florian Mauracher	Testing basic functionality of dogebank	2 h
Florian Mauracher	Configuration and Deploy Management Testing	1 h
Florian Mauracher	Authorization Testing	2 h
Florian Mauracher	Data Validation Testing	2 h
Florian Mauracher	Configure basic LateX template	2 h
Florian Mauracher	Create LaTeX template for OWASP checklist	2 h
	Sum	XX h
Mahmoud Naser	Task1	X h
	Sum	XX h

2 Vulnerabilities Overview

In this chapter, the major security flaws of both the DogeBank and the Goliath National Bank applications will be briefly summarized and compared.
After testing the two web applications thoroughly, we found the following vulnerabilities to be the most serious.

2.1 DogeBank

2.1.1 Bypassing authorizations

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

Any unauthenticated user may approve registrations or transactions using the correct GET request. It is also possible to directly register an employee without even having to login. Considering this security issue, the whole authentication process proves to be useless.

2.1.2 Privilege escalation

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

A logged in customer is able to access employee pages without having the proper privileges, allowing therefore actions which shouldn't be possible. This is also possible the other way around, although this could be considered as a bug, rather than a vulnerability.

2.1.3 Stored XSS in all forms

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

Input values in forms are not validated whatsoever, allowing to store custom scripts inside the database when filling out forms. These scripts will automatically be executed by employees who view the details of the client. This is also valid for stored CSS and HTML injection.

2.1.4 SQL injection in all forms

- Likelihood: *medium*
- Implication: *high*
- Risk: *high*
- Reference: *OWASP OTG-INPVAL-005*

The same concept explained in the XSS vulnerability also applies for SQL injection: since the input values in forms are not validated, it is possible to inject SQL statements. Even though multiple SQL queries are not supported, tricking the server into authenticating a user without valid credentials, or using invalid TANs for that matter, is still easy.

2.1.5 No file extension check during upload

- Likelihood: *medium*
- Implication: *high*
- Risk: *high*
- Reference: *OWASP OTG-BUSLOGIC-008*

During the upload of batch files for multiple transactions, the file extension is not verified, therefore it is possible to upload any potential file or to use Unix commands as the name of the file.

2.1.6 Test Upload of Malicious Files

- Likelihood: *medium*
- Implication: *high*
- Risk: *high*
- Reference: *OWASP OTG-BUSLOGIC-009*

The weak file upload policy leads to another big issue, since the uploaded file may contain malicious code. Exploiting this vulnerability allows to gain complete control of the web application.

2.2 Goliath National Bank

2.3 Comparison

3 Detailed Report

3.1 Tools description

3.2 Configuration and Deploy Management Testing

3.2.1 SAMPLE (OTG-CONFIG-XXX)

	DogeBank
Observation	observation text
Discovery	discovery text
Likelihood	likelihood text
Implication	implication text
CVSS	**AV:** N **AC:** L **PR:** N **UI:** N **S:** U **C:** L **I:** L **A:** L 7.3

	Goliath National Bank
Observation	observation text
Discovery	discovery text
Likelihood	likelihood text
Implication	implication text
CVSS	**AV:** N **AC:** L **PR:** N **UI:** N **S:** C **C:** L **I:** L **A:** L 8.3

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

	DogeBank
Observation	The batch transaction functionality at `http:/IPADDRESS/tran.php` does not seem to check the extension of the uploaded files. Also, while exploring the folder structure of the server, some leftover files were found, as well as hidden folders.
Discovery	If the user tries to upload any file other than a txt file, the server does not provide any error messages. This particular vulnerability will be discussed later in 3.X.X. Additionally, leftover files (e.g. `http:/IPADDRESS/employee_registration.php~`) were not deleted from the server, as well as other files containing sensitive information. More specifically, this is the case of the .git folder.
Likelihood	In order to access the hidden .git folder, however, an attacker must be skilled and know where to search for sensitive information.
Implication	Having access to the data contained inside the hidden .git folder allows to potentially get access to the whole source code of the web application.
CVSS	**AV:** N **AC:** L **PR:** N **UI:** N **S:** C **C:** L **I:** L **A:** L 8.3

	Goliath National Bank
Observation	?
Discovery	?
Likelihood	?
Implication	?
CVSS	**AV:** N **AC:** L **PR:** N **UI:** N **S:** C **C:** L **I:** L **A:** L 8.3

3.2.3 Test HTTP Methods (OTG-CONFIG-006)

	DogeBank
Observation	observation text
Discovery	discovery text
Likelihood	likelihood text
Implication	implication text
CVSS	**AV:** N **AC:** L **PR:** N **UI:** N **S:** C **C:** L **I:** L **A:** L 8.3

	Goliath National Bank
Observation	observation text
Discovery	discovery text
Likelihood	likelihood text
Implication	implication text
CVSS	**AV:** N **AC:** L **PR:** N **UI:** N **S:** C **C:** L **I:** L **A:** L 8.3

