# Lab-8-1: Server Hardening and Advanced Firewall

In today’s lab, we are investigating how to perform server hardening, that is, perform a series of tasks to improve the security of our server. Our goal of today’s lab is to harden the SSH service.

In today’s lab, we need to **start the following two virtual machines**:

1. Kali-Linux
2. Ubuntu-Sever

**Please Note:** Your lecturer has set up a new Virtual Machine (VM) using Ubuntu 16.04-2 64-bit for today's class. Make sure you have the VM available under your D:\IN618\_VMs folder. For your information the following details of the server are:

IP address: 192.168.19.19

Username: opstudent

Password: IN618security

If you are interested in how easy it is to set up SSH on an Ubuntu server, you can install the software (provided you have an Internet connection) using the following command:

opstudent@ubuntu:~$ sudo apt-get install openssh-server

(You do not need to run this command!)

*RESOURCE*: SSH Install on Ubuntu (very basic)

<http://ubuntuhandbook.org/index.php/2016/04/enable-ssh-ubuntu-16-04-lts/>

**Exercise-8-1-1: Configuring a Very Secure SSH Service**

We must implement high security to reduce any potential security issues using a layered approach: that is, configuring multiple layer of security for our server.

We are going to start by hardening our SSH service installation. This is exceptionally important as the SSH service provides remote access and the ability to execute commands and programs on the server.

Make sure you are logged onto Kali Linux. Open a terminal and log into the new Ubuntu-Server using SSH. We will use the same SSH command as last week:

root@kali:~# ssh opstudent@192.168.19.19

Remember, the password for the opstudent user is: IN618security. Also, the ssh command works by specifying a username followed by the "@" symbol and then the target IP address of the host. If we supply no username, we will be logged in as the root user.

You will see some output similar to below. Make sure to accept the server's SSH public key by typing "yes" or simply "y" and pressing enter.

root@kali:~# ssh opstudent@192.168.19.19

The authenticity of host '192.168.19.19 (192.168.19.19)' can't be established.

ECDSA key fingerprint is SHA256:LI/WA4/pm+2YGQ26v4JvSl1fB3VxC/brNksX0NHLxSA.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '192.168.19.19' (ECDSA) to the list of known hosts.

opstudent@192.168.19.19's password:

Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-62-generic x86\_64)

\* Documentation: https://help.ubuntu.com

\* Management: https://landscape.canonical.com

\* Support: https://ubuntu.com/advantage

Last login: Sat Apr 1 20:28:26 2017

Remember, you can view the servers public key in your home folder SSH directory using:

root@kali:~# cat ~/.ssh/known\_hosts

(You do not need to run this command!)

The output should be similar to the key listed below:

|1|Oc2fs2+679rbiAgw1zeWvqUv4F4=|s49BO/7qs1PoZxvPM8EBaWDNYlg= ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBGbgDcuAtWmX+wdUamFtREOgAmip+7oyTlZvOcf2iq/vWPvfDuRrjmkdgb01LKNQHE0gNqDaJK7Bl2OVISGUg6Q=

Now we are going to configure our SSH server. Currently, **SSH has been installed but no other configurations made**. Therefore, we are going to have to investigate the different configurations that we could make to improve the security.

Before this, we are going to make a banner to scare off potential intruders. This will what a user sees when then attempt to connect with the SSH server, so it should contain some information that only authorised users should connect to the server. Use the nano editor to create a new file and enter in some text to the file.

opstudent@ubuntu:~$ sudo nano /etc/ssh/banner.txt

For example, you could use something like the following:

###############################################################

# Welcome to Ubuntu-Server #

# All connections are monitored and recorded #

# Disconnect IMMEDIATELY if you are not an authorized user! #

###############################################################

# If you attempt to illegally access this server you will be #

# breaking Section 250 of the New Zealand Crime Act (1969): #

# Damaging or interfering with computer system which carriers #

# a maximum penalty (imprisonment) of 7 to 10 years. #

###############################################################

To save the file, press and hold "Ctrl" then press "O". The name that the file will be saved as will be displayed at the bottom of the editor. Press "Enter" to accept the default name (that we already provided in the command above: /etc/ssh/banner.txt). Your file will now be saved. To exit, press and hold "Ctrl" then press "X".

