**CTF Challenge Report — TryHackMe / Root Me**

**1. Title Page**

**Challenge:** root me try hack me ctf  
**Platform:** TryHackMe / Root Me  
**Name:** Adia David Kacou  
**Date:** Thursday, 25 September 2025  
**Tools (used, high-level):** Kali Linux, Nmap, GoBuster, nc(listener), and shell  
**Target IP:** 192.168.64.6

**Time spent:** 2 hours

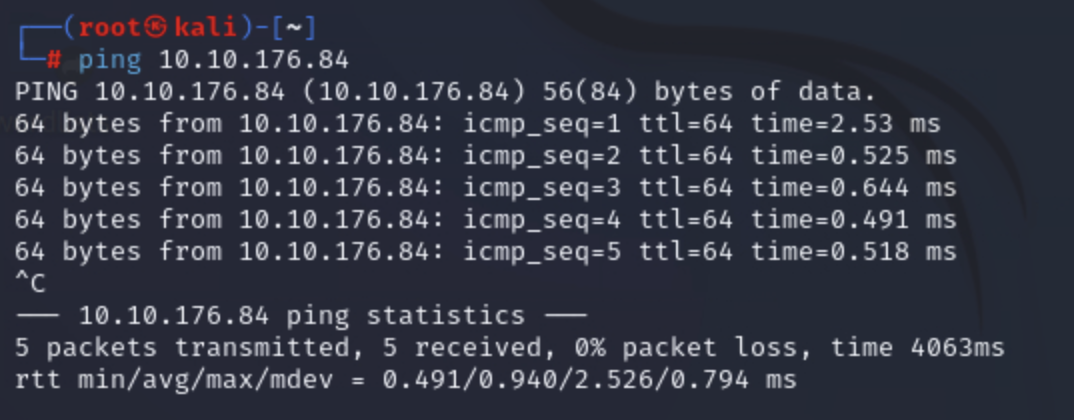
**2. Objective**

Practise network/web reconnaissance, identify and exploit a web upload vector to gain an initial shell (user flag), and escalate privileges to obtain root.txt. Document findings, evidence, impact, and mitigations.

**3. Lab / Network Setup**

* **Network mode:** Shared network (both VMs on same network).
* **Attacker IP:** 10.10.131.186 — Kali VM.
* **Target IP:** 10.10.176.84— target VM.
* **Connectivity verification (high-level):** Obtained interface IPs and verified reachability by pinging each VM from the other.  
  **Evidence:**
* setup-ip.png — screenshot showing ip a / ifconfig output with both IPs.
* setup-ping.png — screenshot(s) showing successful ping(s).

The target IP address was already given by TryHackMe, so I pinged the IP address to make sure my machine was able to communicate with the target machine, and it worked.



**4. Reconnaissance (Network & Services)**

**Objective:** Enumerate open ports and service versions to find attack surfaces.

**High-level action:** Performed a TCP port scan and service version enumeration to determine exposed services and versions (no commands shown here).

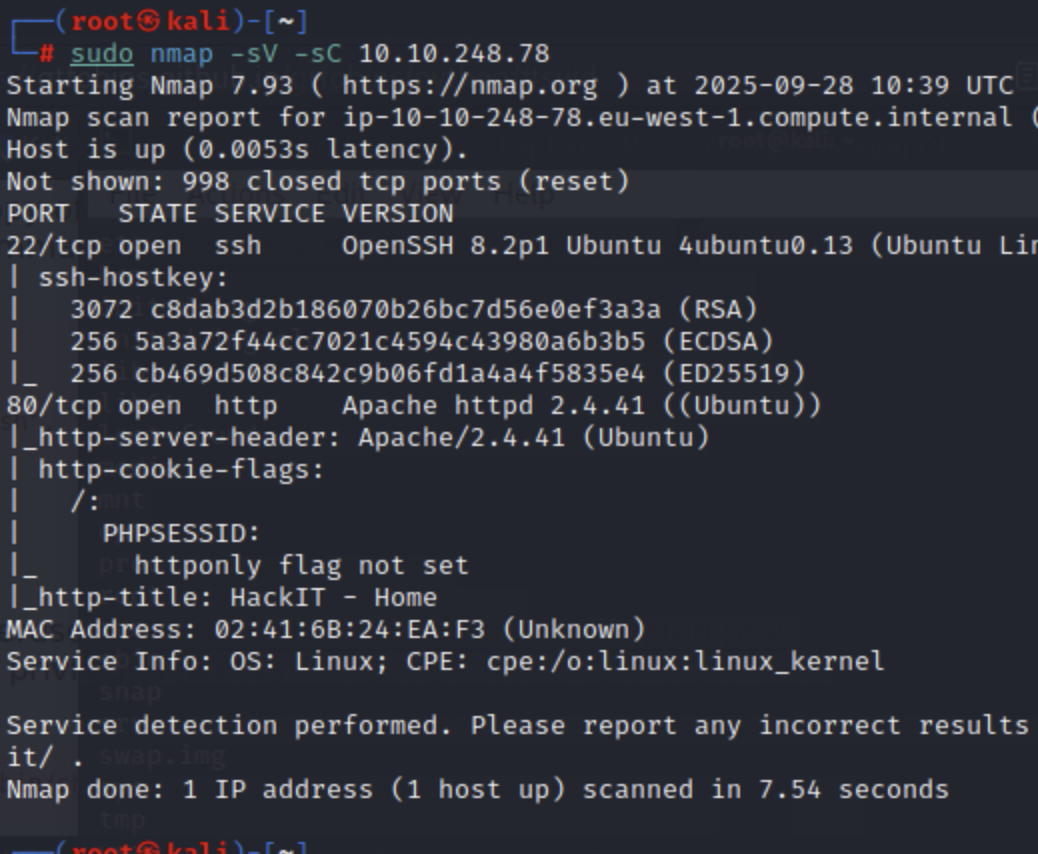
I used nmap with the flags -sV and -sC to check all the ports opened on the target machine and verify with precision their service versions.

