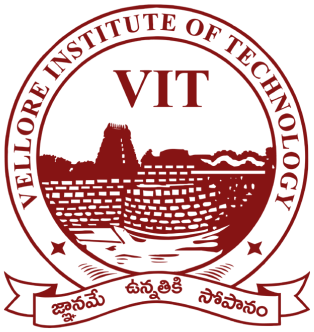
Network Analysis using Pcap Xray Tool

SUBMITTED BY:
P RAMA RAMANA SHARMA



VIT-AP UNIVERSITY

What are pcap files ?

PCAP files are a type of data file that contain captured network traffic. They are commonly used for network analysis, troubleshooting, and security auditing.
The acronym "PCAP" stands for "Packet Capture," and the files typically have a .pcap or .pcapng file extension. They are created by network packet capture software, such as Wireshark, tcpdump, or other similar tools, and contain data about the network packets that were captured.
PCAP files can be used to analyse network traffic for a variety of purposes, including identifying network performance issues, troubleshooting network problems, detecting security threats and attacks, and performing forensic analysis. They are particularly useful for network administrators and security professionals who need to monitor and analyze network traffic in real-time or after the fact.

Installing Pcap Xray Tool

Step 1: If you are using Mac install python using: sudo apt-get install python3

```
ramz@ramz-Standard-PC-Q35-ICH9-2009:~$ sudo apt-get install python3
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.10.6-1~22.04).
python3 set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 111 not upgraded.
ramz@ramz-Standard-PC-Q35-ICH9-2009:~$
```

On Ubuntu use:
apt install python3-pip
apt install python3-tk
apt install graphviz
apt install python3-pil python3-pil.imagetk

```
ramz@ramz-Standard-PC-Q35-ICH9-2009:~$ sudo su
root@ramz-Standard-PC-Q35-ICH9-2009:/home/ramz# apt install python3-pip
apt install python3-tk
apt install graphviz
apt install python3-pil python3-pil.imagetk
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
```

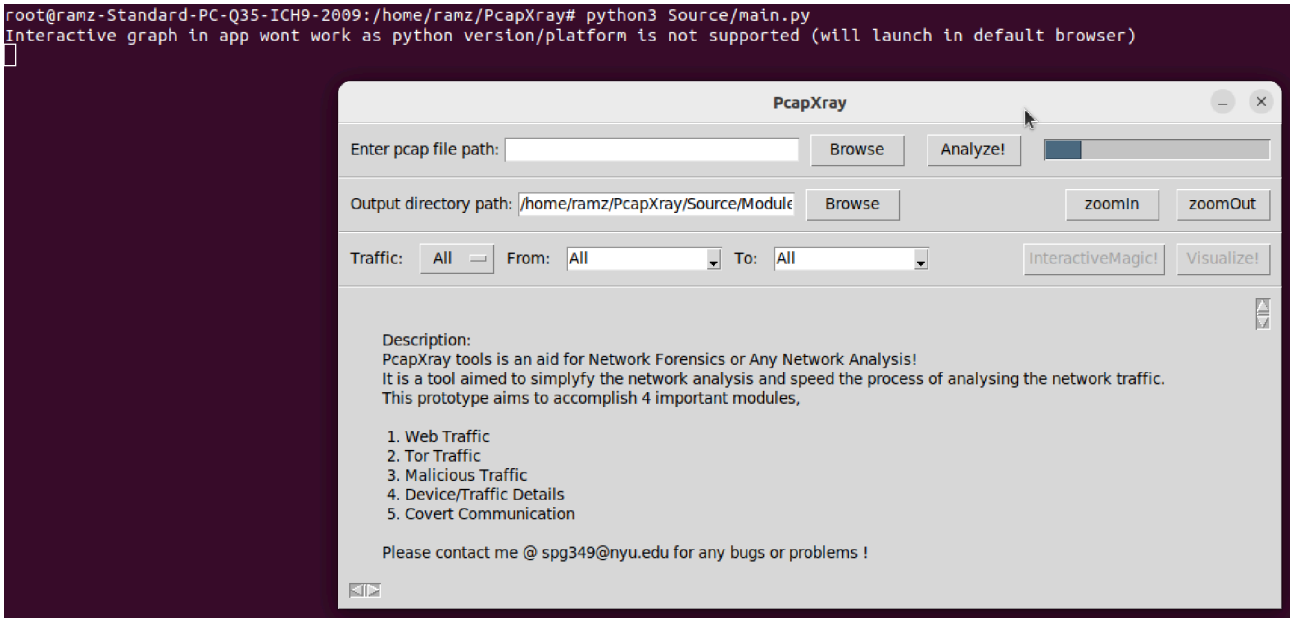
Step 2: Clone the GitHub repository, **https://github.com/Srinivas11789/PcapXray**
git clone https://github.com/Srinivas11789/PcapXray.git

```
root@ramz-Standard-PC-Q35-ICH9-2009:/home/ramz# git clone https://github.com/Srinivas11789/PcapXray.git
Cloning into 'PcapXray'...
remote: Enumerating objects: 1704, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 1704 (delta 3), reused 9 (delta 2), pack-reused 1689
Receiving objects: 100% (1704/1704), 115.75 MiB | 2.72 MiB/s, done.
Resolving deltas: 100% (975/975), done.
root@ramz-Standard-PC-Q35-ICH9-2009:/home/ramz#
```

