# Lab 5-Network Monitoring

Dawn M Inman

CSC-432 Computer and Network Security

Professor Nick Merante

April 17, 2020

Abstract

This lab sets up tcpdump on the router and goes through several small data exercises like comparing traffic on the public and private interfaces and isolating SSH and HTTP traffic. Argus is also set up on the router through quite a few steps: get and install, downloading source code packages, compiling, building the argus-clients package, configure the service by adding to the /etc/argus.conf file, create argus user and groups, set permissions, start the service and verify that it is running and view the network flow. Two different argus clients were used for filtering, rafilteraddr and rasort.

*Keywords*: CentOS 7, Kali Xfce, Proxmox, QEMU, PuTTY, tcpdump, argus, HTTP, SSH, ratop, rafilteraddr, rasort.

# Lab 5-Network Monitoring

Network traffic can be complicated to look at and understand. Two tools that can help with that are tcpdump and argus. These tools are both widely used and can be used easily from the command line. Both tools are standards for cybersecurity programs.

The open source program, tcpdump is a command-line tool that captures and analyzes network traffic that flows through the system, in this case the router. It has multiple options and filters so it can be used for most anything. It can be run on systems with a GUI and without, so it is ideal for those servers that do not have one. Data is often collected to be analyzed later. It can also be scheduled by using other tools along with it, which shows its flexibility. There are many online sources for information about tcpdump but a good one is supported by Red Hat, called opensource.com, which gives a great introduction to tcpdump as well as simple installation instructions for multiple platforms. (Gerardi, 2018)

Argus is an open source tool that is a network flow system that has been adapted for cybersecurity. It addresses several network flow data issues like privacy, performance, applicability and utility. It processes packets into data that can be easily read very quickly. The main system, called argus, is what generates the data and the argus-clients are a collection of data processing programs. Argus was created in the early 1980's at Georgia Tech and is still a critical component of cybersecurity. A new website was just built for Argus and its' clients, released in October 2019. (argus, 2020)

This lab uses the virtual router on the student network, a CentOS 7 router. CentOS 7 is so named because it stands for Community ENTerprise Operating System. It is based on the Linux kernel, free and has been available since 2004. Red Hat Enterprise Linux is the origination of CentOS 7 so it is a compatible option when requiring Linux software. It is very

popular with almost 30% of Linux web servers using it in 2011 and has been one of the most popular in hosting history. (CentOS Blog, 2020)

Kali Xfce is a newer Kali release. It is on the same line of Kali environments that have been created for Penetration Testing. There is a new feature called “Kali Undercover” which can make the display of Kali look like Windows 10. This can happen quickly so it is a type of stealth feature meant for blending in when in public areas. (Abrams, 2019) Other new features include KaliNetHunter KeX for Android which can install a full Kali desktop via Android, upgrading the kernel, Git powered documentation and adding PowerShell. (Elwood, 2019)

Proxmox is also being used by the systems and is used extensively in this lab. Proxmox VE hypervisor is based on GNU/Linux (Debian) and is open source. It has a central web-based management that does not require more installation. (Cheng, 2014) Version 5.4 is built specifically on Debian 9.8 with a “specially modified Linux Kernel 4.15”. (Proxmox, 2019) Proxmox is capable of two types of virtualization: OpenVZ and KVM. OpenVZ needs a patched Linux kernel so Linux guests are the only operating system type that can be created. In OpenVZ, the guests are called containers because they share the same architecture and kernel as the host operating system. (Cheng, 2014) KVM (Kernel-based Virtual Machine) is a modified Linux kernel built with the KVM module so that it can give hardware-assisted virtualization. Virtualization is performed by a software-based emulator (QEMU) which simulates the virtualized environment while KVM only exposes the /dev/kvm interface. (Cheng, 2014) “This converts Linux into a Type 1 (bare-metal) hypervisor.” (What is KVM?, 2020) Then QEMU or the software-based emulator will create the virtual machines on top of KVM. (What is KVM? 2020) Proxmox VE is relatively simple to start working with but can be very in depth as Simon

M.C. Cheng has authored a book called Proxmox High Availability which goes into more detail when setting up a high availability virtual cluster. (Cheng, 2014)

PuTTY is used several times during the lab for double checking past router command. PuTTY is an SSH client for Windows, Mac and Linux. It has a terminal window for access to the server used in this lab, the GNU/Linux server named router. (How to use PuTTY on Windows, 2020) SSH is a software package and means Secure Shell. It secures system administration and file transfers even though the networks are insecure. Tatu Ylonen is the inventor of SSH and OpenSSH which is an open source SSH program is based off of his free versions. (SSH(Secure Shell), 2020)

## Objective

This labs purpose is to get the router VM ready to watch traffic on the network. This is accomplished through the installation of tcpdump, which shows the packets that move through the router, and argus, which shows the network flow in real time as it moves through the router.

The computer that is being used is a 2011 HP Pavillion dv7, i7 quad core processor and 16GB RAM with Windows10Pro operating system. Google Chrome is the internet browser being used for connecting to Proxmox. KaliVM is on Proxmox. The KaliVM then runs Fire Fox internet browser, PuTTY as the SSH connection. The router is a CentOS 7 system with tcpdump and argus as the router network tools being experimented with during the lab.

## Results and Analysis[1]

### Tcpdump

For this lab, tcpdump needed to be installed on the router. Using the instructions tcpdump was installed on the router with

Yum install tcpdump

To put the same program onto the KaliVM, it needs to be typed into the command line differently.

apt-get update && apt-get install tcpdump

The man page (manual) is found at

man tcpdump

To listen to a certain interface, the option needs to be listed as -i. Two different interfaces were desired, the internal router interface, eth1 and the external or gateway, eth0. The commands are below with a portion of what was returned during the session:

tcpdump -i eth0

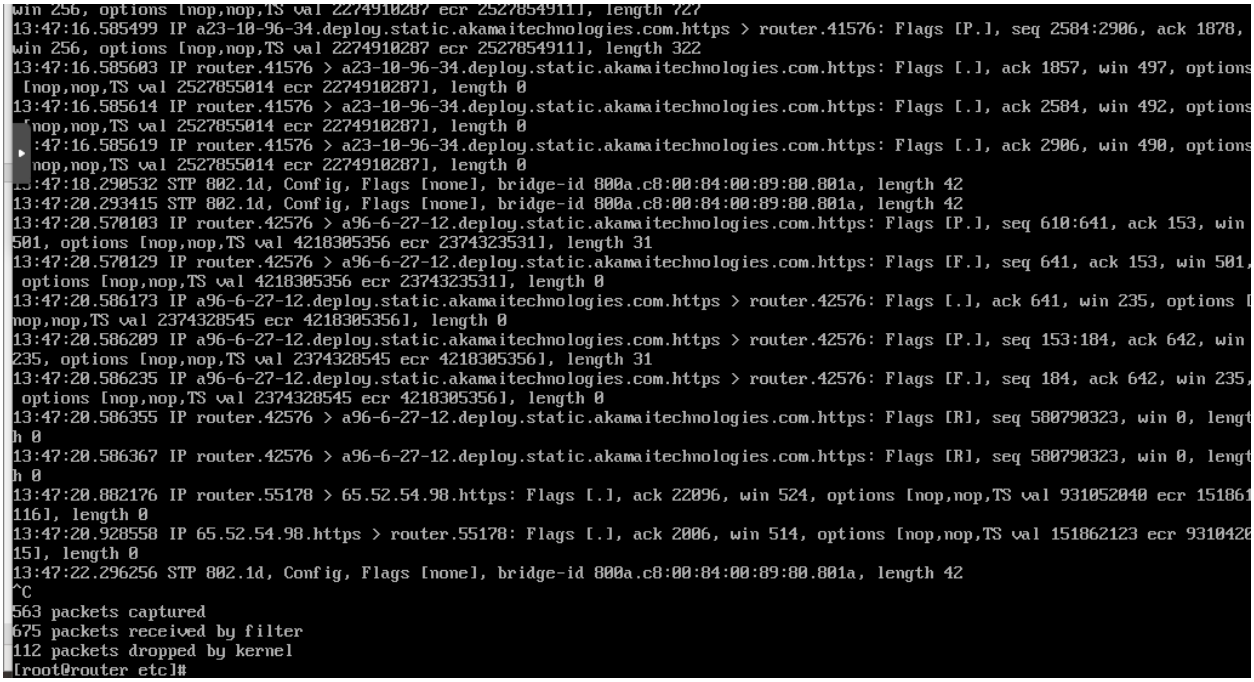



*Figure 1 eth0 tcpdump*

tcpdump -i eth1

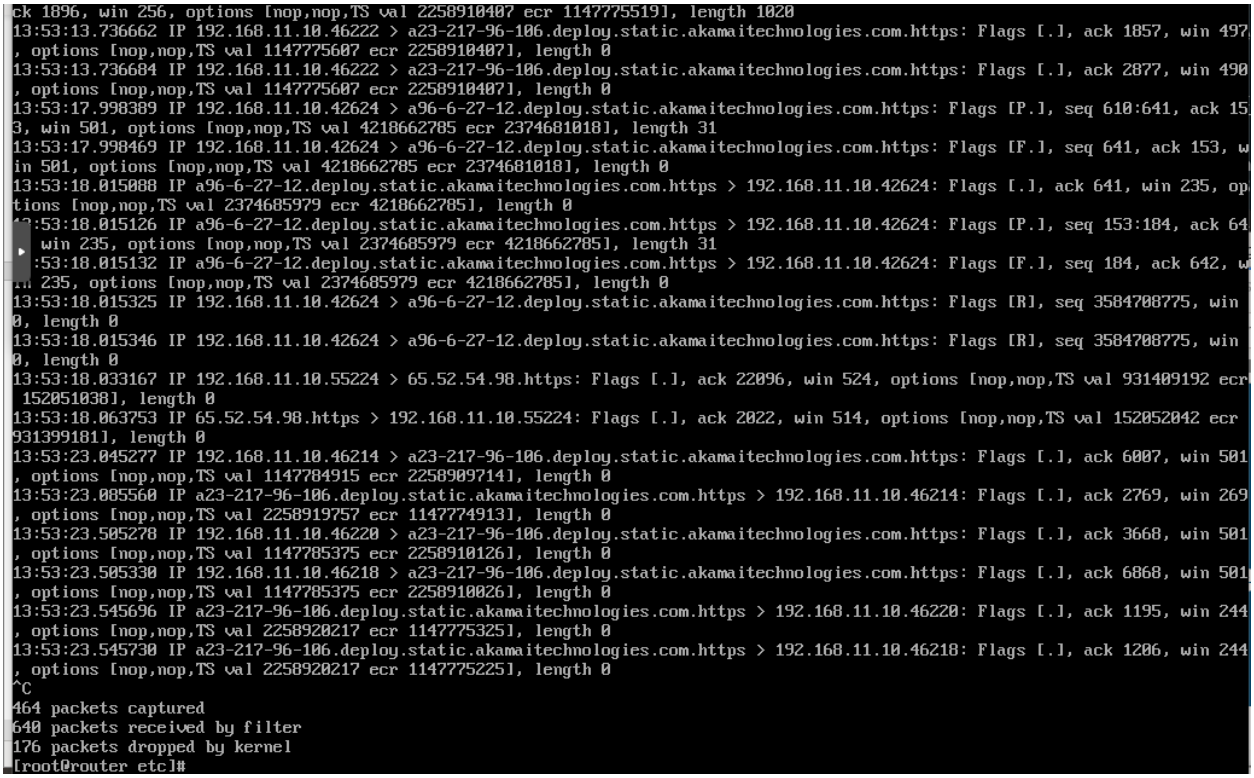



*Figure 2 eth1 tcpdump*

There are two differences between the interfaces to point out. When the KaliVM searches the internet, eth1(internal) shows the IP address of the machine but eth0 (external) shows the address as coming from the router instead. This keeps the IP address of the KaliVM private from online sources that could use the information maliciously. Another difference is how much traffic is coming into eth0 because it is the external router. The packets are requests are not always answered by the router as the packets don't apply to it, but it still creates a constant stream of incoming traffic.

To change the setting so that only SSH traffic is seen, the command line needs to be:

tcpdump port 22

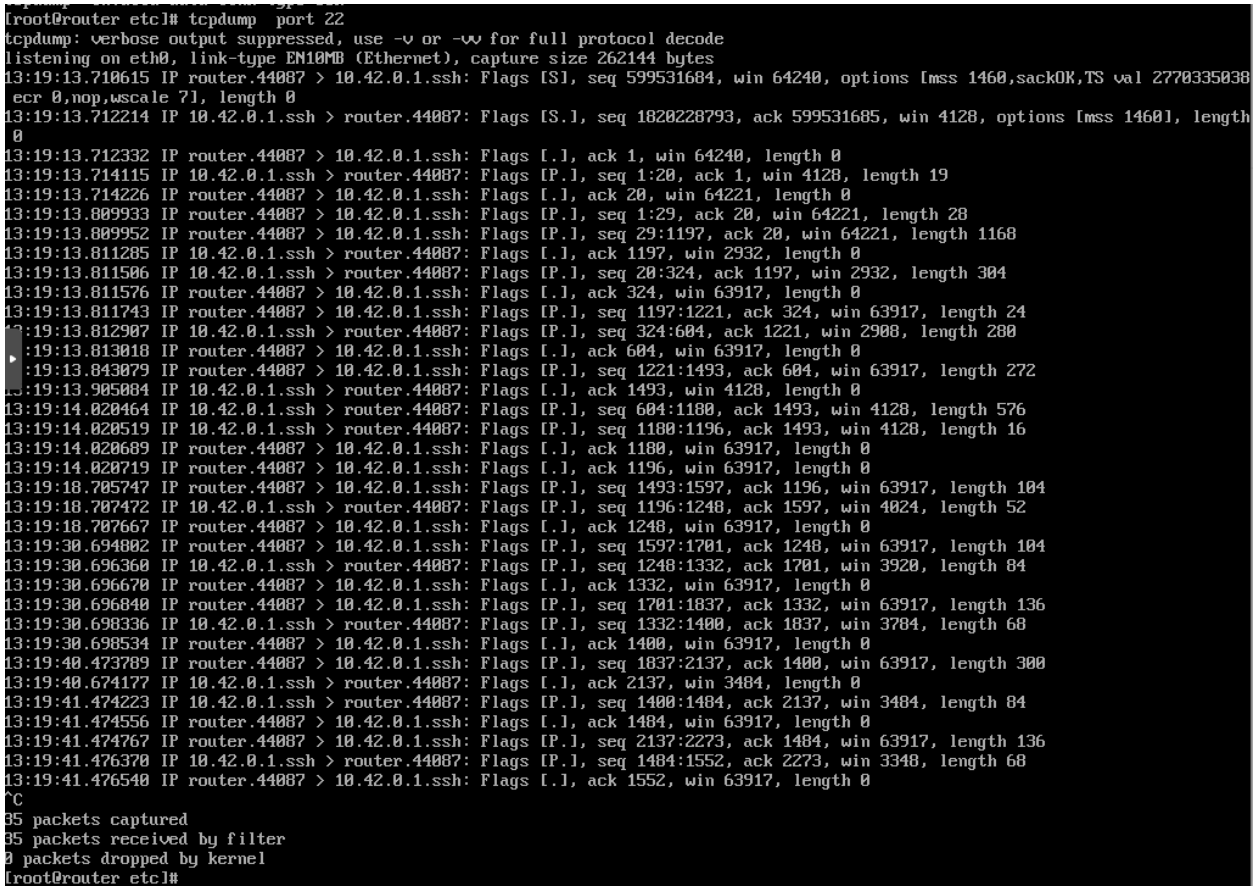



*Figure 3 eth0 ssh traffic*

This shows the traffic for login information from the KaliVM to darknet.

To see the http traffic the command line needs to read what is on port 80.

tcpdump port 80

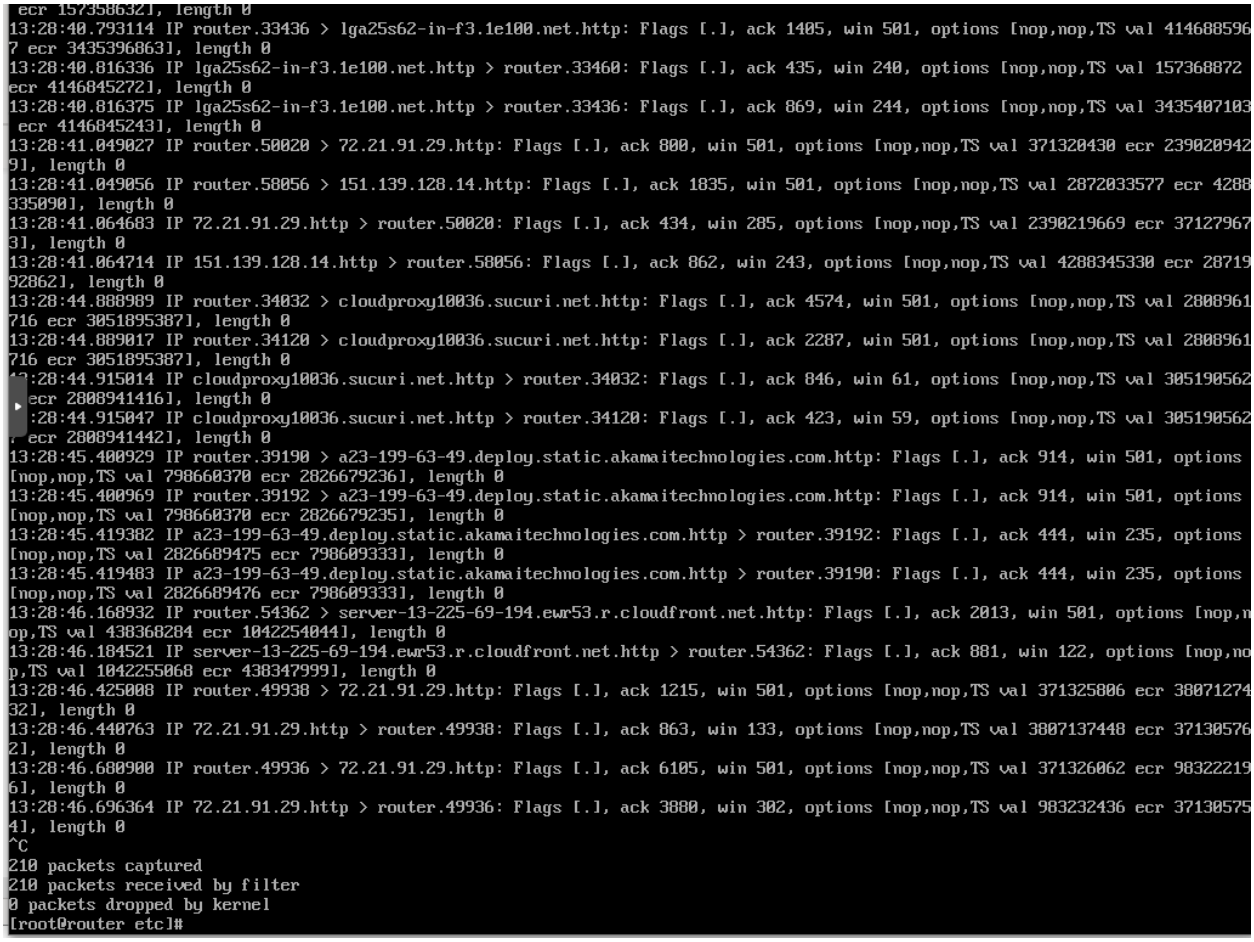



*Figure 4 tcpdump port 80*

When a specific interface is not assigned, tcp dump reads the lowest interface, in these cases, eth0. Port 22 was used for watching ssh traffic and port 80 was watched for http traffic.

When using PuTTY to go over to the WebServerVM, the results did not show up on the server.

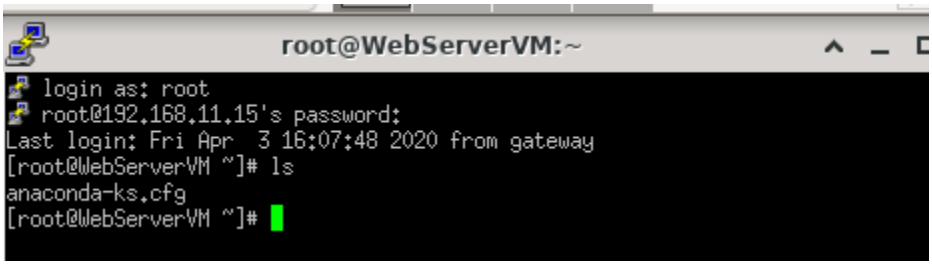



*Figure 5 WebServerVM doesn't show on results for ssh*

That is because the SSH connection did not pass through the eth0 interface.