I found 2 open ports:

Port. Services. Version

22. Ssh. OpenSSH 8.2p1

80. http. Apache2.4.41

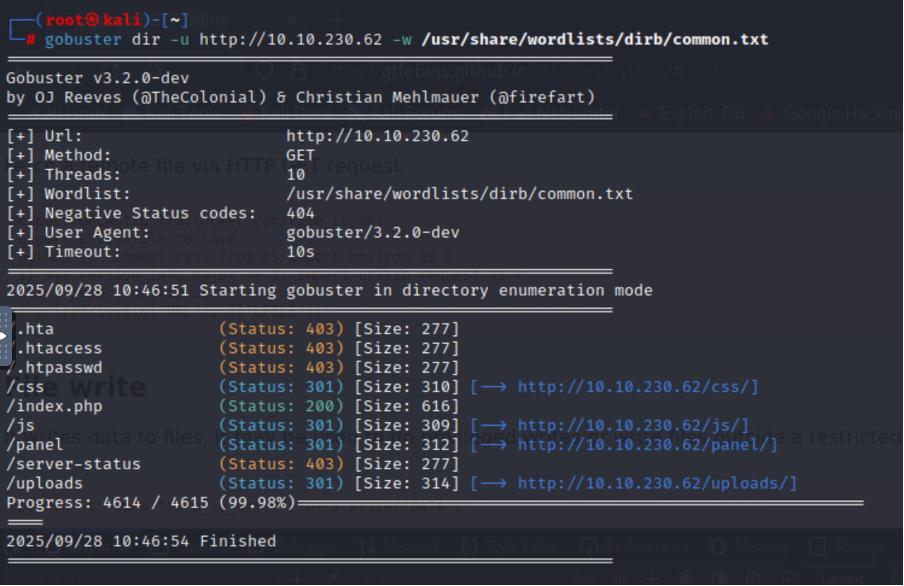


**5. Web Enumeration**

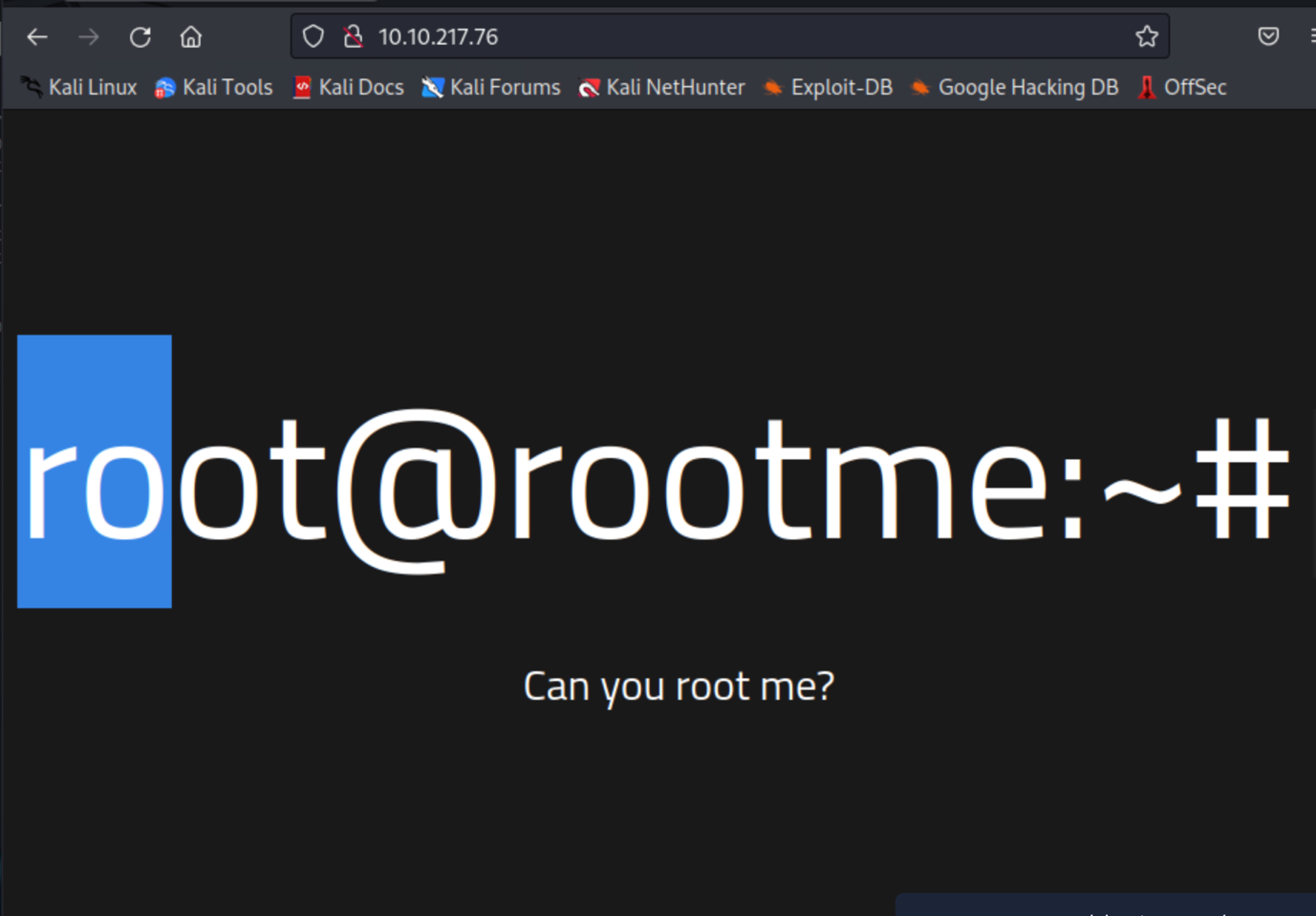
**Objective:** Enumerate web directories, discover admin panels, and identify upload or other functionality.

I used the gobuster command (gobuster dir -u url -w wordlists) to find all the directories, and I found two:

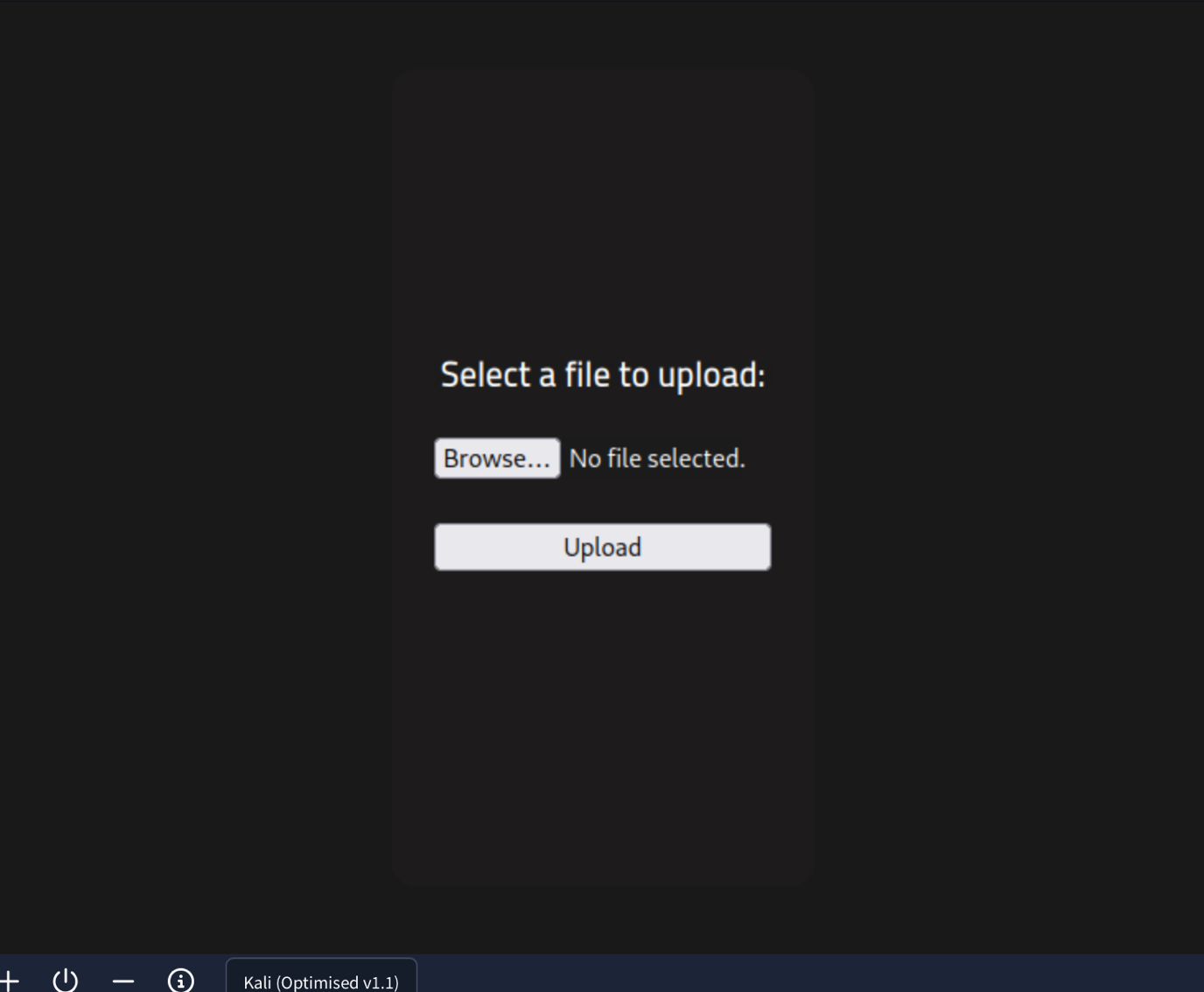
1. <http://10.10.230.62/panel>/
2. <http://10.10.230.62/uploads/>



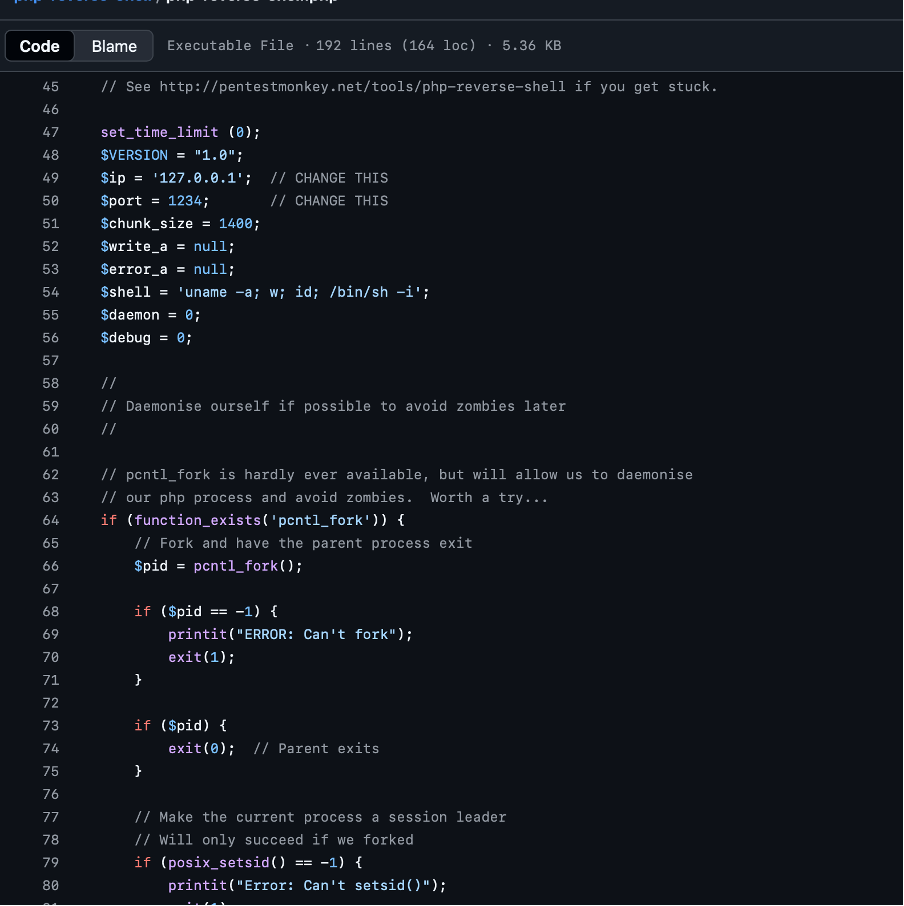
I then proceeded to browse the web page and see what information I could get, but I did not get anything useful on the web page.