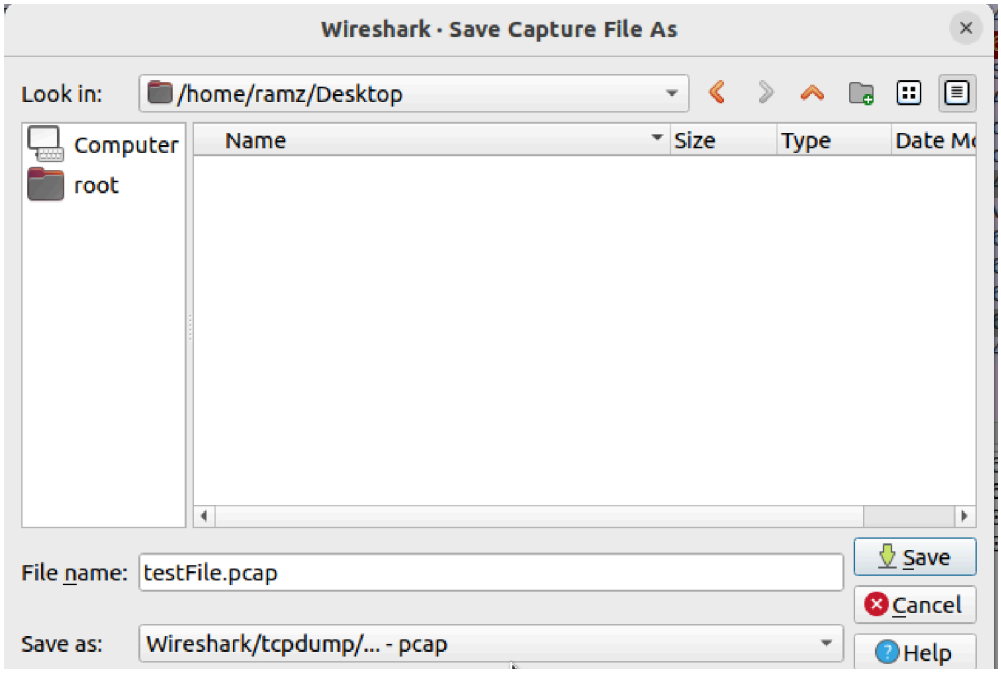
Change the directory to clone folder
Install the requirements using: sudo pip3 install -r requirements.txt

```
root@ramz-Standard-PC-Q35-ICH9-2009:/home/ramz# ls
Desktop  Documents  Downloads  Music  PcapXray  Pictures  Public  snap  Templates  Videos
root@ramz-Standard-PC-Q35-ICH9-2009:/home/ramz# cd PcapXray/
root@ramz-Standard-PC-Q35-ICH9-2009:/home/ramz/PcapXray# pip3 install -r requirements.txt
Collecting scapy
  Downloading scapy-2.5.0.tar.gz (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 2.1 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting pyshark
  Downloading pyshark-0.5.3-py3-none-any.whl (41 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41.1/41.1 KB 3.5 MB/s eta 0:00:00
Collecting ipwhois
  Downloading ipwhois-1.2.0-py2.py3-none-any.whl (73 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 73.5/73.5 KB 3.4 MB/s eta 0:00:00
Collecting netaddr
```

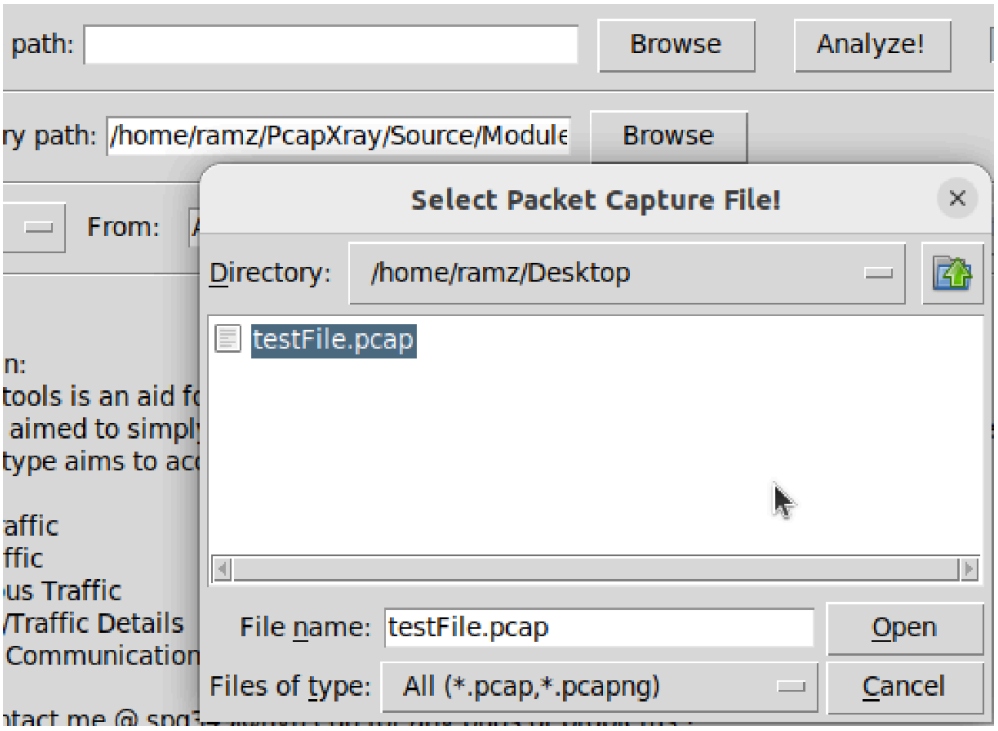
Step 3: Start the application using: sudo python3 Source/main.py



Step 4: Open Wireshark, and save the capture the packets and save the file as cap file.



Step 5: Click in Browse in PcapXray application and select the saved pcap file.



Step 6: Once it is done click on Analyze, now you can Visualise the Pcap files.

Thus you can use GUI based for Pcap Analyser.

Real Time Solution

Previous we saw analysis of a pcap file that is already saved, we can run the PcapXray tool continuously on pcap file that is generated by zeek, in this pcap it is continuously saved to the file so that analysis can be done continuously in a real time.

