Student Name: Azeem Arshad Vasanwala  
Email: avasanwa@asu.edu  
Submission Date: 13 March 2022  
Class Name and Term: CSE468 Spring 2022

Lab 02

# Project Overview

I set up a firewall between a client and server network, and created limited access between the server and the outside world. These firewall rules allowed me to protect the server against unwanted access.

# Network Setup

Diagram

Description automatically generated

# Software

The following services were set up on the server:

* FTP
* SSH
* Apache Web Server

The following services were set up on gateway

* iptables

# Project Description

Server setup:

I set up the apache and ftp server on the server VM. I had to configure additional parameters beyond the FTP lab, adding pasv\_address=<NAT\_FTP\_ADDRESS>, which in my case was 192.168.0.100, and change default parameters of listen=YES and listen\_ipv6=NO.

Gateway setup:

I used the existing gateway machine from the previous lab and added my own firewall rules. The attached ruleset can be found below in Appendix B

# Conclusion

I spent a lot of time on this project and learnt how to use tools like tcpdump to try and understand what was incorrect in my rules. I learnt how to set up a firewall properly, and create proper rules to protect an internal network. I had trouble initially figuring out how to route packets destined for the server directly by the client away, and had trouble figuring out how to get the ftp client to connect to the right IP address in passive mode. I got assistance from Dr Deng in figuring these things out.

# Appendix B: Attached files

Video demo:

<https://nus-sg.zoom.us/rec/share/pMIueaOOoWFFLJIbEawkW3dhpD-Kniu0ZRKI7cK6sMSe20fMSyoj6Vzexgd6NKvn.WeDoHyXDsSI_SoWp>

Access Passcode: p%+38+x!

Rules

#!/bin/sh

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

# #

# rc.firewall - Initial SIMPLE IP Firewall script for Linux and iptables #

# #

# 02/17/2020 Created by Dijiang Huang ASU SNAC Lab #

# #

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

# #

# #

# Configuration options, these will speed you up getting this script to #

# work with your own setup. #

# #

# your LAN's IP range and localhost IP. /24 means to only use the first 24 #

# bits of the 32 bit IP address. the same as netmask 255.255.255.0 #

# #

# #

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

#

# 1. Configuration options.

# NOTE that you need to change the configuration based on your own network setup.

# The defined alias and variables allow you to manage and update the entire

# configurations easily, and more readable :-)

#

# Lab Network Topology

#

# --------- ---------- ---------

# |Client |\_\_Client\_NET\_\_|Gateway |\_\_Server\_NET\_\_|Server |

# --------- ---------- ---------

# |

# |Internet

# | \_\_\_\_\_\_\_\_

# ---------- / \

# |External|\_\_\_\_\_\_\_\_|Internet|

# |Router | \\_\_\_\_\_\_\_\_/

# ----------

#

####

# 1.1. Internet ip address

#

#

Internet\_IP="172.16.165.128"

Internet\_IP\_RANGE="172.16.165.0/24"

Internet\_BCAST\_ADRESS="172.16.165.255"

Internet\_IFACE="ens224"

####

# 1.2 Client network configuration.

#

#

#

# IP addresses of the client-side network

#

Client\_NET\_IP="192.168.0.10"

Client\_NET\_IP\_RANGE="192.168.0.0/24"

Client\_NET\_BCAST\_ADRESS="192.168.0.255"

Client\_NET\_IFACE="ens160"

####

# 1.3 Server Network Configuration.

#

#

# IP addresses of the server-side network

#

Server\_NET\_IP="10.0.0.10"

Server\_NET\_GW="10.0.0.100"

Server\_NET\_IP\_RANGE="10.0.0.0/8"

Server\_NET\_BCAST\_ADRESS="10.255.255.255"

Server\_NET\_IFACE="ens192"

#

# IP aliases for the server (server's IP address)

#

LO\_IFACE="lo"

LO\_IP="127.0.0.1"