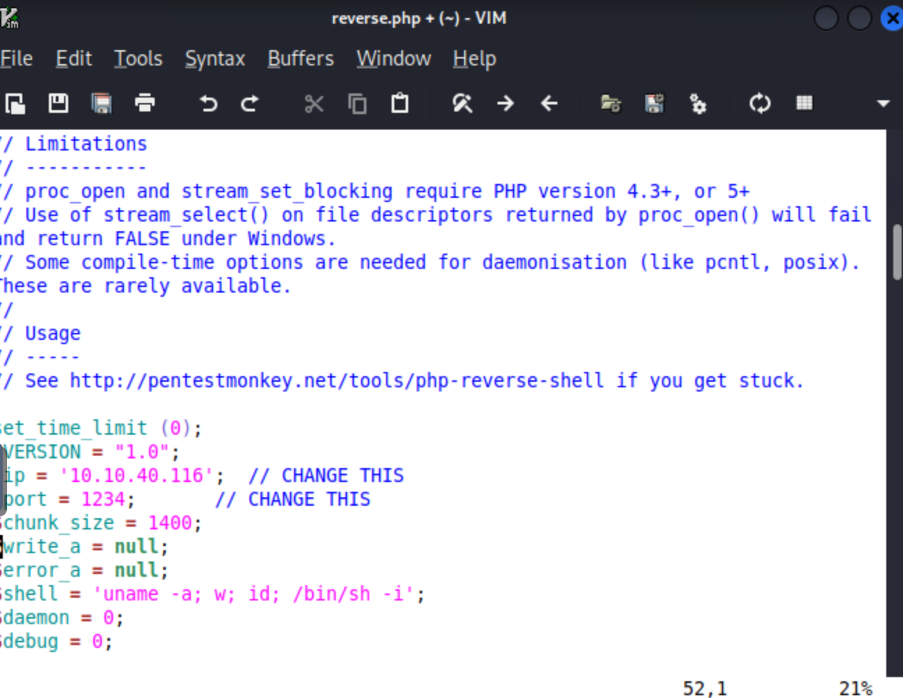


So I decided to browse the panel directory and saw that there was an uploads form. I decided to try to upload a reverse PHP shell code that will allow me to remotely have access to the target network shell.

