Forcepoint NGFW Administrator Course Lab Configuration Guide

Matt McKinley

May 2, 2017

Contents

List of Figures

1 Introduction

This document is designed to guide an instructor through the logic and function of the NGFW VILT Administrator course. The labs were designed to provide a more realistic experience for students by presenting scenarios that actual administrators would encounter while working with the Forcepoint NGFW or any firewall. To that end, the labs incorporate the generation of attack-laden traffic, a functioning external LDAP database, a non-Forcepoint firewall for use in the VPN section, and use of the SMC client that more closely resembles how it would be used by an administrator.

Understanding how the different components in the labs work is crucial to being able to fix a problem should it arise. The most complex aspect of these labs is the interaction of the “pitcher“ and “catcher“ machines, the two that are responsible for generating traffic for use in logging labs and deep inspection. The second aspect to understand is what services are in place and their respective functions in the labs. By knowing which processes are responsible for which labs will enable you to restart the appropriate services should the need arise.

The last part of this document will cover some of the specific changes that were made to the Linux VMs that enables the proper functioning of the labs. Also covered in the last section will be a complete documentation of the scripts. While this is not completely necessary to successfully teach the course, it will, nonetheless, provide a record of the scripts for use in future courses.

2 Lab Design

The main goal of these labs is to make them more realistic compared to some of the other NGFW courses. To that end, although simple, the design of the labs incorporates several elements that have not been present before:

1. The use of locations
2. The lack of static routing. NATing and routing is used as it would be under real conditions
3. The use of a fully configured name server
4. The use of external user storage (OpenLDAP)
5. The use of a DMZ
6. The use of a 3^{rd} party firewall for the VPN lab
7. The use of an extensible traffic generation and receiving architecture for generating interesting logs

2.1 The Virtual Machines

This section details each VM in the environment and their functions in the lab environment:

1. **HQ Firewall**
   - Firewall between the HQ location and simulated internet. In the beginning of the labs, it is preconfigured with a policy to allow traffic to the **Receiver** on the DMZ
   - NAT IP address in use:
     - 212.20.0.101 de-nats to 172.31.200.101
     - 212.20.0.150 de-nats to 192.168.0.150 (DMZ)
     - 212.20.0.60 is the dynamic source NAT ip for the 172.31.200.0/24 network
     - 172.31.200.1 is the default gateway for the 172.31.200.0/24 network
   - Uses a customized management template that should not be modified
2. **HQ SMC**
   - Runs the DNS server (named)
   - Runs OpenLDAP
   - Runs the management and log servers
   - IP Address: 172.31.200.101
3. **HQ Workstation**
   - A Windows 7 workstation that students use to administer the NGFW environment
   - Has shortcuts on the desktop used to start and stop the generation of network traffic
4. **Partner Firewall**

- Based on an open source firewall called IPFire
- Serves as the termination point for the site-to-site VPN lab
- NATs traffic for the Partner Internal Server, which is behind this firewall
- External IP: 130.207.200.79

5. **Partner Internal Server**
    - Resides behind the Partner Firewall and is used for testing connections
    - Used in the VPN labs for testing connections through the VPN
    - Can be used to test locations, but this is a future use
    - Internal IP: 10.100.100.50
6. **Receiver**
    - A small linux server that resides on the DMZ of the HQ Firewall
    - Responsible for running netcat to receive traffic for testing the logging capabilities of the firewall
    - The script that it runs is documented below, *start-catcher.sh*
    - Runs a web server to test connectivity from external to internal using NAT
    - Internal IP: 192.168.0.150
    - External IP: 212.20.0.150
7. **TrafficGen**
    - This is a small Linux server that generates traffic that is sent to the Receiver
    - The script that generates the traffic is documented below, *Pitcher.sh*
    - Runs a web server for testing external connectivity through the firewall
    - External IP: 87.35.16.200

2.2 Important Services

The most important VM in the environment is the **HQ SMC**. It is responsible for the following tasks:

- DNS Server
- OpenLDAP
- Management and Log Servers

Each of these services has a specific purpose for the labs, and should a problem arise, there are a few things that could be checked. First, let's look at how to restart each service.