An alternate way to scan for SSH is with this command line, which looks for all SSH on any port.

tcpdump 'tcp[(tcp[12]>>2):4] = 0x5353482D'

```
210 packets captured
210 packets received by filter
0 packets dropped by kernel
[root@router etc]# tcpdump 'tcp[(tcp[12]>>2):4]=0x5353482D'
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
13:41:45.085971 IP 4.26.24.234.ssh > router.45273: Flags [P.], seq 2919790516:2919790535, ack 1641418739, win 4128, length 19
13:41:45.157935 IP router.45273 > 4.26.24.234.ssh: Flags [P.], seq 1:29, ack 19, win 64221, length 28
^[[A^[[B^C
2 packets captured
```

An alternate way to scan for HTTP traffic is with this command line which looks for all HTTP traffic on any port.

tcpdump -vAls0

```
14:36:48.224922 IP (tos 0x0, ttl 58, id 60686, offset 0, flags [DF], proto TCP (6), length 40)
    104.17.68.176.https > router.57724: Flags [.], cksum 0x5927 (correct), ack 46, win 68, length 0
E..(..@.:...h.D.... ...¦..i...G.P..DY'........
14:36:48.224990 IP (tos 0x0, ttl 58, id 60687, offset 0, flags [DF], proto TCP (6), length 86)
    104.17.68.176.https > router.57724: Flags [P.], cksum 0x5732 (correct), seq 1:47, ack 46, win 68, length 46
E..V..@.:...h.D.... ...¦..i...G.P..DW2......).............HB¦..{"-......7{..    .^...$...
14:36:48.225136 IP (tos 0x0, ttl 63, id 61788, offset 0, flags [DF], proto TCP (6), length 40)
    router.57724 > 104.17.68.176.https: Flags [.], cksum 0x4b0d (incorrect -> 0x5748), ack 47, win 501, length 0
E..(.\@.?...... h.D..¦....G...i.P...K...
14:36:48.229917 IP (tos 0x0, ttl 121, id 41515, offset 0, flags [none], proto UDP (17), length 135)
    dns.google.domain > router.40571: 7035 NXDomain 0/1/0 (107)
E....+..y......... .5.{.s...{..,.......204.212.17.104.in-addr.arpa..............2.cruz.ns
cloudflare.com..dns.Ay2....'... `.   :.....
14:36:48.241212 IP (tos 0x0, ttl 64, id 53289, offset 0, flags [DF], proto UDP (17), length 72)
    router.41160 > dns.google.domain: 23555+ PTR? 176.68.17.104.in-addr.arpa. (44)
E..H.)@.@..:... ......5.4..\...........176.68.17.104.in-addr.arpa.....
14:36:48.264987 IP (tos 0x0, ttl 121, id 57164, offset 0, flags [none], proto UDP (17), length 134)
    dns.google.domain > router.41160: 23555 NXDomain 0/1/0 (106)
E....L..y......... .5...r.T\.,.........176.68.17.104.in-addr.arpa.............1.2.cruz.ns
cloudflare.com..dns.@y2....'... `.   :.....
^C
331 packets captured
487 packets received by filter
156 packets dropped by kernel
[root@router etc]#
```

**Argus**

Since the router has a broken argus, this line was used to try to repair and replace what is missing:

Yum install gcc make bison libpcap libpcap-devel readline-devel flex wget

When it installs correctly, it will say complete and show what was installed.

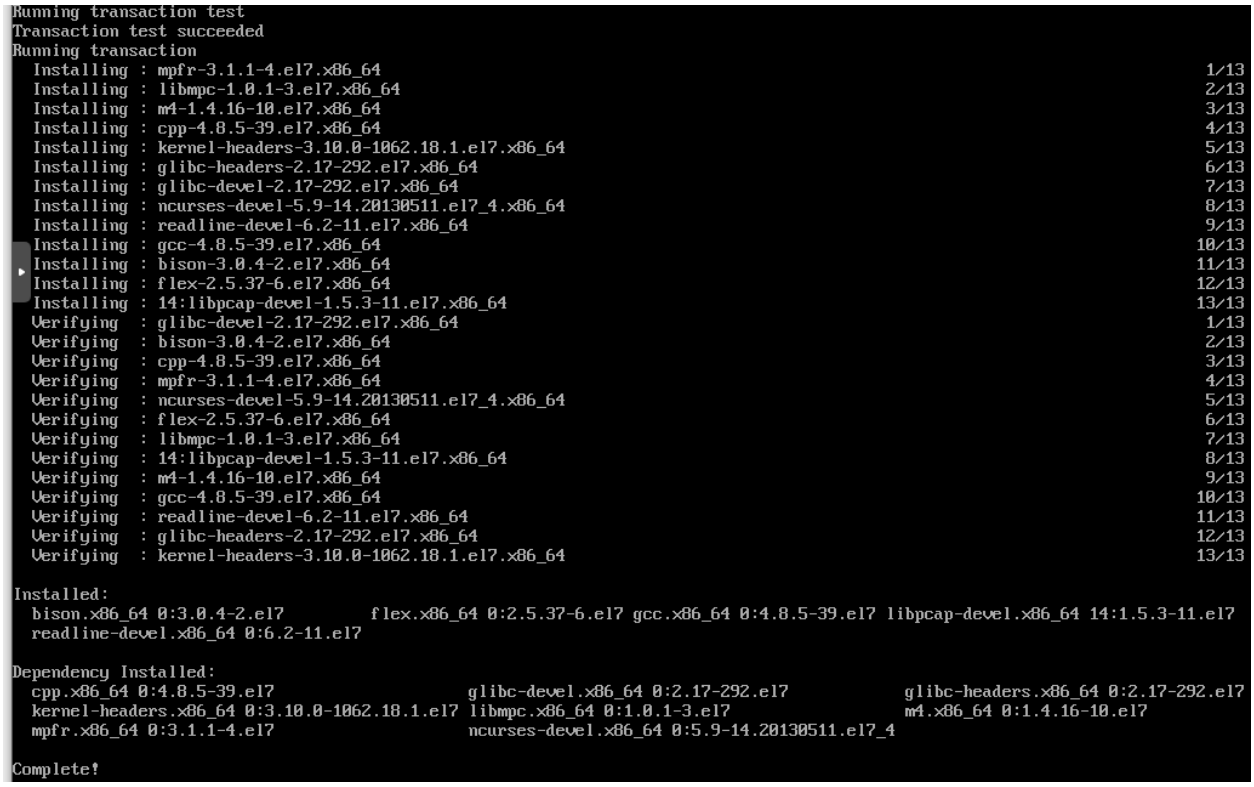



*Figure 6 argus complete install*

The next steps were easy to follow, but a couple mistakes were made along the way which will be described after the errors came up in the return.

Source code packages for argus and argus clients are downloaded:

cd /usr/src

wget http://qosient.com/argus/src/argus-3.0.8.2.tar.gz

wget http://qosient.com/argus/src/argus-clients-3.0.8.2.tar.gz

tar -xaf argus-3.0.8.2.tar.gz

tar -xaf argus-clients-3.0.8.2.tar.gz

Argus is compiled:

cd /usr/src/argus-3.0.8.2

./configure

make && make install

The argus-clients packages are built:

```
cd /usr/src/argus-clients-3.0.8.2
./configure
make && make install
```

Argus service is configured by making a file in vi, the contents are below:

```
vi /etc/argus.conf
```

```
ARGUS_FLOW_TYPE=”Bidirectional”
ARGUS_FLOW_KEY=”CLASSIC_5_TUPLE”
ARGUS_DAEMON=yes
ARGUS_MONITOR_ID=”eth1”
ARGUS_ACCESS_PORT=561
ARGUS_INTERFACE=eth1
ARGUS_SETUSER_ID=argus
ARGUS_SETGROUP_ID=argus
ARGUS_OUTPUT_FILE=/var/lib/argus/argus.out
ARGUS_FLOW_STATUS_INTERVAL=5
ARGUS_MAR_STATUS_INTERVAL=60
```

Argus user and group are created:

```
echo “argus:x:6000:6000:Argus:/home/argus:/xbin/nologin” >> /etc/passwd
echo “argus:x:6000:” >> /etc/group
```

Permissions are set on the output directory to read for all users except the owner:

```
mkdir /var/lib/argus
```

touch /var/lib/argus/argus.out

chown -R argus:argus /var/lib/argus

The Argus service is started:

/usr/local/sbin/argus -F /etc/argus.conf

Verification is needed that the service is running. This is done through netstat pulling up all the numbers that are running and grep finding the correct one on port 561.

netstat -an | grep 561

The first time this was ran, an error showed up, shown below.