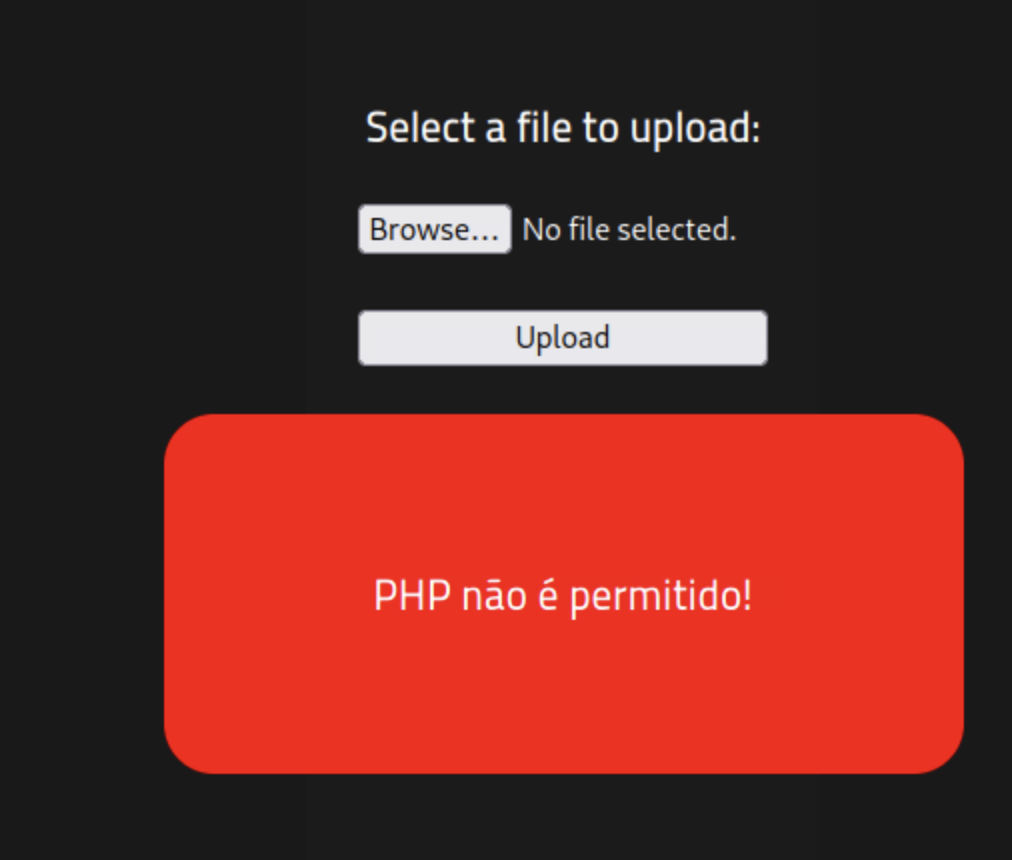


So I went on google and found this php reverse shell code, called phpmonkey (<https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php>) that I decided to use





