

Introduction to Linux and security tools

License



This work by [Z. Cliffe Schreuders](#) at Leeds Metropolitan University is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Contents

[General notes about the labs](#)

[Preparation and logging in](#)

[The password challenge](#)

[Rules of engagement:](#)

[VMware and promiscuous mode](#)

[Familiarisation with the environment](#)

[Basic Linux command skills](#)

[Getting unstuck](#)

[VI](#)

[Piping between programs](#)

[Redirecting to/from files](#)

[Basic Linux networking](#)

[Virtualisation and our use of virtual machines \(VMs\)](#)

[A note about working from home](#)

[Introduction to Backtrack and Kali Linux](#)

[Sniffing and password capturing](#)

[Sniffing Telnet](#)

[Sniffing FTP](#)

[Sniffing conclusion](#)

[Remote shell access and SSH](#)

[Saving your work \(or not\)](#)

[Conclusion](#)


General notes about the labs


Many of the tasks you complete within our labs could be considered illegal if targeted at a computer that you do not have explicit permission to interact with, do security tests on, and attack. In short, keep all activity contained to our labs and to computers you have legal permission to attack. Use common sense, and act within the law, ethically, and according to your own morals. With power comes responsibility, use it wisely.

One of the interesting and inevitable things about working with security attacks, is that because we are often intentionally “breaking” things and making them misbehave for our own intentions, sometimes things do not go exactly according to plan, software may crash or behave erratically. This adds to the challenge, and may require some troubleshooting.

The labs are written to be informative and, in order to aid clarity, instructions that you should actually execute are generally **written in this colour**. Note that all lab content is assessable for the module, but the colour coding may help you skip to the “next thing to do”, but make sure you dedicate time to read and understand everything. Coloured instructions in *italics* indicates you need to change the instructions based on your environment: for example, using your own IP address.

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 and logging in

For 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.

The LinuxZ image will be used directly for some tasks, and serves as a launching point for the various virtual machines (VMs) we will use throughout the course. The VMs include both ethical hacking targets and attack systems.

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

Note: Each of you will be assigned a separate username and password for the LinuxZ image. Don't let anyone else know your password.

The root password -- **which should NOT be used to log in graphically** -- is "tiaspbique2r" (this is a secure password but is quite easy 2 remember). Again, never log in to the desktop environment using the root account -- that is bad practice, and should always be avoided.

The password challenge

Challenge for the semester: attempt to obtain each others LinuxZ passwords.

Rules of engagement:

- No hardware hacks allowed in the labs
- All attacks should be network, software, or social -- obviously not "physical attacks" or "psychological attacks"
- Demonstrate respect for eachother at all times
- The scope is to obtain the LinuxZ password of fellow students within this module, and report your success -- not to obtain any other information or passwords

- If you accidentally capture any other sensitive information or passwords, notify the tutor immediately
- Show the password as proof to your tutor and/or the student whose password it is
- Do not share the password with other students

Shame on those that fail to protect their password, and accolades to anyone who manages to determine another student's assigned password to the image. Extra prestige to those that use a **technical** attack to obtain the password.

VMware and promiscuous mode

Open a terminal console.

One way to do this is to start Konsole from KDE Menu → Applications → System → Terminal → Konsole.

Enable VMware player VMs to put the NIC into promiscuous mode... (If you don't know what this means, look up the meaning of promiscuous mode on Google.) From the host OS (the LinuxZ image) run the following in a console (such as Konsole from KDE Menu → System → Terminal → Konsole):

```
sudo chmod a+rw /dev/vmnet*
```

Familiarisation with the environment

LinuxZ is our custom Linux system, which is based on openSUSE (a distribution of GNU/Linux), and is configured with the KDE desktop environment. Note that many of the VMs you will use are based on different Linux distros, with various desktop environments. Although different distributions may appear to be visually very different, the commands used for each are typically almost identical. By the end of your studies you will be familiar with many flavors of Linux. We will also be using other operating systems (OSs), such as Windows.

Side note: Linux distros have various levels of support for different desktop environments (such as KDE and Gnome), which can often be chosen upon install, and these change the user interface quite dramatically. Note that servers, are often headless; that is, they have no graphical interface, and are typically configured via the command line.

A major difference between various distros is the way that software is organised and installed, and what is installed by default. Some distros are more suited to specific use cases, since they are designed to do certain things well. Beyond these differences, there are many great distributions, and comparisons often come down to a matter of preference.

Why study Linux in this module? Familiarity with Linux and Unix tools are valuable skills. Often Linux is the right tool for the job. Linux servers are common in large organisations, or any serious ICT setup. For example, Google and Facebook run most of their servers on Linux, and even Apple and Microsoft have been known to use Linux servers (as much as Microsoft may rather you not know that). Many security testing tools are only available on Linux, and there are many Linux distros that specialise in security testing, and are industry-standard.