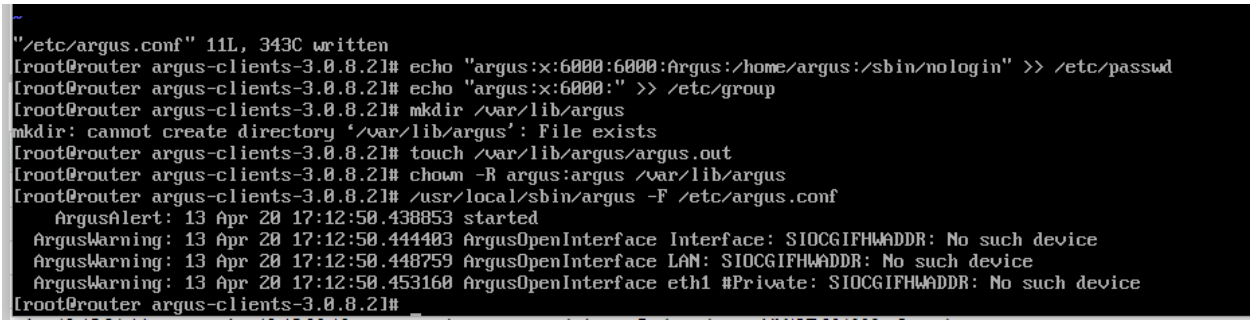



*Figure 7 Argus Issues*

At this point argus is not working. The clue is that there is no such device, but it has been told the device to use in the /etc/argus.conf file. What happened was there were two different typos, one was that the word DAEMON was spelled with the a and e switched, and on the ARGUS_SETGROUP_ID=argus there was a – instead of a _. These two mistakes were difficult to see so that took time to find. During that time a great website was found that shows why this file looks and works like it does. It is part of the linux man pages and is found here: https://www.systutorials.com/docs/linux/man/5-argus.conf/

Another set of mistakes was made because of trying the process more than once to find the errors. As found out in past labs, often trying commands again will create extra lines in files that need to be erased. There were extra lines in the /etc/passwd file as well as the /etc/group

file. Once those were erased down to one line as was proper, everything works properly when argus was tested using netstat:

Netstat -an | grep 561

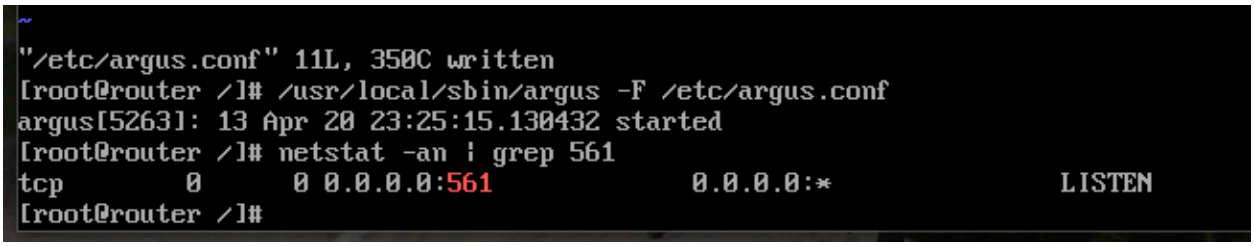



*Figure 8 Argus running correctly*

The next command uses ratop and from it the network flow can be seen in real time as it passes through the router.

Ratop -S 127.0.0.1:561

At first there is no traffic so firefox is started again on the Kali VM.

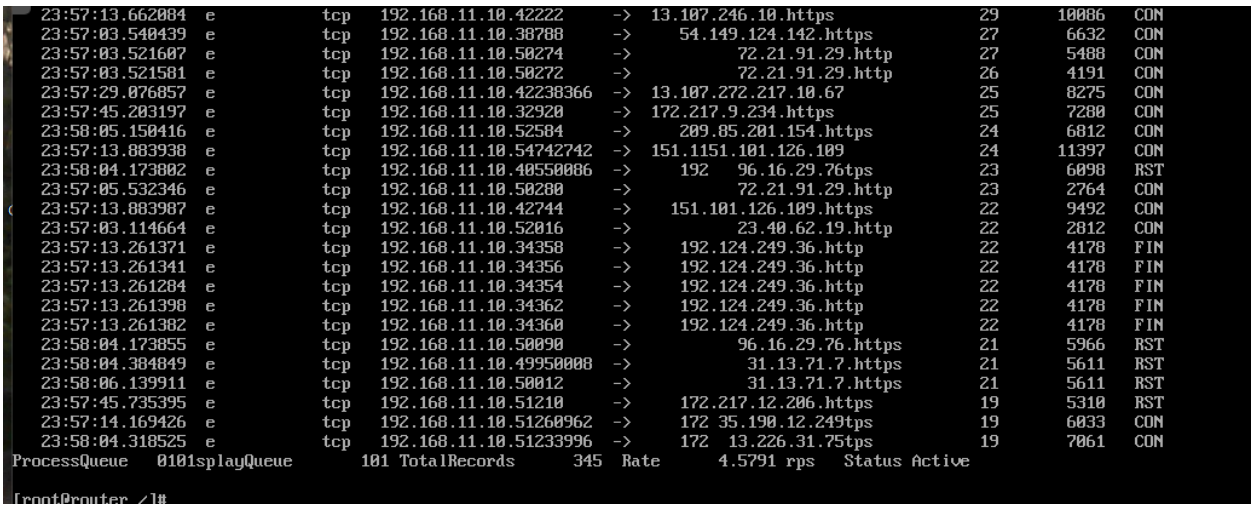
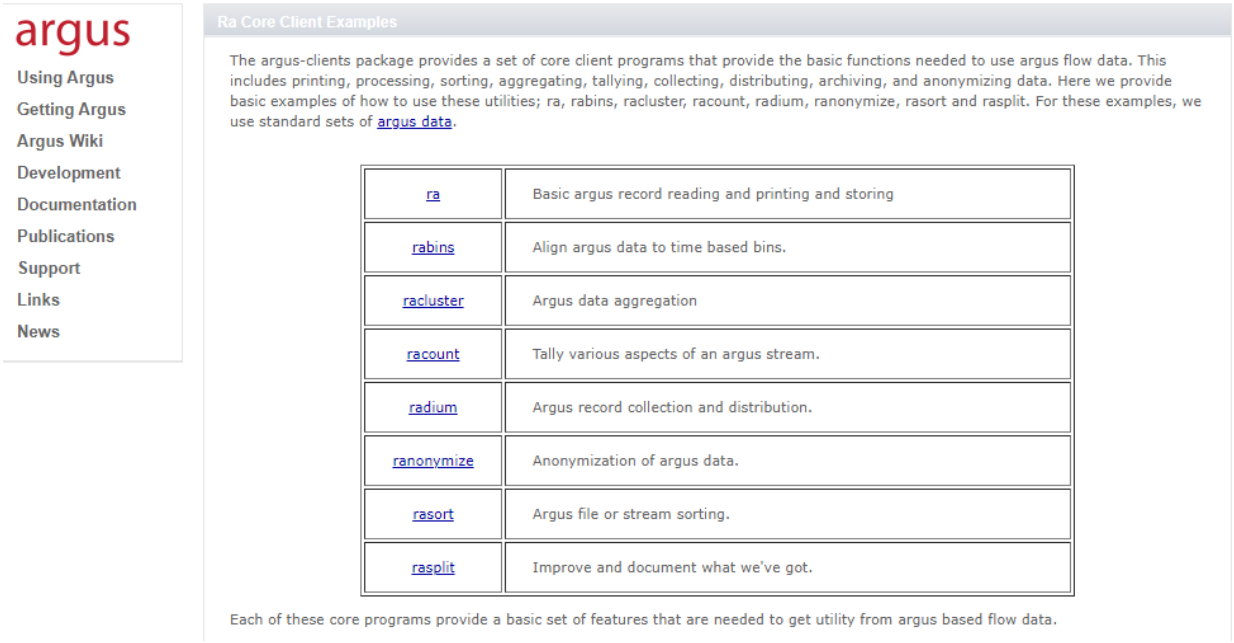



