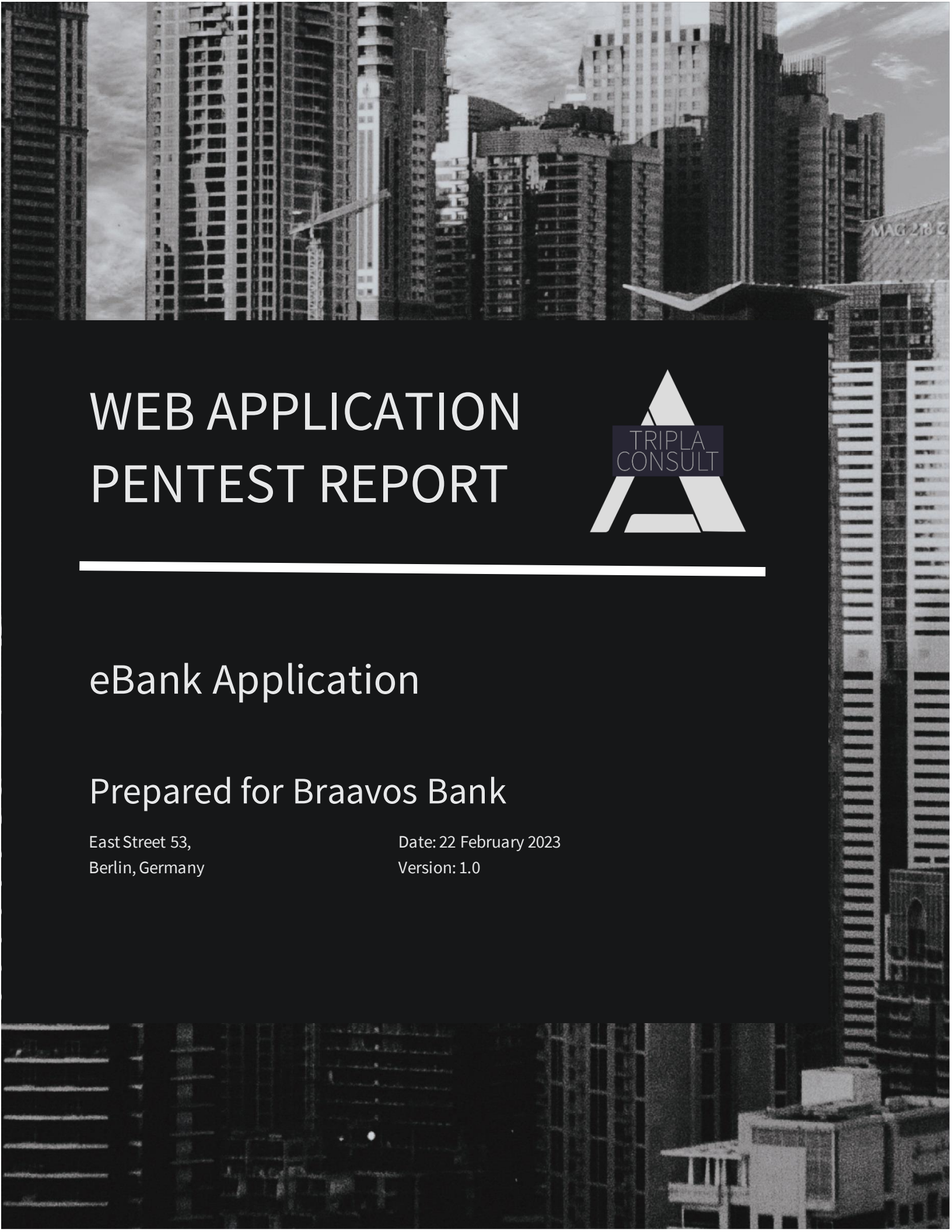


WEB APPLICATION PENTEST REPORT

TRIPLA CONSULT

eBank Application

Prepared for Braavos Bank

East Street 53,
Berlin, Germany

Date: 22 February 2023
Version: 1.0

Table of Contents

1. Confidentiality Statement

This document is the exclusive property of Braavos Bank Corp and Tripla Consult. This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires consent of both Braavos Bank Corp and Tripla Consult.

Braavos Bank Corp may share this document with auditors under non-disclosure agreements to demonstrate penetration test requirement compliance.

2. Disclaimer

A penetration test is considered a snapshot in time. The findings and recommendations reflect the information gathered during the assessment and not any changes or modifications made outside of that period.