Now, let us configure our SSH server. First, we should make a backup of the original configuration file. If we accidently make any changes, we could use this backup to restore the default settings. This is a good practice to maintain. We can copy the file using the copy, or cp, command. This command works the same in Linux and Windows. Use the cp command followed by the location of the file to copy, and the copy destination. Use the following command to copy the configuration file to a backup called "sshd\_config.factory".

opstudent@ubuntu:~$ sudo cp /etc/ssh/sshd\_config /etc/ssh/sshd\_config.factory

Excellent, now we are ready to get started. We can edit the SSH configuration file using the following command (again using the nano text editor):

opstudent@ubuntu:~$ sudo nano /etc/ssh/sshd\_config

To make life easier for you, the entire SSH configuration file is provided at the end of this lab document. There are a wide range of configurations available. First, look over the SSH configuration file. For a complete listing of all configuration commands available in the SSH conf file with a very useful description of each see: <https://linux.die.net/man/5/ssh_config>

NOTE: This syntax of the line changes are:

Original value > New value

So if we have the line:

Port 23 > Port 22345

This means delete “Port 21” and replace it with “Port 22345”

Now we are going to change some default settings. First, find the line that has "Port 22", this is the default port number that SSH is going to use. To change the default port change the following line:

Port 23 > Port 22345

**Q1.** We have talked before about changing port numbers from default values a few classes ago (e.g., changing FTP from default port 21 to 2121). How do you think this increases security (there are a variety of answers). Use google if you are stuck (e.g., "why change default port numbers").

Changing the port numbers increases security because only you know what you changed the port number to. So if you changed the SSH port from 22 to 22345 then if an attacked tries to hack via SSH, they will not know the correct SSH port number.

Now, we should change some more settings to improve security. We can hide the default message to display the last login time. This is useful to disable so we do not give away interesting and potentially useful information. To remove the last login time change the following line:

PrintLastLog yes > PrintLastLog no

The “LoginGraceTime” value specifies how long after a connection request the server will wait before disconnecting. It is recommended to reduce it to 60 seconds. For this, change the line

LoginGraceTime 120 > LoginGraceTime 60

Setting up a proper maximum number of concurrent connections to the SSH daemon can be helpful against a brute-force attack. For this, change the line

#MaxStartups 10:30:60 > MaxStartups 2

NOTE: THE “#” or hash in front of the line means the line is commented out. We need to remove the “#” to enable a configuration.

**Q2.** Do some research using Google to find what SSH MaxStartups are. Explain what #MaxStartups 10:30:60 means? If you are stuck, try googling the following term "MaxStartups 10:30:60".

The 10 is the number of unauthenticated connections before it will start dropping.

The 30 is the percentage chance of dropping once it reaches 10

The 60 is the maximum number of connections at which it starts dropping everything

**Q3.** What do you think would be a good number of MaxStartups would be? This could be dependent on the number of users we have, so include how many users your answer is tailored for (e.g., we have two users who consistently use SSH, so MaxInstances should be 2 because they may connect at the same time).

3:50:2

The port forwarding technique available in SSH is sometimes used by attackers to tunnel network connections through an SSH session to log into systems. This is a dangerous scenario and should be blocked if possible. It is recommend to disable this option. To accomplish this, change the following line to:

X11Forwarding yes > X11Forwarding no

By default, SSH logs a lot of information. However, if you want to log even more information such as failed login attempts you can change the "LogLevel" value from "INFO" to “VERBOSE.” For this, change the following line:

LogLevel INFO > LogLevel VERBOSE

One more thing, change the Banner configuration to the banner we have previously made:

# Banner /etc/issue.net > Banner /etc/ssh/banner.txt

We have finally finished our configuration. We must save our text document and exit. To save hold down "Ctrl" and "O". Then press, "Enter" to save the file using the provided file name. After this, you should have returned to the terminal in our SSH session.

Finally, restart the SSH server so that our new configuration can be used. The following command should get this done. For your information, the "service" command can be used in Ubuntu (and some other Linux systems) to start, stop and restart services. We will restart SSH using the following command:

opstudent@ubuntu:~$ sudo service ssh restart

All done with setting up SSH, well almost!

