OPS245 Lab 7

Contents

LAB PREPARATION

Purpose / Objectives of Lab 7

Setting up a computer network is very important, but the Linux system administrator must also perform networking maintenance which includes **trouble-shooting**, **repairing network connection issues** and **maintaining network security**. System administrators need to **protect or "harden" their computer networks from "penetration" from unauthorized computer users**. Hardening a computer system can range from running an **IDS** (Intrusion Detection System) to monitoring and flagging suspicious activity to implementing security policies which could range from running firewalls to setting locked screen savers on workstations.

Protecting a computer network from unauthorized access is one of the many day-to-day operations for a Linux system administrator and/or security specialist

In this lab, you will learn how to install and configure the SSH service on a VM to allow users to securely access and share data between authorized personnel. In addition, you will learn various methods of running and configuring an ssh server which include: using **Public Key Authentication**, setting up an **SSH tunnel** in order to securely run graphical applications safely among computers in the network, and **disabling root login** into a Linux machine. You will also learn how to set up a firewall using the **iptables** command in order to control the flow of packets throughout your computer server.

Main Objectives

1. To use the **ssh** and **scp** to access and copy data among Linux servers in a secure manner
2. Set up, configure, and start the Secure Shell Service (**sshd**)
   - To refuse root login from remote Linux servers or limit users that are permitted to ssh into Linux servers
3. Generate Public and Private keys to ensure secure connections between Linux servers
4. Use ssh to **tunnel Xwindow applications**
5. Learn about the Linux firewall (via iptables):
   - Use **iptables** command used to configure and maintain a firewall for protection and troubleshooting
   - Configure **iptables** to set a default policy and add exceptions to the default policy

Minimum Required Materials

Solid State Drive

USB key
(for backups)

Lab7 Log Book

Linux Command Reference

Networking Utilities

ssh (http://man7.org/linux/man-pages/man1/ssh.1.html)
ssh-keygen (http://man7.org/linux/man-pages/man1/ssh-keygen.1.html)
ssh-copy-id (http://linux.die.net/man/1/ssh-copy-id)
scp (http://man7.org/linux/man-pages/man1/scp.1.html)
sftp (http://man7.org/linux/man-pages/man1/sftp.1.html)
ss (http://man7.org/linux/man-pages/man8/ss.8.html)
ip (http://man7.org/linux/man-pages/man8/ip.8.html)
ping (http://man7.org/linux/man-pages/man8/ping.8.html)
arp (http://man7.org/linux/man-pages/man8/arp.8.html)
iptables (http://zenit.senecac.on.c

Additional Utilities

hostname (http://man7.org/linux/man-pages/man7/hostname.7.html)
restorecon (http://linux.die.net/man/8/restorecon)

Managing Services

systemctl (http://www.dsm.fordham.edu/cgi-bin/man-cgi.pl?topic=systemctl)

Configuration Files

ssh_config (https://www.freebsd.org/cgi/man.cgi?query=ssh_config&sektion=5)
sshd_config (https://www.freebsd.org/cgi/man.cgi?sshd_config(5))

SSH Reference

A good ssh tutorial (http://support.suso.com/supki/SSH_Tutorial_for_Linux)
A good HOW-TO to make ssh more secure (http://it.toolbox.com/blogs/locutus/shh-securing-ssh-howto-10640)

a/wiki/inde
x.php/IPTa
bles)

INVESTIGATION 1: INSTALLING AND MAINTAINING AN SSH SERVER

So far, you have learned to use the **ssh** utility to establish a secure connection to a remote server in order to perform Linux administration tasks. You have issued the *ssh* command, which is actually the **client** application for ssh. In order to connect to a remote server (like your VMs, Matrix, etc) it needs to run the **SSH service** (i.e. the **ssh daemon**).

In this section, you will learn how to configure an SSH server and restart the ssh service for an existing VM. You will also learn how to configure, restart, and use SSH in order to create secure connections between your Linux machines (host as well as VMs).

Part 1: Confirming sshd service is Running on VMs.

Perform the following steps:

Some tasks in this part of the investigation **require you to be connected to Seneca's VPN**.