Basic Linux command skills

We will start with a crash course in building your way to Linux command-line wizardry (you have to start somewhere), then quickly jump into network sniffing, password grabbing, remote administration of machines via ssh (secure shell), and other exciting things.

Here's a brief cheat sheet of some common Linux/Unix commands:

Command	Simplified description	Example usage
ls	Lists all the files in your current working directory, similar to "dir" on windows (-la shows details)	ls -la
cp	Copies files (-r to recursively copy directory contents)	cp file1 copy_of_file1
mv	Moves (or renames) a file or directory	mv file1 file2
cat	Prints the contents of a file to the console	cat file1
echo	Prints a message to the screen	echo "hello, world!"
mkdir	Makes a directory	mkdir newdirectory

cd	Change working directory, to “move us into a different directory”	cd newdirectory
----	---	-----------------

If you haven't already, [open a terminal console](#).

One way to do this is to start Konsole from KDEMenu → Applications → System → Terminal → Konsole.

Start by making a new directory, moving into it, and creating another directory there:

```
mkdir mydir
```

```
cd mydir
```

```
mkdir mydir2
```

Display at the contents of your working directory:

```
ls
```

Note, this is lowercase “LS”.

Ok, that lists the directory name, but doesn't tell you much else...

Getting unstuck

If you don't know what a command does, or what flags (command arguments) to use, then look it up in the manual, using the **man** command.

Try it now:

```
man ls
```

(and press enter when prompted which ls you are interested in learning about)

The cheat sheet above mentions how to use **ls** to display more details. Scroll through the manual page you have just opened, and [read about what “-a” does, and what “-l” does](#). You may wish to read about some of the other options and/or take some notes.

When you are done reading, [press “q” to quit](#).

Now try it. Run:

```
ls -la
```

Note, this is lowercase "LS -LA".

This gives a much more satisfying output. The output includes permissions, the user who owns the file, filesize (in bytes), last time each file/directory was modified, and more. We will come back to the meaning of all this information in more detail another time.

You can also read the info page, which like the man page, also provides a summary of usage:

```
info ls
```

(press "q" to quit)

VI

Next, lets edit a file, using vi, the "paragon of editing perfection", and most importantly the editor available on nearly every Unix and Linux system. Run:

```
vi mynewfile
```

Vi is 'modal': it has an insert mode, where you can type text into the file, and normal mode, where what you type is interpreted as commands. Press the "i" key to enter "insert mode". Type a message (such as "hello there"), then exit back to "normal mode" by pressing the Esc key. Now to exit and save the file press the ":" key, followed by "wq" (write quit), and press Enter.

Now you know everything you need to edit files using vi!

Side note: if you want to become a serious guru, take 10-20 mins to run through an online tutorial on vi. You will soon be impressing people with your wizard-like editing skills. What's that you say? Want to delete 30 lines? In normal mode, just type "30dd".

Open the file using vi again, and add another line of text to the file. Save and quit.

Next, list the files in your working directory again. Then, refer to the cheat sheet to copy your new file to a file named "mynewfile2".

Print the contents of the file to the console.

Piping between programs

Another important command line skill, is piping between programs. You can send the output (known as stdout) of one program to the input (known as stdin) of another. For example, imagine you had a large text file, and wanted to find all the lines that matched a certain pattern, and then sort those alphabetically. Doing this manually could take an impossibly long time, depending on the size of the file. Using some simple Unix commands we can do this easily.

Lets start by printing the contents of a file on our system. Run:

```
cat /etc/passwd
```

Note that the passwd file contains information about each of the user accounts on the local system.

Now if we wanted to see only the lines that contained the text "/home/", we can use the grep command, to filter the output to only those lines. Run:

```
cat /etc/passwd | grep /home/
```

Note that there are user accounts that are not for normal users (with home directories), so this command will filter those out.

And to put these in *reverse* alphabetical order, run:

```
cat /etc/passwd | grep /home/ | sort -r
```

What does this command do?:

```
dmesg | grep idVendor
```

Hint: try plugging something into the USB port before and after running the command

Redirecting to/from files

The final command line skill we will cover in this quick crash course is redirection to and from files. You can send the output (known as stdout) of a program to a file, or send the contents of a file to the input (known as stdin) of a program.

So for example, it is easy to save the output from a command to a file. Run:

```
ps aux > processes
```

Note, the **ps** command lists the processes (running programs) on the system, and the “aux” flags tell it to show all of the processes in some detail. The command above writes a list of all the processes on the machine to a file named 'processes'.

