Student Name: Gus (August) Fowler

Email: amfowle3@asu.edu  
Submission Date: 3/15/22  
Class Name and Term: CSE468 Spring 2022

Project 2

# Project Overview

I recreated the 3 VM setup from Project 1 on my hypervisor, and then wrote firewall rules using the given rc.firewall as a starter. The only thing I did add is a serial port to each vm, and modified /etc/default/grub to allow for login over serial, so that I could SSH into the physical host and get a shell for each vm that way. Made my life significantly easier.

I have two recording links- one is initial demo, in the second I show that my ftp mkdir works. There is a timestamp past the 10 minutes of waiting in the description of the initial demo.

Initial Demo - <https://youtu.be/VJ1TVLEXUM8>

FTP Mkdir demo - https://youtu.be/duQykRZzULs

# Network Setup

See project 1 submission

# Software

I used Proxmox VE, vsftpd, Openssh Server, Apache2, iptables, and grub.

# Project Description

I used the preexisting rules in the rc.firewall to get an idea of what needed to be done to configure the firewall correctly. I then created an rc.default, which would reset iptables to Project 1 conditions, for debugging. Since I had a serial connection to each VM, I could utilize copy and paste, so I would do my editing in VS Code locally, and then echo “” > rc.firewall; nano rc.firewall, paste and run script. I started by getting port 80 to forward from the gateway to server, after that it was fairly easy to get the rest of the services figured out. There was some ftp configuration hiccups, but google helped out with that.

# Conclusion

At the application level, I actually ended up leaving ftp pretty insecure, but seeing as this project was more focused on the firewall, I wasn’t too concerned. Also if I commented out one unneeded it would remove that vulnerability. Other than that, I conclude that small quality of life setups can make development significantly easier.

# Appendix B: Attached files

~/rc.firewall

#!/bin/sh

# $SYSTEMCTL="/bin/systemctl"

# $SYSTEMCTL disable apache2.service

# $SYSTEMCTL disbale sshd

# $SYSTEMCTL disable ftpd

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

# #

# 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="10.0.0.186"

Internet\_IP\_RANGE="10.0.0.0/24"

Internet\_BCAST\_ADRESS="10.0.0.255"

Internet\_IFACE="ens18"

####

# 1.2 Client network configuration.

#

#

#

# IP addresses of the client-side network

#

Client\_NET\_IP="192.168.0.100"

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

Client\_NET\_BCAST\_ADRESS="192.168.0.255"

Client\_NET\_IFACE="ens19"

####

# 1.3 Server Network Configuration.

#

#

# IP addresses of the server-side network

#

Server\_NET\_IP="16.0.0.100"

Server\_NET\_IP\_RANGE="16.0.0.0/8"

Server\_NET\_BCAST\_ADRESS="16.255.255.255"

Server\_NET\_IFACE="ens20"

Server\_IP="16.0.0.10"

#

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

#

LO\_IFACE="lo"

LO\_IP="127.0.0.1"

WEB\_IP\_ADDRESS=$Server\_IP

MAIL\_IP\_ADDRESS=$Server\_IP

SSH\_IP\_ADDRESS=$Server\_IP

DNS\_IP\_ADDRESS=$Server\_IP

FTP\_IP\_ADDRESS=$Server\_IP

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

NAT\_WEB\_IP\_ADDRESS=$Client\_NET\_IP

NAT\_MAIL\_IP\_ADDRESS=$Client\_NET\_IP

NAT\_SSH\_IP\_ADDRESS=$Client\_NET\_IP

NAT\_DNS\_IP\_ADDRESS=$Client\_NET\_IP

NAT\_FTP\_IP\_ADDRESS=$Client\_NET\_IP

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 default 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

$IPTABLES -N bad\_tcp\_packets

#

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

##ICMP rules for ping from server to client and reply back

$IPTABLES -A icmp\_packets -p icmp --icmp-type echo-request -s $Server\_IP -j ACCEPT

$IPTABLES -A icmp\_packets -p icmp --icmp-type echo-reply -s $Client\_NET\_IP\_RANGE -j ACCEPT

$IPTABLES -A icmp\_packets -p icmp --icmp-type echo-reply -s 16.0.0.100 -j ACCEPT

#

# 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

$IPTABLES -A FORWARD -p icmp -j icmp\_packets

#####

# 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 tcp -j allowed

$IPTABLES -A INPUT -p icmp -j icmp\_packets

#####

# 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 tcp -j allowed

$IPTABLES -A OUTPUT -p icmp -j icmp\_packets

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

# #

# 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

# ssh

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

# ftp

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

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

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

#####

# 5.2 POSTROUTING chain.

#

#

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

#

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

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

# 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

# ssh

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

# ftp

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

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

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

~/rc.default

#!/bin/sh

IPTABLES="/sbin/iptables"

