Member: Michael Gonzalez 4.1 & 4.2, Mohammed Mamun 4.6, Devin Polichettiu 4.4, Cambrian Hausoe 4.3 & 4.5

4.1 Snort IDS and SETUP:

NS FILE:

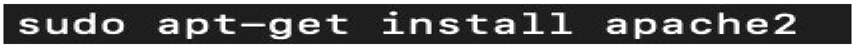
```
#NS File for User Nodes on our network:
set ns [new Simulator]
source tb_compat.tcl


set MICHAEL [$ns node]
set DEVIN [$ns node]
set MOHAMMED [$ns node]
set CAM [$ns node]
set HONEY [$ns node]


set lan0 [$ns make-lan "$MICHAEL $DEVIN $MOHAMMED $CAM" 100Mb 0ms]
tb-set-lan-loss $lan0 .02
tb-set-node-os $MICHAEL Ubuntu1604-STD
tb-set-node-os $DEVIN Ubuntu1604-STD
tb-set-node-os $MOHAMMED Ubuntu1604-STD
tb-set-node-os $CAM Ubuntu1604-STD
$ns rtproto Static
$ns run
```

NOTE:

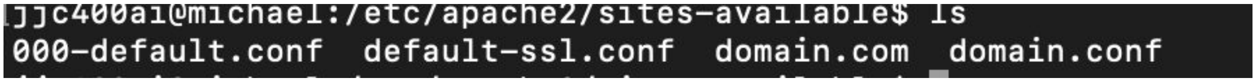
Before beginning the IDS using Snort there are some requirements to generate network traffic. The requirements are an apache2, which is a web server running an example http.conf file which will serve as a web server. The next requirement is an imap server which will use the provided Linux accounts as an email address. Next up, is the IRC server we will be using ircd-irc2 as the package to configure our example IRC server. Lastly, we will need to create a NFS mount point.

Configuring Apache 2 Server:

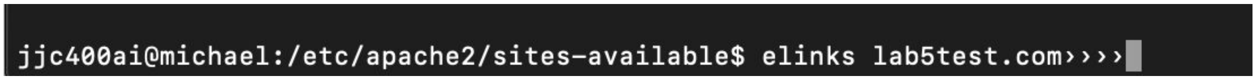
To configure the Apache 2 Web Server I first needed to install apache2 on my Ubuntu1604 environment:



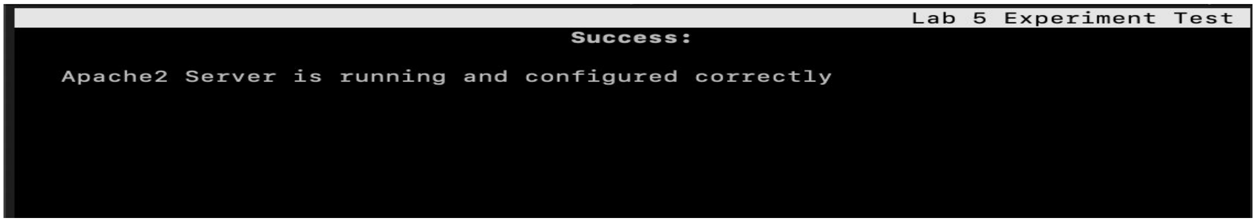
```
sudo apt-get install apache2
```



```
jjc400ai@michael:/etc/apache2/sites-available$ ls
000-default.conf  default-ssl.conf  domain.com  domain.conf
```



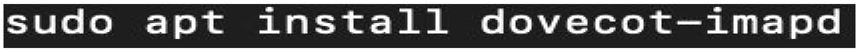
```
jjc400ai@michael:/etc/apache2/sites-available$ elinks lab5test.com››››
```



```
                                                              Lab 5 Experiment Test
                                  Success:

  Apache2 Server is running and configured correctly
```

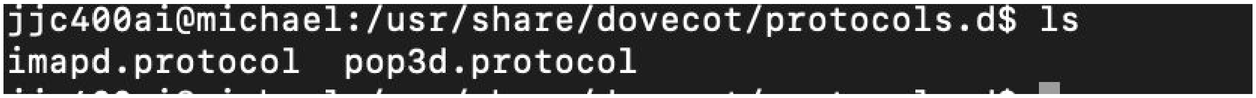
Configuring an IMAP server:

First we had to install dovecot-imapd package:



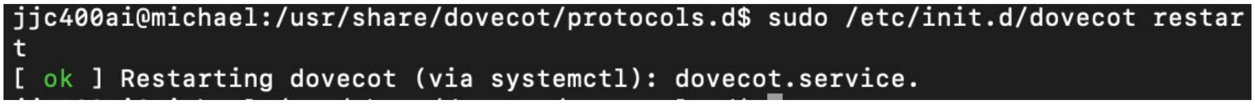
```
sudo apt install dovecot-imapd
```

Once installing this package, I was able to verify that the dovecot was configured using imap and pop3 by looking at the protocols.d file inside of /etc/dovecot/



```
jjc400ai@michael:/usr/share/dovecot/protocols.d$ ls
imapd.protocol  pop3d.protocol
```

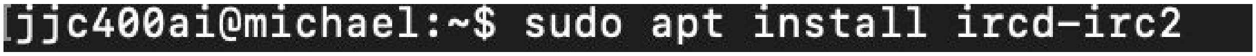
After verifying the protocols were set I reset the service for dovecot:



```
jjc400ai@michael:/usr/share/dovecot/protocols.d$ sudo /etc/init.d/dovecot restar
t
[ ok ] Restarting dovecot (via systemctl): dovecot.service.
```

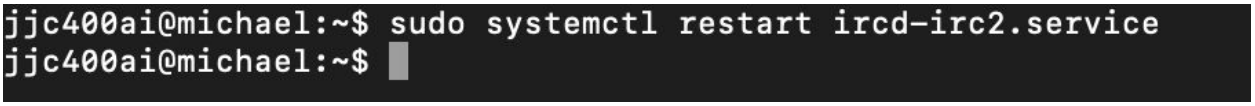
Configuring an IRC server:

Using the instructions found here I was able to set up a simple IRC server.
https://help.ubuntu.com/lts/serverguide/irc-server.html.en



```
jjc400ai@michael:~$ sudo apt install ircd-irc2
```

After installation I restarted the service and left it running. No further configuration would be necessary.

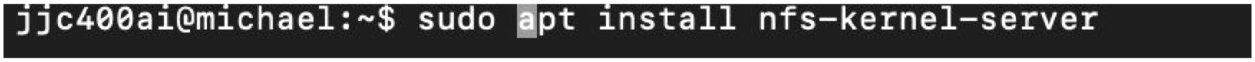


```
jjc400ai@michael:~$ sudo systemctl restart ircd-irc2.service
jjc400ai@michael:~$
```

NFS Mount Point:

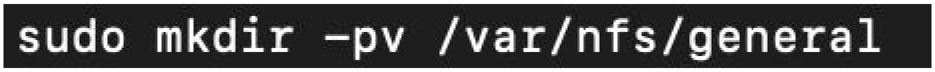
Using the instructions found here I was able to set up an NFS Mount Point.
https://www.digitalocean.com/community/tutorials/how-to-set-up-an-nfs-mount-on-ubuntu-16-04

Firstly I needed to install the nfs-kernel-server:



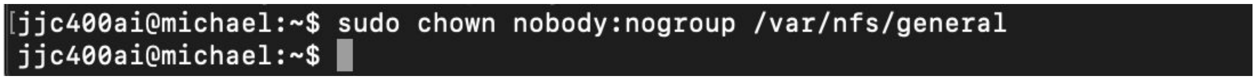
```
jjc400ai@michael:~$ sudo apt install nfs-kernel-server
```

Next I began to make a directory



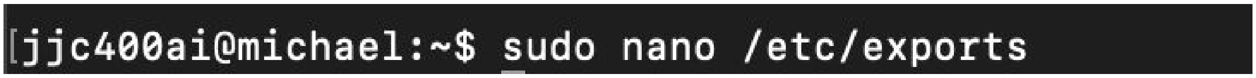
```
sudo mkdir -pv /var/nfs/general
```

Then I needed to modify the permissions:

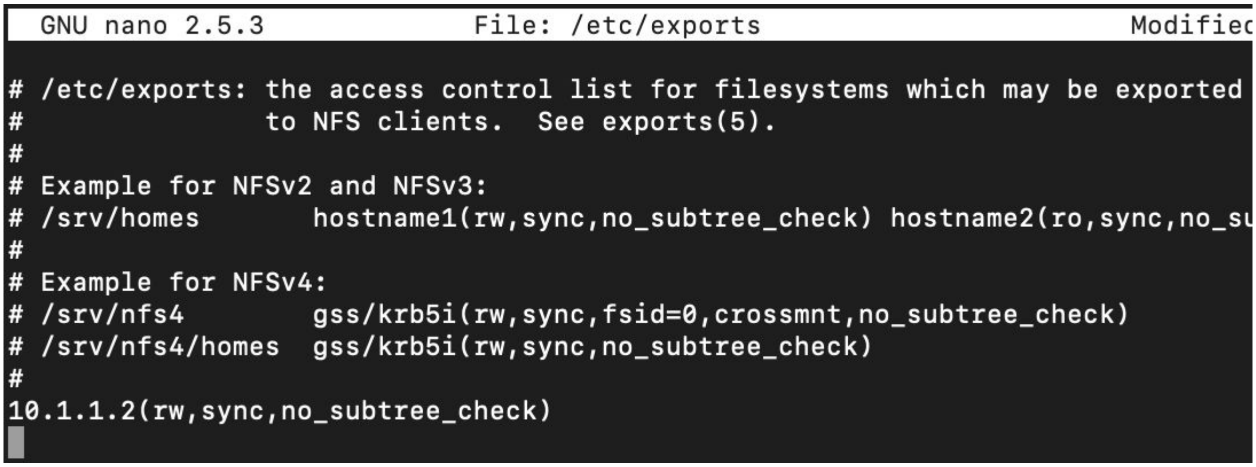


```
[jjc400ai@michael:~$ sudo chown nobody:nogroup /var/nfs/general
 jjc400ai@michael:~$
```

To edit and create a mount point I used nano to modify the configuration of nfs-kernel-server:



```
[jjc400ai@michael:~$ sudo nano /etc/exports
```



```
  GNU nano 2.5.3              File: /etc/exports                     Modifiec

# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_su
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
10.1.1.2(rw,sync,no_subtree_check)
```

Using Snort:

The two rules needed to be triggered are:

- log tcp any :1024 -> 192.168.1.0/24 500: log tcp traffic from privileged ports less than or equal to 1024 going to ports greater than or equal to 500
- alert tcp any any -> 192.168.1.0/24 143 (content: "|90C8 C0FF FFFF|/bin/sh"; msg: "IMAP buffer overflow!";)

In order to do this, it would be smart to setup a simple TCP server script to act as a listener. Then to trigger the two alerts we would need client TCP connections to send to the snort machines ip address using a port greater than or equal to 500 and less than or equal to 1024. This can be a recv inside of the tcp server rather than the client, that way you don't have to re-initiate the connection after the connection is established.

For the TCP network socket with message content we need to follow the same general guideline however, inside snort we would need to set a rule to detect TCP/IP Headers on the imap protocol. In doing so, while sending this specific header would trigger the rule due to when evaluated the number 159191905861631 is significantly higher than the highest bit allowed for a char value to be sent (the highest being 10ffff) The overflow occurs in the last four characters.

4.2 Bro Installation

To install bro we have to get the tar.gz package from executing this command:

```
sudo apt-get install cmake make gcc g++ flex bison libpcap-dev libssl-dev python-dev swig zlib1g-dev sendmail sendmail-bin
```

Next we must extract bros contents:

```
Tar -xvzf bro-2.5.2.tar.gz
```

Then bro must be configured using:

```
./configure --prefix= etc/bro
```

Next I needed to modify the config file for bro:

Located at etc/broctl.cfg

In this file I had to change the interface in use to my interface which was eth3.