To display that file we can send the contents of the file to the **cat** program:

```
cat < processes
```

Alternatively you can tell cat to read the file itself:

```
cat processes
```

The difference is that the first version sends the contents of the file to the cat process, which just receives the lines of text and displays them, whereas the second command tells cat that you want it to open the file and read out its contents.

Another handy program is **less**, which gives you the ability to scroll through the file (as well as lots of other handy features). Run:

```
less processes
```

Scroll through, and press “q” to quit when you are ready.

Write a one line command that only prints the processes that contain the text 'bash' to a file named 'bashes'.

Hint: pipe the output from ps to grep, then redirect the output to a file.

Basic Linux networking

Find your system’s IP address(es). Run:

```
/sbin/ifconfig
```

Note, this is similar in purpose to the ipconfig command on Windows.

Hint: In the lab room, your IPv4 address will start with “192.168”.

On many Linux systems you can leave off the `/sbin/` part, depending on the `$PATH` environment variable, which lists where to look for programs to run. To see what your `$PATH` is currently set to, run:

```
echo $PATH
```

Note, the **echo** command simply outputs something to the screen (so for example, “echo hello, world!” would print “hello, world!” to the console), and variables start with “\$” in Bash, the Linux shell.

Note that the command will display each of the network interfaces, including physical interfaces, and virtual ones. Ethernet ports will be represented as “eth0”, “eth1”, and so on.

If you ever find that your ethernet port has not been allocated an appropriate IP address (for example, there was a problem with the DHCP server), you can usually remedy the problem by running:

```
dhclient eth0
```

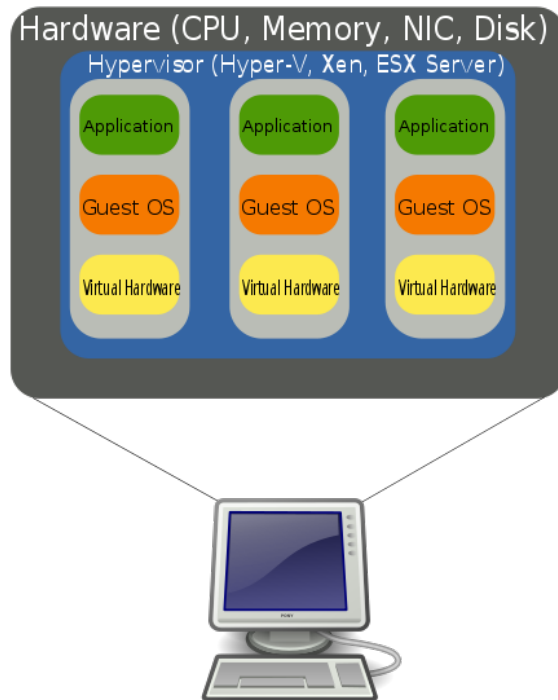
or (depending on distro)

```
dhcpcd eth0
```

You can also try restarting the entire network service, or bring up/down interfaces (ask Google or your tutor).

Virtualisation and our use of virtual machines (VMs)

Virtualisation is a very powerful tool, that can provide important security features, such as isolation, and is an important component of many modern cloud infrastructures. Virtualisation can create virtual environments, and can even run entire operating systems as though they were on separate hardware. This type of virtualisation is known as platform virtualisation, or hardware virtualisation. As illustrated in the figure below, virtualisation allows one set of hardware (a computer), to host a number of guest virtual machines (VMs), each with their own operating systems, applications, and virtual hardware.



Hardware Virtualization ([Image](#): public domain by [John Apledged](#))

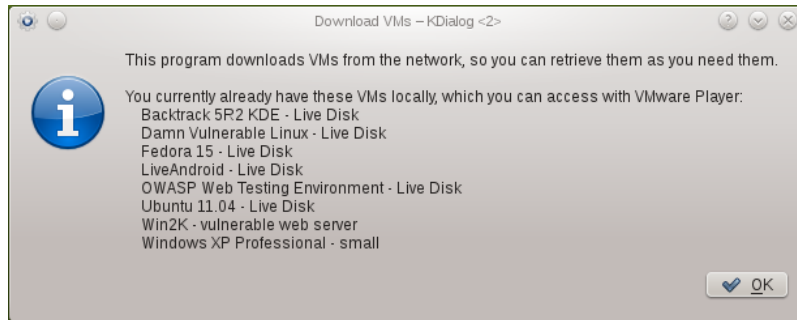
During the module we will make use of various VMs to recreate realistic security scenarios.

To make it easy to set up the various lab tasks and hacking scenarios, we have a network share containing a number of different VMs, including some systems to launch security attacks, and some to be attack victims.

To download VMs:

On the LinuxZ desktop, click the "enable network drive" icon.