ICMP\_IP\_ADDRESS="10.0.0.10"

WEB\_IP\_ADDRESS="10.0.0.10"

MAIL\_IP\_ADDRESS="10.0.0.10"

SSH\_IP\_ADDRESS="10.0.0.10"

DNS\_IP\_ADDRESS="10.0.0.10"

FTP\_IP\_ADDRESS="10.0.0.10"

#IP aliases for NATed services (this is the GW's ip on client network)

NAT\_ICMP\_IP\_ADDRESS="192.168.0.100"

NAT\_WEB\_IP\_ADDRESS="192.168.0.100"

NAT\_MAIL\_IP\_ADDRESS="192.168.0.100"

NAT\_SSH\_IP\_ADDRESS="192.168.0.100"

NAT\_DNS\_IP\_ADDRESS="192.168.0.100"

NAT\_FTP\_IP\_ADDRESS="192.168.0.100"

PASV\_FTP\_PORT\_RANGE="30000:30099"

####

# 1.4 IPTables Configuration.

#

IPTABLES="/sbin/iptables"

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

# #

# 2. Module loading. #

# #

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

#

# Needed to initially load modules

#

/sbin/depmod -a

#

# flush iptables

#

$IPTABLES -F

$IPTABLES -X

$IPTABLES -F -t nat

#####

# 2.1 Required modules

#

/sbin/modprobe ip\_tables

/sbin/modprobe ip\_conntrack

/sbin/modprobe iptable\_filter

/sbin/modprobe iptable\_mangle

/sbin/modprobe iptable\_nat

/sbin/modprobe ipt\_LOG

/sbin/modprobe ipt\_limit

/sbin/modprobe ipt\_state

#####

# 2.2 Non-frequently used modules

#

#/sbin/modprobe ipt\_owner

#/sbin/modprobe ipt\_REJECT

#/sbin/modprobe ipt\_MASQUERADE

#/sbin/modprobe ip\_conntrack\_ftp

#/sbin/modprobe ip\_conntrack\_irc

#/sbin/modprobe ip\_nat\_ftp

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

#

# 3. /proc set up.

#

#

# 3.1 Required proc configuration

#

#

# Enable ip\_forward, this is critical since it is turned off as defaul in

# Linux.

#

echo "1" > /proc/sys/net/ipv4/ip\_forward

#

# 3.2 Non-Required proc configuration

#

#

# Dynamic IP users:

#

#echo "1" > /proc/sys/net/ipv4/ip\_dynaddr

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

#

# 4. rules set up.

#

# The kernel starts with three lists of rules; these lists are called firewall

# chains or just chains. The three chains are called INPUT, OUTPUT and FORWARD.

#

# The chains are arranged like so:

#

# \_\_\_\_\_

# / \

# -->[Routing ]--->|FORWARD|------->

# [Decision] \\_\_\_\_\_/ ^

# | |

# v \_\_\_\_

# \_\_\_ / \

# / \ |OUTPUT|

# |INPUT| \\_\_\_\_/

# \\_\_\_/ ^

# | |

# ----> Local Process ----

#

# 1. When a packet comes in (say, through the Ethernet card) the kernel first

# looks at the destination of the packet: this is called `routing'.

# 2. If it's destined for this box, the packet passes downwards in the diagram,

# to the INPUT chain. If it passes this, any processes waiting for that

# packet will receive it.

# 3. Otherwise, if the kernel does not have forwarding enabled, or it doesn't

# know how to forward the packet, the packet is dropped. If forwarding is

# enabled, and the packet is destined for another network interface (if you

# have another one), then the packet goes rightwards on our diagram to the

# FORWARD chain. If it is ACCEPTed, it will be sent out.

# 4. Finally, a program running on the box can send network packets. These

# packets pass through the OUTPUT chain immediately: if it says ACCEPT, then

# the packet continues out to whatever interface it is destined for.

#

#####

# 4.1 Filter table

#

#

# 4.1.1 Set policies

#

#

# Set default policies for the INPUT, FORWARD and OUTPUT chains

#

# Whitelist (Whitelist is preferred)

$IPTABLES -P INPUT DROP