- If you are running your installation through VMWare, then you can use the instructions provided by ITS (https://students.senecacollege.ca/spaces/186/it-services/wiki/view/1025/student-vpn) to connect to it from your Windows machine (your c7host and its nested VMs will use the VPN through the windows machine without further configuration).
- If you installed your c7host **directly onto a machine without using VMWare** as an intermediary (or the steps above do not work for you), use the following instructions:
    - Install the package openconnect
    - Run the following command as root (or with sudo): openconnect --protocol=gp studentvpn.senecacollege.ca **-**b
    - This should prompt you for your username and password (you could also put the user name in the command with **-**p)
    - You'll know it is working if you check your ip address and see something in the 10.0.0.0/8 range.
    - To disconnect, as root (or with sudo): killall openconnect

Once you have connected to the VPN with either method you may continue

1. Launch your **c7host machine** and your **centos1** and **centos3** VMs.
2. Switch to your **c7host** VM.
3. Create a file in your current directory of your c7host machine with some text in it called: **myfile.txt**
4. Issue the following command (using your Matrix login id):
   `scp myfile.txt yourmatrixid@matrix.senecacollege.ca:/home/yourmatrixid`
   (followed by your Matrix password)
   What did this command do?
5. Issue the following single command (arguments are separated by a space **-** use your Matrix login id):
   `ssh yourmatrixid@matrix.senecacollege.ca ls /home/yourmatrixid/myfile.txt`
   (followed by your Matrix password)
   What did this command do?
   Issue the following Linux command:
   `ssh yourmatrixid@matrix.senecacollege.ca cat /home/yourmatrixid/myfile.txt`
   How do these commands differ from using issuing the ssh command without the ls or cat command? How is this useful?

   The client ssh application contains the utlities: **ssh**, **scp** and **sftp** (learned in ULI101) to connect to remote Linux servers in order to issue commands or transfer files between Linux servers. You can install the SSH service on your Linux server, although this has already been performed upon installation. We will now confirm that the ssh service is running on all of your VMs.

6. OpenSSH should have been installed by default. Let's confirm this by issuing the command:
   `rpm -qa | grep ssh`
7. You should see a number of packages installed including **openssh-clients** and **openssh-server**
8. The **openssh-server** package installs a service called **sshd**.
9. Confirm that this service is running by issuing the command:
   `systemctl status sshd`

10. Now that you know the service is running, investigate what **port number** and **protocol** sshd uses by issuing the command:
    `ss -atunp | more`
    What protocol and port is the sshd process using? What is the state of the port? Why would you think that UDP ports don't have a state?
11. Reissue the `ss` command without the **-n** option. What is the difference?
12. You can refer to the **/etc/services** file in order to determine a port number for a service. Issue the following command to confirm that port 22 is associated with ssh:
    `grep ssh /etc/services`
13. Make sure the **sshd** service is running on **all 3 of your VM's**

Part 2: SSH Server Security Configuration

Any time that you configure your computer to allow logins from the network you are leaving yourself **vulnerable to potential unauthorized access** by penetration testers or even hackers. Running the sshd service is a fairly common practice but care must be taken to make things more difficult for those individuals that attempt to use **brute force attacks** to gain access to your system. Hackers use their knowledge of your system and can use **password guessing programs** help to gain access. They know which port is likely open to attack (TCP:22), the administrative account name (root).

The Linux system administrator can **configure the SSH server** in order to make the SSH server less vulnerable to attacks. Examples include not permitting root login, and changing the default port number for the ssh service.

Perform the following steps:

1. For this section, you will still be using your **c7host** and **centos1** VMs.

   The next change you can make is to prevent the root account from logging in to sshd altogether.

2. Change to your **centos1** VM and open a terminal.
3. Edit the file **/etc/ssh/sshd_config** (you will need elevated privileges to do so) and look for the option `PermitRootLogin`. **Un-comment the option** (or add the option if it does not appear) and change the option value to `no`.

   NOTE: Now any hacking attempt also has to guess an account name as well as the password.
   If you need root access on the machine you ssh to, ssh as a regular user and then use **sudo**, just like on a local machine.