#Flushes IP tables

$IPTABLES -F

$IPTABLES -X

$IPTABLES -F -t nat

#Blacklist

$IPTABLES -P INPUT ACCEPT

$IPTABLES -P OUTPUT ACCEPT

$IPTABLES -P FORWARD ACCEPT

#Allow internet access via nat postrouting masquerade

$IPTABLES -t nat -A POSTROUTING -o ens18 -j MASQUERADE

~/client-test.sh

#!/bin/bash

nmap -sT --max-retries 0 -p- 192.168.0.100

nmap -sU --max-retries 0 -p- 192.168.0.100

ping -c 5 -i 0.1 8.8.8.8

ping -c 5 -i 0.1 192.168.0.100

ping -c 5 -i 0.1 16.0.0.10

~/server-test.sh

#!/bin/bash

nmap -sT --max-retries 0 -p- 16.0.0.100

nmap -sU --max-retries 0 -p- 16.0.0.100

ping -c 5 -i 0.1 8.8.8.8

ping -c 5 -i 0.1 16.0.0.100

ping -c 5 -i 0.1 192.168.0.10

~/gateway-test.sh

#!/bin/bash

ping -c 5 8.8.8.8

ping -c 5 192.168.0.10

ping -c 5 16.0.0.10

ubuntu@ubuntu:~$ cat /var/www/html/index.html

Welcome! This is the server! -Gus Fowler

ubuntu@ubuntu:~$ cat /etc/vsftpd.conf

# Example config file /etc/vsftpd.conf

#

# The default compiled in settings are fairly paranoid. This sample file

# loosens things up a bit, to make the ftp daemon more usable.

# Please see vsftpd.conf.5 for all compiled in defaults.

#

# READ THIS: This example file is NOT an exhaustive list of vsftpd options.

# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's

# capabilities.

#

#

# Run standalone? vsftpd can run either from an inetd or as a standalone

# daemon started from an initscript.

listen=YES

#

# This directive enables listening on IPv6 sockets. By default, listening

# on the IPv6 "any" address (::) will accept connections from both IPv6

# and IPv4 clients. It is not necessary to listen on \*both\* IPv4 and IPv6

# sockets. If you want that (perhaps because you want to listen on specific

# addresses) then you must run two copies of vsftpd with two configuration

# files.

listen\_ipv6=NO

#

# Allow anonymous FTP? (Disabled by default).

anonymous\_enable=YES

#

# Uncomment this to allow local users to log in.

local\_enable=YES

#

# Uncomment this to enable any form of FTP write command.

write\_enable=YES

#

# Default umask for local users is 077. You may wish to change this to 022,

# if your users expect that (022 is used by most other ftpd's)

#local\_umask=022

#

# Uncomment this to allow the anonymous FTP user to upload files. This only

# has an effect if the above global write enable is activated. Also, you will

# obviously need to create a directory writable by the FTP user.

anon\_upload\_enable=YES

#

# Uncomment this if you want the anonymous FTP user to be able to create

# new directories.

anon\_mkdir\_write\_enable=YES

#

# Activate directory messages - messages given to remote users when they

# go into a certain directory.

dirmessage\_enable=YES

#

# If enabled, vsftpd will display directory listings with the time

# in your local time zone. The default is to display GMT. The

# times returned by the MDTM FTP command are also affected by this

# option.

use\_localtime=YES

#

# Activate logging of uploads/downloads.

xferlog\_enable=YES

#

# Make sure PORT transfer connections originate from port 20 (ftp-data).

connect\_from\_port\_20=YES

#

# If you want, you can arrange for uploaded anonymous files to be owned by

# a different user. Note! Using "root" for uploaded files is not

# recommended!

#chown\_uploads=YES

#chown\_username=whoever

#

# You may override where the log file goes if you like. The default is shown

# below.

#xferlog\_file=/var/log/vsftpd.log

#

# If you want, you can have your log file in standard ftpd xferlog format.

# Note that the default log file location is /var/log/xferlog in this case.

#xferlog\_std\_format=YES

#

# You may change the default value for timing out an idle session.

#idle\_session\_timeout=600

#

# You may change the default value for timing out a data connection.

#data\_connection\_timeout=120

#

# It is recommended that you define on your system a unique user which the

# ftp server can use as a totally isolated and unprivileged user.

#nopriv\_user=ftpsecure

#

# Enable this and the server will recognise asynchronous ABOR requests. Not

# recommended for security (the code is non-trivial). Not enabling it,

# however, may confuse older FTP clients.

#async\_abor\_enable=YES