Time-limited engagements do not allow for a full evaluation of all security controls. Tripla Consult prioritized the assessment to identify the weakest security controls an attacker would exploit. Tripla Consult recommends conducting similar assessments on an annual basis by internal or third-party assessors to ensure the continued success of the controls.

3. Contact Information

Name	Title	Contact Information
Braavos Bank Corp		
John Adams	Project Manager	Email: adams@braavosbank.corp
Tripla Consult		
Andrei Agape	Lead Consultant	Email: contact@tripla.dk

4. Executive Summary

Tripla Consult evaluated Braavos' eBank application through penetration testing from January 22, 2023 to February 5, 2023. The following sections provide a high-level overview of vulnerabilities discovered, successful and unsuccessful attempts, and strengths and weaknesses.

It was found that the Braavos' eBank application is lacking input validation on critical functions. Malicious actors can abuse these vulnerabilities to read emails, plain text passwords, credit card numbers and source code. The findings have high impact on confidentiality, integrity and availability (CIA) which could directly affect Braavos' brand and its customers.

Additionally, the verbose error messages and test default credentials suggest improper separation between the development and production environments.

Scoping and Time Limitations

Scoping during the engagement did not permit denial of service or social engineering across the testing components. Time limitations were in place for testing. The web application penetration testing was permitted for ten business days.

Testing Summary

The network assessment evaluated Braavos Bank's eBank security posture. The following list presents a high-level description of the tests performed against the target application:

- Broken Access Control
- Cryptographic Failures
- Injection Attacks
- Insecure Design
- Security Misconfiguration
- Vulnerable and Outdated Components
- Identification and Authentication Failures
- Software and Data Integrity Failures
- Security Logging and Monitoring Failures
- Server-Side Request Forgery

Key Strengths and Weaknesses

The following key strengths were identified during the assessment:

- The Web Application Firewall (WAF) blocked some of the payloads
- The code injection vulnerability was not exploitable

The following key weaknesses were identified during the assessment:

- Lack of input validation
- Lack of security best practices
- Cleartext storage of sensitive information

5. Scope

The following IP and URLs were defined to be in scope during the assessment:

IP/URL	Description
http://braavosbank.corp/	eBank web application - production
http://braavosbank.dev/	eBank web application - staging

Scope Exclusions

Per client request, Tripla Consult did not perform any of the following attacks during testing:

- Phishing
- Social Engineering
- Denial of Service

All other attacks not specified above were permitted by Braavos Bank.

Client Allowances

Braavos Bank provided Tripla Consult the following allowances:

- Test pplication user accounts
- API Documentation
- Source code

6. Vulnerability Summary

A detailed description of each finding follows below. For information about how severity rating has been estimated, see the Appendix.

Finding	Severity
SQL Injection	High
Local File Inclusion	High
Account Takeover	High
Test Account in Production	Medium
Verbose Error Messages	Low

6.1 SQL Injection [High]

Description

A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. The application is affected by SQL injection in multiple input fields where user provided information is used as part of a SQL query.

URL: http://testphp.vulnweb.com/search.php?test=<USER_INPUT>

Impact

Malicious actors can abuse this vulnerability to alter the original SQL query and change the application behavior. It is possible to read information about the databases, tables, columns and dump all the data which could lead to sensitive data exposure and GDPR issues. The table "users" in the "acuart" database is storing credit card information, password, full name, address, etc.

Replication

1. Identify input fields which might be used by the application as part of a SQL query (i.e: the search function http://testphp.vulnweb.com/search.php?test=123)

```
$dummyResults = mysql_query("SELECT * FROM guestbook WHERE
sender='".$_GET["test"]."';");
while($row = mysql_fetch_array($dummyResults)){
    echo "<!--".$row["mesaj"]."-->";
}
```

2. Check if the field is vulnerable using sqlmap

```
python3 sqlmap.py -u http://testphp.vulnweb.com/search.php?test=123
```

3. Once **sqlmap** confirms the vulnerability, attempt to extract information about the backend database (database names: "acuart" and "information_schema")