$IPTABLES -P OUTPUT DROP

$IPTABLES -P FORWARD DROP

# Blacklist

# $IPTABLES -P INPUT ACCEPT

# $IPTABLES -P OUTPUT ACCEPT

# $IPTABLES -P FORWARD ACCEPT

#

# 4.1.2 Create user-specified chains

#

#

# Example of creating a chain for bad tcp packets

#

#$IPTABLES -N bad\_tcp\_packets

#

# Create separate chains for allowed (whitelist), ICMP, TCP and UDP to traverse

#

#$IPTABLES -N allowed

#$IPTABLES -N tcp\_packets

#$IPTABLES -N udp\_packets

#$IPTABLES -N icmp\_packets

#

# In the following 4.1.x, you can provide individual user-specified rules

#

# 4.1.3 Example of create content in user-specified chains (bad\_tcp\_packets)

#

#

# bad\_tcp\_packets chain

#

#$IPTABLES -A bad\_tcp\_packets -p tcp --tcp-flags SYN,ACK SYN,ACK -m state --state NEW -j REJECT --reject-with tcp-reset

#$IPTABLES -A bad\_tcp\_packets -p tcp ! --syn -m state --state NEW -j LOG --log-prefix "New not syn:"

#$IPTABLES -A bad\_tcp\_packets -p tcp ! --syn -m state --state NEW -j DROP

#

# 4.1.4 Example of allowed chain (allow packets for initial TCP or already established TCP sessions)

#

#$IPTABLES -A allowed -p TCP --syn -j ACCEPT

#$IPTABLES -A allowed -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT

#$IPTABLES -A allowed -p TCP -j DROP

#####

# 4.2 FORWARD chain

#

#

# Provide your forwarding rules below

#

# example of checking bad tcp packets

#$IPTABLES -A FORWARD -p tcp -j bad\_tcp\_packets

# Allow http traffic from client network to server network

#$IPTABLES -A FORWARD -p TCP --dport 80 -i $Client\_NET\_IFACE -s $Client\_NET\_IP\_RANGE -d $WEB\_IP\_ADDRESS -j ACCEPT

#$IPTABLES -A FORWARD -p TCP --sport 80 -i $Server\_NET\_IFACE -s $WEB\_IP\_ADDRESS -d $Client\_NET\_IP\_RANGE -j ACCEPT

# example of using allowed

#$IPTABLES -A FORWARD -p tcp -j allowed

# Allow ICMP Client -> Server

$IPTABLES -A FORWARD -p icmp -i $Client\_NET\_IFACE -s $Client\_NET\_IP -d $ICMP\_IP\_ADDRESS -j ACCEPT

# Allow ICMP Server -> Client

$IPTABLES -A FORWARD -p icmp -i $Server\_NET\_IFACE -s $ICMP\_IP\_ADDRESS -d $Client\_NET\_IP -j ACCEPT

# Consider allowing only established/related connections outgoing

# Allow Web Client -> Server

$IPTABLES -A FORWARD -p TCP --dport 80 -i $Client\_NET\_IFACE -s $Client\_NET\_IP\_RANGE -d $WEB\_IP\_ADDRESS -j ACCEPT

# Allow Web Server -> Client

$IPTABLES -A FORWARD -p TCP --sport 80 -i $Server\_NET\_IFACE -s $WEB\_IP\_ADDRESS -d $Client\_NET\_IP\_RANGE -j ACCEPT

# Allow FTP Client -> Server - Cmd

$IPTABLES -A FORWARD -p TCP --dport 20:21 -i $Client\_NET\_IFACE -s $Client\_NET\_IP\_RANGE -d $FTP\_IP\_ADDRESS -j ACCEPT