4. Even better, it is possible to restrict access to just specific users that require it:
   Edit the file **/etc/ssh/sshd_config** and **add** a new option of `AllowUsers yourAccountName` (where "yourAccountName" is your regular user accountname for your centos1 VM)
5. In order for these changes to take affect, you need to restart the sshd daemon. Issue the following command to restart the **sshd** service:
   `sudo systemctl restart sshd`
6. Try SSHing from your **c7host** VM to your **centos1** VM as **root**. Where you successful?
7. Try SSHing from your c7host VM to your centos1 VM as your regular user accountname. Did it work?
8. Create another regular user on centos1 called: **other**
9. Set the password for the newly-created called **other**
10. Try SSHing from your c7host VM to your centos1 VM for the account called **other**. Why didn't it work?
11. Edit the file **/etc/ssh/sshd_config** to add the account **other** for the **AllowUsers** option (use a space to separate usernames instead of a comma).
12. Restart the ssh service.
13. Try SSHing from your c7host VM to your centos1 VM for the account called **other**. Did it work this time?
14. Issue the following command to make a backup copy of your sshd_config file to your original regular user's home directory:
    `sudo cp /etc/ssh/sshd_config /home/regularuserid/sshd_config.bk`
15. Issue the following command to allow same group and other group members to view the file contents:
    `chmod og+r /home/regularuserid/sshd_config.bk`

16. Finally, as a system administrator, you should periodically monitor your system logs for unauthorized login attempts.
17. On CentOS systems the log file that is used is **/var/log/secure**
18. This is the same file that logs all uses of the **su** and **sudo** commands.
19. Attempt to connect to all of your VM's as root and other users using both public key and password authentication. Use some **su** and **sudo** commands also.
20. Inspect the log to see what kind of information is logged.

Answer INVESTIGATION 1 observations / questions in your lab log book.

INVESTIGATION 2: ADDITIONAL METHODS TO SECURE YOUR SSH SERVER

Part 1: Generating Private and Public Keys (Public Key Infrastructure)

As a system administrator, you have the ability to generate or create **public** and **private** keys to ensure safe and secure ssh connections. This will require a user to prove who they say they are in order to access a Linux server via SSH (i.e. **authentication**). The system administer can generate these keys for the first time, or if the system administrator suspects that a hacker has compromised or trying to penetrate the server, they can remove the existing keys and generate new keys.

A common type of attack, **Arp Poisoning (Man in the Middle Attack)**, can be used to redirect packets to a third party while maintaining the illusion that the connection is secure. Therefore, understanding about the generation and management of public/private keys are important to the security of servers.

Storing Fingerprints
When a user connects to a host using ssh, the host sends a fingerprint or digital signature to the client to establish its identity. The first time a connection is established the identity must be stored for subsequent connections. The fingerprints are stored separately for each user in a file called `~/.ssh/known_hosts` .

From now on when you connect to that host the client will compare the received fingerprint against the list of known hosts before connecting. If the fingerprint does not match it could indicate somebody had setup a system to impersonate the computer you wish to connect.

Perform the following steps:

1. Switch to your **centos3** VM.

   We can use the **ss** utility as a trouble-shooting tool to view the SSH service and determine which STATE the SSH service is performing: **LISTENING**, **ESTABLISHED**, **CLOSED** , or **WAITING**

2. Run the **ss -atunp** command (pipe to "grep sshd") to check the state of a possible *ssh connection*. What is the state (i.e. LISTENING or ESTABLISHED)?
3. While in your **centos3** VM, issue the following command to connect to **your same VM** via ssh: `ssh ops245@centos3`
4. Enter yes at the prompt, and enter your OPS245 password.
   The output should appear similar as what is shown below:

```
The authenticity of host 'centos3 (192.168.245.13)' can't be established.
RSA key fingerprint is 53:b4:ad:c8:51:17:99:4b:c9:08:ac:c1:b6:05:71:9b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'centos3' (RSA) to the list of known hosts.
```

5. Issue the following command to confirm that you connected to your centos3 VM: `hostname`
6. Re-run that same **ss pipeline command**. Any change to the connection status?
7. Log-out of your ssh connection by typing `exit`.
8. Run that same **ss** command again. Wait a few minutes and then check again. Record your observations.

