

2022

# Final Report Network Sniffer

CW258 CYBERCRIME & IT SECURITY  
ERLANDAS BACAUSKAS

INSTITUTE OF TECHNOLOGY CARLOW | Supervisor: Paul J. Barry

## Acknowledgements

I would like to express my gratitude to my supervisor, Paul J. Barry, for his feedbacks, comments and ongoing support throughout the year on this project. In addition, I would like to thank all my other lecturers who have helped me and provided great insight and advice throughout the year and helped shape me as a professional individual.

## Table of Contents

Acknowledgements.....	1
Abstract.....	3
Introduction .....	4
Project Description.....	4
Application Flow .....	5
Sniffer .....	5
Host Discovery .....	12
Port Scan .....	13
ARP Poison .....	14
DHCP Starvation.....	17
Denial of Service (DoS, DDoS) .....	17
Use Case Examples.....	18
ARP Attack Capturing Traffic from Target.....	18
Inspect packet flow during DHCP starvation attack .....	20
Inspect packet flow during DDoS attack.....	21
Original Design and Specifications Vs. Current Design and Specifications .....	23
Learning Outcomes .....	24
Technology.....	24
Personal .....	24
Achievement Review .....	25
Plagiarism Detection .....	26

## Abstract

The purpose of this report is to create a network analyzer tool which captures network traffic in real time and provides some network-based attacks. In this report I will outline the decisions made when completing this project, functionality of the application, differences between original design, what was implemented and key the outcomes that I have found while working on this project.

## Introduction

In this final report I will provide detailed analysis of my application flow. This includes network capturing, which will have detailed steps outlined how the application operates with visual guides provided. I will also provide visual guides and details for all the functionality included in the application such as host discovery, port scan, ARP poisoning, DHCP starvation and Denial of Service (DoS/DDoS). I will also provide a step-by-step use case examples to achieve the desired outcomes. I will also discuss my original design and specification of my application and compare this to what I was able to achieve. This will be accompanied by my learning outcomes that I have achieved while completing this project, both technological and personal learning outcomes.

## Project Description

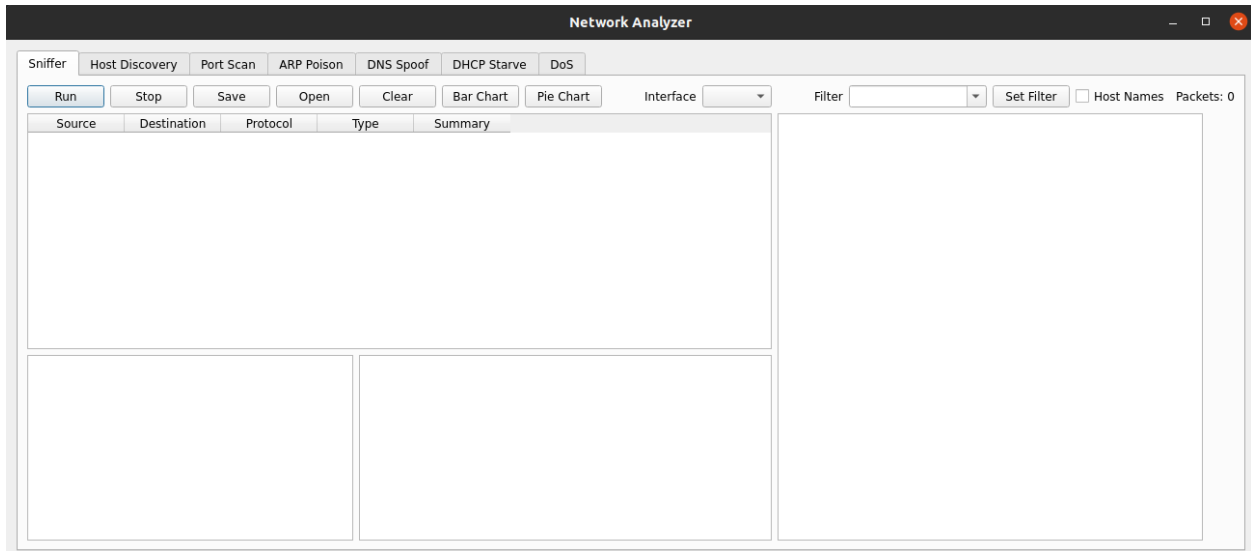
The network analyzer application is capable of capturing network traffic on any chosen interface that is available on a host computer, thus the sniffer component also is able to save and open network captured files in a well known pcap format. Pcap files can also be analyzed with other tools such as Wireshark or Tcpcap. The sniffer component also has a variety of filtering options and is also capable of translating IP addresses to host names. In the details panel, packets can be analyzed by its fields.

The host discovery component is able to identify alive end devices on any given network or just checking if given end device is active. The network analyzer has the functionality to scan for opened ports for a given IP address or domain name. It allows for the range to be specified.