Now, on the desktop, click the "download VMs" icon.



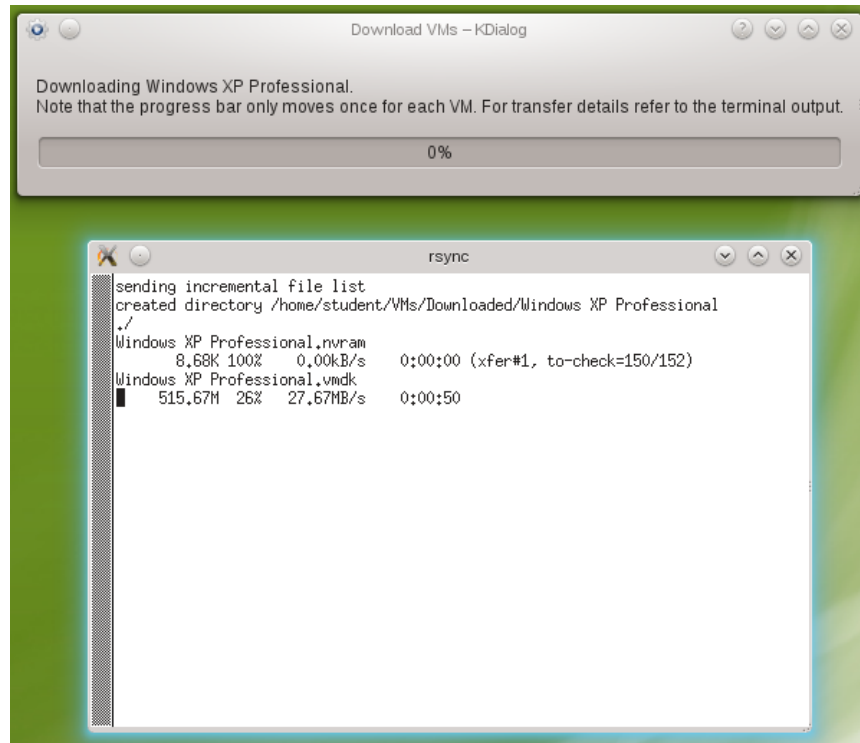
VM download script: already installed list

As shown in the figure above, you will be greeted with a list of VMs already on your system. These VMs can be launched from the “launch VMs” desktop icon. Note some VMs run as Live disks off the ISOs on the network share. These are very lightweight, since you do not have to download the VM, it just runs over the network. However, with the Live disk VMs your changes will not persist when restarting the VM: you will lose any changes within the VM. Click “OK”.

If you want to download VMs, you can do so at the next step. Simply select which of the VMs you want to download to your local system (you can select multiple at a time).

In this case, select the Kali Live Disk VM, which we will use as our attack system, and also select Metasploitable2, which we will use as our victim server.

Click “OK” to begin the download. As shown below, you will then be able to view the download progress. While waiting on the download, please read ahead. Once complete, you will be asked whether to launch the VMs now. Click “Yes”. If prompted, tell VMware “I copied it”.



VM download script: downloading

A note about working from home

If you want to work on labs from home, in many cases this is possible if you are willing to do a little bit of extra work to set up your own lab environment, and so long as you are willing to figure things out when they are a bit different to the instructions given. However, **choosing to do work outside of our lab environment is at your own risk and responsibility.**

The most important thing to consider is to **ensure you are not acting illegally.** As mentioned, many of the tasks we do in our sandboxed labs could be considered illegal if targeted at someone else's servers. Any tasks done in your own environment should be contained to your own personal computer(s), and to be sure you should disconnect your personal lab from all external network connections. **If in any doubt, only complete the exercises within our secure labs.**

Ideally you would have a PC with VMware Player and Virtualbox installed. To make things simplest, you could install openSUSE as the host OS on your PC; alternatively, you could have an openSUSE VM to run the commands intended for the LinuxZ image. Keep in mind that your mileage will vary, and we cannot provide support for your own home setups.

Note that after enabling the network share, it is possible to access all the VMs and ISO files used in the labs via /mnt/NetworkDrive.

Introduction to Backtrack and Kali Linux

In 2013, Offensive Security released Kali Linux, the successor to the very popular BackTrack Linux distribution.

Both BackTrack and Kali Linux are Linux distributions especially designed for penetration testing, and forensics. These distros have become the industry standard for ethical hacking.

If you have not already done so (it should already be running), [start the Kali VM](#).


Assuming you downloaded the Live disk VM, [select the Live boot option](#), by pressing enter. Tip: to get your mouse back to your host OS, press "Ctrl-Alt".

Before long you are greeted by the Kali Linux desktop.