1. To restart **named**, the DNS server: ***systemctl restart named***
2. To restart **OpenLDAP**, ***systemctl restart slapcat***
3. To restart the management server, ***/etc/init.d/sgMgtServer restart***

4. To restart the log server, ***/etc/init.d/sgLogServer restart***

Under normal operating circumstances, these services should not need to be restarted unless something bad happens. The DNS server is critical because all of the VM traffic relies on name resolution.

The **Receiver** and **TrafficGen** VMs also are quite important to the function of the labs. In addition to running the scripts that generate and receive traffic for logging purposes, they also run web servers for testing. In the event that a student tests something and they do not get the expected result, there are a series of commands that could be run to resolve the problems. The following commands will work on both VMs:

- To restart the web server: ***/etc/init.d/httpd restart***
- To restart networking: ***/etc/init.d/network restart***

2.3 Configuration Files

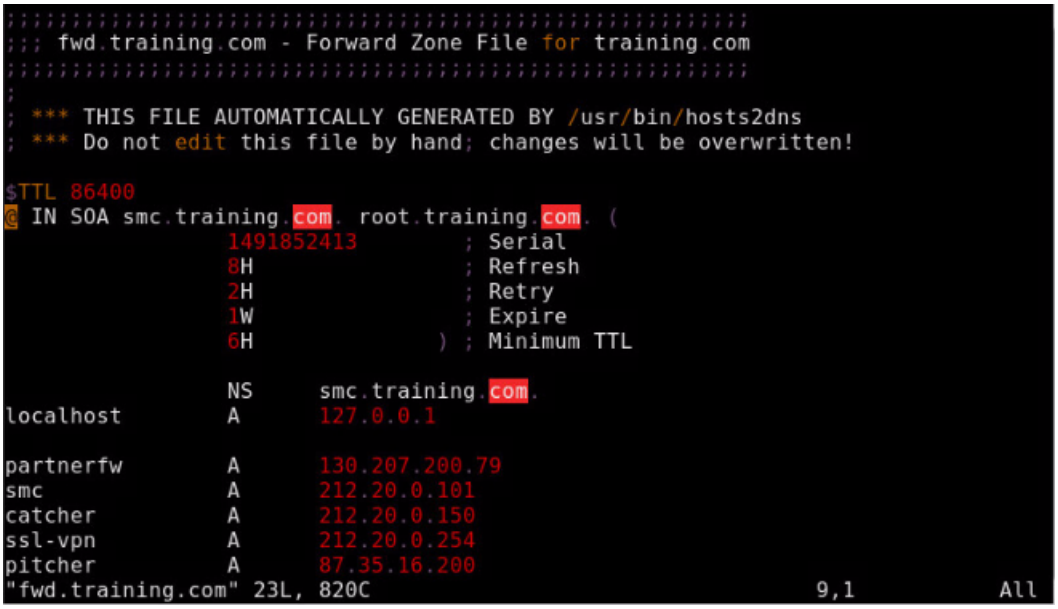
These are the most important configuration files in the lab environment and are included here for backup purposes as well as to lend a general understanding of how the lab environment works and is configured.