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

	DogeBank
Observation	The application is only accessible over HTTP.
Discovery	No HTTPS is enforced, therefore all data sent between the server and client is not encrypted.
Likelihood	An attacker could perform a man in the middle attack.
Implication	Sniffing the network traffic, all data exchanged between the server and the client can be read as clear text. No confidentiality at all is supported on this end.
CVSS	**AV:** N **AC:** L **PR:** N **UI:** N **S:** U **C:** L **I:** L **A:** L 7.3

	Goliath National Bank
Observation	The application is only accessible over HTTP.
Discovery	No HTTPS is enforced, therefore all data sent between the server and client is not encrypted.
Likelihood	An attacker could perform a man in the middle attack.
Implication	Sniffing the network traffic, all data exchanged between the server and the client can be read as clear text. No confidentiality at all is supported on this end.
CVSS	**AV:** N **AC:** L **PR:** N **UI:** N **S:** U **C:** L **I:** L **A:** L 7.3

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

	DogeBank
Observation	The web application doesn't support additional technologies like Flash, Silverlight or Java.
Discovery	No cross-domain policy files were found.
Likelihood	N/A
Implication	N/A
CVSS	N/A

	Goliath National Bank
Observation	The web application doesn't support additional technologies like Flash, Silverlight or Java.
Discovery	No cross-domain policy files were found.
Likelihood	N/A
Implication	N/A
CVSS	N/A

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)

	DogeBank
Observation	When accessing the application, a randomly generated PHPSESSID session cookie is set. The cookie doesn't have an expiration date nor is it tagged as secure. Apparently the session cookie is already set before logging into the application. This cookie is simply replaced by a new one once the user logs out of the application. No other cookies are set. Also if the cookie is tampered with, the server automatically generates a new cookie, containing a new session ID.
Discovery	The PHPSESSID cookie has been discovered while intercepting HTTP requests/responses using Burp. The same cookie details were later on confirmed using the Cookies plugin for browser.
Likelihood	N/A
Implication	Since the only used cookie only contains the session ID, even though it is easy to change the value of the cookie, no other session values are exposed to the user. It is still possible to hijack another session by changing the entire value of the cookie with the one associated to another user (see Session 004).
CVSS	N/A

	Goliath National Bank
Observation	The same observations made for the DogeBank application apply.
Discovery	The PHPSESSID cookie was analyzed using the Cookies plugin for browser.
Likelihood	N/A
Implication	The same implications mentioned for the DogeBank application apply.
CVSS	N/A

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

	DogeBank
Observation	observation text
Discovery	discovery text
Likelihood	likelihood text
Implication	implication text
CVSS	**AV:** N **AC:** L **PR:** N **UI:** N **S:** U **C:** L **I:** L **A:** L 7.3

	Goliath National Bank
Observation	observation text
Discovery	discovery text
Likelihood	likelihood text
Implication	implication text
CVSS	**AV:** N **AC:** L **PR:** N **UI:** N **S:** C **C:** L **I:** L **A:** L 8.3

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

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)

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

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

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 LDAP Injection (OTG-INPVAL-006)

3.7.7 Testing for ORM Injection (OTG-INPVAL-007)

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

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

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

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

3.7.12 Testing for Code Injection (OTG-INPVAL-012)

Testing for Local File Inclusion
Testing for Remote File Inclusion

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

3.7.14 Testing for Buffer overflow (OTG-INPVAL-014)

Testing for Heap overflow
Testing for Stack overflow
Testing for Format string

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

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

3.8 Error Handling

3.8.1 Analysis of Error Codes (OTG-ERR-001)

3.8.2 Analysis of Stack Traces (OTG-ERR-002)

3.9 Cryptography

3.9.1 Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection (OTG-CRYPST-001)

3.9.2 Testing for Padding Oracle (OTG-CRYPST-002)

3.9.3 Testing for Sensitive information sent via unencrypted channels (OTG-CRYPST-003)

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

This section was prioritized as low, therefore the client side was not tested in depth. Furthermore, as stated in the OWASP testing guide, black box testing of the client side is usually not performed, since access to the source code is always available, as it needs to be sent to the client to be executed. Nevertheless, some potential vulnerabilities were found and briefly analyzed.

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

3.11.2 Testing for JavaScript Execution (OTG-CLIENT-002)

3.11.3 Testing for HTML Injection (OTG-CLIENT-003)

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

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

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

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

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

3.11.9 Testing for Clickjacking (OTG-CLIENT-009)

3.11.10 Testing WebSockets (OTG-CLIENT-010)

3.11.11 Test Web Messaging (OTG-CLIENT-011)

3.11.12 Test Local Storage (OTG-CLIENT-012)