Lets first install zeek in Ubuntu as follows
Step 1: Update and Upgrade the ubuntu machine using: sudo apt-get update
sudo apt-get upgrade

```
ramz@ramz:~$ sudo apt-get update
sudo apt-get upgrade
[sudo] password for ramz:
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:2 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://download.opensuse.org/repositories/security:/zeek/xUbuntu_22.04  InRelease [1,554 B]
Err:3 http://download.opensuse.org/repositories/security:/zeek/xUbuntu_22.04  InRelease
  The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 69D1B2AAEE3D166A
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:5 https://ppa.launchpadcontent.net/oisf/suricata-stable/ubuntu jammy InRelease
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [293 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1,069 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [800 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [156 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [41.4 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [9,144 B]
Get:13 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [729 kB]
Get:14 http://in.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [487 kB]
Get:15 http://security.ubuntu.com/ubuntu jammy-security/universe i386 Packages [524 kB]
Get:16 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [101 kB]
Get:17 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [14.3 kB]
Get:18 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [611 kB]
Get:19 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [912 kB]
Get:20 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [18.5 kB]
Get:21 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [14.3 kB]
Get:22 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 Metadata [269 kB]
Get:23 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [18.7 kB]
Get:24 http://in.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 DEP-11 Metadata [940 B]
Get:25 http://in.archive.ubuntu.com/ubuntu jammy-backports/main amd64 DEP-11 Metadata [7,976 B]
Get:26 http://in.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 DEP-11 Metadata [12.9 kB]
Reading package lists... Done
```

Step 2: Install the dependencies using: sudo apt-get install cmake make gcc g++ flex bison libpcap-dev libssl-dev python3-dev swig zlib1g-dev

```
ramz@ramz:~$ sudo apt-get install cmake make gcc g++ flex bison libpcap-dev libssl-dev python3-dev swig zlib1g-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
bison is already the newest version (2:3.8.2+dfsg-1build1).
flex is already the newest version (2.6.4-8build2).
g++ is already the newest version (4:11.2.0-1ubuntu1).
gcc is already the newest version (4:11.2.0-1ubuntu1).
libpcap-dev is already the newest version (1.10.1-4build1).
make is already the newest version (4.3-4.1build1).
swig is already the newest version (4.0.2-1ubuntu1).
cmake is already the newest version (3.22.1-1ubuntu1.22.04.1).
libssl-dev is already the newest version (3.0.2-0ubuntu1.9).
python3-dev is already the newest version (3.10.6-1~22.04).
zlib1g-dev is already the newest version (1:1.2.11.dfsg-2ubuntu9.2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ramz@ramz:~$
```

Step 3: Go to **https://zeek.org/get-zeek/** and download source code of zeek.
Step 4: Change the directory to downloaded folder, and run tar -xzf zeek-5.0.8.tar.gz

```
ramz@ramz:~$ cd Downloads/
ramz@ramz:~/Downloads$ tar -xzf zeek-5.0.8.tar.gz
ramz@ramz:~/Downloads$
```

Step 5: Change the directory to zeek folder and run ./configure

```
ramz@ramz:~/Downloads/zeek-5.0.8$ ./configure
Build Directory : build
Source Directory: /home/ramz/Downloads/zeek-5.0.8
Using cmake version 3.22.1

-- The C compiler identification is GNU 11.3.0
-- The CXX compiler identification is GNU 11.3.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Performing Test test_arch_x64
-- Performing Test test_arch_x64 - Success
-- Performing Test test_arch_aarch64
-- Performing Test test_arch_aarch64 - Failed
-- Performing Test test_arch_arm
-- Performing Test test_arch_arm - Failed
-- Performing Test test_arch_power
```

Step 6: Now run make and then sudo make install

```
ramz@ramz:~/Downloads/zeek-5.0.8$ make
make -C build all
make[1]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build'
make[2]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build'
make[3]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build'
make[3]: Leaving directory '/home/ramz/Downloads/zeek-5.0.8/build'
make[3]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build'
[  0%] Building libkqueue
gmake[4]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build/libkqueue-build'
gmake[5]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build/libkqueue-build'
gmake[6]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build/libkqueue-build'
gmake[6]: Leaving directory '/home/ramz/Downloads/zeek-5.0.8/build/libkqueue-build'
gmake[6]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build/libkqueue-build'
[  6%] Building C object CMakeFiles/kqueue.dir/src/common/debug.c.o
[ 12%] Building C object CMakeFiles/kqueue.dir/src/common/filter.c.o
[ 18%] Building C object CMakeFiles/kqueue.dir/src/common/kevent.c.o
[ 25%] Building C object CMakeFiles/kqueue.dir/src/common/knote.c.o
[ 31%] Building C object CMakeFiles/kqueue.dir/src/common/kqueue.c.o
[ 37%] Building C object CMakeFiles/kqueue.dir/src/common/libkqueue.c.o
[ 43%] Building C object CMakeFiles/kqueue.dir/src/common/map.c.o
[ 50%] Building C object CMakeFiles/kqueue.dir/src/linux/platform.c.o
[ 56%] Building C object CMakeFiles/kqueue.dir/src/linux/read.c.o
[ 62%] Building C object CMakeFiles/kqueue.dir/src/linux/signal.c.o
[ 68%] Building C object CMakeFiles/kqueue.dir/src/linux/timer.c.o
[ 75%] Building C object CMakeFiles/kqueue.dir/src/linux/user.c.o
[ 81%] Building C object CMakeFiles/kqueue.dir/src/linux/vnode.c.o
[ 87%] Building C object CMakeFiles/kqueue.dir/src/linux/write.c.o
[ 93%] Building C object CMakeFiles/kqueue.dir/src/linux/proc.c.o
[100%] Linking C static library libkqueue.a
gmake[6]: Leaving directory '/home/ramz/Downloads/zeek-5.0.8/build/libkqueue-build'
```

```
ramz@ramz:~/Downloads/zeek-5.0.8$ sudo make install
[sudo] password for ramz:
make -C build all
make[1]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build'
make[2]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build'
make[3]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build'
make[3]: Leaving directory '/home/ramz/Downloads/zeek-5.0.8/build'
make[3]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build'
[  0%] Building libkqueue
gmake[4]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build/libkqueue-build'
gmake[5]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build/libkqueue-build'
gmake[6]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build/libkqueue-build'
Consolidate compiler generated dependencies of target kqueue
gmake[6]: Leaving directory '/home/ramz/Downloads/zeek-5.0.8/build/libkqueue-build'
[100%] Built target kqueue
gmake[5]: Leaving directory '/home/ramz/Downloads/zeek-5.0.8/build/libkqueue-build'
gmake[4]: Leaving directory '/home/ramz/Downloads/zeek-5.0.8/build/libkqueue-build'
[  1%] Completed 'project_kqueue'
make[3]: Leaving directory '/home/ramz/Downloads/zeek-5.0.8/build'
[  1%] Built target project_kqueue
make[3]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build'
Consolidate compiler generated dependencies of target binpac_lib
make[3]: Leaving directory '/home/ramz/Downloads/zeek-5.0.8/build'
[  1%] Built target binpac_lib
make[3]: Entering directory '/home/ramz/Downloads/zeek-5.0.8/build'
```