After this the process is relatively straightforward, you just need to navigate sudo in the right direction to run the program as follows:

```
sudo etc/broctl
then install
```

Then you can exit broctl.

The way in which Bro works is through a command line system that handles all requests to the bro service called BroControlShell.

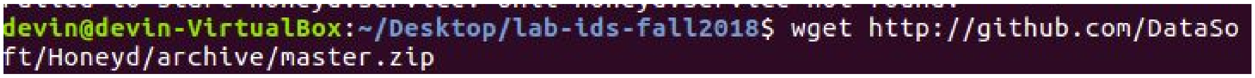
4.3

Installing and configuring the OSSEC-HIDS application took some time. I used my assigned node as the server and provided my email address for alerts pertaining to that server. Next, I tried configuring a different node as an agent, however I was unsuccessful in my attempts to do so. The first rule that was triggered was a new user logging into the server. The second rule was the successful login into the server. As far as I can tell, there is no way to completely disable the triggering of alerts by the IDS. However, you are able to modify at which level the alerts would be triggered by adding a few lines to the server's ossec.conf file. For example, you can modify the file to where any level 3 alerts and below (system low priority notifications & successful authorized events) would be ignored.

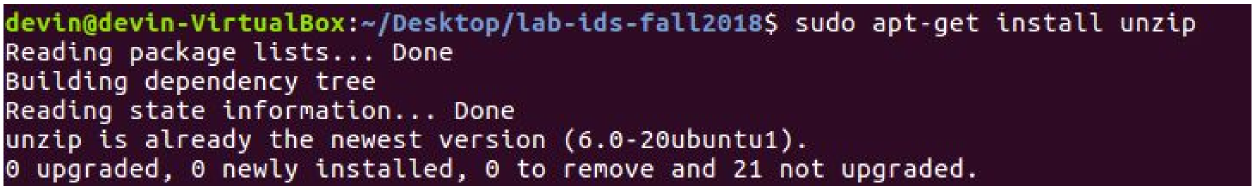
4.4

My first step for 4.4 was choosing to use Ubuntu to install the Honeyd packages. The given guidelines provided in the lab-ids zip file to install Honeyd had given me many issues when trying to install. I would get outputs stating that Honeyd is not available and that the package “honeyd” has no installation candidate. After hours of research I managed to locate a

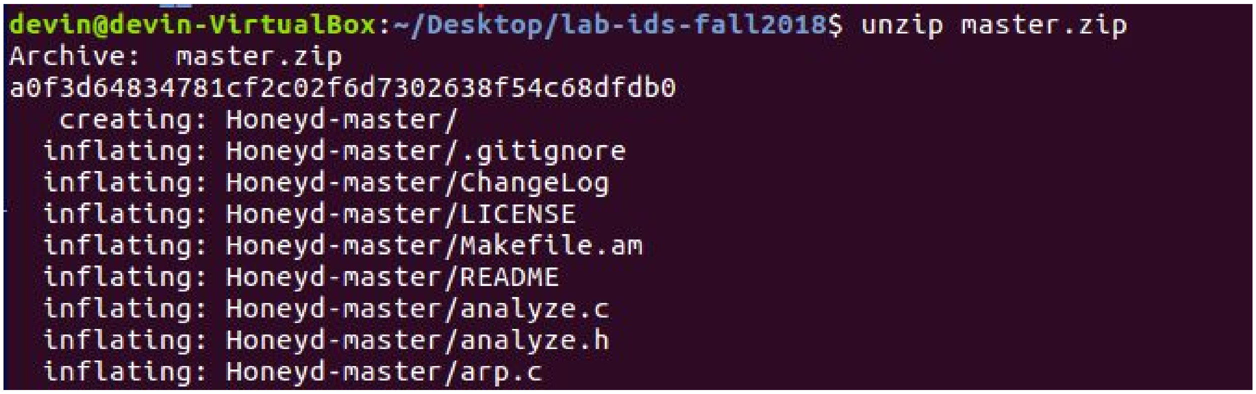
website that had the correct lines of code to install Honeyd.
https://singhgurjot.wordpress.com/2014/09/22/how-to-install-honeyd-on-ubuntu-13-10/
This link made me use the github link provided to obtain the master.zip file which was holding me back from installing Honeyd. Once using the line “unzip master.zip it loaded all the packages needed for Honeyd. Then using a libtool automaker and then using “cd Honeyd-master” which then made it possible to access the tools of Honeyd.



```
devin@devin-VirtualBox:~/Desktop/lab-ids-fall2018$ wget http://github.com/DataSo
ft/Honeyd/archive/master.zip
```