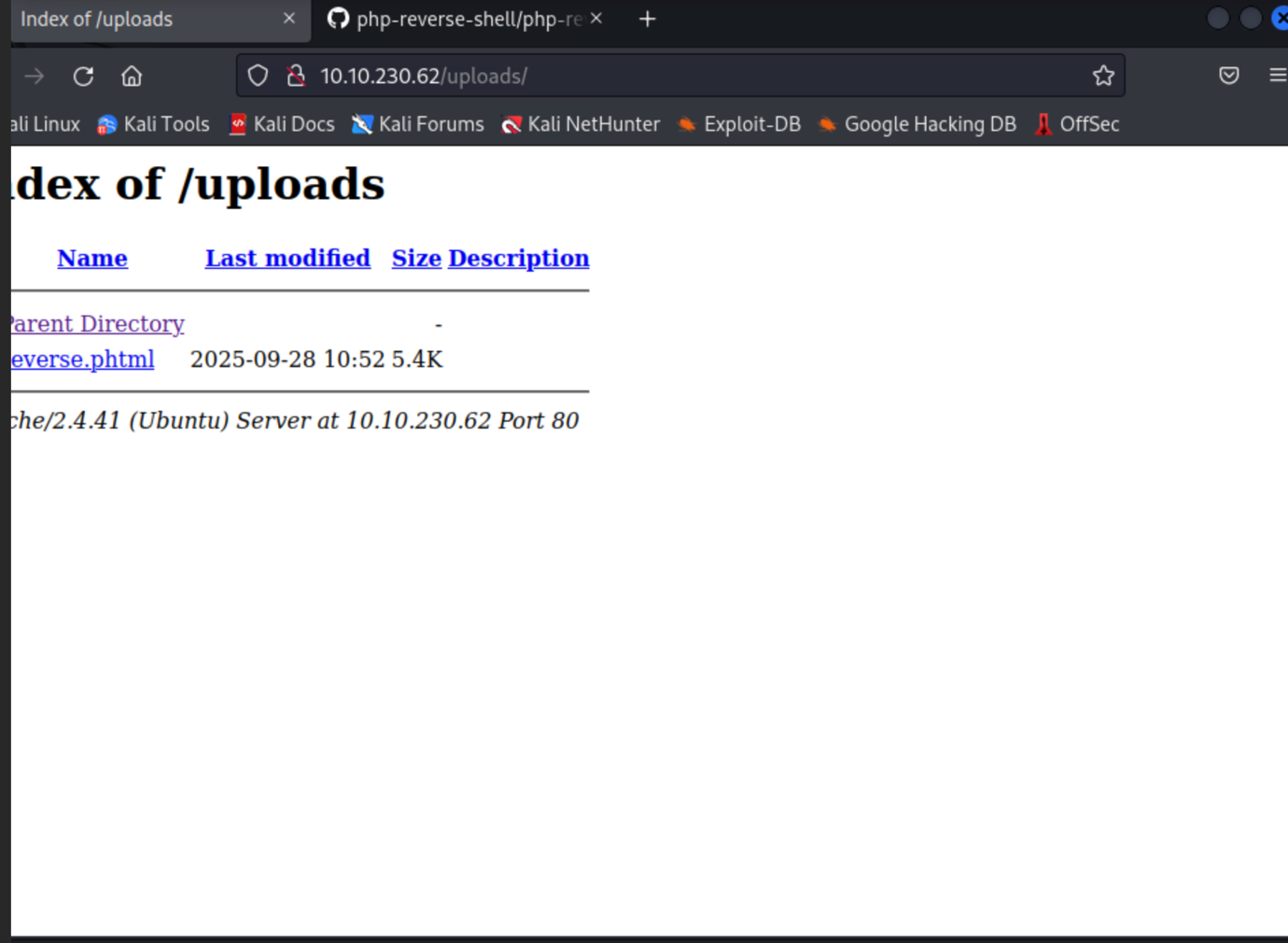
I put my own ip address and the port I want to be listening to, that tells the I am the destination of the reverse shell.