So far, we have learned to establish an ssh connection to another host using a password to establish your identity. But **passwords are not the only or even the best way of authenticating your identity**. We can also use **Public/Private key encryption**.

Public Key authentication is a method of establishing identity using a **pair of encryption keys that are designed to work together**. One key is known as your **private key** (which as the name suggests should remain private and protected) and the other is known as the **public key** (which as the name suggests can be freely distributed) The keys are designed to work together to encrypt data asymmetrically, that is to say that when we **encrypt data with one of the keys it can only be decrypted with the other key** from the pair.

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: POSSIBLE DNS SPOOFING DETECTED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The RSA host key for centos3 has changed,
and the key for the according IP address 192.168.235.13
is unchanged. This could either mean that
DNS SPOOFING is happening or the IP address for the host
and its host key have changed at the same time.
Offending key for IP in /home/user1/.ssh/known_hosts:10
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
96:92:62:15:90:ec:40:12:47:08:00:b8:f8:4b:df:5b.
Please contact your system administrator.
Add correct host key in /home/user1/.ssh/known_hosts to get rid of this message.
Offending key in /home/user1/.ssh/known_hosts:53
RSA host key for centos3 has changed and you have requested strict
checking.
Host key verification failed.
```

If you ever receive a message like the one displayed above, you should investigate why it is happening as it could indicate a **serious security issue**, or it could just mean that something on **the host has changed**(i.e. the OS was reinstalled)

While it doesn't mean the message is secure as anybody could decrypt it with the public key, it does establish my identity, if the host can successfully decrypt the message then it must have come from the one person in possession of the private key.

10. Switch to your **centos2** VM.
11. Confirm you are in your centos2 VM by entering the command: `hostname`
12. Make certain that you are in your centos2 VM and that you are logged in as a **regular user** (i.e. NOT root!) (you have been warned!)
13. To generate a keypair (public/private keys), issue the following command: `ssh-keygen`
14. After generating the keys it prompts you for the location to save the keys. The default is **~/.ssh** Your private key will be saved as **id_rsa** and your public key will be saved as **id_rsa.pub**. Press ENTER to accept the default.
15. You will then be prompted for a **pass-phrase**. The pass-phrase must be entered in order to use your private key. Pass-phrases are more secure than passwords and should be lengthy, hard to guess and easy to remember. For example one pass-phrase that meets this criteria might be *"seneca students like to dance at 4:00am"*. Avoid famous phrases such as *"to be or not to be"* as they are easy to guess. It is possible to leave the pass-phrase blank but this is dangerous. It means that if a hacker were able to get into your account they could then use your private key to access other systems you use.

    The output should appear similar to what is shown below:

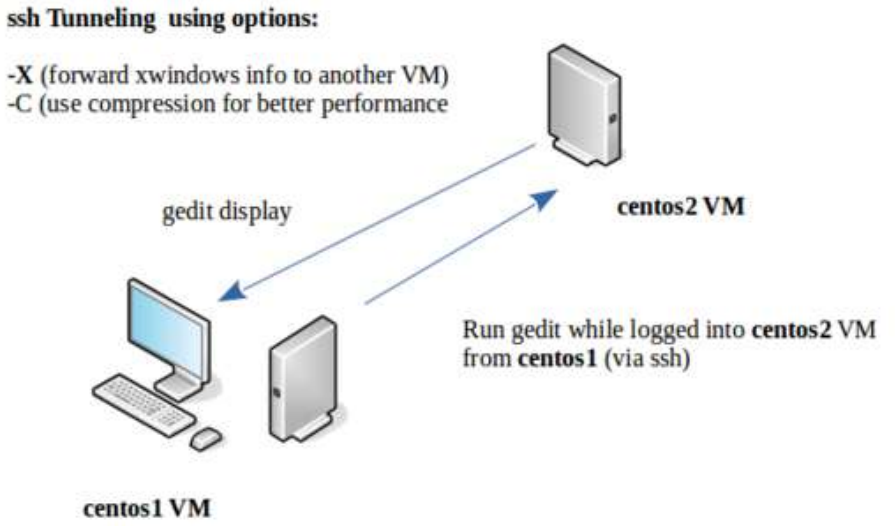
```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/yoursenecaid/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter passphrase again:
Your public key has been saved in /home/yoursenecaid/.ssh/id_rsa.pub.
The key fingerprint is:
ef:de:31:67:f7:15:a4:43:39:15:5d:78:1b:e8:97:74 yoursenecaid@centos2
The key's randomart image is:
+--[ RSA 2048]----+
|              .+=|
|             .+oE|
|            .+.o=|
|            ..++ |
|        S    o.. |
|         .    . .|
|          . o o o|
|         . . = .o|
|         .o .   .|
+-----------------+
```