Step 7: Once the above commands are done executing now head to bashrc and add the path as follows, **export PATH=/usr/local/zeek/bin:$PATH** and execute source ~/.bashrc
Check the zeek path using, which zeek and version using zeek —version

```
ramz@ramz:~$ nano ~/.bashrc
ramz@ramz:~$ source ~/.bashrc
ramz@ramz:~$ which zeek
/usr/local/zeek/bin/zeek
ramz@ramz:~$ zeek --version
zeek version 5.0.8
```

Step 8: Change the directory to /usr/local/zeek/etc folder using: cd /usr/local/zeek/etc
Use nano to edit node.cfg, Change the interface to the system assigned interface, use ifconfig to find the interface in my case it is enp0s1

```
ramz@ramz:/usr/local/zeek/etc$ ifconfig
enp0s1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.19.61.203  netmask 255.255.224.0  broadcast 172.19.63.255
        inet6 fe80::b063:6883:632a:a0ce  prefixlen 64  scopeid 0x20<link>
        ether ce:de:24:9b:3e:2c  txqueuelen 1000  (Ethernet)
        RX packets 9866  bytes 888946 (888.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 189  bytes 19734 (19.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
  GNU nano 6.2                                                    node.cfg
# Example ZeekControl node configuration.
#
# This example has a standalone node ready to go except for possibly changing
# the sniffing interface.

# This is a complete standalone configuration.  Most likely you will
# only need to change the interface.
[zeek]
type=standalone
host=localhost
interface=enp0s1

## Below is an example clustered configuration. If you use this,
## remove the [zeek] node above.

#[logger-1]
#type=logger
#host=localhost
#
#[manager]
#type=manager
#host=localhost
#
#[proxy-1]
#type=proxy
#host=localhost
#
#[worker-1]
#type=worker
#host=localhost
#interface=eth0
#
#[worker-2]
#type=worker
#host=localhost
#interface=eth0
```

Now change directory to bin file as shown below and run sudo ./zeekctl check

```
ramz@ramz:/usr/local/zeek/etc$ cd ..
ramz@ramz:/usr/local/zeek$ cd bin
ramz@ramz:/usr/local/zeek/bin$ sudo ./zeekctl check
WARNING: ********************************************************************************
WARNING: You're using Linux with the default ZeekPort setting 47760. This configuration
WARNING: is know to cause persistent worker failures with error messages as follows:
WARNING:
WARNING:     error in <...>/cluster/setup-connections.zeek, lines 94-96: Failed to listen on INADDR_ANY:47764 (...)
WARNING:
WARNING: Starting with Zeek 5.2, the default ZeekPort used by zeekctl will
WARNING: change from 47760 to 27760 in order to avoid potential port collisions
WARNING: with other processes due to 47760 falling right into Linux's default
WARNING: ephemeral port range.
WARNING:
WARNING: Consider changing the ZeekPort option in your zeekctl.cfg to 27760
WARNING: now to prepare for this change. Doing so will silence this warning.
WARNING:
WARNING:     ZeekPort = 27760
WARNING:
WARNING: Note, if you're employing strict firewall rules between Zeek nodes,
WARNING: you'll likely need to update these rules. If you're using Zeek on
WARNING: a single physical host, no further action should be required.
WARNING: If possible do test the change in a non-production environment.
WARNING:
WARNING: To silence this warning without changing the ZeekPort option,
WARNING: set zeek_port_warning.disable = 1 in zeekctl.cfg.
WARNING:
WARNING: See the following PR for more details:
WARNING:     https://github.com/zeek/zeekctl/pull/41
WARNING:
WARNING: Feel free to reach out on zeekorg.slack.com or community.zeek.org if
WARNING: you have any questions around this change.
WARNING: ********************************************************************************
Hint: Run the zeekctl "deploy" command to get started.
zeek scripts are ok.
```