HQ SMC: */etc/hosts*

```
127.0.0.1 training training.com localhost4 localhost4.localdomain4
::1       localhost localhost.localdomain localhost6 localhost6.localdomain6
172.31.200.101 smc smc.training.com
172.31.200.60  wrk1 wrk1.training.com

#!DNS: DOMAIN training.com
#!DNS:START
212.20.0.101    smc
130.207.200.79  partnerfw
87.35.16.200    pitcher
212.20.0.150    receiver
212.20.0.150    catcher
212.20.0.254    ssl-vpn
#!DNS:END
```

HQ SMC: */var/named/fwd.training.com*



```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; fwd.training.com - Forward Zone File for training.com
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; *** THIS FILE AUTOMATICALLY GENERATED BY /usr/bin/hosts2dns
; *** Do not edit this file by hand; changes will be overwritten!

$TTL 86400
@ IN SOA smc.training.com. root.training.com. (
                1491852413        ; Serial
                8H                ; Refresh
                2H                ; Retry
                1W                ; Expire
                6H              ) ; Minimum TTL

                NS      smc.training.com.
localhost       A       127.0.0.1

partnerfw       A       130.207.200.79
smc             A       212.20.0.101
catcher         A       212.20.0.150
ssl-vpn         A       212.20.0.254
pitcher         A       87.35.16.200
"fwd.training.com" 23L, 820C                                  9,1           All
```

Figure 1: Forward Zone File, training.com

HQ SMC: */var/named/rev.training.com*

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; rev.training.com -- Reverse Zone File for training.com
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; *** THIS FILE AUTOMATICALLY GENERATED BY /usr/bin/hosts2dns
; *** Do not edit this file by hand; changes will be overwritten!

$TTL 86400
@ IN SOA smc.training.com. root.training.com. (
        1491852413        ; Serial
        8H                ; Refresh
        2H                ; Retry
        1W                ; Expire
        6H              ) ; Minimum TTL

                NS      smc.training.com.