This application is also able to perform OSI layer 2 attacks, such as ARP poisoning, DHCP starvation and DoS/DDoS attacks. The ARP attack creates ARP packets and send one to a gateway and others to the target. This attack can also be utilized for DoS attacks when the packet forwarding is disabled. A DHCP starvation attack broadcasts forged DHCP requests until whereby servers are exhausted. Finally, DoS/DDoS created multiple threads with random source IP's and ports thus constantly sending packets to a targeted IP or domain.

## Application Flow

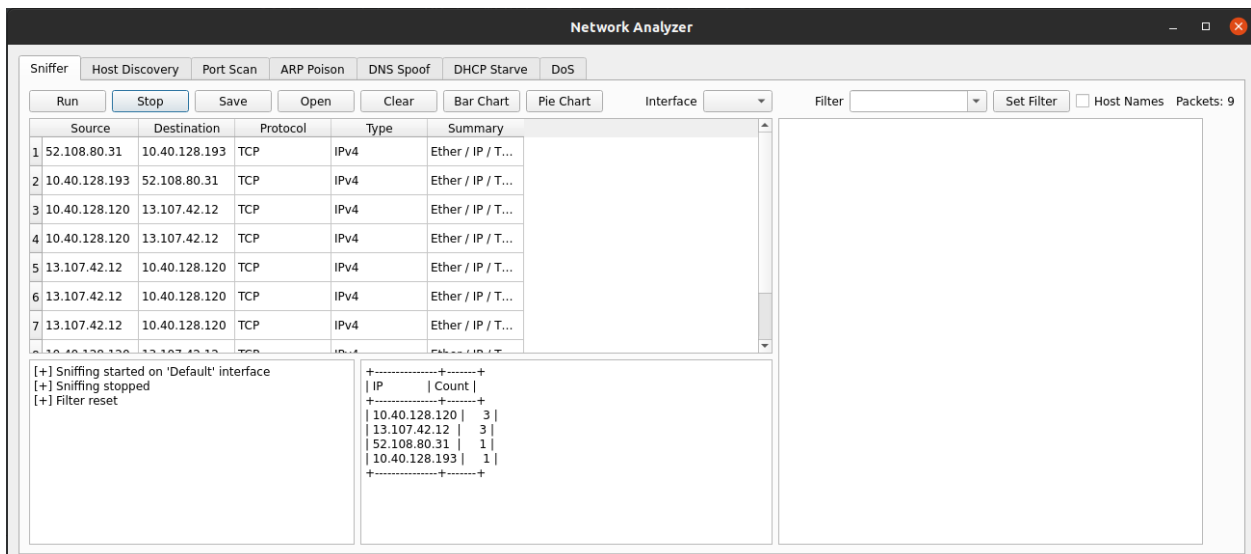
The below will outline the Network Sniffer application flow in detail. All functionality and options will be specified under each category with examples of usage.



## Sniffer

Sniffer option works by capturing traffic on a given or default interface and filter options can be applied. The application has an ability to import and export captured files in a well-known capture format pcap for later analysis or by using different tools. Below will be examples of different options that the sniffer part of an application will have.

**Run:** this button will start application in the sniffer mode by capturing active traffic on a host that it is running on. As currently no option has been chosen for an active interface the application will take currently active default interface to capture traffic on.



1	52.114.92.45	192.168.1.14	TCP	IPv4	Ether / IP / TCP 52.114.92.45:https > 192.168.1.14:54868 PA / Raw
---	--------------	--------------	-----	------	---