Once you get zeek scripts are ok as output run sudo ./zeekctl deploy

```
ramz@ramz:/usr/local/zeek/bin$ sudo ./zeekctl deploy
WARNING: ********************************************************************************
WARNING: You're using Linux with the default ZeekPort setting 47760. This configuration
WARNING: is know to cause persistent worker failures with error messages as follows:
WARNING:
WARNING:     error in <...>/cluster/setup-connections.zeek, lines 94-96: Failed to listen on INADDR_ANY:47764 (...)
WARNING:
WARNING: Starting with Zeek 5.2, the default ZeekPort used by zeekctl will
WARNING: change from 47760 to 27760 in order to avoid potential port collisions
WARNING: with other processes due to 47760 falling right into Linux's default
WARNING: ephemeral port range.
WARNING:
WARNING: Consider changing the ZeekPort option in your zeekctl.cfg to 27760
WARNING: now to prepare for this change. Doing so will silence this warning.
WARNING:
WARNING:     ZeekPort = 27760
WARNING:
WARNING: Note, if you're employing strict firewall rules between Zeek nodes,
WARNING: you'll likely need to update these rules. If you're using Zeek on
WARNING: a single physical host, no further action should be required.
WARNING: If possible do test the change in a non-production environment.
WARNING:
WARNING: To silence this warning without changing the ZeekPort option,
WARNING: set zeek_port_warning.disable = 1 in zeekctl.cfg.
WARNING:
WARNING: See the following PR for more details:
WARNING:     https://github.com/zeek/zeekctl/pull/41
WARNING:
WARNING: Feel free to reach out on zeekorg.slack.com or community.zeek.org if
WARNING: you have any questions around this change.
WARNING: ********************************************************************************
checking configurations ...
installing ...
creating policy directories ...
installing site policies ...
generating standalone-layout.zeek ...
generating local-networks.zeek ...
generating zeekctl-config.zeek ...
generating zeekctl-config.sh ...
stopping ...
stopping zeek ...
starting ...
starting zeek ...
ramz@ramz:/usr/local/zeek/bin$
```

To check the status of zeek run sudo ./zeekctl status

```
ramz@ramz:/usr/local/zeek/bin$ sudo ./zeekctl status
WARNING: ********************************************************************************
WARNING: You're using Linux with the default ZeekPort setting 47760. This configuration
WARNING: is know to cause persistent worker failures with error messages as follows:
WARNING:
WARNING:     error in <...>/cluster/setup-connections.zeek, lines 94-96: Failed to listen on INADDR_ANY:47764 (...)
WARNING:
WARNING: Starting with Zeek 5.2, the default ZeekPort used by zeekctl will
WARNING: change from 47760 to 27760 in order to avoid potential port collisions
WARNING: with other processes due to 47760 falling right into Linux's default
WARNING: ephemeral port range.
WARNING:
WARNING: Consider changing the ZeekPort option in your zeekctl.cfg to 27760
WARNING: now to prepare for this change. Doing so will silence this warning.
WARNING:
WARNING:     ZeekPort = 27760
WARNING:
WARNING: Note, if you're employing strict firewall rules between Zeek nodes,
WARNING: you'll likely need to update these rules. If you're using Zeek on
WARNING: a single physical host, no further action should be required.
WARNING: If possible do test the change in a non-production environment.
WARNING:
WARNING: To silence this warning without changing the ZeekPort option,
WARNING: set zeek_port_warning.disable = 1 in zeekctl.cfg.
WARNING:
WARNING: See the following PR for more details:
WARNING:     https://github.com/zeek/zeekctl/pull/41
WARNING:
WARNING: Feel free to reach out on zeekorg.slack.com or community.zeek.org if
WARNING: you have any questions around this change.
WARNING: ********************************************************************************
Name         Type       Host          Status    Pid    Started
zeek         standalone localhost     running   2401   03 May 09:12:19
ramz@ramz:/usr/local/zeek/bin$
```

Now to capture packets in same directory run, sudo ./zeek -i enp0s1 -C -w /home/ramz/capture.pcap Here /home/ramz/capture.pcap is the destination to save capture.pcap, change the destination as per your requirements.

Step 9: As we are successful in capturing packets now lets clone git hub repository of AutomatedPcap tool which can be found at
https://github.com/ramz-021002/AutomatedPcapXray
Use git clone **https://github.com/ramz-021002/AutomatedPcapXray.git** to clone this repository.

```
ramz@ramz:~$ git clone https://github.com/ramz-021002/AutomatedPcapXray.git
Cloning into 'AutomatedPcapXray'...
remote: Enumerating objects: 55, done.
remote: Counting objects: 100% (55/55), done.
remote: Compressing objects: 100% (48/48), done.
remote: Total 55 (delta 11), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (55/55), 330.95 KiB | 1.33 MiB/s, done.
Resolving deltas: 100% (11/11), done.
ramz@ramz:~$
```

Before we start running the tool we need make some modifications in the codes,
Go to AutomatedPcapXray folder then go to Module folder there you can find a python file named user_interface.py in this file and change self.pcap_file and self.destination_report as per your choice as shown below. Note pcap file must be present at the given location.

```
class pcapXrayCLI():
    def __init__(self):
        self.pcap_file = '/home/ramz/AutomatedPcapXray/zeek/capture.pcap'
        self.destination_report = '/home/ramz/AutomatedPcapXray/Module/Report'

        self.engine = str()
        self.engine = 'scapy'

        self.option = str()
        self.options = {'All', 'HTTP', 'HTTPS', 'Tor', 'Malicious', 'ICMP', 'DNS'}
```

Now go to plot_lan_network.py and change pcapfile as done in user_interface.py, shown below.

```
598         r.render()
599         interactive_graph.save_graph(self.filename+".html")
600
601 def main():
602     # draw example
603     import pcap_reader
604     pcapfile = pcap_reader.PcapEngine('/home/ramz/AutomatedPcapXray/zeek/capture.pcap',
    "scapy")
605     details = communication_details_fetch.trafficDetailsFetch("sock")
606     import sys
607     print(sys.path[0])
608     network = plotLan("test", sys.path[0])
609
610 main()
```