16. Now issue the command `ssh-copy-id -i ~/.ssh/id_rsa.pub ops245@centos3`
17. When prompted for password, enter OPS245's password

18. Try using ssh to now log into your **centos3** VM from your **centos2** VM. What happens? Were you required to use your pass-phrase?
19. Issue the **hostname** command to verify that you are successfully logged into your **centos3** VM.
20. Make certain to logout of your **centos3** system. Use the **hostname** command to verify you are back in your centos2 server.

Part 2: Securely Running Graphical Applications Between Linux Servers

You can also use ssh to **tunnel window and bitmap information**, allowing us to login to a remote desktop host and **run a Xwindows application** such as **gedit** or **firefox** and the application will run on the remote host but be displayed on the local host.



You can use an SSH tunnel with options to allow running of applications on remote Linux servers.

Perform the following steps:

1. For this section, you will be using your **c7host** and **centos1** VMs.
2. Switch to your c7host VM, open a terminal and remain logged in as a regular user.
3. Issue the following command to connect to your **centos1** VM:
   `ssh -X -C yourUserID@centos1` (where 'yourUserID' is your user account name on centos1)
   (The **-X** option enables the forwarding of X window information, and the **-C** option enables compression for better performance).

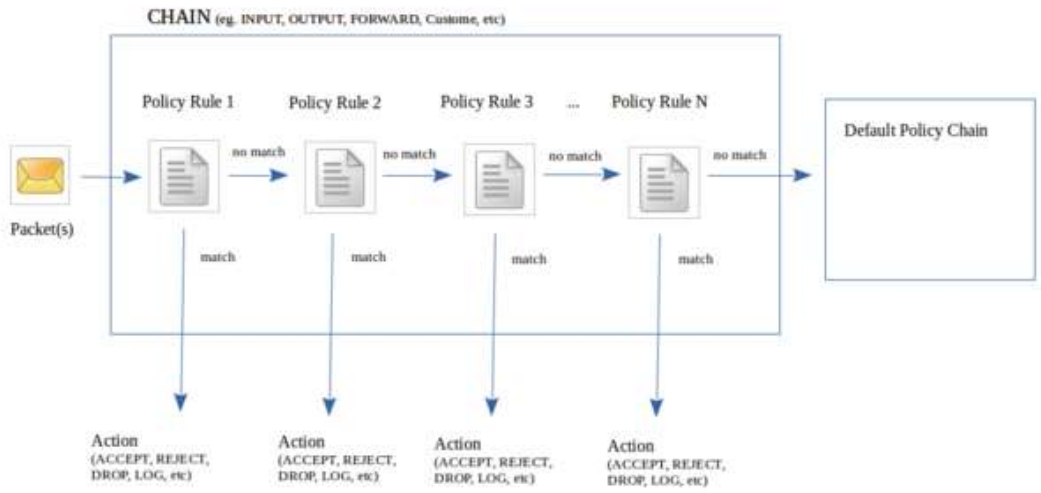
4. Once the connection is properly established, run the command `gedit`
5. The *gedit* window will display on your **c7host** VM, but in reality, this application is running on your **centos1** VM!
6. Enter some text and save your editing session.
7. Exit the **gedit** application.
8. In which VM was the file saved? What does that tell you about the use of tunneling for this section?
9. Run the graphical program remotely by issuing only one Linux command:
   `ssh -X -C yourUserID@centos1 gedit` (Note: ignore warning messages).
10. Exit the gedit application.
11. Experiment with running other GUI applications (in the /bin directory with applications starting with the letter "x" via **ssh** (for example: xev or xchat).

Answer INVESTIGATION 2 observations / questions in your lab log book.

INVESTIGATION 3: MANAGING FIREWALLS FOR PROTECTION & TROUBLESHOOTING

Linux Firewall (iptables)
Concepts

Since Linux servers can be connected to the Internet, it is very important to run a **firewall** to control what packets might come into the computer system, what packets might go out of the computer system, and what packets might be forwarded to another computer. We are currently using the utility called **iptables** can be used to set the firewall rules on a Linux server.



When using iptables packets must pass-through "a chain of policy rules" in order to handle packets. If a packet matches a rule, then an action is taken (some examples include: **ACCEPT**, **DROP**, **REJECT**, or **LOG**); otherwise, the packet will be directed to the default policy chain.

Basically, there is a list (**chain**) of policy rules that **packets** must pass-through in order to handle packets. If a packet matches a rule, then an action is taken (some examples include: **ACCEPT**, **DROP**, **REJECT**, or **LOG**). If the packet passes through the chain of rules without a match, then the packet is directed to the default policy chain (for example: *ACCEPT*, *REJECT*, or *DROP*).

You can create your own **customized chains** (which you will learn in the OPS335 course) but to keep thing simple, we only deal with 3 **common predefined chains**:

- **INPUT**: Packets coming into current Linux server
- **OUTPUT**: Packets leaving current Linux server
- **FORWARD**: Packets being routed between Linux servers

Part 1: Listing & Clearing Existing iptables Rules

Let's get some practice using the iptables command such as listing CHAIN rules, and clearing the CHAIN rules:

Perform the following steps:

1. For the remainder of this section, use your **c7host** machine.
2. Issue the following command to list the existing iptables policy rules: `sudo iptables -L`
3. Were there already iptables policy rules that already existed by default?
4. Note that normal users are not permitted to view iptables settings. Try running the previous command without sudo. You must use sudo to use the iptables command.
5. Before we proceed, we need to understand various methods to list iptables rules:

Listing iptables Rules:

iptables -L	List all iptables rules (eg. INPUT, OUTPUT. FORWARD, and any customized chains (if any)
iptables -L -v	Verbosely List all iptables rules including information such as total size of packets affected by rules
iptables -L CHAIN-NAME	List all iptables rules for that particular chain-name for less clutter (eg. INPUT or OUTPUT, etc)

6. Issue the following Linux command: `iptables -L INPUT`
   What do you notice is different with this command compared to the previous iptables command?
7. Issue the iptables command separately to display the rules for the **OUTPUT** chain and for the **FORWARD** chain.
8. Issue the following command: `iptables -L -v`
   What do you notice about this command as opposed to the first iptables command you issued?
   What sort of additional information does this command provide regarding affected packets?

9. Sometimes it may be useful to completely clear the rules for all or a particular chain. Note the options that can be used to clear (or flush) the iptables rules,

   Clearing (Flushing) iptables Rules:

iptables -F	Clears the rules for ALL of the chains
iptables -F CHAIN-NAME	Clears the rules for only the specified CHAIN-NAME (eg. INPUT or OUTPUT)

10. Issue the following command to reset the iptables rules for the INPUT chain: `iptables -F INPUT`
11. Issue the **iptables -L INPUT** command to verify that the iptables rules for the INPUT chain have been cleared.
12. Now, issue the command: `iptables -F`
    and then issue the command: `iptables -L`
    What do you notice?

Part 2: Setting a Default Policy / Setting Policy Exceptions (iptables)

Usually when setting policy rules with iptables, a general "overall" policy is set (default policy chain). A good way to think about setting policies is to have a **"safety-net"** to take some sort of action to prevent un-handled packets from passing through the firewall by mistake. After the default policy is set-up, then specific exceptions to the default policy can be added to control specific network traffic.

An example would be to set a default policy for incoming network traffic (INPUT chain) to DROP everything, and then set an exception certain exceptions (like ssh connections). Note the following table below for policy setting examples.

Policy Setting Examples:

iptables -P INPUT DROP	Drops all incoming packets regardless of protocol (eg. tcp, udp, icmp), port numbers (eg. 22, 80) or source or destination IP Addresses. Setting a default rule to DROP all incoming traffic would make it easier to specify a few exceptions.
iptables -P INPUT ACCEPT	Accepts all incoming packets regardless of protocol (eg. tcp, udp, icmp), port numbers (eg. 22, 80) or source or destination IP Addresses. It would seem that setting a default rule to ACCEPT all incoming traffic would require A LOT of exceptions to help "lock-down" the server for protection! The better practice is to use DROP as the default action, and only ACCEPT the traffic you actually want.

Perform the following steps:

1. Make certain you are in your **c7host** machine.
2. Issue the following Linux command: `iptables -P INPUT DROP`
3. Issue the **iptables -L** command. Can you see the policy to DROP all incoming connections?
4. Although you have set a default policy to DROP all incoming connections, there is a problem: now, you cannot browse the Internet. You can confirm that by opening a SEPARATE web-browser and perform a Net-search.