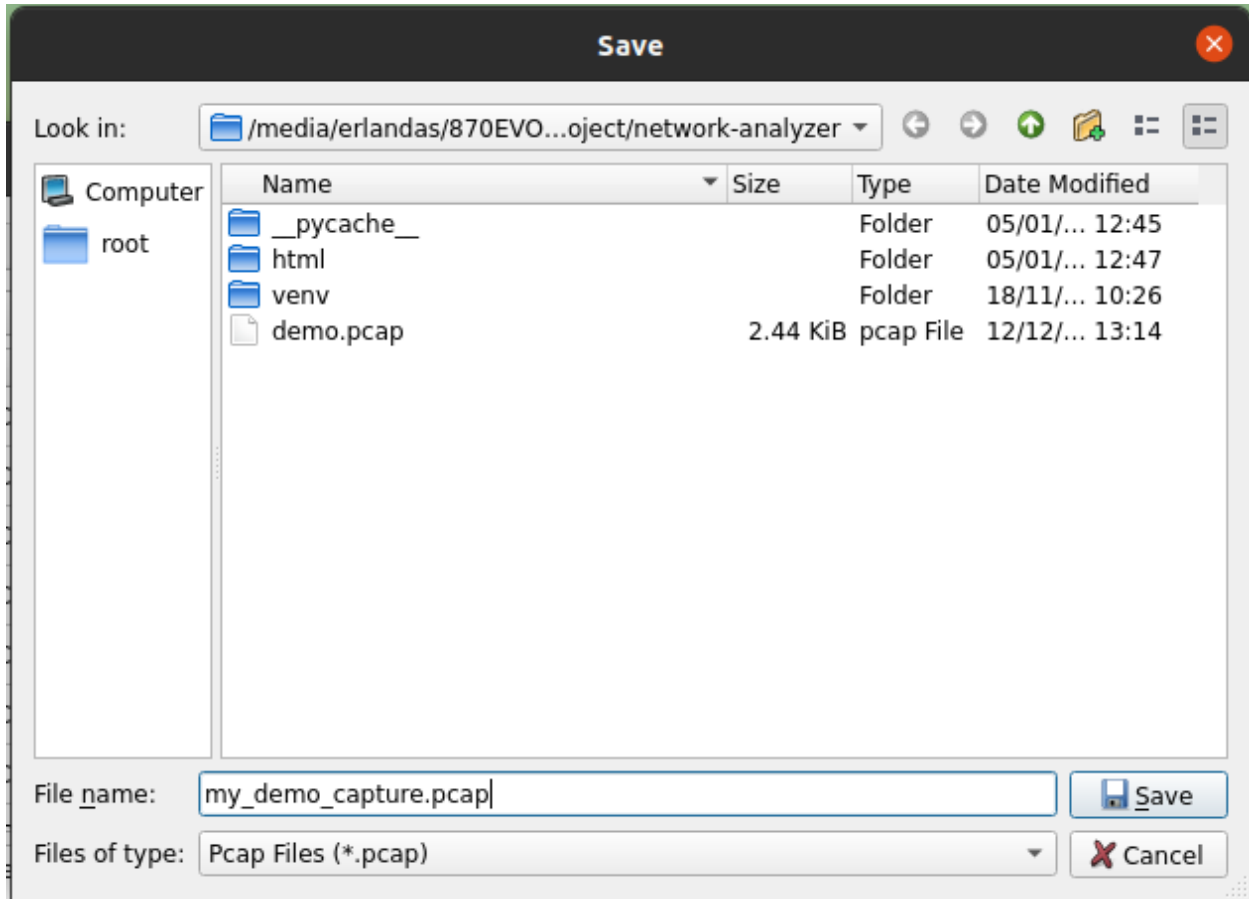
**Details Field:** at the right-hand side of the application, details field will outline all the components of a selected packet.

```
###[ Ethernet ]###
  dst    = a0:a4:c5:1c:0b:88
  src    = e4:fb:5d:19:2b:df
  type   = IPv4
###[ IP ]###
  version = 4
  ihl     = 5
  tos     = 0x0
  len     = 373
  id      = 58162
  flags   = DF
  frag    = 0
  ttl     = 114
  proto   = tcp
  chksum  = 0xd1fa
  src     = 52.114.92.45
  dst     = 192.168.1.14
  \options \
###[ TCP ]###
  sport   = https
  dport   = 54868
  seq     = 2438236270
  ack     = 996908879
  dataofs = 5
  reserved = 0
  flags   = PA
  window  = 2049
  chksum  = 0xedd2
  urgptr  = 0
  options = []
###[ Raw ]###
  load    = '\x17\x03\x03\x01H\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00/\^\x8dj\x04\x04\x06
\x0c\x07\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20\x21
~\x02+\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a
\x01"%\xaaU\xfa\xab\x89Yj\x0f3[\x0fb2وQ\x19|\x07Z\x11\x07|\x0f8c\x0a0A6s|\x0f
\x035ē.@|\x04Z|\x05\x08o*eY|\x0b1|\x07|\x0fb|\x0c1|\x0da|\x0d3|\x014|\x09eq"\x0f|\x0eeK|\x02|\x0e2|
\x086|\x0d9VTη/7"ω"\x0bc|\x0c3_|\x0ce|\x0cb|\x010G|\x09d|\x0de4FxG|\x0ae|\x0eb|\x0be|\x05|\x09cĭAŷ|
\x0e3?|\x0fai|\x017|\x0e3|\x0fd|\x0b1|\x05|\x0bayhu|\x09a6|\x0yLD#\x015~|\x03|\x088|\x0ac|\x0a8|\x0c25|
\x0b2.*|\x0e|\x01d|\x09fĉ3[|\x0e8|\x013|\x0de|\x00|\x090|\x0875|\x096|\x09a}|\x01f|\x093|\x0f0:4|\x010ZC|
\x0e9Z&0iP|\x084|\x086|\x082_|\x0f1|\x013|\x098|\x0f0|\x0d9M|\x0e1"\x0feL|\x080|\x010|\x0ef|\x0c0V|\x0db|
|\x0fb7|\x019D|\x0f0@|\x0b3aC|\x013gXB4|\x02|\x02|\x08b|\x094)|\x0ee7[##|eR|\x0e1|\x0cfe|\x0d2|
|\x0c2|t|\x012|\x0f9|\x0c9-|\x0f2|\x087|\x0a6|\x0fd|\x0bc| |\x0d3x9|\x0d6q=|\x0f5|\x08b%|\x086E|\x087|\x0a0|
|\x0f3|\x01e0|\x0c|\x0bb|\x013|\x0c7N'
```

**Stop:** this button will stop current capture session, also it will append action, letting the user know what the application does at the current moment. In addition, it will reset filter options for future use.

```
[+] Sniffing started on 'Default' interface
[+] Sniffing stopped
[+] Filter reset
```