79      PTR     partnerfw.training.com.
101     PTR     smc.training.com.
150     PTR     catcher.training.com.
254     PTR     ssl-vpn.training.com.
200     PTR     pitcher.training.com.
~
"rev.training.com" 22L, 726C                                  1,1           All
```

Figure 2: Reverse Zone File, training.com

3 Management Template

3.1 Access Rules

These labs accurately simulate what it is actually like to remotely manage a distributed firewall infrastructure. To that end, there is a management template that has been created upon which all student policies must be created. This template contains the following:

1. Non-local networks to Management for HTTP, HTTPS, Ping, SG Control, SG Data Browsing and SSH
2. Management Server to remote firewalls. This is only one remote firewall, the router
3. Remote firewalls to Management, SG Engine to Log, SG Engine to Management, and SG Initial Log Contact
4. Remote firewalls to the HQ Firewall's external IP for ESP and IKE. This is so that the 3^{rd} party firewall can form a VPN
5. Non-local networks to the Receiver's NAT address, 212.20.0.150. This rule ensures that traffic generation will work properly
6. SSH traffic between the HQ Workstation and the DMZ host. This is so that traffic generation can be started and stopped

This template should not be modified, and it is *critical* that the students base their policies on this template. If they do not, the symptoms are clear: traffic generation will not work, the remote firewalls will have "red" as their status, DNS will not work, etc.

Automatic Rules Insert Point					
2	NOT-HQ-NETS	MGMT-NAT	DNS HTTP HTTP proxy HTTPS Ping SG Control SG Control (Monitoring Client) SG Data Browsing SSH	Allow	
3	Receiver	Management Server	DNS	Allow	