#

# By default the server will pretend to allow ASCII mode but in fact ignore

# the request. Turn on the below options to have the server actually do ASCII

# mangling on files when in ASCII mode.

# Beware that on some FTP servers, ASCII support allows a denial of service

# attack (DoS) via the command "SIZE /big/file" in ASCII mode. vsftpd

# predicted this attack and has always been safe, reporting the size of the

# raw file.

# ASCII mangling is a horrible feature of the protocol.

#ascii\_upload\_enable=YES

#ascii\_download\_enable=YES

#

# You may fully customise the login banner string:

#ftpd\_banner=Welcome to blah FTP service.

#

# You may specify a file of disallowed anonymous e-mail addresses. Apparently

# useful for combatting certain DoS attacks.

#deny\_email\_enable=YES

# (default follows)

#banned\_email\_file=/etc/vsftpd.banned\_emails

#

# You may restrict local users to their home directories. See the FAQ for

# the possible risks in this before using chroot\_local\_user or

# chroot\_list\_enable below.

#chroot\_local\_user=YES

#

# You may specify an explicit list of local users to chroot() to their home

# directory. If chroot\_local\_user is YES, then this list becomes a list of

# users to NOT chroot().

# (Warning! chroot'ing can be very dangerous. If using chroot, make sure that

# the user does not have write access to the top level directory within the

# chroot)

#chroot\_local\_user=YES

#chroot\_list\_enable=YES

# (default follows)

#chroot\_list\_file=/etc/vsftpd.chroot\_list

#

# You may activate the "-R" option to the builtin ls. This is disabled by

# default to avoid remote users being able to cause excessive I/O on large

# sites. However, some broken FTP clients such as "ncftp" and "mirror" assume

# the presence of the "-R" option, so there is a strong case for enabling it.

#ls\_recurse\_enable=YES

#

# Customization

#

# Some of vsftpd's settings don't fit the filesystem layout by

# default.

#

# This option should be the name of a directory which is empty. Also, the

# directory should not be writable by the ftp user. This directory is used

# as a secure chroot() jail at times vsftpd does not require filesystem

# access.

secure\_chroot\_dir=/var/run/vsftpd/empty

#

# This string is the name of the PAM service vsftpd will use.

pam\_service\_name=vsftpd

#

# This option specifies the location of the RSA certificate to use for SSL

# encrypted connections.

rsa\_cert\_file=/etc/ssl/certs/ssl-cert-snakeoil.pem

rsa\_private\_key\_file=/etc/ssl/private/ssl-cert-snakeoil.key

ssl\_enable=NO

#

# Uncomment this to indicate that vsftpd use a utf8 filesystem.

#utf8\_filesystem=YES

pasv\_address=192.168.0.100

allow\_writeable\_chroot=YES

ubuntu@ubuntu:~$ cat /etc/default/grub

# If you change this file, run 'update-grub' afterwards to update

# /boot/grub/grub.cfg.

# For full documentation of the options in this file, see:

# info -f grub -n 'Simple configuration'

GRUB\_DEFAULT=0

GRUB\_TIMEOUT\_STYLE=hidden

GRUB\_TIMEOUT=0

GRUB\_DISTRIBUTOR=`lsb\_release -i -s 2> /dev/null || echo Debian`

GRUB\_CMDLINE\_LINUX\_DEFAULT="maybe-ubiquity"

GRUB\_CMDLINE\_LINUX="console=tty1 console=ttyS0,115200"

# Uncomment to enable BadRAM filtering, modify to suit your needs

# This works with Linux (no patch required) and with any kernel that obtains

# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)

#GRUB\_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)

GRUB\_TERMINAL="console serial"

GRUB\_SERIAL\_COMMAND="serial --speed=115200 --unit=0 --word=8 --parity=no --stop=1"

# The resolution used on graphical terminal

# note that you can use only modes which your graphic card supports via VBE

# you can see them in real GRUB with the command `vbeinfo'

#GRUB\_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux

#GRUB\_DISABLE\_LINUX\_UUID=true

# Uncomment to disable generation of recovery mode menu entries

#GRUB\_DISABLE\_RECOVERY="true"

# Uncomment to get a beep at grub start

#GRUB\_INIT\_TUNE="480 440 1"

# References

Reference is optional, but nice to have to allow others to read your report with additional linked source for validation and learning.

1. Wireshark, available at <https://www.wireshark.org/>, accessed by 8/31/2018.
2. Postel, Jon. "RFC 791: Internet protocol." (1981).