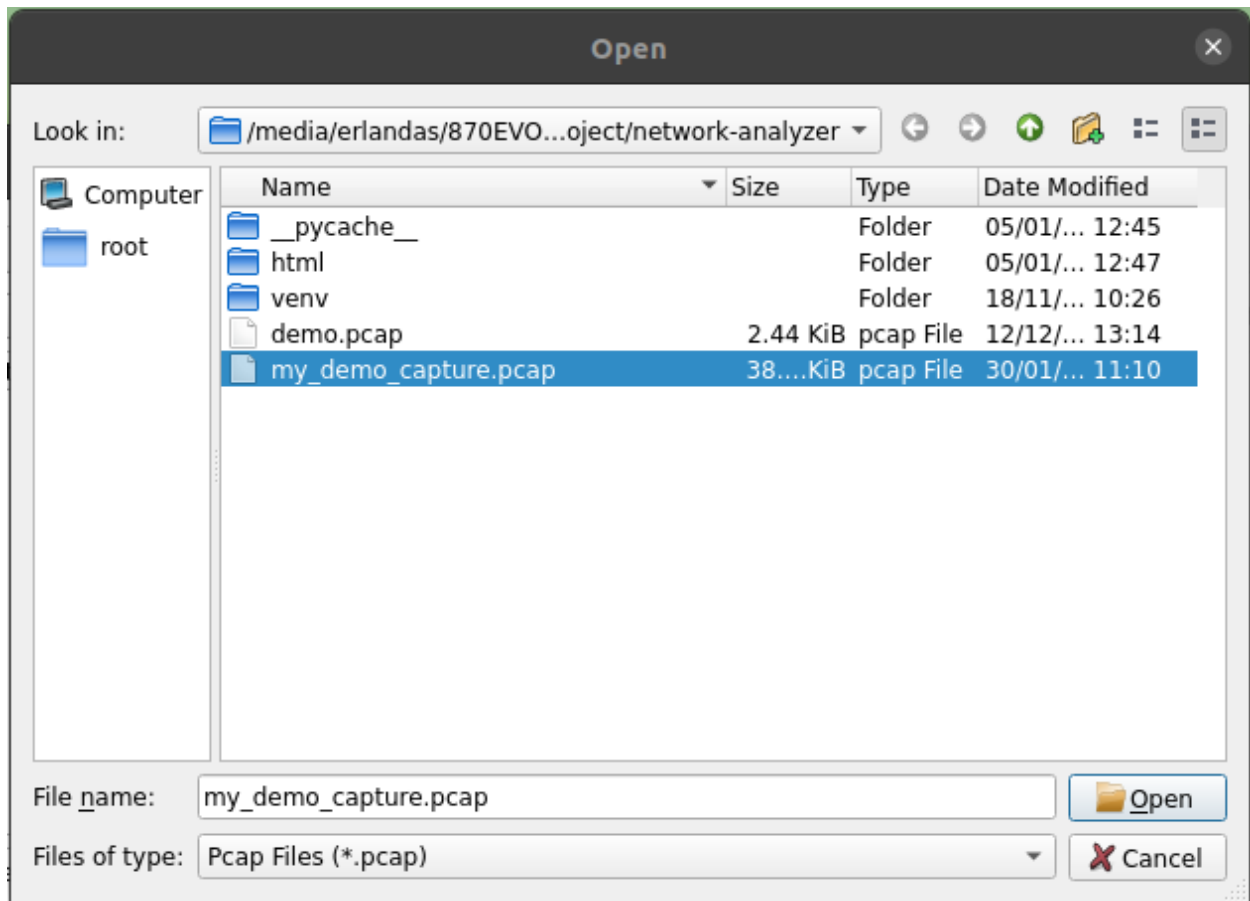
**Save:** this option will allow user to save capture file in pcap format so that a capture could be analyzed using other tools if needed.



```
erlandas@lx:/media/erlandas/870EVO/college_y4/05-project/network-analyzer$ ls
app_data.db  Doxyfile          gut_main_window.py  html/             README.md
db_controls.py  gut_arp_poison.py  gut_port_scan.py    main.py*          requirements.txt
db_main.py     gut_dns_spoof.py   gut_quit_note.py    my_demo_capture.pcap  venv/
demo.pcap      gut_host_discovery.py  gut_sniffer.py      __pycache__/
erlandas@lx:/media/erlandas/870EVO/college_y4/05-project/network-analyzer$ sudo tcpdump -tttnnr my_demo_capture.pcap
reading from file my_demo_capture.pcap, link-type EN10MB (Ethernet)
2022-01-30 11:02:41.646500 IP 192.168.1.14.38738 > 192.168.1.12.8009: Flags [P.], seq 742432711:742432821, ack 3270147482, wln 501, options [nop,nop,TS val 2617111711 ecr 105975655], length 110
2022-01-30 11:02:41.649597 IP 192.168.1.20.50418 > 192.168.1.12.8009: Flags [P.], seq 4008793869:4008793979, ack 2355330557, wln 1025, length 110
2022-01-30 11:02:41.765663 IP 192.168.1.12.8009 > 192.168.1.14.38738: Flags [P.], seq 11111, ack 110, wln 729, options [nop,nop,TS val 105976930 ecr 2617111711], length 110
2022-01-30 11:02:41.765739 IP 192.168.1.14.38738 > 192.168.1.12.8009: Flags [P.], ack 111, wln 501, options [nop,nop,TS val 2617111830 ecr 105976930], length 0
2022-01-30 11:02:41.766062 IP 192.168.1.12.8009 > 192.168.1.20.50418: Flags [P.], seq 1:111, ack 110, wln 712, length 110