I saved the file as reverse.php, but the website probably has lines of code that stop PHP code from being uploaded to the website.

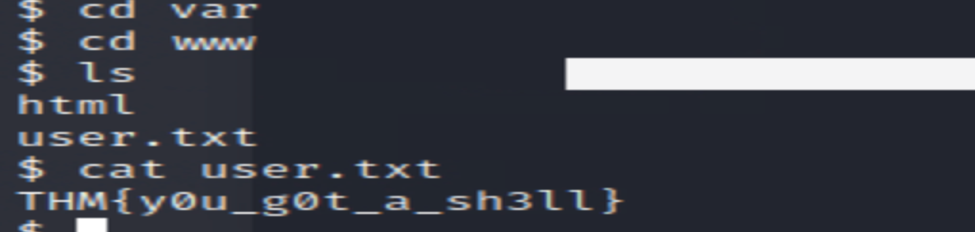
**6. Initial Exploitation — Upload & Reverse Shell (High Level)**

I tried to upload the same file again, but using another PHP file extension (phtml), and it worked. The web developer probably used a blacklist instead of a whitelist. That means instead of stopping other files such as txt , pdf , img from being uploaded, he just wrote a code that stops only the file extension php from being uploaded. That doesn’t stop the PHP file from being uploaded because we can just use another variant of PHP.

****

****

I used nc -lvp to listen, and after executing the PHP file on the website, I was able to listen and get access to the shell remotely. And listed all the files.



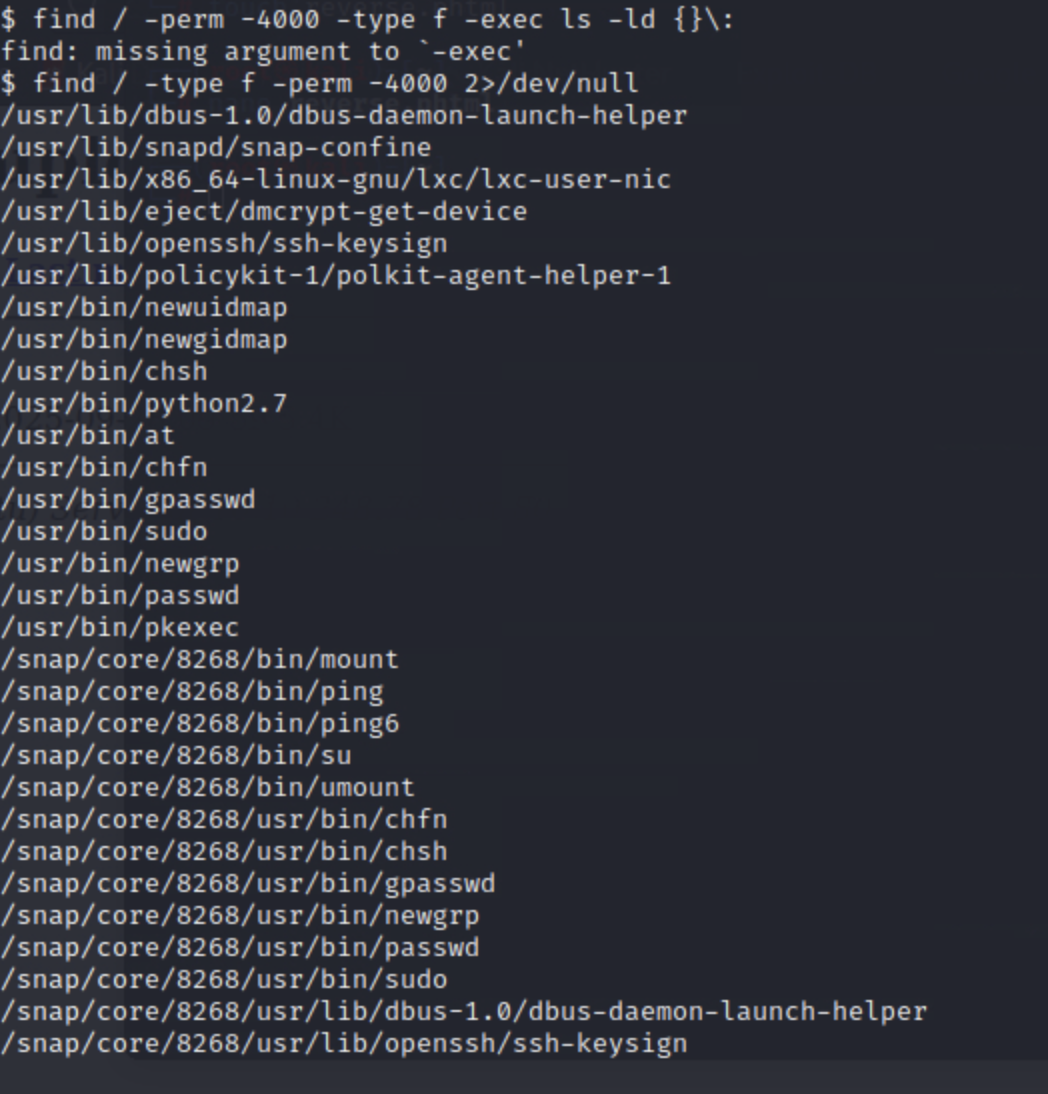
To find the flag, I just went to the var/www directories and listed all the files.