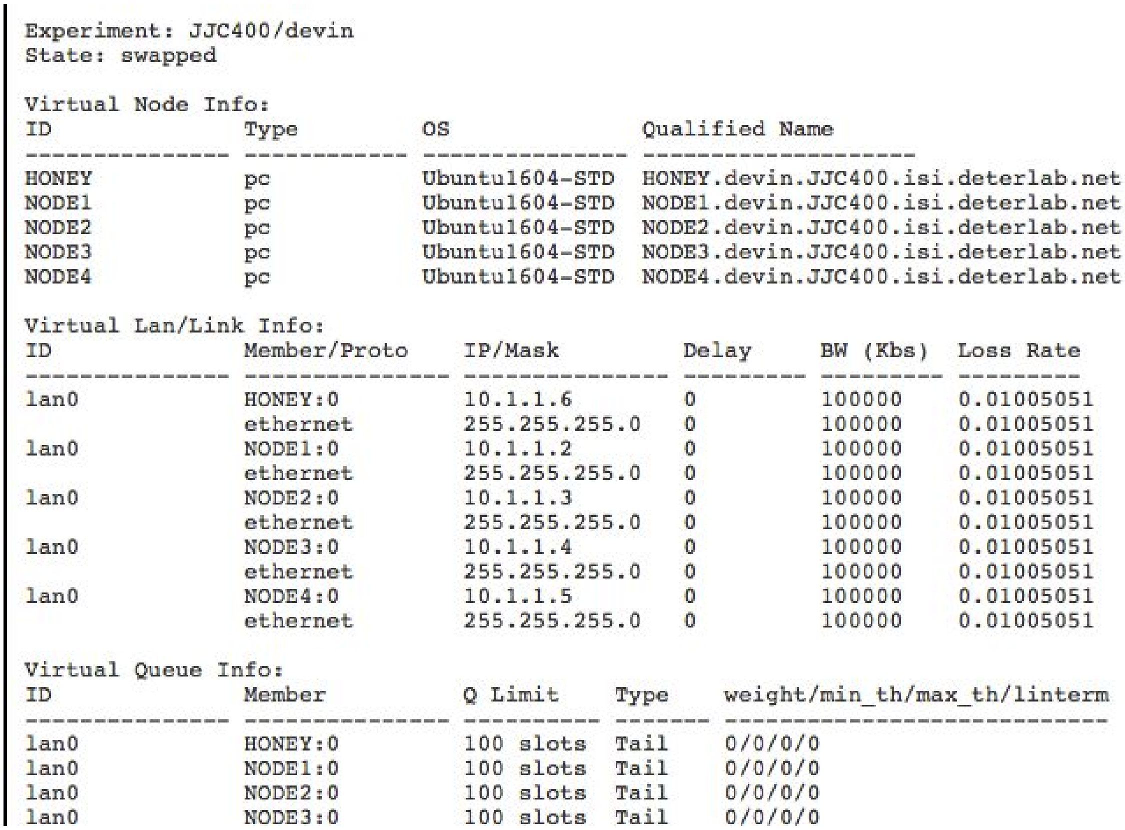


```
devin@devin-VirtualBox:~/Desktop/lab-ids-fall2018$ sudo apt-get install unzip
Reading package lists... Done
Building dependency tree
Reading state information... Done
unzip is already the newest version (6.0-20ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 21 not upgraded.
```