# Allow FTP Client -> Server - Negotiated port

$IPTABLES -A FORWARD -p TCP --dport $PASV\_FTP\_PORT\_RANGE -i $Client\_NET\_IFACE -s $Client\_NET\_IP\_RANGE -d $FTP\_IP\_ADDRESS -j ACCEPT

# Allow FTP Server -> Client - Cmd

$IPTABLES -A FORWARD -p TCP --sport 20:21 -i $Server\_NET\_IFACE -s $FTP\_IP\_ADDRESS -d $Client\_NET\_IP\_RANGE -j ACCEPT

# Allow FTP Server -> Client - Negotiated port

$IPTABLES -A FORWARD -p TCP --sport $PASV\_FTP\_PORT\_RANGE -i $Server\_NET\_IFACE -s $FTP\_IP\_ADDRESS -d $Client\_NET\_IP\_RANGE -j ACCEPT

# Allow SSH Client -> Server

$IPTABLES -A FORWARD -p TCP --dport 22 -i $Client\_NET\_IFACE -s $Client\_NET\_IP\_RANGE -d $SSH\_IP\_ADDRESS -j ACCEPT

# Allow SSH Server -> Client

$IPTABLES -A FORWARD -p TCP --sport 22 -i $Server\_NET\_IFACE -s $SSH\_IP\_ADDRESS -d $Client\_NET\_IP\_RANGE -j ACCEPT

# Allow Client -> Internet

$IPTABLES -A FORWARD -p all -i $Client\_NET\_IFACE -s $Client\_NET\_IP\_RANGE -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

# Allow Internet -> Client

$IPTABLES -A FORWARD -p all -i $Internet\_IFACE -d $Client\_NET\_IP\_RANGE -m state --state ESTABLISHED,RELATED -j ACCEPT

# Allow Server -> Internet

$IPTABLES -A FORWARD -p all -i $Server\_NET\_IFACE -s $Server\_NET\_IP -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

# Allow Internet -> Server

$IPTABLES -A FORWARD -p all -i $Internet\_IFACE -d $Server\_NET\_IP\_RANGE -m state --state ESTABLISHED,RELATED -j ACCEPT

# Block anything going to Internet IP

# $IPTABLES -A FORWARD -p all -d $Internet\_IP -j DROP

#####

# 4.3 INPUT chain

#

#

# Provide your input rules below

#

#

# Example of checking bad TCP packets we don't want.

#

#$IPTABLES -A INPUT -p tcp -j bad\_tcp\_packets

$IPTABLES -A INPUT -p icmp -i $Client\_NET\_IFACE -m state --state ESTABLISHED,RELATED -j ACCEPT

$IPTABLES -A INPUT -p icmp -i $Server\_NET\_IFACE -m state --state ESTABLISHED,RELATED -j ACCEPT

#####

# 4.3 OUTPUT chain

#

#

# Provide your output rules below

#

#

# Example of checking bad TCP packets we don't want.

#

#$IPTABLES -A OUTPUT -p tcp -j bad\_tcp\_packets

$IPTABLES -A OUTPUT -p icmp -o $Client\_NET\_IFACE -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

$IPTABLES -A OUTPUT -p icmp -o $Server\_NET\_IFACE -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

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

# #

# 5. NAT setup #

# #

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

#####

# 5.1 PREROUTING chain.

#

#

# Provide your NAT PREROUTING rules (packets come into your private domain)

#

#

# Example of enable http to internal web server behind the firewall (port forwarding)

#

# web

#$IPTABLES -t nat -A PREROUTING -p tcp -d $NAT\_WEB\_IP\_ADDRESS --dport 80 -j DNAT --to $WEB\_IP\_ADDRESS

# Any situation where client directly contacts server or server directly contacts client is routed out of the network

$IPTABLES -t nat -A PREROUTING -p all -i $Client\_NET\_IFACE -d $Server\_NET\_IP -j DNAT --to $Internet\_IP