Start by browsing through the tools that are available.

[Click the "Applications" menu → "Kali Linux"](#)

There are an amazing amount of security/hacking tools included with Kali. Take a few minutes to familiarise yourself with the layout of this menu.

Most of these programs are command line tools. [Open a terminal](#), by clicking the console icon ()

This is where the magic happens!

To further emphasise the sheer amount of ~~awesomeness~~ tools, run:

```
ls /usr/bin
```

Have a quick scroll through the vast "arsenal" of tools. Do you already recognise any of these programs?

Sniffing and password capturing

For those of you hanging out to hack something...

When the Internet's underlying network protocols were designed security was not a high priority. The focus was on reliability and stability, rather than communications security. Most networks and hosts were considered to be trusted, and almost all of the

early protocols (including many still used today) do not use any encryption to protect information. That is, information is sent “in the clear”, meaning that anyone who happens to have access to the network can view (and maybe even modify) Internet traffic.

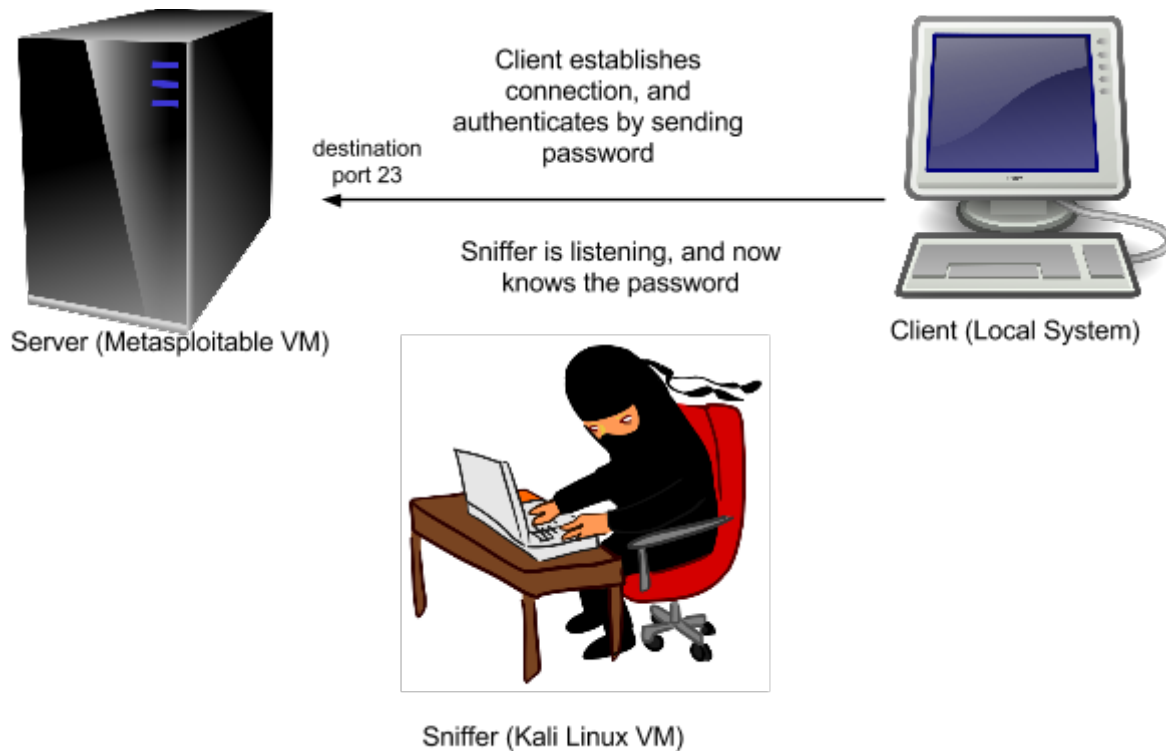
Typically unencrypted communication includes transport protocols, such as TCP/IP and UDP/IP, and also application protocols, such as the Web (http), file transfers (ftp), and email (pop3 and smtp), to name but a few. Only in recent years there has been an increasing awareness of the importance of securing communications.

Sniffing Telnet

An early popular remote login protocol is Telnet. Telnet has an extremely simple client-server architecture. The server listens on TCP port 23. When a client connects to the port they are typically greeted with a login to a shell session, enabling them to run commands on the server. The Telnet protocol is particularly straightforward, essentially all text is sent over TCP practically raw (other than a few special sequences for virtual terminals), with no encryption or other encoding. Remember, Telnet was designed before such things were considered essential.

Lets see some of the major security consequences.

We are going to simulate a three computer situation. One acts as the server, one as the Telnet client, while another malicious host on the network listens in.