In order to fix that problem, you can make an exception to allow incoming web-based traffic (via port 80). Those iptables commands to create exceptions are more complex since you need to determine:

- **Where each rules appears in the chain**? (order can be important)
- **Which protocol(s)** are affected (eg. tcp, udp, icmp)
- **What source or destination IP Addresses** are affected?
- **What port numbers** are affected?
- **What action to take** if all of the above conditions are met? (eg. ACCEPT, REJECT, DROP, or LOG)

iptables Command Structure (for setting exceptions):
(NOTE: If element in column is not specified in the iptables command, then rule relates to ALL elements)

Place Rule in Chain	Chain Name	Specify Protocol	Source/Destination IPADDR	Port Number	Action ->	Target
-A (add / Append to bottom of chain) **-I** (insert at top of chain) **-I CHAIN-NAME 5** (insert before line 5)	**INPUT** **OUTPUT** **FORWARD** **CHAIN-NAME**	**-p tcp** (tcp packets) **-p udp** (datagram packets) **-p tcp,udp,icmp** (combined) (refer to **/etc/protocols**)	**-s IPADDR** (originating IPADDR) **-d IPADDR** (destination IPADDR)	**--sport 22** (originating port 22 - SSH) **--dport 80** (destined port 80 - http) (refer to **/etc/services**)	**-j**	**ACCEPT** **REJECT** **DROP** **LOG**

5. Issue the following Linux command to ensure the loopback interface is not affected by these rules. The computer needs to be able to communicate with itself with any state and protocol:
   `iptables -A INPUT -i lo -p all -j ACCEPT`
6. Issue the following Linux command to ensure the system can get responses to traffic it sends out:
   `iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT`
7. Note that both of the previous rules actually already existed in the default iptables rules before we deleted them.
8. Use your other web-browser to confirm that you can now browse the Internet. If you cannot, contact your lab assistant or professor for help.
9. Determine the **external facing address** of your c7host machine.
   (**Tip:** in a web-browser, enter the term: **"ip address"**. The external facing IP Address should start with **"10."**).
10. Using your Windows machine running VMware (or another machine in your network if you did a bare-metal installation for c7hsost) try to ping that external facing address. Were you successful?
11. Issue the following iptables command to allow an exception for pings from the machine you are using to contact your c7host:
    `iptables -A INPUT -p icmp -s {other machine's external facing address} -j ACCEPT`
12. Repeat pinging your c7host's external facing IP Address. What happened? Why?
13. Try to SSH into YOUR c7host. Were you Successful?
14. Issue the following Linux command to ADD an exception to the INPUT chain to allow incoming ssh traffic (ie. port 22):
    `iptables -A INPUT -p tcp --dport 22 -j ACCEPT`
15. Issue an iptables command to confirm that there is an exception rule to handle incoming tcp packets using port 22.
16. Try to SSH into your c7host (at least to get a password prompt). Were you Successful? If so, why?
17. Note that the rule we entered would allow **anyone** who can reach your c7host to try to log in with ssh. This is **not** a good idea. What would you have to change about that rule to only allow the one machine you are connecting from?
18. Shutdown all VMs and restart your c7host Linux machine.
19. List the iptables rules for the INPUT chain. What happened to your iptables rules for the INPUT chain?
20. Proceed to the next part to learn how to learn how to make your iptables rules persistent.

Part 3: Making iptables Policies Persistent

Any changes to your iptables policy rules will be lost when you restart your Linux server, unless you make your iptables rules persistent. Failure to perform the following steps after setting up your firewall rules can cause confusion and wasted time.

Don't save copies of rules that libvirtd will auto-add every boot.