2022-01-30 11:02:41.809511 IP 192.168.1.20.50418 > 192.168.1.12.8009: Flags [P.], ack 111, wln 1025, length 0
2022-01-30 11:02:43.060855 IP 52.113.194.132.443 > 192.168.1.20.50500: Flags [R.], seq 1100212444, ack 4280055239, wln 0, length 0
2022-01-30 11:02:43.759492 IP 192.168.1.14.52592 > 192.168.1.1.53: 36780* A? presence.services.sfb.trafficmanager.net. (58)
2022-01-30 11:02:43.766900 IP 192.168.1.1.53 > 192.168.1.14.52592: 36780 2/0/0 CNAME a-ups-presence4-prod-azsc.francecentral.cloudapp.azure.com., A 52.114.104.169 (140)
2022-01-30 11:02:43.767500 IP 192.168.1.14.56438 > 192.168.1.1.53: 21000* A? a-ups-presence4-prod-azsc.francecentral.cloudapp.azure.com. (76)
2022-01-30 11:02:43.788086 IP 192.168.1.1.53 > 192.168.1.14.56438: 21000 1/0/0 A 52.114.104.169 (92)
2022-01-30 11:02:43.788502 IP 192.168.1.14.51714 > 52.114.104.169.443: Flags [S], seq 1915737515, wln 04240, options [nss 1400,sackOK,TS val 178285146 ecr 0,nop,wscale 7], length 0
2022-01-30 11:02:43.810500 IP 52.114.104.169.443 > 192.168.1.14.51714: Flags [S.], seq 3182765098, ack 1915737516, wln 65535, options [nss 1412,nop,wscale 8,nop,nop,sackOK], length 0
2022-01-30 11:02:43.810597 IP 192.168.1.14.51714 > 52.114.104.169.443: Flags [P.], ack 1, wln 502, length 0
2022-01-30 11:02:43.810954 IP 192.168.1.14.51714 > 52.114.104.169.443: Flags [P.], seq 1:518, ack 1, wln 502, length 517
2022-01-30 11:02:43.834932 IP 52.114.104.169.443 > 192.168.1.14.51714: Flags [P.], seq 1:1413, ack 518, wln 2049, length 1412
2022-01-30 11:02:43.834979 IP 192.168.1.14.51714 > 52.114.104.169.443: Flags [P.], ack 1413, wln 493, length 0
```