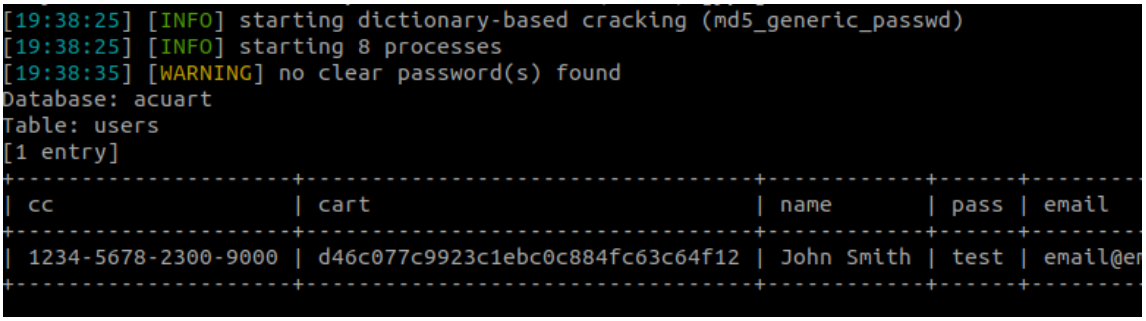
```
andrei@SRTVM:~/Tools/sqlmap-dev$ python3 sqlmap.py -u
http://testphp.vulnweb.com/search.php?test=123 --dbs
[19:36:30] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema
```

4. Extract information about the tables of the non-default databases

```
andrei@SRTVM:~/Tools/sqlmap-dev$ python3 sqlmap.py -u
http://testphp.vulnweb.com/search.php?test=123  --tables -D acuart

[19:37:07] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+-----------+
| artists   |
| carts     |
| categ     |
| featured  |
| guestbook |
| pictures  |
| products  |
| users     |
+-----------+
```

5. Read information stored in the tables that might contain sensitive information



Remediation

- Use prepared statements with variable binding (parameterized queries) such as PHP Data Objects - PDO
- Apply input validation for user supplied input
- Consider using stored procedures
- Review and consult the OWASP SQL injection prevention cheat sheet

References

- PHP: PDO - Manual
- SQL Injection Prevention - OWASP Cheat Sheet Series

6.2 Local File Inclusion [High]

Description

The application is affected by Local File Inclusion on the **showimage.php** page where user provided information is used as part of loading images from disk. Malicious actors can abuse this vulnerability to alter the original function and read arbitrary files from the local server, but limited to the following paths:

```
/hj/:/tmp/:/proc/
```

URL: http://testphp.vulnweb.com/showimage.php?file=<USER_INPUT>

Impact

Using the **/proc/self/cwd/<PATH>** it was possible to read files from application's current directory. This allowed for access to the source code of the application and discover:

- plain text credentials to connect to the database
- possible remote code execution (RCE) vulnerability
- detect and confirm the existence of SQL injection

Replication

1. To validate the LFI vulnerability: use an intermediary proxy (BurpSuite), access the following path http://testphp.vulnweb.com/showimage.php?file=/proc/self/cwd/showimage.php and inspect the server response:

```
if( isset($_GET["file"]) && !isset($_GET["size"]) ){
    header("Content-Type: image/jpeg");
    $name = $_GET["file"];
    if (filter_var($name, FILTER_VALIDATE_URL)) {
          exit();
    }
    $fp = fopen($name, 'rb');
    header("Content-Type: image/jpeg");
    fpassthru($fp);
    exit;
}
```

2. To read the DB connection details: use an intermediary proxy (BurpSuite), access the following path http://testphp.vulnweb.com/showimage.php?file=/proc/self/cwd/database_connect.php and inspect the server response:

```
<?PHP
    $connection = mysql_connect('127.0.0.1', 'acuart', 'trustno1') or die('Website is
out of order. Please visit back later. Thank you for understanding.');
    mysql_select_db('acuart', $connection) or die('Website is out of order. Please
visit back later. Thank you for understanding.');
?>
```

3. To validate the potential RCE that uses the **eval()** function: use an intermediary proxy (BurpSuite), access the following path http://testphp.vulnweb.com/showimage.php?file=/proc/self/cwd/comment.php and inspect the server response:

```
echo "<p class='story'>".$_POST['name']."", thank you for your comment.</p>";
echo "<p class='story'><i>";
if ($_POST["phpaction"] == "printf(md5(acunetix_wvs_security_test));exit;//")
    eval($_POST["phpaction"]);
echo "</p></i>";
```

Remediation

- Ensure that user supplied input is not directly used as part of loading files
- Maintain a white list of files that may be included by the page, and then use an identifier (for example the index number) to access to the selected file
- Limit the loading paths only to the images folder, and ensure that path traversal is not possible
- Review and consult the OWASP LFI cheat sheet

References

- WSTG - v4.2 | OWASP Foundation
- Input Validation - OWASP Cheat Sheet Series

6.3 Account Takeover [High]

Description

The application is affected by the following vulnerabilities that can be chained together to take over the accounts of other users:

- persistent cross-site scripting (XSS) in the guestbook section
- lack of content security policy (CSP)
- lack of session cookies security flags (HttpOnly/SecureOnly)

URL: http://testphp.vulnweb.com/guestbook.php

Impact

A malicious actor with anonymous access to the guestbook section can inject an XSS payload that reads the session cookie of the authenticated users, re-use the cookie value to impersonate the victim and gain access to his/her account. Additionally, the application does not enforce 2FA and access to the victim's account grants details about the credit card information.