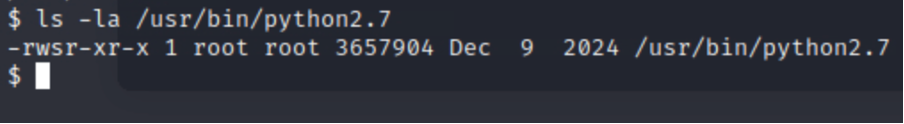
**The flag was: THM{y0u\_g0t\_a\_sh3ll}**

**7. Post-Exploitation & Privilege Escalation (High Level)**

To have privilege escalation, I decided to look for all the files with executing permissions that my user has. For that I used: **find / -perm -4000 type f -exec ls -ld**

Find starts searching from the root directories; -4000 is the number corresponding to execution permissions, and 'f' indicates that I am looking for a file

****

****

I found a python file that I could execute; I will use it to have privilege escalation.

**A screenshot of a computer

AI-generated content may be incorrect.**

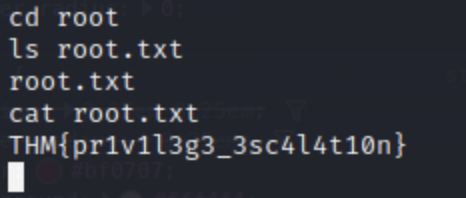
I searched on <https://gtfobins.github.io> a website used to bypass local security restrictions, and I found this line of code that will help me get a privilege escalation using the Python file.

**A computer screen with white text

AI-generated content may be incorrect.**

Running that command allowed me to get a root access in the shell target machine

I knew from TryHackMe that the flag file was located in root and was called “root.txt”, so I used “ls root.txt” to find the flag, which was: THM {pr1v1l3g3\_3sc4l4t10n}

****

**8. Recap / What I Learned**

With this challenge, I cemented my understanding of the whole penetration testing process: starting with reconnaissance to identify available services, then moving into web enumeration to discover hidden directories and upload functionality. I learned how file upload validation is easily attacked by bypassing blacklists using alternative extensions, and how a simple reverse shell can provide initial access. From here, I practiced privilege escalation, enumerating SUID binaries and using GTFOBins methods to root. Overall, this experience learned me the importance of full enumeration, outside-the-box thinking when dealing with filters, and the worth of possessing a clear, step-by-step methodology to compromise and escalate.

**Hardening recommendations:** The site should verify uploads with a strict whitelist (allowed extensions + MIME type checking and magic-byte scanning) and upload files to places outside the webroot (or inside a non-executable directory) so that the webserver cannot execute them; disable server-side scripting in the uploads directory (e.g., php\_admin\_flag engine Off / SetHandler none), use least-privilege file and directory permissions, and remove unnecessary SUID bits on local scripts/binaries. Use input validation and output encoding to prevent injection, log and alert (monitor uploads and suspicious POST requests), configure a WAF and rate-limiting to mitigate bot attacks, keep software and libraries current, and use secure headers (CSP, X-Content-Type-Options, etc.) and HTTPS to reduce attack surface.