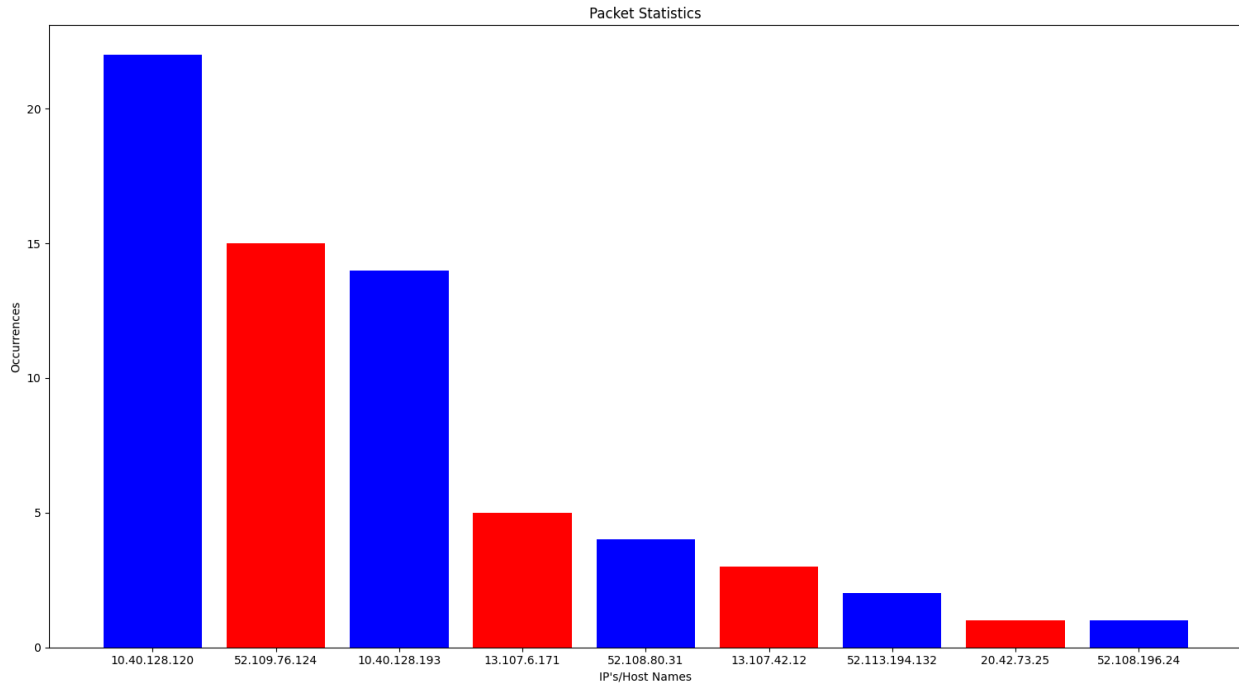


**Open:** will open chosen pcap file and import into application for further analysis.

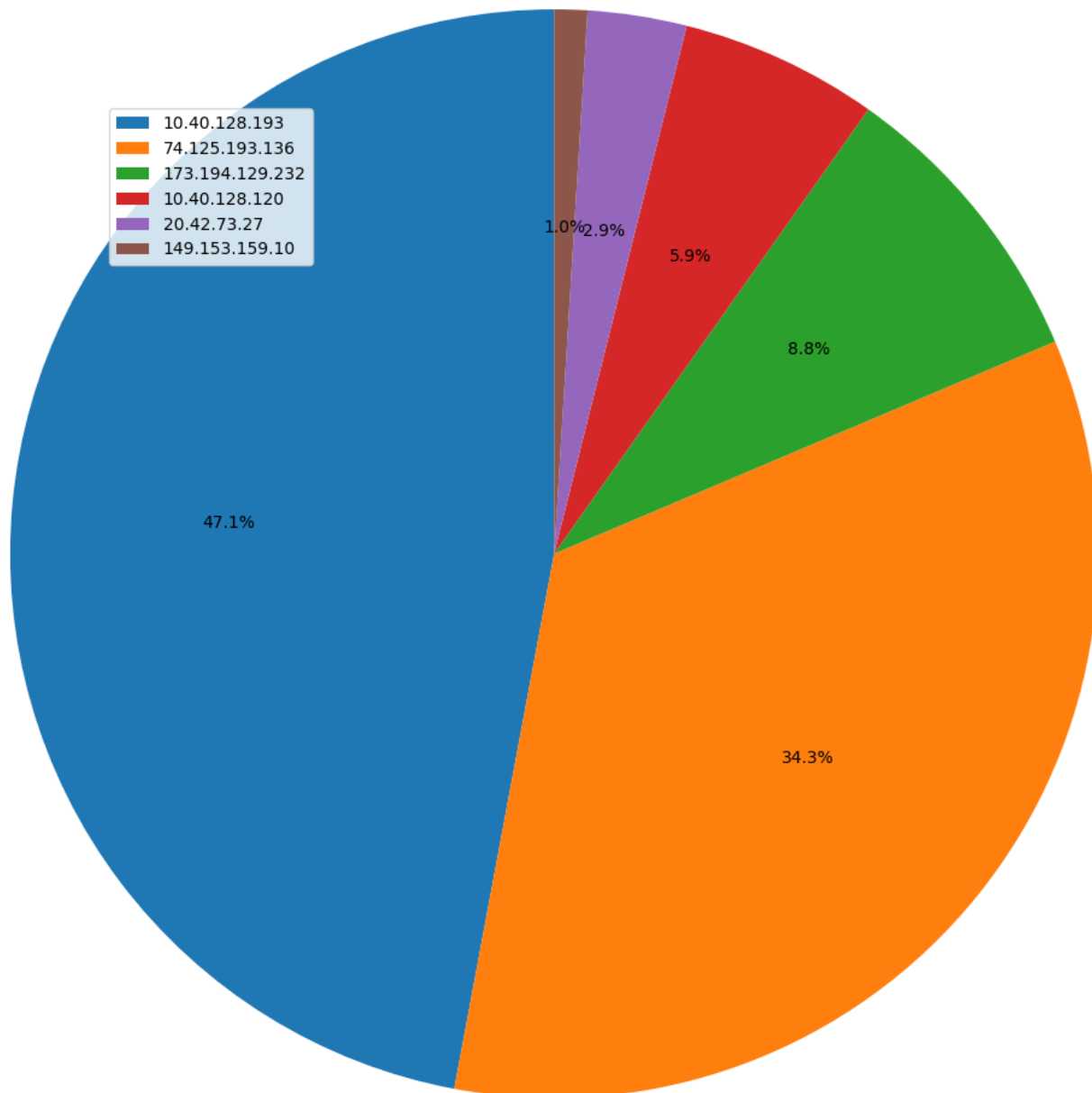


**Clear:** all output from network capture will be cleared. In addition, the user will be notified of actions taken by the application.

**Bar Chart:** Summary of packet capture will be displayed in a bar chart. All data packets will be sorted by frequency. Bar chart could be displayed while application actively working on capturing network traffic, when application is stopped but has data collected from previous inspection or from loaded capture file.