Replication

1. Navigate to http://testphp.vulnweb.com/guestbook.php
2. Post the following comment:

```
<script>alert(document.cookie)</script>
```

3. Refresh the page
4. Note that the application displays a pop-up message that contains the current user's session cookie
5. Modify the script to send the value to our own server
6. Wait for victim to visit the guestbook section and collect their cookie information
7. Using the browser's developer tools, load these cookie
8. Refresh the page and note that we are logged in to the victim's account

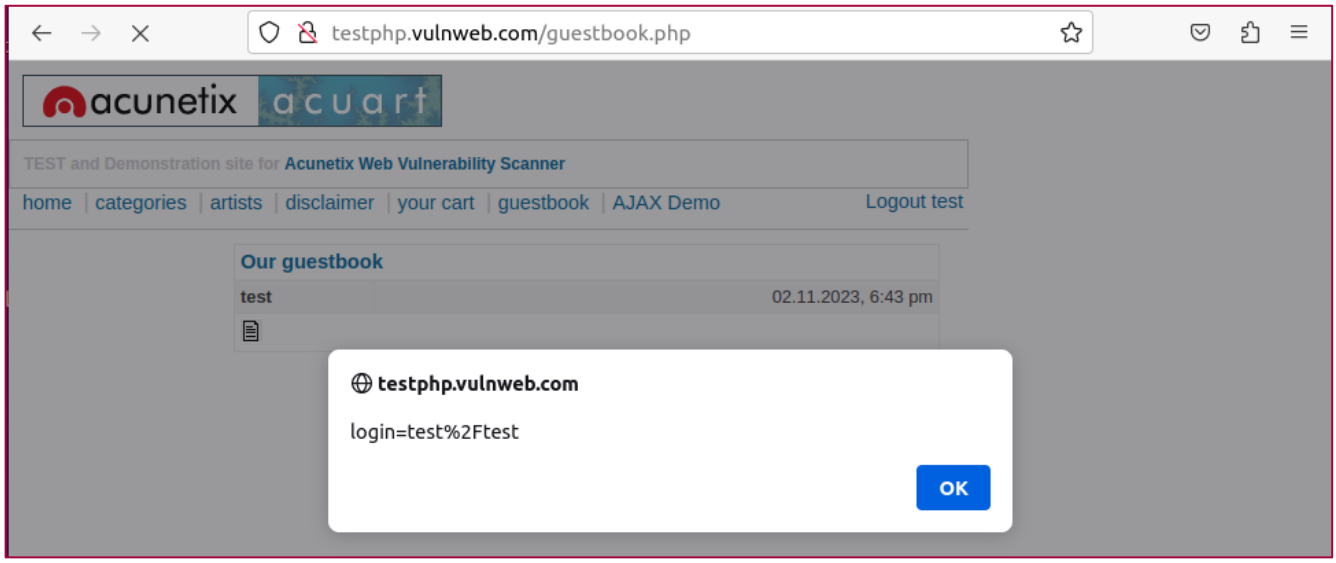


Figure 1: XSS payload reading user's session cookie

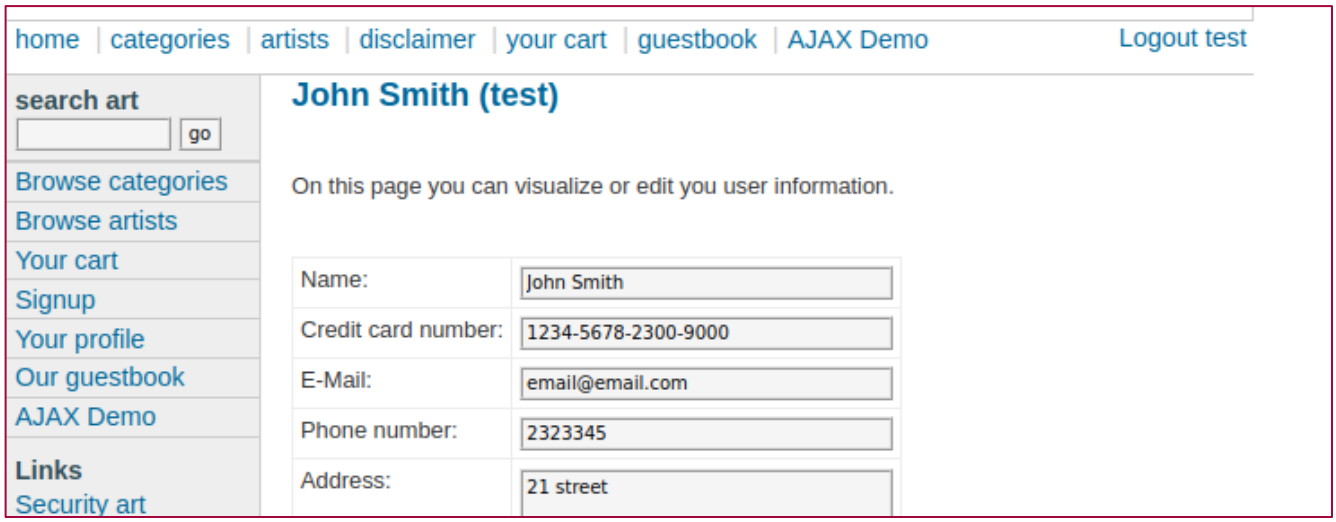


Figure 2: Access to victim's account reveals credit card information

Remediation

- Apply input validation and output encoding
- Implement a strong content security policy (CSP)
- Set the **HttpOnly** and **SecureOnly** to the session cookie
- Enforce two factor authentication (2FA)

References

- Cross Site Scripting Prevention - OWASP Cheat Sheet Series
- Content Security Policy - OWASP Cheat Sheet Series
- Multifactor Authentication - OWASP Cheat Sheet Series