4	Management Server	RemoteFirewalls	SG Control SG Management to Firewall	Allow	
5	RemoteFirewalls	MGMT-NAT	SG Engine to Log SG Engine to Management SG Log Initial Contact	Allow	
6	RemoteFirewalls	$$ Local Cluster(NDI addresses only)	ESP ISAKMP (UDP) NAT-T (Destination)	Allow	
7	NOT-HQ-NETS	DMZ-WWW-NAT	ANY	Continue	
8	win-wrkstn	Receiver	SSH	Allow	
HQ-Rules					
Discard all					

Figure 3: Management Template Access Rules

3.2 NAT Rules

There are several NAT rules that are critical to proper communication. The function of these rules is as follows:

1. No-NAT rules between HQ Workstation and the Receiver. This prevents communication from being NAT out
2. Non-local networks destined for 212.20.0.150 are de-NATed to 192.168.0.150
3. Non-local networks destined for 212.20.0.101 are de-NATed to 172.31.200.101
4. Traffic from DMZ host 192.168.0.150 (Receiver) is NATed to 212.20.0.150
5. Traffic from the SMC (172.31.200.101) is NATed to 212.20.0.101

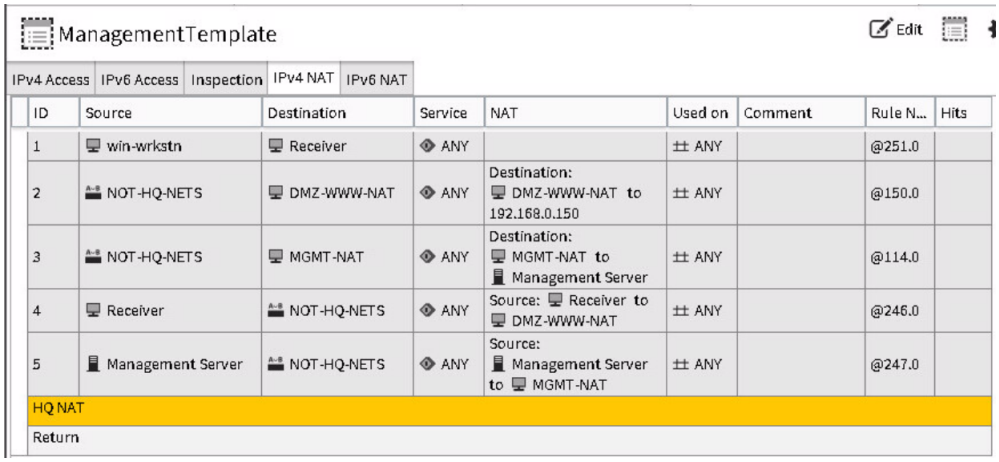


Figure 4: Management Template NAT Rules

4 Locations and Contact Addresses

While this section is not critical to understanding the labs, the contents of this section can be shown to students as a way of highlighting how remote, distributed management is done. There are 3 locations defined: Default, HQ, and Remote. Below is a list if the locations and their contents:

- **HQ**: contains HQ FW, Log server, and Management Server. In this location, the addresses that are used are the “real“ IP address, where no NATing is necessary

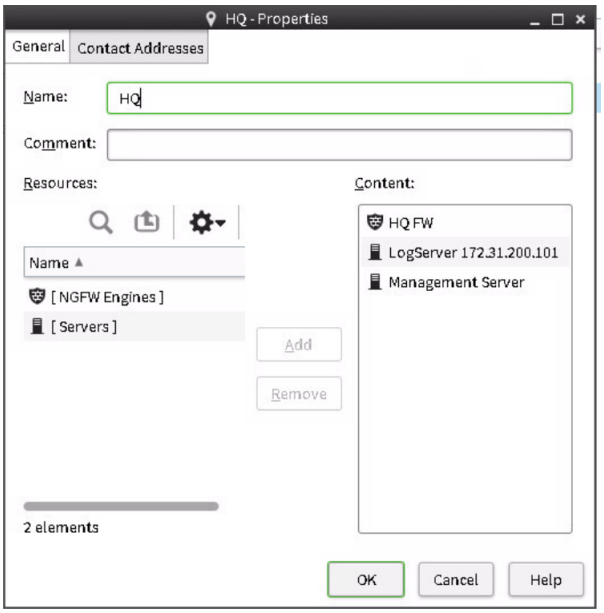


Figure 5: HQ Location Properties

- **Remote**: contains only the Router. Because the Router is in the remote location, it will use the NAT IP addresses for communication to the Management and Log Servers