*RESOURCE:* If you are interested, the following links are some good tutorials and discussions on configuring SSH on Ubuntu (including security settings):

<https://help.ubuntu.com/community/SSH/OpenSSH/Configuring>

<https://www.maketecheasier.com/secure-ssh-server-ubuntu/>

**Exercise-8-1-2: Setting up SSH Access Using Public Key Cryptography**

To improve our security even further we are going to generate a public/private key pair for Kali Linux machine, and use it to logon to the server without a password. Make sure you are on the Kali Linux machine (and not in an SSH session to the Ubuntu server, if you are still in an SSH session, use “exit” to leave). Run the SSH key generation using the following command.

Press enter once to accept the default path to save your key, and enter a passphrase twice (you must remember this password!).

You should see output that is similar to that listed below:

root@kali:~# ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /root/.ssh/id\_rsa.

Your public key has been saved in /root/.ssh/id\_rsa.pub.

The key fingerprint is:

SHA256:gL4zGP09GPu68gqSE+2l/M1V6y7anEggVvmCz56kyZg root@kali

The key's randomart image is:

+---[RSA 2048]----+

| |

| o |

| + . |

| . = . . |

|. \* \* o S . |

| \* O = = . . |

|+ \* B = + . |

| \* B.B \*.+ |

|E + =+O+=oo |

+----[SHA256]-----+

**Q5.** Document your password so you do not forget (obviously this a very bad method to remember passwords!).

password

Now we have a public/private key of our very own. The only thing we have to do now is get our public key to the server. Luckily, for us, there is a very simple program to achieve this, called ssh-copy.

root@kali:~# ssh-copy-id [opstudent@192.168.19.19](mailto:opstudent@192.168.19.19) –p 22345

/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed

/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys

opstudent@192.168.19.19's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'opstudent@192.168.19.19'"

and check to make sure that only the key(s) you wanted were added.

Since we have an SSH key now we can disable the use of passwords to authenticate with SSH to our server. We can make this modification by changing the "PasswordAuthentication" value to "no" in the SSH configuration file on the server. But to accomplish this, we need to log back into the server using SSH… but this time we need to specify the correct port number that we changed last time:

root@kali:~# ssh [opstudent@192.168.19.19](mailto:opstudent@192.168.19.19) –p 223345

Once again edit the SSH configuration file using nano:

opstudent@ubuntu:~$ sudo nano /etc/ssh/sshd\_config

And now, change the following line so that no-one can use passwords to logon:

# PasswordAuthentication yes > PasswordAuthentication no

Restart the SSH service…… again!

opstudent@ubuntu:~$ sudo service ssh restart

Exit the SSH session by typing “exit”. Now logon again…

**Q6.** What password did you enter? Why was this password required to logon to the server?

No password was required.

**Q7.** Think about the use of passwords compared to the use of encryption keys. What would you suggest using and why? For more information, have a quick read of the following StackExchange post:

<https://security.stackexchange.com/questions/69407/why-is-using-an-ssh-key-more-secure-than-using-passwords>

Passwords are definitely useful to have over encryption keys as it is easier to have multiple trusted users if they all know the same password rather than having encryption keys for each individual that wants to have access to the system.

*RESOURCE:* A quick guide to setup SSH keys on Linux

<https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys--2>

**Exercise-8-1-3: The Final Step: Firewall Configuration**

We did some basic firewall configuration last week. This week we will use the knowledge we have to set up a selection of firewall rules specifically for our very secure SSH service. Make sure you are logged onto Kali Linux. Open a terminal and log into the new Ubuntu-Server using SSH. We will use the same SSH command as last week:

root@kali:~# ssh [opstudent@192.168.19.19](mailto:opstudent@192.168.19.19) –p 22345

First, we will create an exception to accept all traffic that is part of an established connection or is related to an established connection

opstudent@ubuntu:~$ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

This rule uses the conntrack extension, which provides internal tracking so that iptables has the context it needs to evaluate packets as part of larger connections instead of as a stream of discrete, unrelated packets. TCP is a connection-based protocol, so an established connection is fairly well-defined.

To allow SSH connections only from 192.168.19.11 (our Kali Linux machine) run the following command:

opstudent@ubuntu:~$ sudo iptables -A INPUT -p tcp -m state --state NEW --source 192.168.19.11 --dport 22345 -j ACCEPT

Now, disable SSH connection from all other hosts by running the following command:

opstudent@ubuntu:~$ sudo iptables -A INPUT -p tcp --dport 22345 -j DROP

**Q8.** Briefly explain (using your own words) what the previous two firewall rules achieve.

The first rule allows the Kali Linux machine (192.168.19.11) to have access to the port (22345).