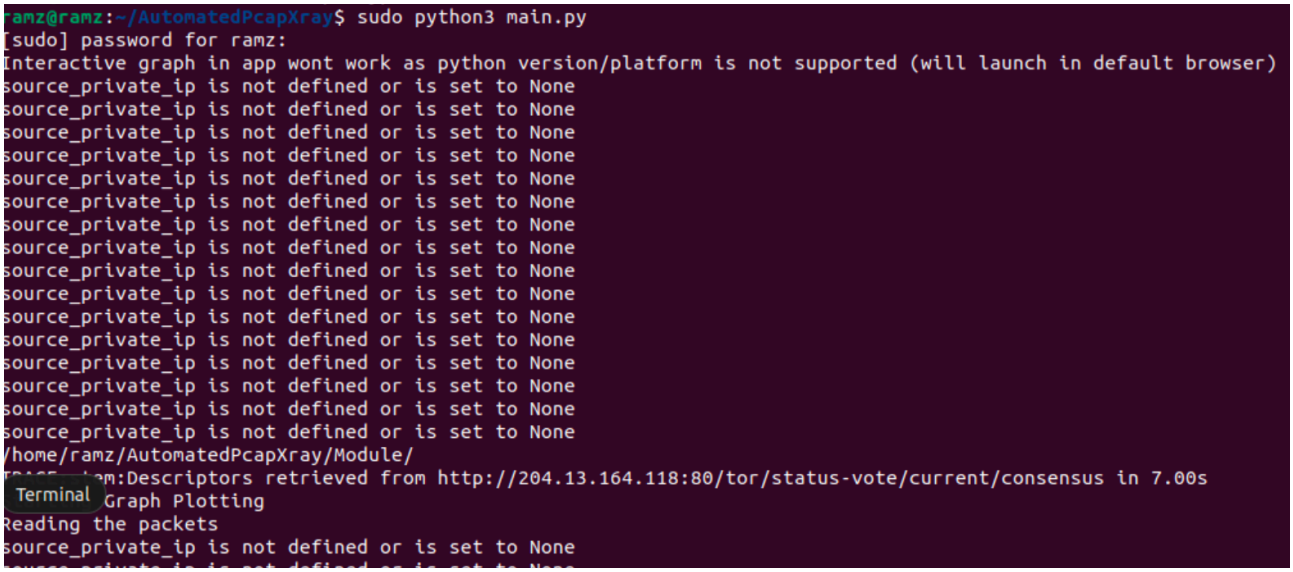
Step 10: Now in terminal change to directory to AutomatePcapXray using cd AutomatedPcapXray/
Let us install all the requirements now use, sudo pip3 install -r requirements.txt
If you don't have pip3 install use sudo apt install python3-pip to install pip3.

```
ramz@ramz:~/AutomatedPcapXray$ sudo pip3 install -r requirements.txt
Collecting scapy
  Downloading scapy-2.5.0.tar.gz (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 1.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting pyshark
  Downloading pyshark-0.6-py3-none-any.whl (41 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 41.4/41.4 KB 655.0 kB/s eta 0:00:00
Requirement already satisfied: ipwhois in /usr/local/lib/python3.10/dist-packages (from -r requirements.txt (line 12)) (1.2.0)
Collecting netaddr
  Downloading netaddr-0.8.0-py2.py3-none-any.whl (1.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 2.8 MB/s eta 0:00:00
Requirement already satisfied: pillow in /usr/lib/python3/dist-packages (from -r requirements.txt (line 18)) (9.0.1)
Collecting stem
  Downloading stem-1.8.1.tar.gz (2.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.9/2.9 MB 3.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting networkx
  Downloading networkx-3.1-py3-none-any.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 4.9 MB/s eta 0:00:00
Collecting graphviz
  Downloading graphviz-0.20.1-py3-none-any.whl (47 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 47.0/47.0 KB 815.0 kB/s eta 0:00:00
Collecting matplotlib
  Downloading matplotlib-3.7.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.6/11.6 MB 6.4 MB/s eta 0:00:00
Collecting cefpython3
  Downloading cefpython3-66.0-py2.py3-none-manylinux1_x86_64.whl (79.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.6/79.6 MB 6.2 MB/s eta 0:00:00
Collecting pyvis
  Downloading pyvis-0.3.2-py3-none-any.whl (756 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 756.0/756.0 KB 6.4 MB/s eta 0:00:00
Requirement already satisfied: cryptography in /usr/lib/python3/dist-packages (from -r requirements.txt (line 33)) (3.4.8)
Collecting lxml
  Downloading lxml-4.9.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_24_x86_64.whl (7.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.1/7.1 MB 4.4 MB/s eta 0:00:00
Collecting packaging
  Downloading packaging-23.1-py3-none-any.whl (48 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.9/48.9 KB 666.6 kB/s eta 0:00:00
Collecting appdirs
  Downloading appdirs-1.4.4-py2.py3-none-any.whl (9.6 kB)
Collecting termcolor
  Downloading termcolor-2.3.0-py3-none-any.whl (6.9 kB)
Requirement already satisfied: dnspython<=2.0.0 in /usr/local/lib/python3.10/dist-packages (from ipwhois->-r requirements.txt (line 12)) (2.0.0)
Collecting cycler>=0.10
```