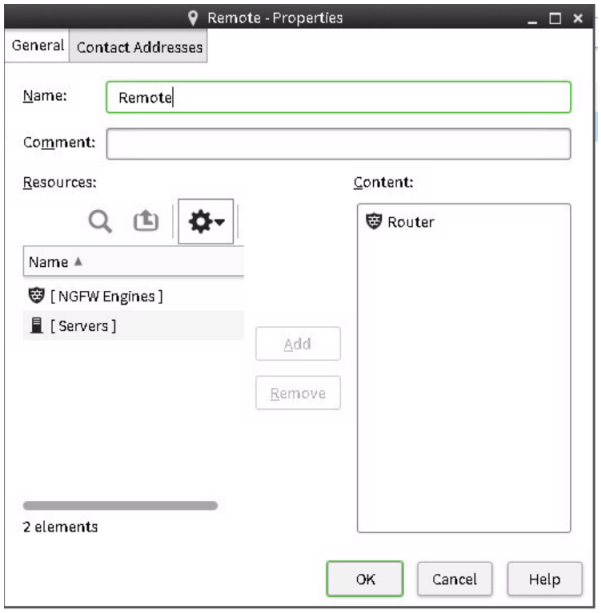


Figure 6: Remote Location Properties

For the Management and Log servers, contact addresses are defined so that devices outside of the HQ location have the correct IP address for communication with management components. These addresses are listed below.

- Viewed from the Remote location, the contact IP address for the Management server is 212.20.0.101

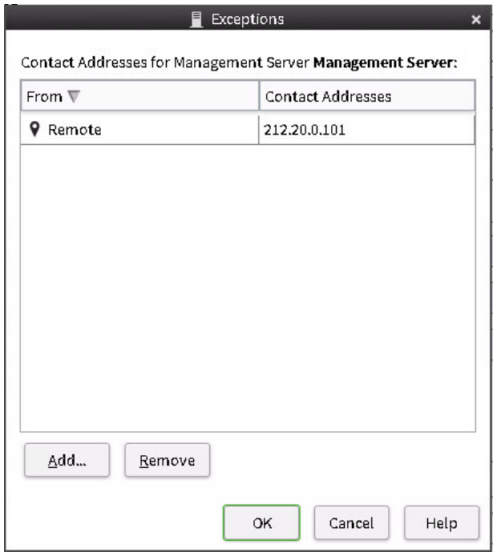


Figure 7: Management Server Contact Address

- Viewed from the Remote location, the contact IP address for the Log server is 212.20.0.101

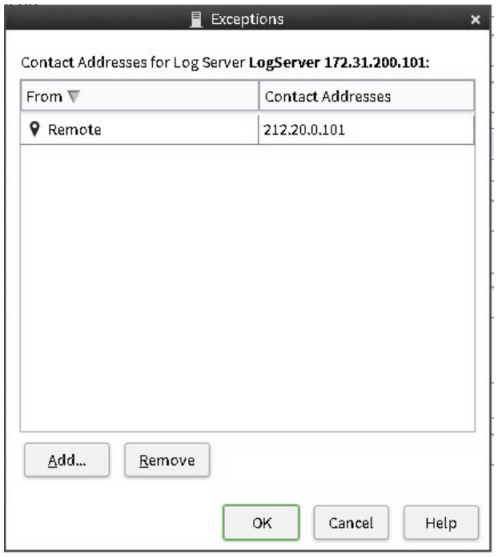


Figure 8: Log Server Contact Address

Provided that the Management template is being used, these contact addresses and locations need not be adjusted. When explaining this to students, make sure to explain that in order to change the location, depending on where the environment is being managed, the proper location must be chosen in the Management Client.

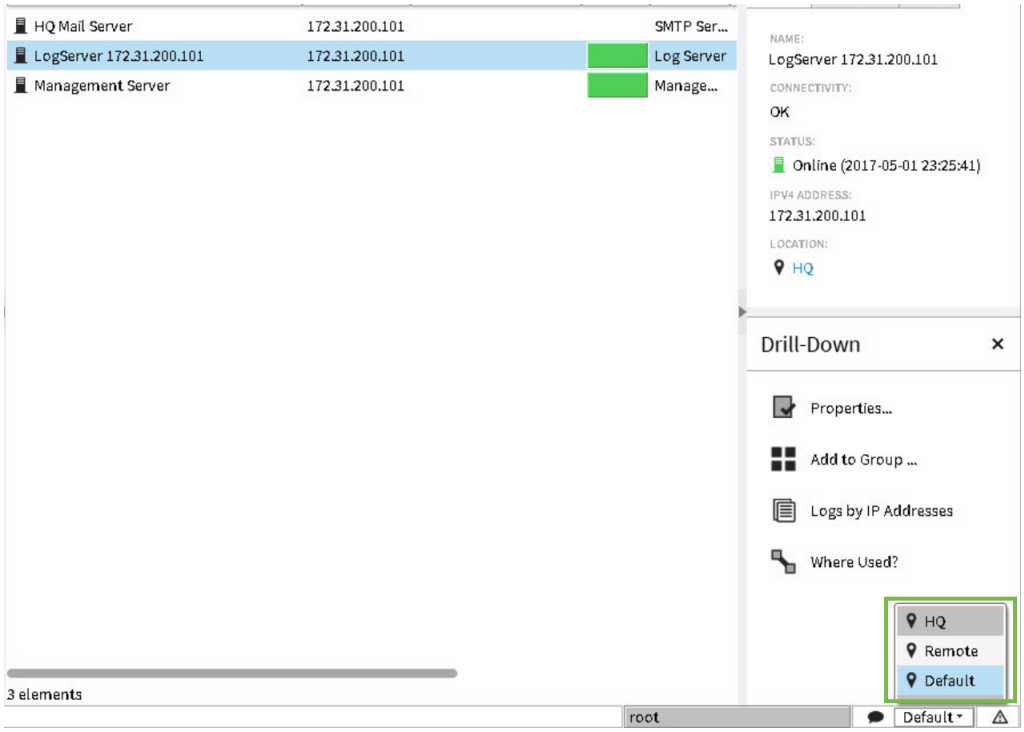


Figure 9: Select the Client Location

5 Traffic Generation Scripts

5.1 Starting and Stopping Traffic

Starting and stopping the traffic for labs that require it is simple: there are two shortcuts on the HQ Workstation desktop, one to start and one to stop the traffic. These two shortcuts link to *putty*, the Windows SSH client. For this reason, communication between the HQ Workstation and the DMZ host, 192.168.0.150, must be allowed. When clicked, putty has the capability of executing commands on the remote host. These commands are in a text file on the HQ Workstation. The following is the contents of those files:

To start the traffic, the following putty script is run from the HQ Workstation desktop shortcut "Start-Traffic":