Sniffing Telnet using Kali to intercept passwords that are sent in the clear

The figure above illustrates what you are about to do. Using Kali Linux you will eavesdrop on the password being sent by the client (on our local system) to the server, which will be the Metasploitable VM.

If they are not already running (they will be if you have completed the steps above), **start the Kali Linux and Metasploitable VMs**, which you downloaded previously.

Network settings: both VMs should be set to use Host Only networking (which should be the case if you have followed the steps above). Kali Linux should be allowed to enter promiscuous mode (if you skipped the instructions in the preparation section, run `sudo chmod a+rw /dev/vmnet*` on the host OS, LinuxZ).

A note about the Metasploitable VM: Metasploitable is a Linux system that is intentionally vulnerable to attack. Not to be confused with *Metasploit* which is a framework for exploiting/attacking vulnerable systems. In this example the Metasploitable VM will be our 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 (172.xxx.xxx), and can therefore communicate with each other.

There are many different sniffing tools available on Kali, but lets start with a particularly easy one to use: **dsniff**. Dsniff listens to network traffic, and extracts authentication information. In this case it will simply extract the text from a Telnet session, including any passwords that are typed.

On the Kali Linux VM (the attacker), in the terminal you opened earlier, run:

```
ifconfig
```

Identify the interface on the same subnet (starting the same as the Metasploitable IP address). For example, “eth0” or “eth1”.

Start the sniffer:

```
dsniff -m -i eth0
```

Where *eth0* is the interface you noted earlier.

Note: if you get an error message about entering promiscuous mode, then you need to run the “sudo chmod a+rw /dev/vmnet*” command on the host OS, and restart the Kali Linux VM.

Now your sniffer is running, and will attempt to detect any useful information on the local host-only network (between your PC and the VMs).

On your local system (LinuxZ) (the client), connect to the server using Telnet:

```
telnet IP-address-of-Metasploitable-server
```

Where *IP-address-of-Metasploitable-server* is the IP address you noted earlier. For example, “telnet 172.16.165.128”.

Log in to the Telnet server using user: msfadmin, password: msfadmin. You can now run commands on the server. Run “whoami”.

Now run “exit”, to logout of the Telnet session.

On the Kali Linux VM (the attacker), in the terminal you should now see the text from the Telnet session. This third computer was able to eavesdrop on the communications, and steal a copy of the password!

Tip: if you were not successful, this is likely because you did not select the correct network interface when running dsniff on Kali Linux.

Take a screenshot showing a captured Telnet password, as evidence that you have completed this part of the task.

Label it or save it as "Intro-1".

Sniffing FTP


File Transfer Protocol (FTP) is, as the name suggests, a protocol for transferring files over a network. FTP is typically used to transfer files onto Web servers.

FTP has a client-server architecture, and involves two ports: a command port and a data port. The client initiates the session by connecting to TCP port 21 on the server, which is the command port. After authenticating by sending "USER *username*", "PASS *password*" (where username and password are the user credentials), the data connection is established.

There are two ways the data connection can be established, depending on whether the server allows "passive" or "active" ftp. For active FTP, the server connects to a port that the client is listening on, and for passive FTP the client connects to a port that the server is listening to. Each of these approaches have pros and cons, and affects firewall rules.

Like many early protocols FTP is susceptible to eavesdropping to obtain authentication credentials.

On the Kali Linux VM (the attacker), start Wireshark. You can start it with the command "**wireshark &**", or through the Kali menu.

Click the capture interfaces button (), and select the interface you identified earlier (eth0 or eth1). Click "Start".

Wireshark is a powerful sniffer, with a graphical interface.

On your local system (LinuxZ) (the client), connect to the server using ftp:

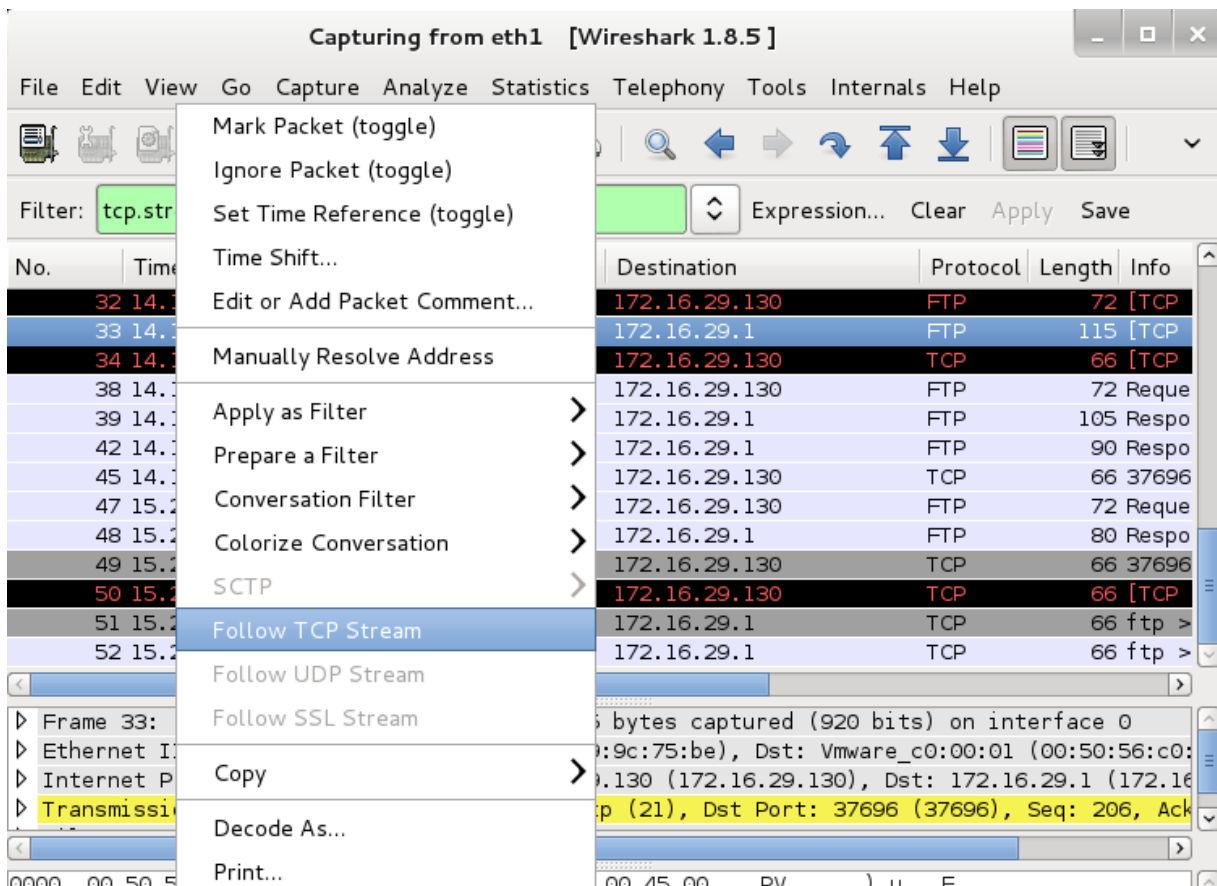
ftp IP-address-of-Metasploitable-server

Where *IP-address-of-Metasploitable-server* is the IP address you noted earlier. For example, "ftp 172.16.165.128".

When you are prompted, enter the username and password "msfadmin", "msfadmin".

You can then type some FTP commands if you like, such as "ls". Once you are finished, type "exit".

On the Kali Linux VM (the attacker), inspect the Wireshark output. The output includes detailed information about the traffic that was recorded. Note that some traffic is identified as using the FTP protocol. As shown below, right click one such entry, and select "Follow TCP Stream".



Read through the captured traffic. Note that the eavesdropping attacker has once again managed to obtain the username and password.

Take a screenshot showing a captured FTP password, as evidence that you have completed this part of the task.

Label it or save it as "Intro-2".

Because Telnet is such a raw protocol you can use the Telnet client to connect to other ports, and interact with other protocols. For example, you can connect to the FTP server using Telnet:

On your local system (LinuxZ) (the client), connect to the FTP server using the Telnet client:

```
telnet IP-address-of-Metasploitable-server 21
```

Where *IP-address-of-Metasploitable-server* is the IP address you noted earlier. For example, "telnet 172.16.165.128 21".

Type "USER msfadmin" (press enter). Note that the server prompts you to provide a password. Type "PASS msfadmin" (press enter). The server will tell you that login was successful. From here you could continue to talk to the server by typing the raw FTP commands, as specified in the FTP protocol. Type "quit".

An even better client to use for experimenting with raw TCP connections is Netcat, the "swiss army knife for TCP/IP". You could repeat the above using: "nc 172.16.165.128 21".

Is dsniff still running? Did it detect the FTP password?

Sniffing conclusion

All the while, the attacker (the Kali Linux VM) has been completely passive, and did not interact with the server or client at all, and was just listening to what was happening on the network. This kind of attack is possible on hub networks (where messages are sent out to everyone), when the attacker has compromised a computer that they want to spy on, or when an attacker is placed in a central location on the network, such as when they have access to any routers. Other tricks are required, such as using man-in-the-middle attacks, when the network is switched (where messages are not broadcast to everyone on the network, as is now common) or the attacker does not have direct access to the same part of the network.