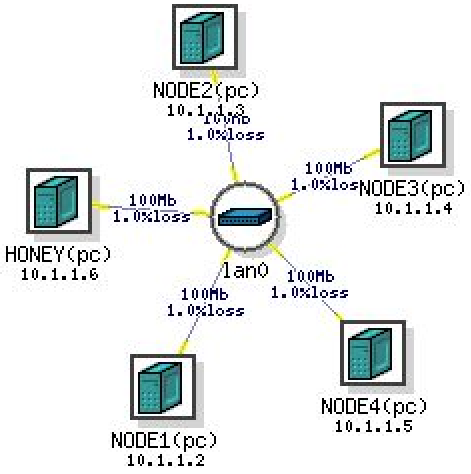


```
devin@devin-VirtualBox:~/Desktop/lab-ids-fall2018$ unzip master.zip
Archive:  master.zip
a0f3d64834781cf2c02f6d7302638f54c68dfdb0
   creating: Honeyd-master/
  inflating: Honeyd-master/.gitignore
  inflating: Honeyd-master/ChangeLog
  inflating: Honeyd-master/LICENSE
  inflating: Honeyd-master/Makefile.am
  inflating: Honeyd-master/README
  inflating: Honeyd-master/analyze.c
  inflating: Honeyd-master/analyze.h
  inflating: Honeyd-master/arp.c
```

For Deter I began with the ns file that supported the 5 nodes that had to be used within. That started by creating the ns file as a .txt file and then using the syntax to ensure the ns file is valid. Everything checked out fine with the ns file.txt and my experiment was able to be created through deter.



```
Experiment: JJC400/devin
State: swapped

Virtual Node Info:
ID              Type         OS              Qualified Name
--------------- ------------ --------------- --------------------
HONEY           pc           Ubuntu1604-STD  HONEY.devin.JJC400.isi.deterlab.net
NODE1           pc           Ubuntu1604-STD  NODE1.devin.JJC400.isi.deterlab.net
NODE2           pc           Ubuntu1604-STD  NODE2.devin.JJC400.isi.deterlab.net
NODE3           pc           Ubuntu1604-STD  NODE3.devin.JJC400.isi.deterlab.net
NODE4           pc           Ubuntu1604-STD  NODE4.devin.JJC400.isi.deterlab.net

Virtual Lan/Link Info:
ID              Member/Proto    IP/Mask         Delay     BW (Kbs)  Loss Rate
--------------- --------------- --------------- --------- --------- ---------
lan0            HONEY:0         10.1.1.6        0         100000    0.01005051
                ethernet        255.255.255.0   0         100000    0.01005051
lan0            NODE1:0         10.1.1.2        0         100000    0.01005051
                ethernet        255.255.255.0   0         100000    0.01005051
lan0            NODE2:0         10.1.1.3        0         100000    0.01005051
                ethernet        255.255.255.0   0         100000    0.01005051
lan0            NODE3:0         10.1.1.4        0         100000    0.01005051
                ethernet        255.255.255.0   0         100000    0.01005051
lan0            NODE4:0         10.1.1.5        0         100000    0.01005051
                ethernet        255.255.255.0   0         100000    0.01005051

Virtual Queue Info:
ID              Member          Q Limit    Type    weight/min_th/max_th/linterm
--------------- --------------- ---------- ------- ---------------------------
lan0            HONEY:0         100 slots  Tail    0/0/0/0
lan0            NODE1:0         100 slots  Tail    0/0/0/0
lan0            NODE2:0         100 slots  Tail    0/0/0/0
lan0            NODE3:0         100 slots  Tail    0/0/0/0
```