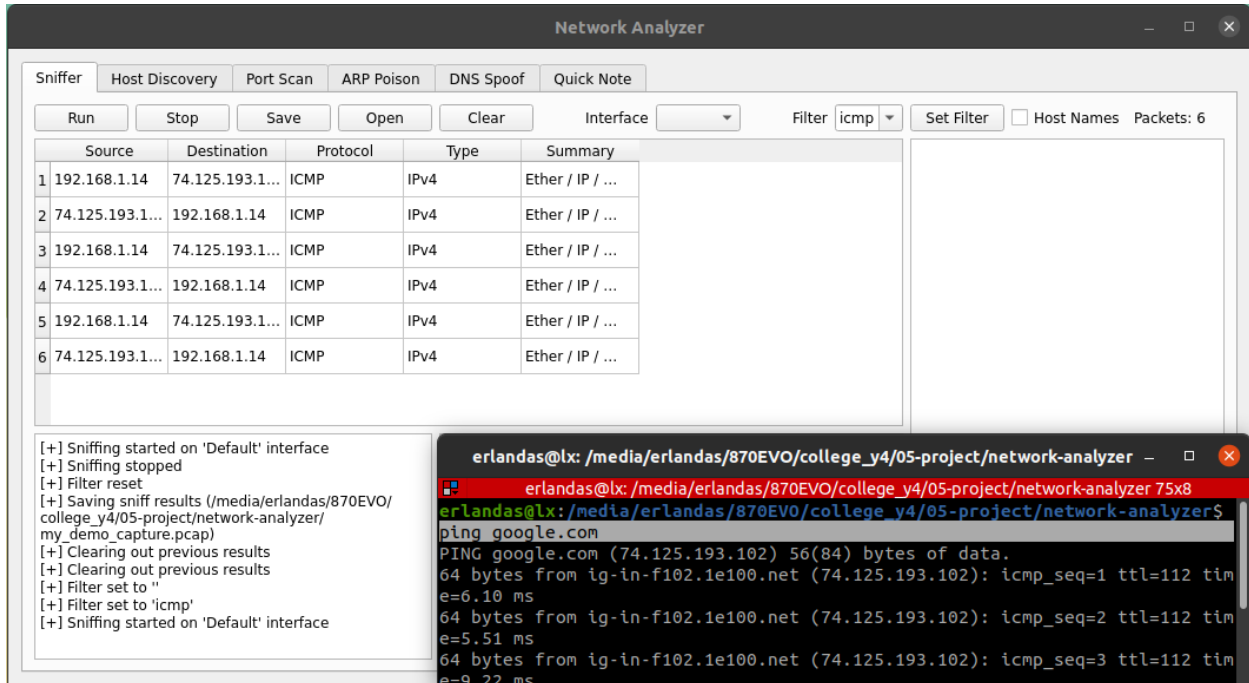
**Pie Chart:** Summary of packet capture will be displayed in a bar chart. All data packets will be sorted by frequency. Bar chart could be displayed while application actively working on capturing network traffic, when application is stopped but has data collected from previous inspection or from loaded capture file.



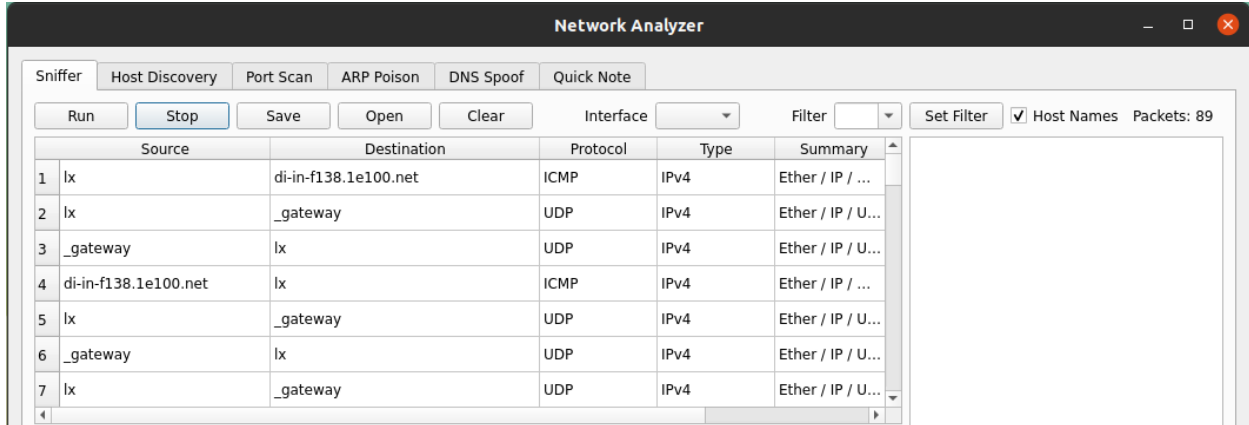
**Interface:** the user has an option to choose an interface on which network capture will be initiated. It is useful if the device has more than one network interface card. However, if the interface is not chosen by default, the application will use a currently active default interface. Choice made by the user will be outlined in the bottom feedback field.