- Secure Cookie Attribute | OWASP Foundation

6.4 Test Account in Production [Medium]

Description

The application has test accounts with default passwords enabled in the production. The account can be accessed by signing up to on the following page using the **test:test** combination:

URL: http://testphp.vulnweb.com/login.php

Impact

A malicious actor gaining access to the test account can, for example: perform authenticated enumeration, get access to the application functionality, read potentially sensitive information, launch attacks against the other members, etc.

Replication

1. Navigate to http://testphp.vulnweb.com/login.php
2. Use the test: test combination for username and password
3. Note that the account is enabled and the login is successful

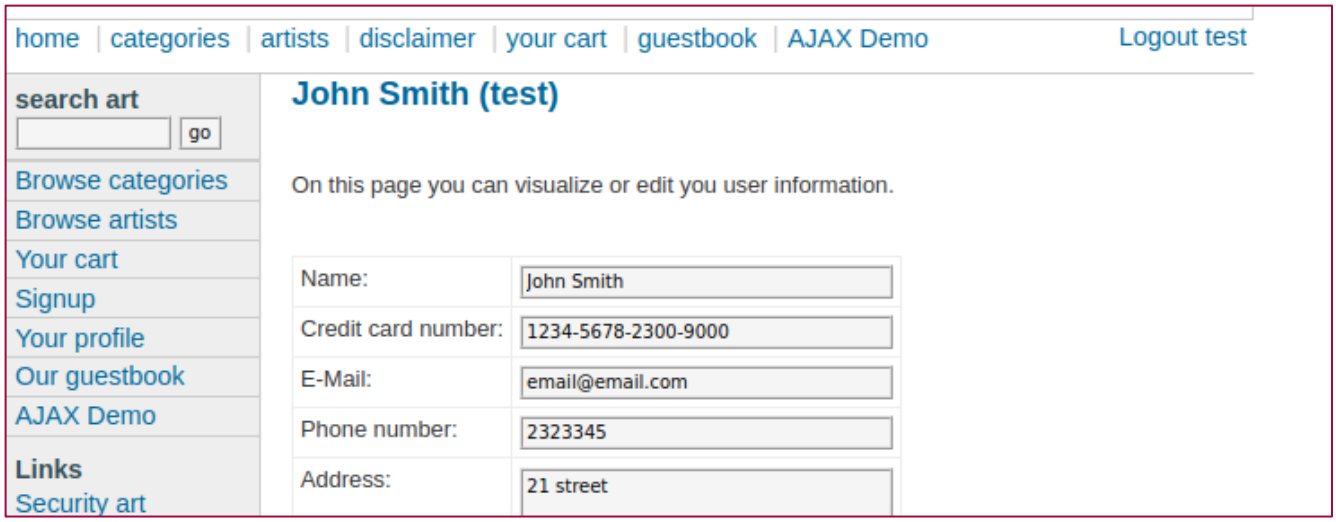


Figure 3: Access to the test account reveals credit card information

Remediation

- Change default passwords
- Use a strong password policy
- Implement multi-factor authentication
- Review, monitor and log access to the test accounts
- Consider disabling test accounts in production when not necessary