The libvirtd service running on your c7host automatically adds some rules to iptables to allow the machines in your virtual network comunicate with each other and the outside world. We don't want to save these, or you will end up with two (or more) copies of them in your firewall. This won't actually break anything, but it does clutter up your iptables and make them harder to read. Before you continue with this investigaion, restart your iptables. This will leave you with just the rules that exist in its saved configuration, and not the ones added by libvirtd.

Perform the following steps:

1. Make a backup of the file **/etc/sysconfig/iptables** by issuing the command:
   `cp /etc/sysconfig/iptables /etc/sysconfig/iptables.bk`
2. Make sure the default policy of your INPUT and FORWARD chains are both set to DROP.
3. Delete the rule in the INPUT chain that allows SSH traffic from **anyone**, and replace it with one that only allows ssh traffic sent by your other machine (that is, your windows host, or other machine in your network).
4. Note that this now means your VMs won't be able to ssh to your c7host, so add a rule that allows the entire network your VMs are on (192.168.245.0/24) to ssh to your c7host.
5. Delete the rule in the INPUT chain that allows ICMP traffic from **anyone**, and replace it with one that only allows ssh traffic sent by your other machine.
6. Delete the rule in your **INPUT'** ***and*** *FORWARD* ***chains that*** *REJECT***s any traffic you haven't** *ACCEPT*ed. You are better protected by the default **DROP** policy you set.
7. To make the iptables rules **persistent** (i.e. keeps rules when system restarts), you issue the command: `sudo iptables-save > /etc/sysconfig/iptables` (read the next step)
8. You will notice that even when running the command with sudo, it isn't letting you write to `/etc/sysconfig/iptables`. Use `sudo -i`, then try to save them again. When done, log out of root user (exit sudo).
9. Verify that the file **/etc/sysconfig/iptables** exists.
10. Restart your iptables service and test your configuration.
11. Restart the libvirtd service, and note the rules it adds to your iptables. It will do this automatically every time it starts.

Answer INVESTIGATION 3 observations / questions in your lab log book.

LAB 7 SIGN-OFF (SHOW INSTRUCTOR)

Follow the submission instructions for lab 7 on Blackboard.

Time for a new backup!
If you have successfully completed this lab, make a new backup of your virtual machines as well as your host machine.

Perform the Following Steps:

1. Make certain ALL of your VMs are running.
2. Switch to your **c7host** VM and change to your user's **bin** directory.
3. Issue the Linux command: `wget https://raw.githubusercontent.com/OPS245/labs/main/lab7-check.bash`
4. Give the **lab7-check.bash** file execute permissions (for the file owner).
5. Run the shell script and if there are any warnings, make fixes and re-run shell script until you receive "congratulations" message.
6. Arrange proof of the following on the screen:

✓ **centos2** VM:

- have logged into centos3 VM using **public key authentication** (with a pass-phrase)

✓ **c7host** Machine:

- have tunneled Xwindows application from **centos1** via ssh
- Run the **lab7-check.bash** script in front of your instructor (must have all `OK` messages)

✓ **Lab7** log-book filled out.

7. Upload a screenshot of proof from the previous step, along with your logbook, and the file generated by **lab7-check.bash**.

Practice For Quizzes, Tests, Midterm & Final Exam

1. What port does sshd use by defaults?
2. What file is used to configure sshd?
3. What kind of files are stored in the "~/.ssh/" directory?
4. How do you determine whether the sshd service is running on your system or not?
5. What is the purpose of the ~/.ssh/known_hosts file?
6. What is the purpose of the ~/.ssh/authorized_keys file?
7. Which system log file records each use of the sudo command?
8. How do you stop the sshd service?
9. How do you tunnel XWindows applications?
10. What port is the default ssh port?
11. What port(s) is/are used by httpd service?