$IPTABLES -t nat -A PREROUTING -p all -i $Server\_NET\_IFACE -d $Client\_NET\_IP -j DNAT --to $Internet\_IP

# Allow ICMP Client -> Server

$IPTABLES -t nat -A PREROUTING -p icmp -d $NAT\_ICMP\_IP\_ADDRESS -j DNAT --to $ICMP\_IP\_ADDRESS

# Allow ICMP Server -> Client

$IPTABLES -t nat -A PREROUTING -p icmp -d $Server\_NET\_GW -j DNAT --to $Client\_NET\_IP

# Allow SSH Client -> Server

$IPTABLES -t nat -A PREROUTING -p tcp -d $NAT\_SSH\_IP\_ADDRESS --dport 22 -j DNAT --to $SSH\_IP\_ADDRESS

# Allow FTP Client -> Server - Cmd

$IPTABLES -t nat -A PREROUTING -p tcp -d $NAT\_FTP\_IP\_ADDRESS --dport 20:21 -j DNAT --to $FTP\_IP\_ADDRESS

# Allow FTP Client -> Server - Data

$IPTABLES -t nat -A PREROUTING -p tcp -d $NAT\_FTP\_IP\_ADDRESS --dport $PASV\_FTP\_PORT\_RANGE -j DNAT --to $FTP\_IP\_ADDRESS

# Allow HTTP Client -> Server

$IPTABLES -t nat -A PREROUTING -p tcp -d $NAT\_WEB\_IP\_ADDRESS --dport 80 -j DNAT --to $WEB\_IP\_ADDRESS

#####

# 5.2 POSTROUTING chain.

#

#

# Provide your NAT PREROUTING rules (packets go to the public domain)

#

# Allow the web server reply to the client go through the NAT

#$IPTABLES -t nat -A POSTROUTING -p tcp -s $WEB\_IP\_ADDRESS -o $Client\_NET\_IFACE --sport 80 -j SNAT --to $Client\_NET\_IP

# Allow client and server network nodes to access to Internet using masquerade

$IPTABLES -t nat -A POSTROUTING -o $Internet\_IFACE -j MASQUERADE

# Allow ICMP Server -> Client

$IPTABLES -t nat -A POSTROUTING -p icmp -s $ICMP\_IP\_ADDRESS -o $Client\_NET\_IFACE -j SNAT --to-source $NAT\_ICMP\_IP\_ADDRESS

# Allow ICMP Client -> Server

$IPTABLES -t nat -A POSTROUTING -p icmp -s $Client\_NET\_IP -o $Server\_NET\_IFACE -j SNAT --to-source $Server\_NET\_GW

# Allow SSH Server -> Client

$IPTABLES -t nat -A POSTROUTING -p tcp -s $SSH\_IP\_ADDRESS -o $Client\_NET\_IFACE --sport 22 -j SNAT --to-source $NAT\_SSH\_IP\_ADDRESS

# Allow FTP Server -> Client - Cmd

$IPTABLES -t nat -A POSTROUTING -p tcp -s $FTP\_IP\_ADDRESS -o $Client\_NET\_IFACE --sport 20:21 -j SNAT --to-source $NAT\_FTP\_IP\_ADDRESS

# Allow FTP Server -> Client - Data

$IPTABLES -t nat -A POSTROUTING -p tcp -s $FTP\_IP\_ADDRESS -o $Client\_NET\_IFACE --sport $PASV\_FTP\_PORT\_RANGE -j SNAT --to-source $NAT\_FTP\_IP\_ADDRESS

# Allow Web Server -> Client

$IPTABLES -t nat -A POSTROUTING -p tcp -s $WEB\_IP\_ADDRESS -o $Client\_NET\_IFACE --sport 80 -j SNAT --to-source $NAT\_WEB\_IP\_ADDRESS

# References