References

- Valid Accounts: Default Accounts, Sub-technique T1078.001 - Enterprise | MITRE ATT&CK®
- Risks of Default Passwords on the Internet | CISA
- WSTG - Latest | OWASP Foundation

6.5 Verbose Error Messages [Low]

Description

The application has debug and verbose error messages enabled in production which can be used by malicious actors to gather information about the underlying technologies and infrastructure, and craft target specific payloads.

URL: http://testphp.vulnweb.com/listproducts.php?cat=1%27

Impact

A malicious actor can gather information about the Bravoos eBank Application such as:

- possible SQL injection
- usage of mysql_fetch_array() function
- internal path disclosure /hj/var/www/listproducts.php

Replication

1. Navigate to http://testphp.vulnweb.com/listproducts.php?cat=1%27
2. Note the verbose error message about invalid SQL syntax

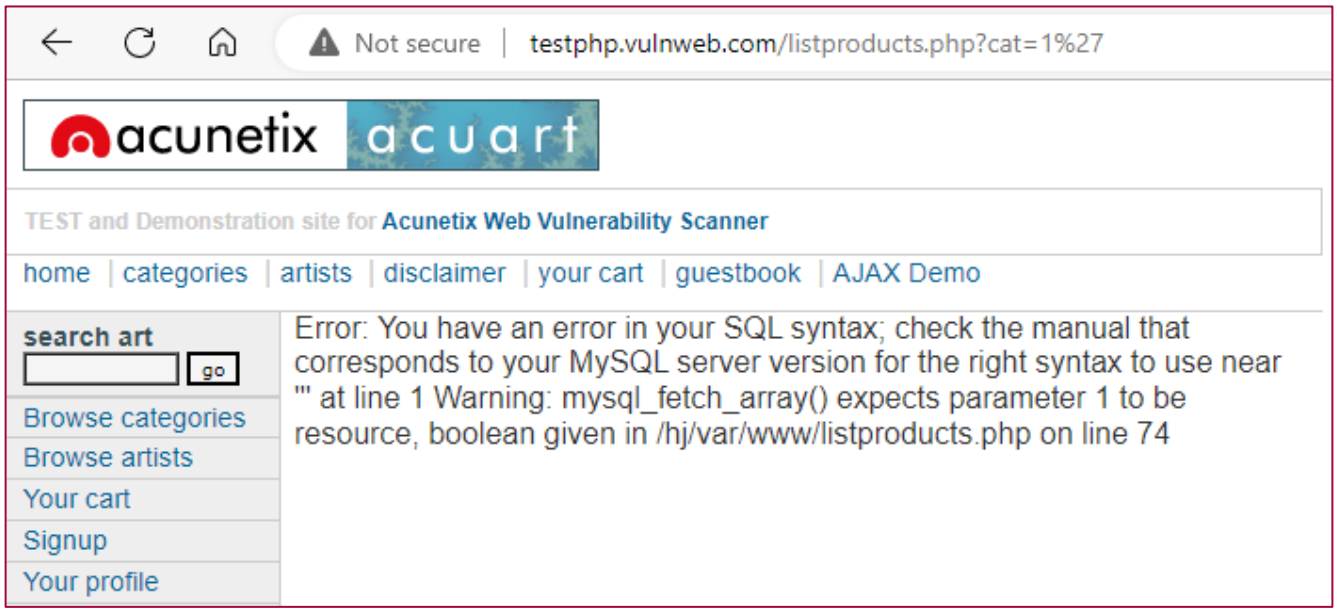


Figure 4: Verbose error message about invalid SQL syntax

Remediation

- Disable debug messages in production
- Replace the verbose messages with generic errors or IDs that can be used to track the issue

References

- Improper Error Handling | OWASP Foundation
- Error Handling - OWASP Cheat Sheet Series
- WSTG - v4.2 | OWASP Foundation

7. Appendix

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

Severity	CVSS Range	
Critical	9.0 – 10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0 – 8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible
Medium	4.0 – 6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1 – 3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	0.0 – 0.0	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

TRIPLA
CONSULT