🛡️ Web Shell Deployment Report

Environment: Lab
Target: Metasploitable2 (DVWA)
Attacker: Parrot OS (VirtualBox)
Author: Alvin Okiya
Date: 27/05/2025

🚀 Executive Summary

This report documents a successful web shell deployment on the DVWA (Damn Vulnerable Web Application) running on Metasploitable2, executed in a controlled lab environment. The objective was to understand real-world exploitation techniques by targeting vulnerable file upload functionality, crafting a malicious payload, and interacting with the system through a reverse shell.

🧪 Environment Setup

Role	System	Configuration
Target	Metasploitable 2	DVWA enabled, NAT network
Attacker	Parrot OS	Owasp zap, PHP

Network: NAT-based VirtualBox network with internal routing between the attacker and target machines.

Objective

- Identify and exploit file upload functionality in DVWA.
- Deploy a PHP-based web shell.
- Establish command execution capability on the target.

Vulnerability Description

The **DVWA "Upload" module** allows uploading image files but lacks:

- Proper MIME type validation
- Restriction on file content
- Execution prevention in the upload directory

This leads to a **Remote Code Execution (RCE)** vulnerability through file upload.

⚔️ Attack Workflow

Set up

I used oracle virtual box as my test environment where I installed metasploitable2 an intentionally vulnerable environment containing vulnerable websites for Security testing such as:

TWiki

phpMyAdmin

Mutillidae

DVWA

WebDAV

Overview of the activities

For this activity I used DVWA. After setting up metasploitable 2 and obtaining the ip address I accessed the Ip address on my parrot vbox to ensure the vulnerable site is up. After that I logged in using using the default credentials for the application and started exploring, my focus was how to upload a shell and gain shell access to the target. This made my scope to be targeting vulnerabilities that can allow me do this, one of them being proper input validation and insecure file handling on web applications. Due to presence of an uploads section where images were being uploaded, I used that as my entry point where I crafted a simple php shell and saved it with .jpeg extension to be able to upload it as an image. I then accessed the uploaded shell and started executing commands on the target system allowing me to view files and contents in the target machine.

The steps below outline the whole process:

1. Upload Interface Discovery

Navigated:

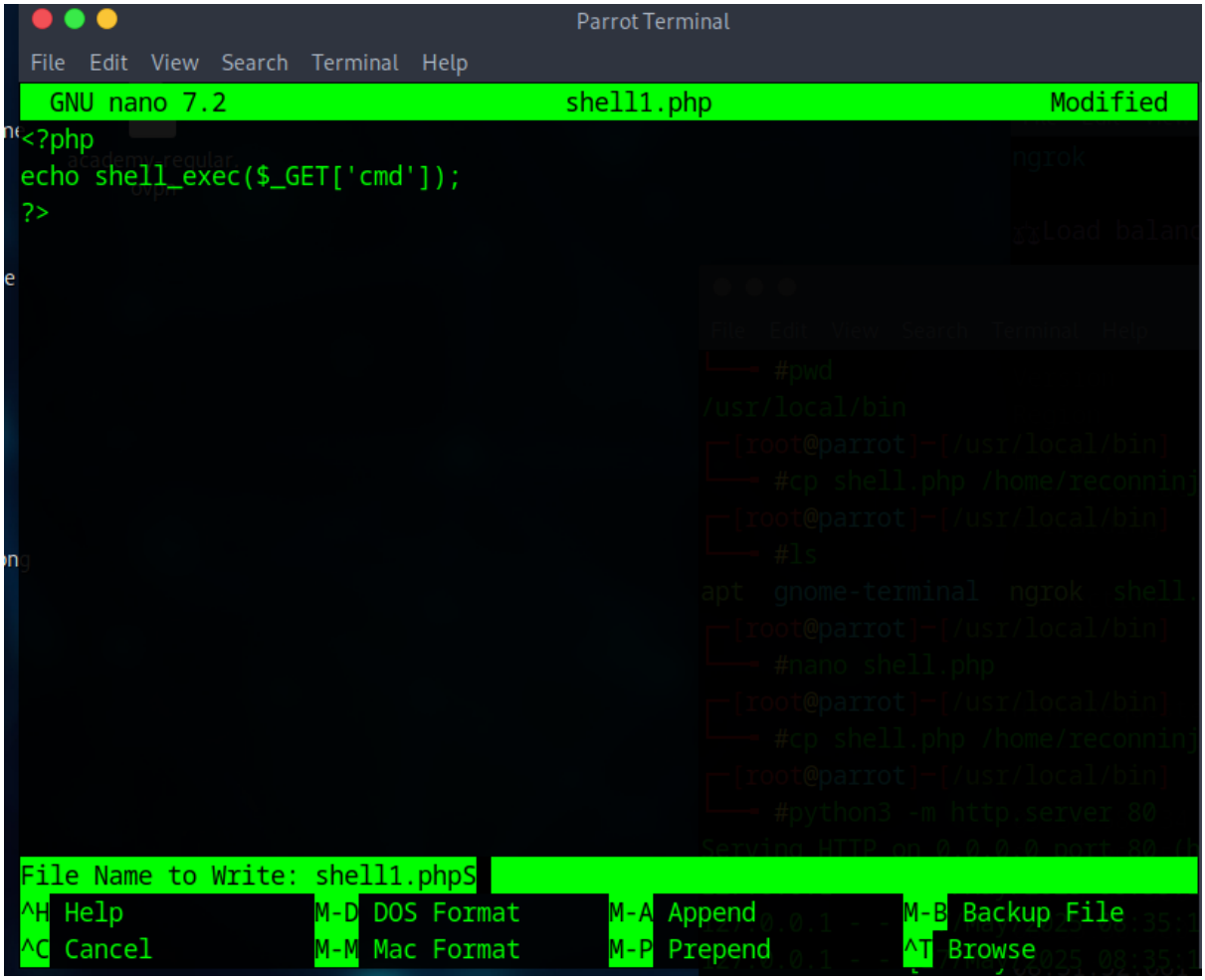
http://192.168.79.8/dvwa/vulnerabilities/upload/

The form accepted image files (.jpg, .png) but did not sanitize file contents.

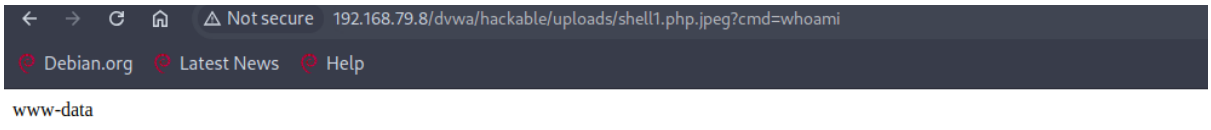
2. Payload Creation

Crafted a **PHP web shell**:

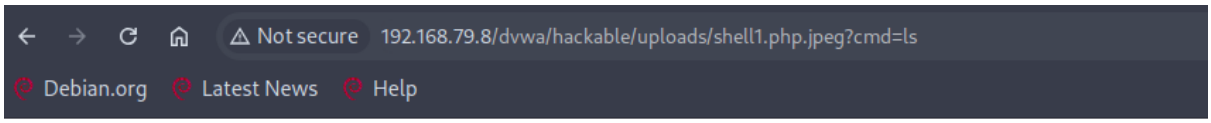
I started with this as I executed some basic commands to check if it is responding:



Which after uploading and gaining access gave me the following output when i executed the whoami command:

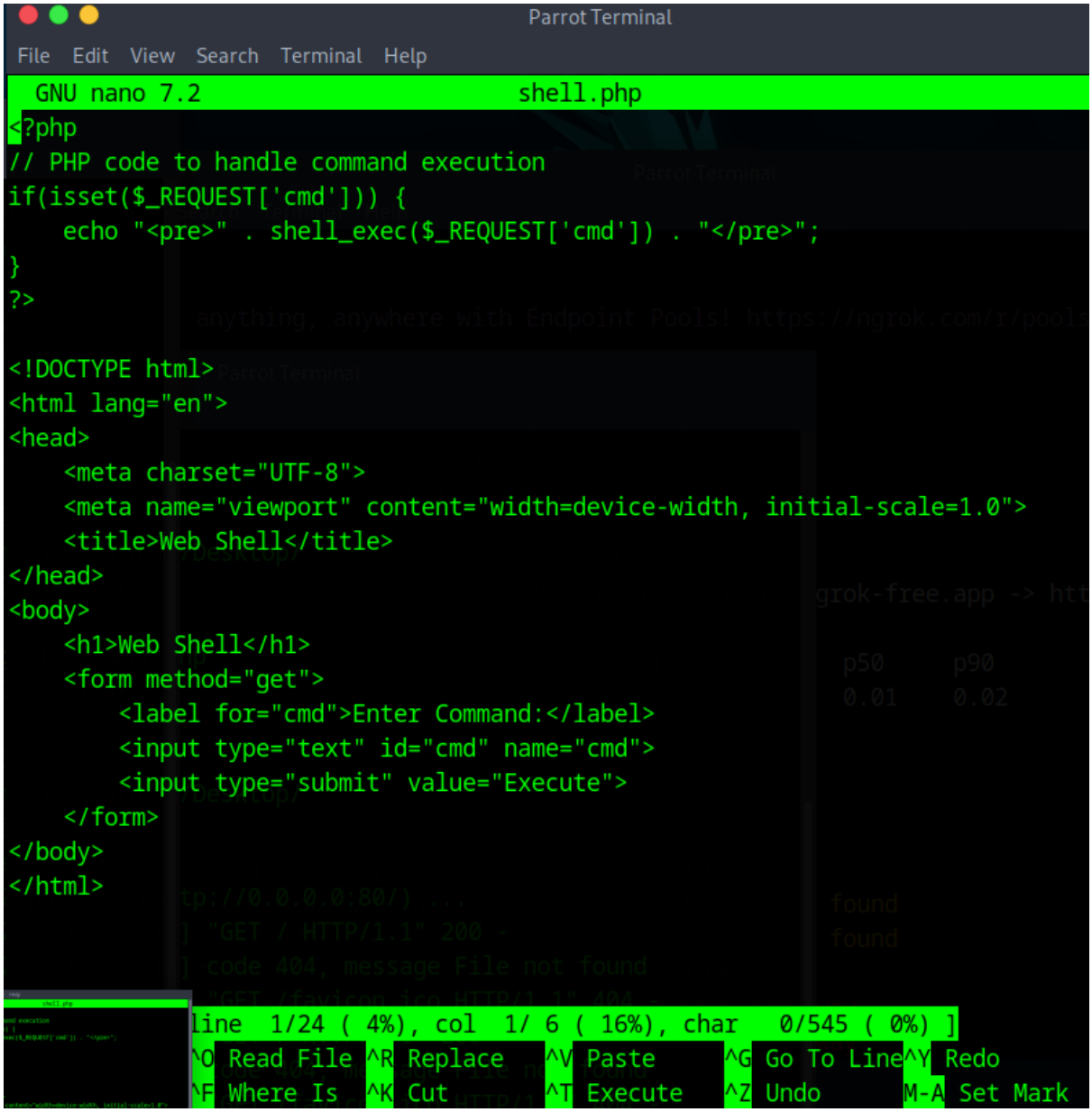


And after executing the ls command I was able to view the content in the directory:



dvwa_email.png shell.php.jpeg shell1.php.jpeg

I then crafted this one which is allows a better view:



Saved as:

shell.php.jpeg

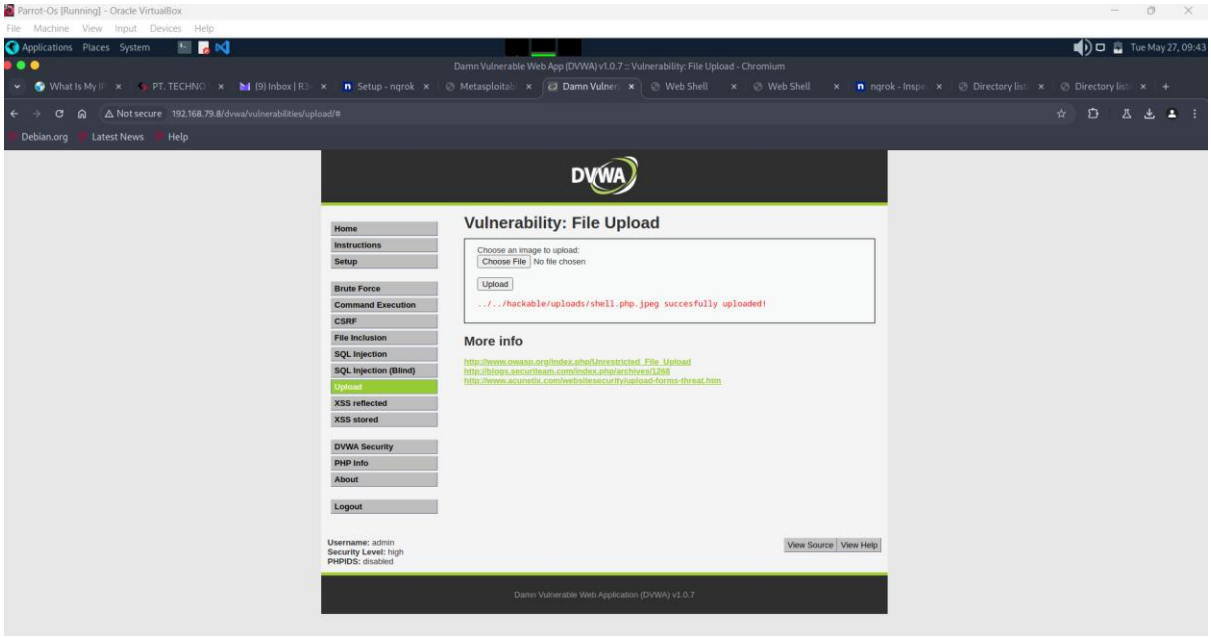
Renamed with a .jpeg extension to bypass simple file extension filters.

---

3. Upload Execution

Uploaded the file using DVWA's form. After upload, the file was accessible via:

http://192.168.79.8/dvwa/hackable/uploads/shell.php.jpeg



Despite the .jpg extension, the server processed the embedded PHP due to Apache's file handling.

---

4. Remote Command Execution

Accessed the webshell:

http://192.169.79.8/dvwa/hackable/uploads/shell.php.jpeg?cmd=whoami

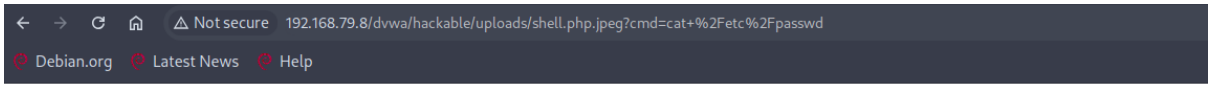
Successful output:

www-data

Executed other commands:

- ls /
- cat /etc/passwd
- uname -a

eg cat /etc/passwd gave me the output below:



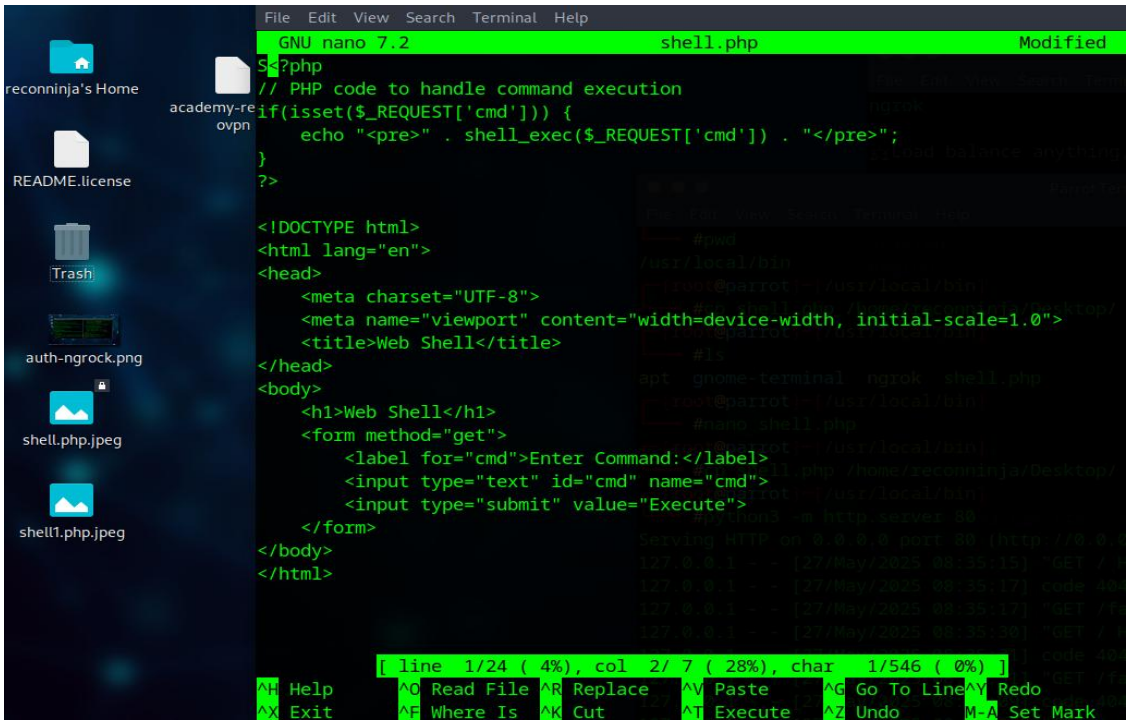

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534::/:/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
service:x:1002:1002:,,,:/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
```

Web Shell

Enter Command: Execute

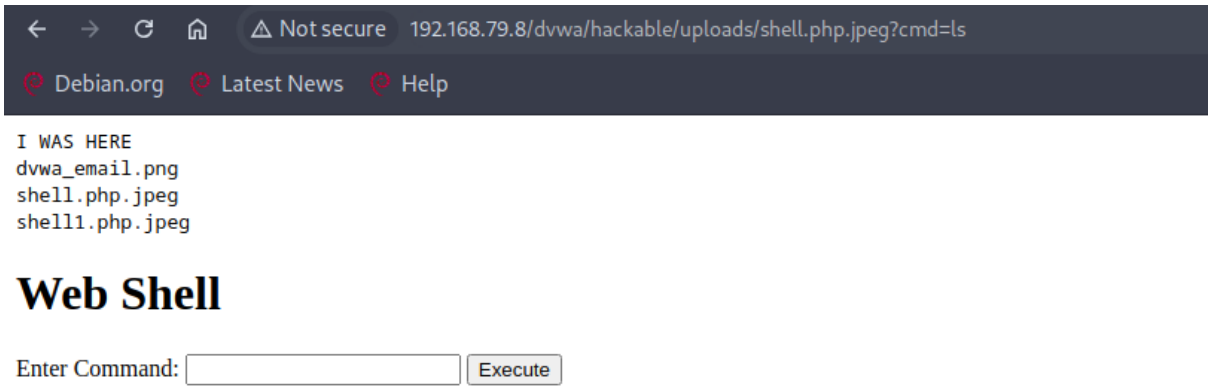
Evidence

📷 Screenshot of uploaded webshell name. After creating the shell.php, i created a copy called shell.php.jpeg

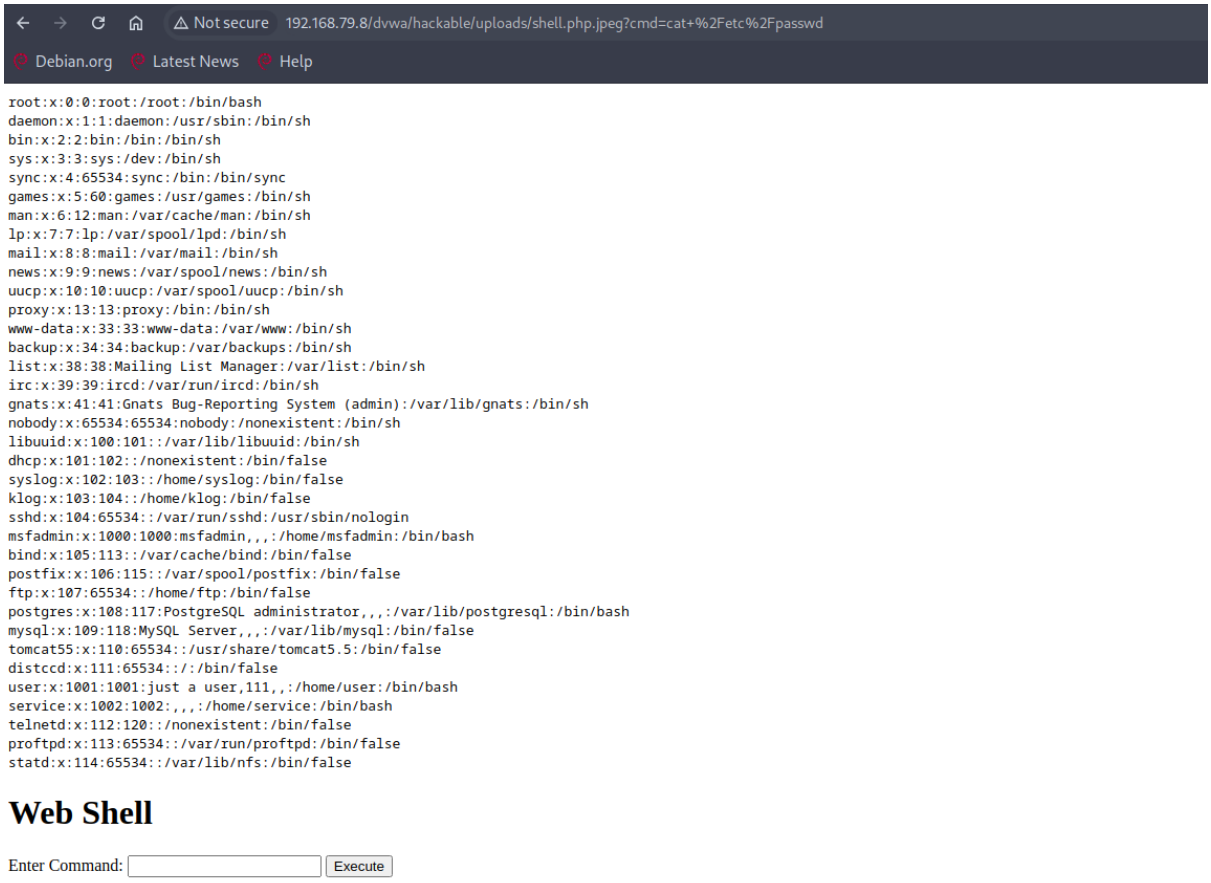


📷 Screenshot of command execution results

I was able to create a directory named “I WAS HERE”:



I was able to view content in the /etc/passwd file:



Impact

An attacker could:

- Execute arbitrary system commands
- Browse and exfiltrate sensitive files
- Establish persistence or pivot into internal systems

Mitigation Recommendations

- Restrict executable permissions in upload directories.
- Implement strict server-side file type and content validation.
- Rename uploaded files with UUIDs and store outside web root.
- Use a WAF or mod_security to filter suspicious uploads.

Conclusion

The DVWA upload module in Metasploitable2 is intentionally vulnerable and allowed successful deployment of a PHP web shell using a double extension trick. This lab confirms the importance of proper input validation and secure file handling on web applications.

Appendix

- Target IP: 192.168.79.8
- Attacker IP: 192.168.79.5
- Webshell path: /dvwa/hackable/uploads/shell.php.jpeg