```
sh -c 'cd VILT-Labs && ./startlabs.sh'
```

When run, the following command is run, which is specified in the shortcut properties:

```
putty.exe -ssh root@192.168.0.150 -pw Pass1234 \
    -m c:\Users\bjones\Documents\start-command.txt
```

The contents of this text file, listed above, runs a script on the remote host, 192.168.0.150 (Receiver), which begins the traffic generation. This script on the remote host contains the following:

```
#!/bin/sh

# This script communicates with the Pitcher, 87.35.16.200,
# and runs the traffic generation script. To do this,
# the screen command is used to detach the traffic
# generation script from the login shell.
# Without screen, when the login to run the script
# closed, the instance of the traffic generation would stop.
# When stopping the traffic generation, the stopping script
# has only to kill the instance of screen.
# startlabs.sh v1.0
#
# Matt McKinley

# Communicate with TrafficGen and run the traffic generation script
# SSH keys are used so that no password is required
ssh -l root 87.35.16.200 'screen -S gensession -d -m \
    ./VILT-Labs/Pitcher.sh'
sleep 2

# Run the local script to listen for incoming traffic,
# the receiving script
~/VILT-Labs/start-catcher.sh > /dev/null 2<&1 &
```

Stopping the traffic is very similar. There are two parts to this, similar to starting the traffic, a putty script and a script on the Receiver that communicates with TrafficGen to stop the traffic.

To stop the traffic, the following putty script is run from the HQ Workstation desktop shortcut "Stop-Traffic":

```
sh -c 'cd VILT-Labs && ./killlabs.sh'
```

When run, the following command is run, which is specified in the shortcut properties:

```
putty.exe -ssh root@192.168.0.150 -pw Pass1234 \
    -m c:\Users\bjones\Documents\kill-command.txt
```

The contents of this text file, listed above, runs a script on the remote host, 192.168.0.150 (Receiver), which stops the traffic generation. The script on the Receiver contains the following:

```
#!/bin/sh

# This script communicates with the Pitcher, 87.35.16.200,
# and issues a command that stops traffic generation.
# The screen session and netcat are killed on the Pitcher.
# Any running instances of netcat are killed locally on
# the Receiver.
# killlabs.sh v1.0
#
# Matt McKinley

# Open a session to the Pitcher and issue a kill command
ssh -l root 87.35.16.200 "killall -KILL screen; killall -KILL nc"

# Kill the traffic reception script and netcat
killall -KILL start-catcher.sh
killall -KILL nc

# Confirming all is shut down
echo Scripts shut down!
```

5.2 *start-catcher.sh*

The script below runs on the Receiver (192.168.0.150), located on the HQ Firewall DMZ. The script has the following functions:

1. An instance of netcat is run on port 65001 to receive notification from the Pitcher that it is done processing a packet capture
2. When the Receiver is ready to receive traffic on a new random port, that is sent to the Pitcher so that it begins sending traffic destined to that port
3. When the run state is TRUE, netcat will continue listening on the chosen port. When the Pitcher finishes processing a packet capture, it will send the run state FALSE flag to the Receiver

4. When the run state sent from the Pitcher is FALSE, the script begins listening on a new random port and sends that to the Pitcher

```
#!/bin/bash

# This is a simple script that listens for communication
# from the server containing port number on which to start
# listening. When the Pitcher communicates that it is done
# processing a packet capture, it sends a run state, which
# is used here to kill the current netcat instance and
# start listening on a new port. This is designed to simulate
# random traffic on random ports.
# Client v1.0
#
# Matt McKinley

# Variables
declare -i DEST_PORT

# Make sure there are no running instances of netcat
killall -KILL nc

# Start the listener for communication from the pitcher
nc -n -l 65001 -k > run &

# Set the initial run state to TRUE
echo TRUE > run
sleep 3

# Generate a random source port
DEST_PORT=${RANDOM%63000+2000}

# Send the port to the pitcher
echo $DEST_PORT | nc -n 87.35.16.200 65000
sleep 2

# Start an instance of netcat to receive traffic on the chosen port
nc -n -l $DEST_PORT -k >> /dev/null 2<&1 &
INIT_NC_PID=!$

while true; do
        RUN=$(tail -1 ./run)
        echo Run state is: $RUN

        # If the run state is false, netcat is killed and restarts
        # on another port. This is to simulate interesting traffic.
        if [ "$RUN" = "FALSE" ]; then
```

```
            kill -KILL $INIT_NC_PID >> /dev/null 2<&1
            kill -KILL $NC_RUN_PID >> /dev/null 2<&1
            echo $RANDOM >> /dev/null 2<&1
            DEST_PORT=${RANDOM%63000+2000}
            echo $DEST_PORT | nc -n 87.35.16.200 65000
            nc -n -l $DEST_PORT -k >> /dev/null 2<&1 &
            NC_RUN_PID=!$
            echo TRUE > run
        fi
        sleep 1
done
```

5.3 Pitcher.sh

The script the follows is run on the Pitcher and is responsible for several tasks:

1. Begin listening on port 65000 for messages from the Receiver
2. Read a directory containing packet captures and tcpstreams
3. Convert the packet captures into data that can be sent, via netcat, to the Receiver
4. Receive port number information, via port 65000, from the Receiver and send out the packet capture data on that port