The second rule is saying to drop all other machines that are trying to access the same port (22345).

These two rules improve security on the new SSH port as the firewall is only allowing one machine to access it while dropping all other machines that try to gain access to that machine from that port.

Last step! Saving iptables rules. At this point, you should test your firewall rules and make sure they cover the block the traffic you want to keep out while not hindering your normal access. Once you are satisfied that your rules are behaving correctly, you can save them so that they will be automatically be applied to your system at boot.

opstudent@ubuntu:~$ sudo sh –c “iptables-save > /etc/iptables/rules.v4”

This will overwrite your /etc/iptables/rules.v4 and /etc/iptables/rules.v6 files with the policies you crafted on the command line. Check that the changes have been saved, they should appear when viewing the following file:

opstudent@ubuntu:~$ cat /etc/iptables/rules.v4

All over, well done! :)

**APPENDIX 1: ENTIRE SSH\_CONFIF FILE:**

# Package generated configuration file

# See the sshd\_config(5) manpage for details

# What ports, IPs and protocols we listen for

Port 22

# Use these options to restrict which interfaces/protocols sshd will bind to

#ListenAddress ::

#ListenAddress 0.0.0.0

Protocol 2

# HostKeys for protocol version 2

HostKey /etc/ssh/ssh\_host\_rsa\_key

HostKey /etc/ssh/ssh\_host\_dsa\_key

HostKey /etc/ssh/ssh\_host\_ecdsa\_key

HostKey /etc/ssh/ssh\_host\_ed25519\_key

#Privilege Separation is turned on for security

UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key

KeyRegenerationInterval 3600

HostKey /etc/ssh/ssh\_host\_rsa\_key

HostKey /etc/ssh/ssh\_host\_dsa\_key

HostKey /etc/ssh/ssh\_host\_ecdsa\_key

HostKey /etc/ssh/ssh\_host\_ed25519\_key

#Privilege Separation is turned on for security

UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key

KeyRegenerationInterval 3600

ServerKeyBits 1024

# Logging

SyslogFacility AUTH

LogLevel INFO

# Authentication:

LoginGraceTime 120

PermitRootLogin prohibit-password

StrictModes yes

RSAAuthentication yes

PubkeyAuthentication yes

#AuthorizedKeysFile %h/.ssh/authorized\_keys

# Don't read the user's ~/.rhosts and ~/.shosts files

IgnoreRhosts yes

# For this to work you will also need host keys in /etc/ssh\_known\_hosts

RhostsRSAAuthentication no

# similar for protocol version 2

ostbasedAuthentication no

# Uncomment if you don't trust ~/.ssh/known\_hosts for RhostsRSAAuthentication

#IgnoreUserKnownHosts yes

# To enable empty passwords, change to yes (NOT RECOMMENDED)

PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with

# some PAM modules and threads)

ChallengeResponseAuthentication no

# Change to no to disable tunnelled clear text passwords

#PasswordAuthentication yes

# Kerberos options

#KerberosAuthentication no

#KerberosGetAFSToken no

#KerberosOrLocalPasswd yes

#KerberosTicketCleanup yes

# GSSAPI options

#GSSAPIAuthentication no

#GSSAPICleanupCredentials yes

X11Forwarding yes

X11DisplayOffset 10

PrintMotd no

PrintLastLog yes

TCPKeepAlive yes

#UseLogin no

#MaxStartups 10:30:60

#Banner /etc/issue.net

# Allow client to pass locale environment variables

AcceptEnv LANG LC\_\*

Subsystem sftp /usr/lib/openssh/sftp-server

Subsystem sftp /usr/lib/openssh/sftp-server

# Set this to 'yes' to enable PAM authentication, account processing,

# and session processing. If this is enabled, PAM authentication will

# be allowed through the ChallengeResponseAuthentication and

# PasswordAuthentication. Depending on your PAM configuration,

# PAM authentication via ChallengeResponseAuthentication may bypass

# the setting of "PermitRootLogin without-password".

# If you just want the PAM account and session checks to run without

# PAM authentication, then enable this but set PasswordAuthentication

# and ChallengeResponseAuthentication to 'no'.

UsePAM yes