As time has gone on, the importance of securing communications has received more attention. Nowadays insecure protocols such as Telnet and FTP are typically avoided (although not always!), and more secure alternatives or secure versions of these protocols can instead be used.

An important takeaway message is that many attacks take advantage of something that the designers did not consider. In this case, that other hosts on the network may be eavesdropping on messages.

There are lots of other attacks that take advantage of unsecured communication, which we will come back to later.

Remote shell access and SSH

Secure shell (SSH) is a network protocol that enables secure communication between two computers on an insecure network. The primary purpose of SSH is to provide an encrypted remote shell (command line) session, for executing commands on a remote server. This is safer than the old insecure method of remote logins using Telnet or rlogin, which have no encryption and can be easily attacked.

In addition to remote shell access, SSH can also be used for encryption of other kinds of connections, and can be used for SSH tunneling, SCP file transfers, VPNs, and so on. In short, it is very useful.

SSH involves a server listening to a port (typically on TCP port 22, it can be a good idea to change this), and a client connects and sends commands. The protocol uses a form of public-key cryptography for authentication and encryption.

Now let's see ssh in action. You can complete these steps on the LinuxZ image.

On your local system (LinuxZ)

Check whether the local SSH server (sshd) is running on your system:

```
sudo /sbin/service sshd status
```

Note that **sudo** executes the command as root, the Unix superuser. On BackTrack/Kali "sudo /sbin/" is not necessary, since you are logged in as root (again, running everything as root is normally a bad idea, but for security testing purposes is ok).

If you wanted to run an sshd server on Backtrack/Kali, you would start by running "sshd-generate", to create a key pair for the encryption.

If the sshd service is not running, you can start it by running:

```
sudo /sbin/service sshd start
```

Once the sshd service is running, other users can SSH into your machine, and issue commands.

To illustrate this, ssh into a separate computer. This could be the computer at the front of the room (if you are completing this in class, and the tutor has shared the IP address). If you are completing this exercise outside of the class, you can connect to one of your classmate's computers, or even connect to your own IP. Run this command to ssh into the server:

```
ssh username@server-ip
```

Where *username* is your own username, as allocated to you by your tutor for the module.

If you leave out the username (and @ symbol), the username you are logged in as is used to connect to the remote system. This will work fine on the LinuxZ image, since each computer has all the same user accounts.

Enter the password given to you at the start of the class.

You should check the fingerprint displayed, to confirm you are connecting to the machine you think you are connecting to. This protects against man-in-the-middle attacks, where you think you are connecting to a trusted server, but are actually being fooled into talking to a malicious host, who may be intercepting or modifying the communications.

To check the fingerprint, on the server run:

```
ssh-keygen -lf /etc/ssh/ssh_host_key.pub
```

Hint: you may also need to check other .pub files in that directory

If the fingerprint presented to you while connecting matches, type "yes", and from then on if you connect to the same machine you won't be prompted.

If all has gone well you are now sharing the computer. Use these commands and interpret the output:

```
hostname
```

```
/sbin/ifconfig
```

```
whoami
```

```
top
```

```
who
```

```
write user
```

```
ps aux
```

Hint: remember, if you are not sure what a command does, check its man page, using “man *command*”.

Try running “xeyes” on the remote system, why doesn't that work?

Log out of the ssh session (run “exit” or press Ctrl-D), but be careful, if you exit one too many times your current terminal console will close. Now try adding the command line argument “-X” which forwards X11 traffic, so that graphical programs have their interfaces forwarded to your local X server:

```
ssh user@host -X
```

Retry running “xeyes”.

Now the program is running on the remote server, but displayed locally!

Take a screenshot showing xeyes running over SSH, as evidence that you have completed this part of the task.

Label it or save it as “Intro-3”.

When you are finished, simply run “exit” or press Ctrl+D to **exit back to your own local terminal.**

Saving your work (or not)

I highly recommend you save anything you need to keep onto a USB drive, or email files to yourself from the LinuxZ image, rather than saving an IMS image at the end of each class. Simply reload a fresh LinuxZ IMS template image and then download the

VMs you need each time you are in the labs.

If you are in the middle of a lab task and want to save the state of the entire LinuxZ image so that you can continue at a later stage, you can do so. However, I highly suggest you **delete all the VMs from your home directory that you don't need to save**. Otherwise, your IMS image may become ridiculously large and take a very long time to save and load.

Conclusion

At this point you have:

- learned about our lab environment for the module, including some of the virtual machines we will be using;
- developed some very important Unix/Linux command line skills, including common commands and piping between processes;
- learned some basic Linux networking commands and skills;
- learned some techniques to sniff passwords off the network; and,
- learned about remote administration using SSH.

Well done! Not bad for the first week!