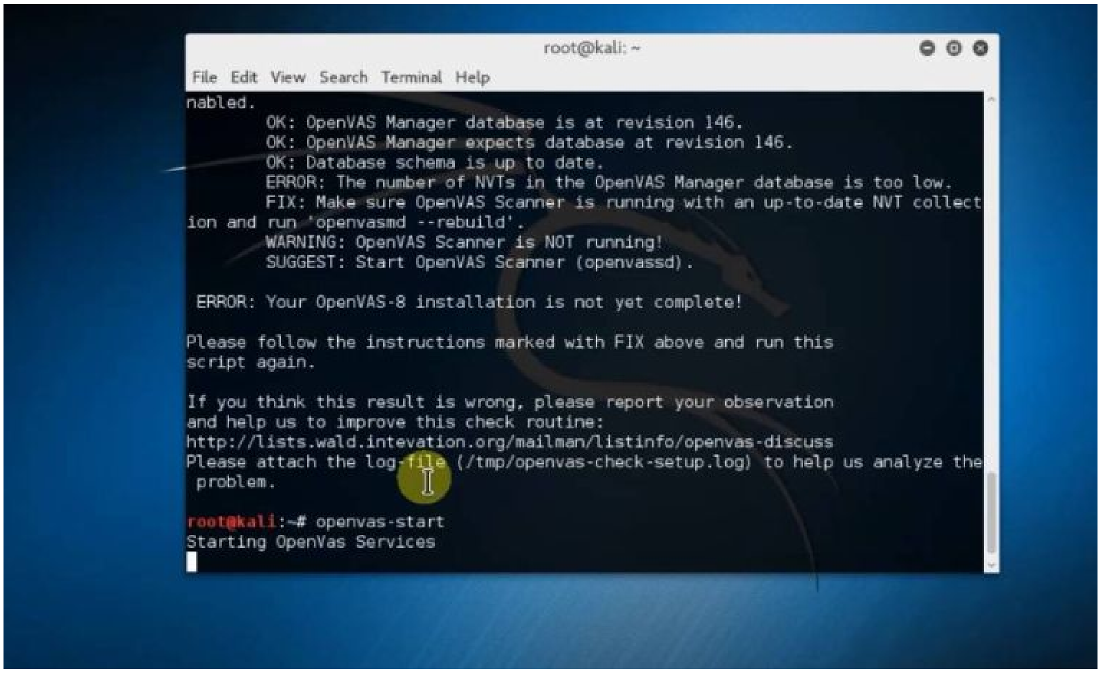
My issue with this part that I couldn't figure out is how to make Honey have every node individually connect to it in the diagram. I played around with the ns file to attempt in doing so but was unable. I then began with using ubuntu and attacking the network with the 5 nodes with operating systems. I began by placing Honeyd within the network and strategically changing the nodes within and giving them the names, MacOS, UNIX, UBUNTU, LINUX and WINDOWS.

They were placed and performed great but then the screen froze on me and causing my laptop to enter a sleep state and taking an hour to turn on. My data got lost and when repeating it would continue to slow down my entire computer and forcing it to turn off.

4.6

OpenVas allows to test penetration in our group by scanning the vulnerabilities.

```
root@kali:~# apt-get update
root@kali:~# apt-get dist-upgrade
root@kali:~# apt-get install openvas
root@kali:~# openvas-setup
```



```
nabled.
        OK: OpenVAS Manager database is at revision 146.
        OK: OpenVAS Manager expects database at revision 146.
        OK: Database schema is up to date.
        ERROR: The number of NVTs in the OpenVAS Manager database is too low.
        FIX: Make sure OpenVAS Scanner is running with an up-to-date NVT collect
ion and run 'openvasmd --rebuild'.
        WARNING: OpenVAS Scanner is NOT running!
        SUGGEST: Start OpenVAS Scanner (openvassd).

 ERROR: Your OpenVAS-8 installation is not yet complete!

Please follow the instructions marked with FIX above and run this
script again.

If you think this result is wrong, please report your observation
and help us to improve this check routine:
http://lists.wald.intevation.org/mailman/listinfo/openvas-discuss
Please attach the log-file (/tmp/openvas-check-setup.log) to help us analyze the
 problem.

root@kali:~# openvas-start
Starting OpenVas Services
```

Using the self assigned kali ssl certificate https://127.0.0.1:9392

After this you can just look at the reports in the ssl site.

Can you trigger this alert: alert tcp any any -> any any (minfrag: 256; msg: "Tiny fragments detected, possible hostile activity";)

For the individuals that want to run Bro directly from the build/ can do that by using this build path

```
./configure
```

```
 make
source
build/bro-path-dev.sh
bro <options
```