*Figure 9 ratop*

Ratop provides information like Sending and Destination IP addresses, direction the information is going, time, TCP or UDP, destination port, total bytes, total packets and the state. It is also in real time where it moves as quickly as the internet browser page does. The output is different from tcpdump as tcpdump will show a lot of lines for something like connection setup

and data transfer where ratop will consolidate that information into one line and give the duration of time it took for the whole transaction. This is especially helpful to answer questions like “who is talking to whom, how often, is one address sending all the traffic, are they doing the bad thing?” (Auditing network activity, 2012) Being able to ask and answer these questions is very helpful to a network security professional or network administrator.

Reading through everything that Argus can do, especially as a beginner with it, can be a large task. A website that helped was openargus.org, specifically orpenargus.org/oldsite/ra.core.examples.shtml . Below is a pic of some of the argus clients that help with data flow. Once this list was found, choosing two clients to use with argus was much easier.




At this point the process to acquire the clients like rasort, racluster and rafilteraddr was unclear. The clients were already installed, however, so once the work has been done to this point of the lab, the clients are already installed and ready to work with .To begin client work by working off

the man pages, rasort was used as it provided very simple directions and gave a focused look at where all the information was at. For example, when wanting to sort by bytes of the total transaction, use:

rasort -r /var/lib/argus/argus.out -m bytes

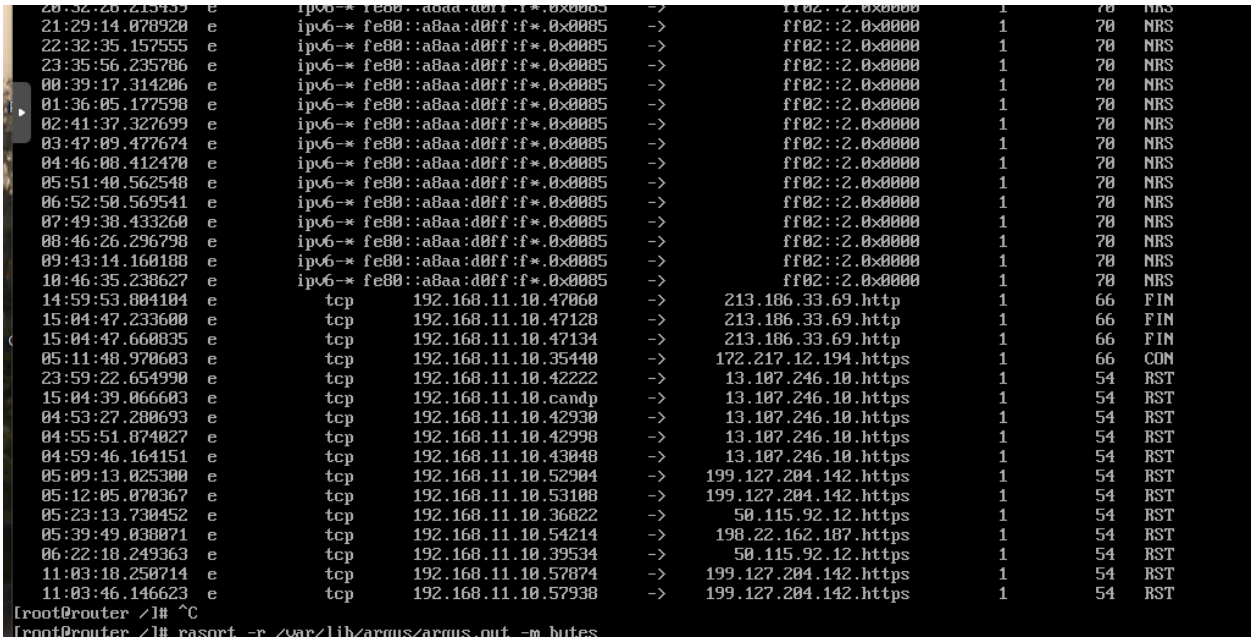



*Figure 10 rasort sorting for bytes*

When sorting for total transaction packets, use:

rasort -r /var/lib/argus/argus.out -m pkts

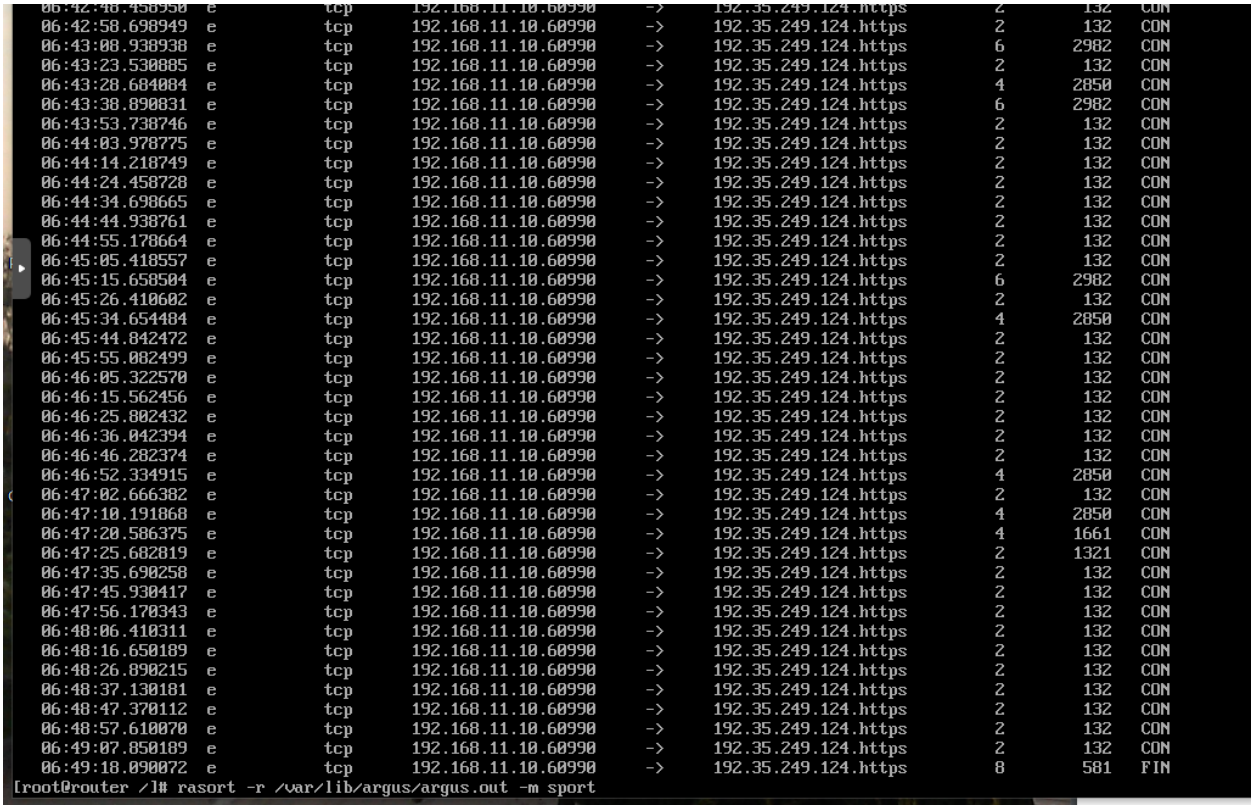



*Figure 11 rasort sorting for packet count*

Below the source port was sorted by using:

rasort -r /var/lib/argus/argus.out -m sport

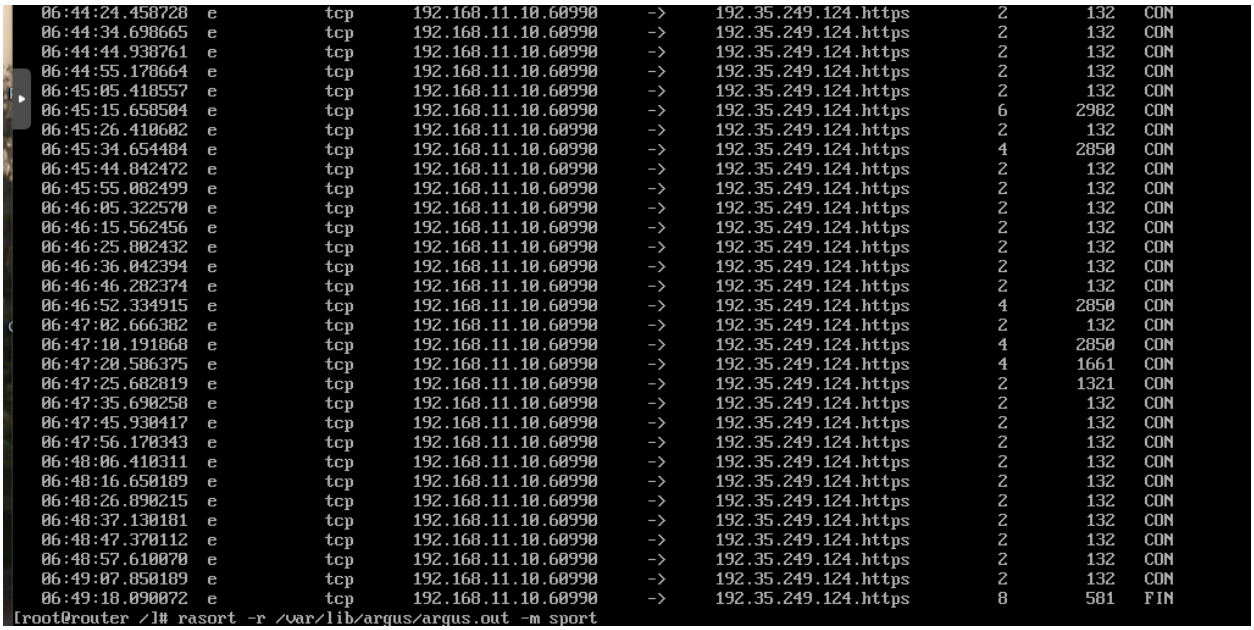



*Figure 12 rasort for source port*

Finally, the oui of the source MAC address was sorted for

rasort -r /var/lib/argus/argus.out -m soui

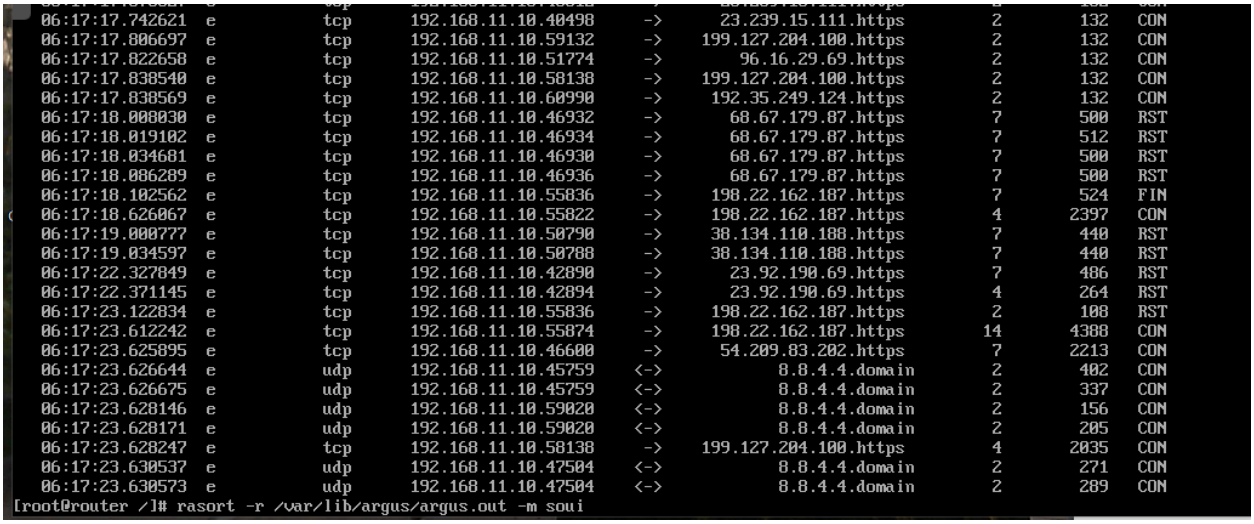



*Figure 13 rasort for oui of the source MAC address*

The next client used is a filter for IP addresses. This client requires another file to be made that has the IP addresses in it that are wanted to be searched for. This file can be anywhere

on the computer but needs to be input into the command line, along with the output argus file. For the first try, just the standard IP address was used of the KaliVM machine, 192.168.11.10

rafilteraddr -r /var/lib/argus/argus.out -f /var/lib/argus/ipfile

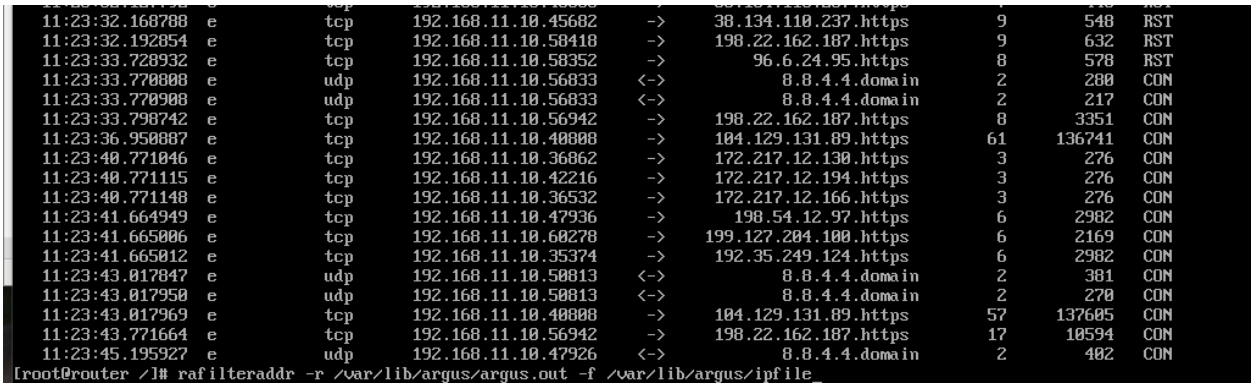



*Figure 14 rafilteraddr for IP address 192.168.11.10*

Since there are a few different IP addresses in the above pic, three were chosen and a new file (ipfile2) was created to filter from using IP addresses 8.8.4.4, 172.217.11.2 and 8.8.8.8.