```
#!/bin/sh

# This script uses netcat (nc) to generate traffic
# from random IP addresses to a netcat listener on
# the other end. tcptrace is used to extract the payload
# data so that netcat can use it to send to the Receiver
# behind the firewall.
# Server v1.0
#
# Matt McKinley

# Variables
declare -i SOURCE_PORT
declare -i SOURCE_ID
declare -i TARGET_PORT

SOURCE_FILE=/root/VILT-Labs/RANDOM_IPS-scrubbed
SOURCE_DAT=/root/VILT-Labs/DAT/
SOURCE_PCAP=/root/VILT-Labs/PCAPS/
TARGET_HOST=212.20.0.150
NC_WAIT_TIME=2
DATA_EXTENSION=dat
RCV_LPORT=65001

# Start Listener, ensure there are no running instances of netcat
killall -KILL nc
rm -f $SOURCE_DAT*

# Start the listener for traffic from the client (Receiver)
nc -n -l 65000 -k > listen_port &

sleep 10

echo running....

### Functions ###
```

```
# Add a random IP to an interface and update the arp
# cache on surrounding devices
function ADD_IP(){
    ip a a $1/32 dev eth0 label eth0:1
    arping -q -c 2 -I eth0 $1
}

# Delete a the previous random IP from an interface
function DEL_IP(){
    ip a d $1/32 dev eth0
}

# Use tcptrace to extrace the data portion of a packet capture
function CONN_TRACE(){
    tcptrace --output_dir="/root/VILT-Labs/DAT" -n -e $1 >> /dev/null 2>&1
}

# Remove all the DAT files that were generated by tcptrace
function CLEAN_DAT(){
    rm -f $SOURCE_DAT*
}

# Get the listening port from the client (Receiver)
function GET_PORT(){
    TARGET_PORT=$(tail -1 /root/VILT-Labs/listen_port)
}

# When done processing a packet capture, send a notification
# to the client (Receiver)
function SEND_KILL(){
    echo FALSE | nc -n $TARGET_HOST $RCV_LPORT
}

SOURCE_PORT=${RANDOM%63000+2000}
SOURCE_ID=${RANDOM%4+1}
SOURCE_IP=`head -$SOURCE_ID $SOURCE_FILE | tail -1`

# Start processing the DAT files generated by tcptrace and send the data to
# the Receiver
for PCAP in $SOURCE_PCAP*; do
    echo Got PCAP: $PCAP

    # If the file extension is pcap, then run tcptrace,
    # if .tcpstream, just send it
    if [ "${PCAP##*.}" == pcap ]; then
        CONN_TRACE $PCAP
```

```
            echo Trace Complete!
            sleep 1
        else
            cp $PCAP $SOURCE_DAT
            echo "Copying $PCAP to $SOURCE_DAT"
            sleep 1
        fi

        # choose a random source IP and add it to the interface
        ADD_IP $SOURCE_IP

        # process the tcpstream or pcap and send it to the client (Receiver)
        for DAT in $SOURCE_DAT*; do
            TARGET_PORT=$(tail -1 ./listen_port)
            echo Got port: $TARGET_PORT
            cat $DAT | nc -n -w $NC_WAIT_TIME -s $SOURCE_IP $TARGET_HOST \
            $TARGET_PORT
            sleep .2
        done
        sleep 1

        # clean up and move to the next pcap or tcpstream
        DEL_IP $SOURCE_IP
        echo $RANDOM >> /dev/null 2>&1
        SOURCE_ID=${RANDOM%4+1}
        SOURCE_IP=‘head -$SOURCE_ID $SOURCE_FILE | tail -1‘
        CLEAN_DAT
        SEND_KILL
        sleep 1
    done
```