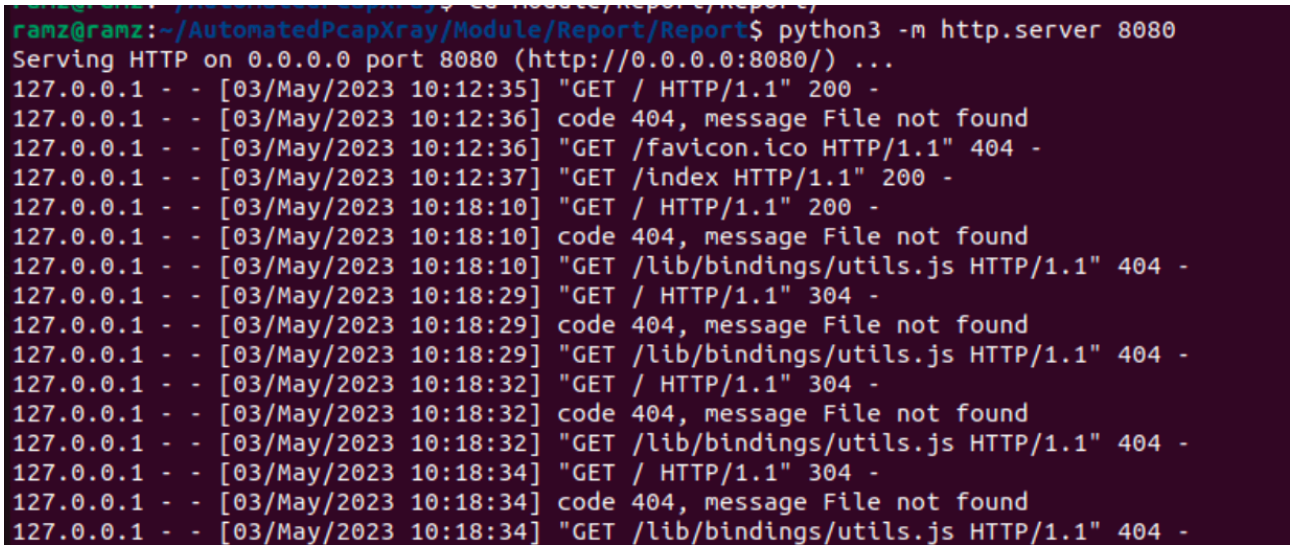
Now for running the tool capture packets to the destination file changed in the code as shown below,

```
ramz@ramz:/usr/local/zeek/bin$ sudo ./zeek -i enp0s1 -C -w /home/ramz/AutomatedPcapXray/zeek/capture.pcap
[sudo] password for ramz:
listening on enp0s1
```

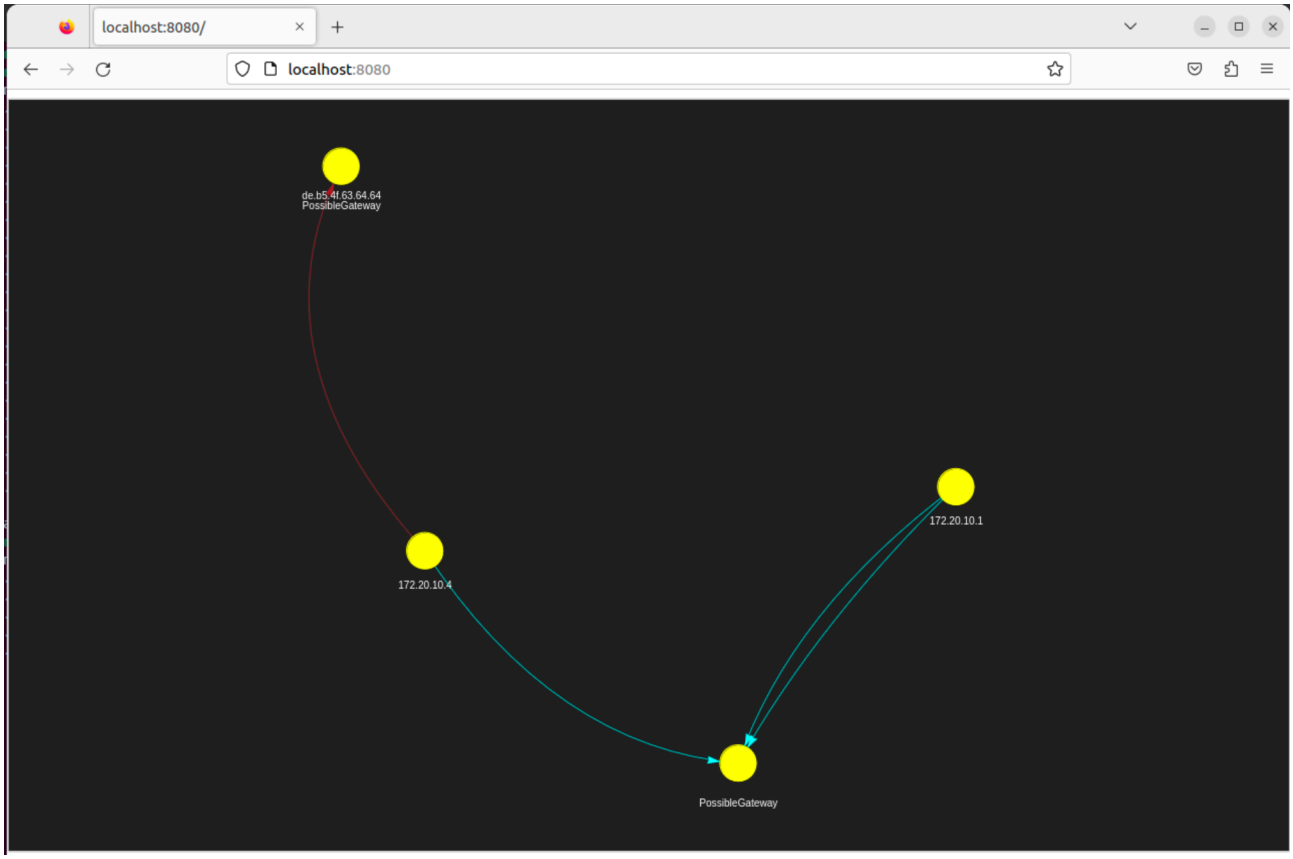
Step 11: Now in another terminal in directory AutomatedPcapXray run main.py using,
sudo python3 main.py
If you are thrown with any errors try installing packages using,
sudo apt install python3-pip
sudo apt install python3-tk
sudo apt install graphviz
sudo apt install python3-pil python3-pil.imagetk
After these installations try running the python file again.