rafilteraddr -r /var/lib/argus/argus.out -f /var/lib/argus/ipfile2

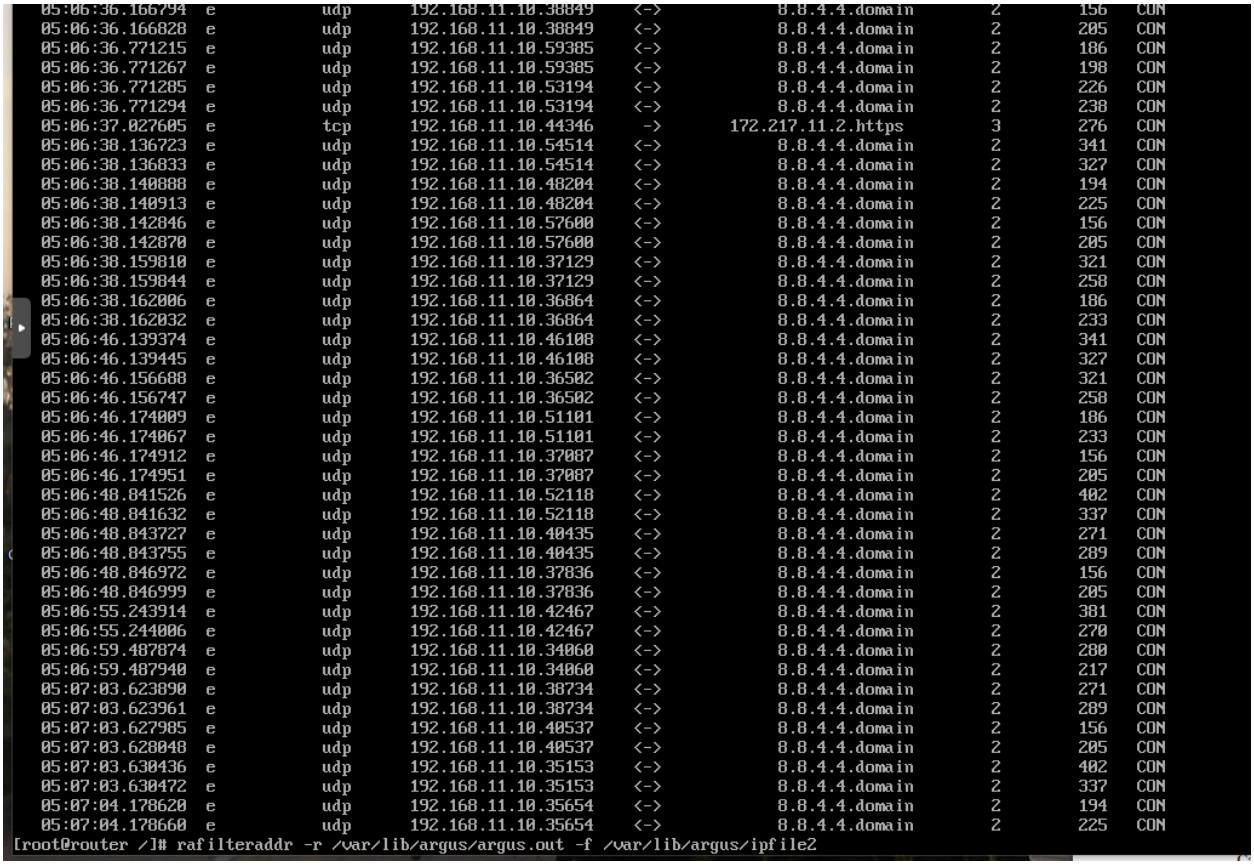



*Figure 15 rafilteraddr IP address 2*

In the right column of IP addresses, 8.8.4.4 is appearing the most but the filter is working properly as the address 172.217.11.2 can be seen toward the top. This feature could be useful when searching for specific known malicious IP addresses, say from a botnet server or a known IP address that has repeatedly attacked a system, to see if it is still happening or to see if mitigation may need to take place or company emails sent out, as in the case of botnet spam.

**Conclusion.**

This lab was successful at teaching how to set up tcpdump and argus on a CentOS 7 server. It also gave more clarity on man pages, how to read them, what to look for in the details and what a program client needs to work properly. Using the tools rasort and rafilteraddr helped the student learn more about what a client program is and how they are attached to a main

program like argus and what they help do. The rasort program was a standard sorting tool that shows more information than just filtering out for one item. This could be useful when looking for malicious activity as hackers can often change one or two details to fool servers into thinking they are legitimate. The rafilteraddr is a filter that scans for specific IP addresses that have been put into a file. When these addresses appear, they are shown on the screen. This would be helpful for security as when malicious IP addresses have been previously been identified, they can be put into the list file and any attempts by them to infiltrate the system could be seen clearly as soon as the attack begins. Overall this lab was an excellent learning experience and a great source for future traffic monitoring.

References

Abrams, Lawrence. (November 29, 2019). *Kali Linux adds 'undercover' mode to impersonate* Windows 10. BleepingComputer. Retrieved from https://www.bleepingcomputer.com/news/security/kali-linux-adds-undercover-mode-to-impersonate-windows-10/

Argus. (2020). Definition from openargus.org. Retrieved from https://openargus.org/

Auditing network activity. (2012). Argus. Retrieved from https://qosient.com/argus/gettingstarted.shtml

Centos Blog (2020). *What is Centos?* Retrieved from https://www.centosblog.com/what-is-centos/

Cheng, Simon M.C. (October 27, 2014). *Basic concept of ProxMox Virtual Environment.* Packt. Retrieved from https://hub.packtpub.com/basic-concepts-proxmox-virtual-environment/

Elwood. (November 26, 2019). *Kali Linux 2019.4 release*. Kali Linux News. Retrieved from https://www.kali.org/news/kali-linux-2019-4-release/

Gerardi, Ricardo. (October 10, 2018). *An introduction to using tcpdump at the Linux command line.* Retrieved from https://opensource.com/article/18/10/introduction-tcpdump

ProxMox. (April 11, 2019). ProxMox.com Retrieved from https://www.proxmox.com/en/news/press-releases/proxmox-ve-5-4

SSH(Secure Shell). (2020). SSH.com. Retrieved from https://www.ssh.com/ssh

What is KVM? (2020) RedHat.com Retrieved from https://www.redhat.com/en/topics/virtualization/what-is-KVM

Lab Network Topology