To view the output using localhost change directory to /AutomatedPcapXray/Module/Report/Report and run python3 -m http.server 8080



Step 12: Now open Firefox and type localhost:8080



Now to visualise this is cloud or public internet space launch an EC2 ubuntu instance in AWS and connect to it using ssh in ubuntu local machine as shown below.

```
ramz@ramz:~$ ssh -i "local.pem" ubuntu@ec2-3-111-213-193.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-3-111-213-193.ap-south-1.compute.amazonaws.com (64:ff9b::36f:d5c1)' can't be established.
ED25519 key fingerprint is SHA256:0SM1OScubKSG0Gevi4KLmiRIVD90GxJYkRsvUBUvEBE.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-111-213-193.ap-south-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-1031-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed May  3 05:48:32 UTC 2023

  System load:  0.0               Processes:             95
  Usage of /:   23.5% of 7.57GB   Users logged in:       0
  Memory usage: 21%               IPv4 address for eth0: 172.31.39.100
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

     https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

42 updates can be applied immediately.
21 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Sat Apr 29 09:27:59 2023 from 106.217.146.43
ubuntu@ip-172-31-39-100:~$ sudo apt-get update
```

Step 13: Change the directory to /etc/ssh using commands below.

```
ubuntu@ip-172-31-39-100:~$ cd ..
ubuntu@ip-172-31-39-100:/home$ cd ..
ubuntu@ip-172-31-39-100:/$
ubuntu@ip-172-31-39-100:/$ ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  lost+found  media  mnt  opt  proc  root  run  sbin  snap  srv  sys  tmp  usr  var
ubuntu@ip-172-31-39-100:/$ cd etc/ssh
ubuntu@ip-172-31-39-100:/etc/ssh$
```

Edit sshd_config using nano or vi(Edit in sudo access), add GatewayPorts yes as shown below.

```
# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO
GatewayPorts yes
```

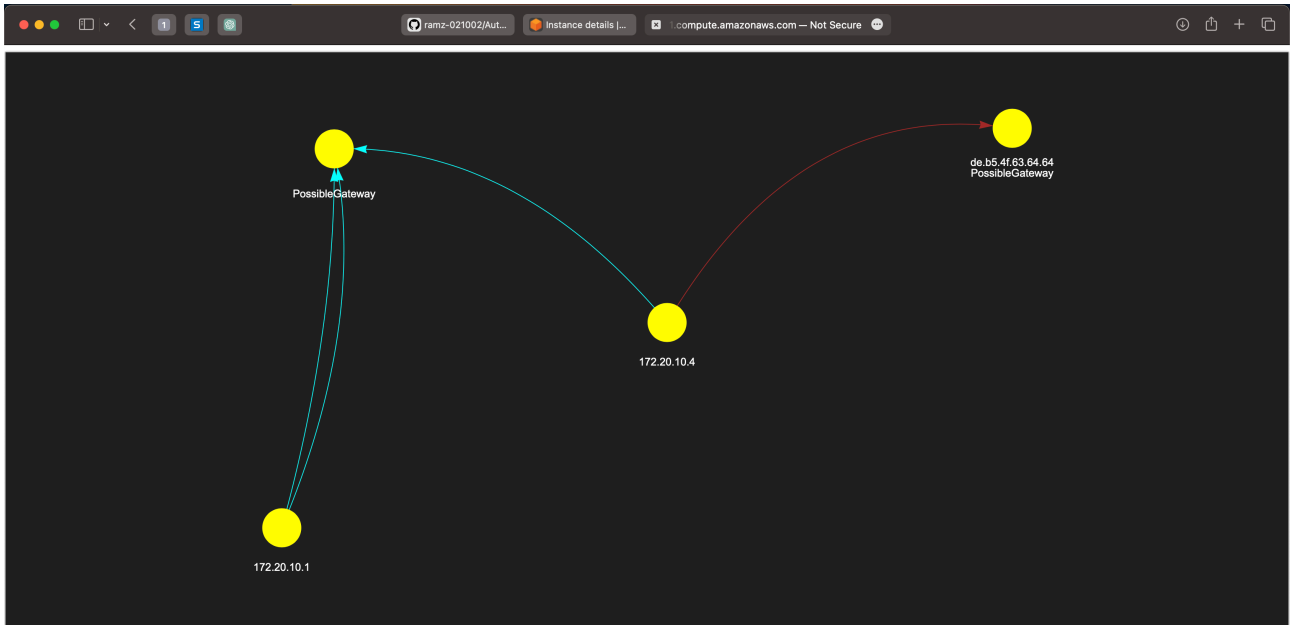
Now after saving the file restart ssh using: sudo service ssh restart

```
ubuntu@ip-172-31-39-100:/etc/ssh$ sudo service ssh restart
ubuntu@ip-172-31-39-100:/etc/ssh$
```

Step 14: Now create an SHH tunnel using the command: ssh -R :8080:localhost:8080 -i "<key>" ubuntu@<Public IPv4 DNS>

```
ramz@ramz:~$ ssh -R :8080:localhost:8080 -i "local.pem" ubuntu@ec2-3-111-213-193.ap-south-1.compute.amazonaws.com
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-1031-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed May  3 06:11:46 UTC 2023

  System load:  0.0               Processes:             98
  Usage of /:   28.2% of 7.57GB   Users logged in:       1
  Memory usage: 27%               IPv4 address for eth0: 172.31.39.100
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

     https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

22 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


*** System restart required ***
Last login: Wed May  3 05:48:33 2023 from 152.58.196.155
ubuntu@ip-172-31-39-100:~$
```

Now access this page using <Public IPv4 DNS>:8080



References

[1] https://github.com/Srinivas11789/PcapXray (source tool)
[2] https://github.com/ramz-021002/AutomatedPcapXray (modified tool)
[3] https://docs.zeek.org/en/lts/install.html
[4] https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-ssh-tunnel-local.html