**Filter/Set Filter:** allows the user to set a filter included from protocol to host. All filters can be applied that most of the popular applications have, such as WireShark. To set filter, criteria must be provided in a filter field and “set filter” button must be pressed. User will be notified when filter is set in the feedback field.

All options that were used for filtering will be stored in a local database and will be present for next time when the application will be used. Example below show filtering only ICMP packets.



**Host Names:** this checkbox will try to translate to domain names if domain name is present.



## Host Discovery

This option has an ability to identify currently active hosts on a given network. It is also able to identify single target if a target internet protocol address is given. This functionality works by issuing ARP requests for given subnet or host using broadcast address.

**Start:** starts host discovery when subnet provided in a form of “192.168.1.0/24” or a single IP address in a form of “192.168.1.1. Below will be examples of a host discovery in action and what information is given when one of the entries are selected from identified hosts table.

The screenshot shows the Network Analyzer interface with the Host Discovery tab selected. The Start field is set to 192.168.1.0/24 and the method is ARP. A table lists 8 active hosts. The selected host 192.168.1.1 is highlighted, and its corresponding request and response details are shown in the right pane.

IP	MAC	Status
1 192.168.1.1	e4:fb:5d:...	ACTIVE
2 192.168.1.20	08:00:27:c2:...	ACTIVE
3 192.168.1.9	a4:08:01:70:...	ACTIVE
4 192.168.1.13	80:6d:...	ACTIVE
5 192.168.1.12	48:b0:2d:...	ACTIVE
6 192.168.1.3	ae:c9:17:62:...	ACTIVE
7 192.168.1.11	00:f3:61:0b:f...	ACTIVE
8 192.168.1.4	20:0b:cf:...	ACTIVE

```
[+] Request
###[ Ethernet ]###
dst = ff:ff:ff:ff:ff:ff
src = a0:a4:c5:1c:0b:88
type = ARP
###[ ARP ]###
hwtype = 0x1
ptype = IPv4
hlen = None
plen = None
op = who-has
hwsrc = a0:a4:c5:1c:0b:88
psrc = 192.168.1.14
hwdst = 00:00:00:00:00:00
pdst = 192.168.1.1

[+] Response
###[ Ethernet ]###
dst = a0:a4:c5:1c:0b:88
src = e4:fb:5d:19:2b:df
type = ARP
###[ ARP ]###
hwtype = 0x1
ptype = IPv4
hlen = 6
plen = 4
op = is-at
hwsrc = e4:fb:5d:19:2b:df
psrc = 192.168.1.1
hwdst = a0:a4:c5:1c:0b:88
pdst = 192.168.1.14
```

The screenshot shows the Network Analyzer interface with the Host Discovery tab selected. The Start field is set to 192.168.1.1 and the method is ARP. A table lists one active host. The selected host 192.168.1.1 is highlighted, and its corresponding request and response details are shown in the right pane.

IP	MAC	Status
1 192.168.1.1	e4:fb:5d:19:2b:df	ACTIVE

```
[+] Request
###[ Ethernet ]###
dst = ff:ff:ff:ff:ff:ff
src = a0:a4:c5:1c:0b:88
type = ARP
###[ ARP ]###
hwtype = 0x1
ptype = IPv4
hlen = None
plen = None
op = who-has
hwsrc = a0:a4:c5:1c:0b:88
psrc = 192.168.1.14
hwdst = 00:00:00:00:00:00
pdst = 192.168.1.1

[+] Response
###[ Ethernet ]###
dst = a0:a4:c5:1c:0b:88
src = e4:fb:5d:19:2b:df
type = ARP
###[ ARP ]###
hwtype = 0x1
ptype = IPv4
hlen = 6
plen = 4
op = is-at
hwsrc = e4:fb:5d:19:2b:df
psrc = 192.168.1.1
hwdst = a0:a4:c5:1c:0b:88
pdst = 192.168.1.14
```

## Port Scan

Port scanner works by firstly checking if a host that application attempts to scan is up and if not, it lets the user know and doesn't take any other actions. If host is up the application starts to scan by sending TCP synchronize packet to destined port that is defined in a range. Port scanner attempts to send TCP Syn packet from a random port number that is created on demand for each port in the range. For the scanner to work, IP address of a target or a domain name must be provided. Also, port range between 1 and 65535 must be provided as min and max ports. If this range is not in the range specified, feedback will be given to the user with an error. To scan for a single port, both min and max must be the same. At the left-hand side will be displayed only opened ports and on the right-hand side all ports that were scanned. Below will be examples of an IP address scan and a domain name scan.

Network Analyzer

Sniffer Host Discovery Port Scan ARP Poison DNS Spoof Quick Note

Start Stop itcarlow.ie 1 65535

Port	Status
1 80	Open
2 113	Open
3 443	Open

Port	Status
1 1	Closed
2 2	Closed
3 3	Closed
4 4	Closed
5 5	Closed
6 6	Closed
7 7	Closed
8 8	Closed
9 9	Closed
10 10	Closed

[+] Checking host at itcarlow.ie .....  
[+] Host itcarlow.ie is UP

Network Analyzer

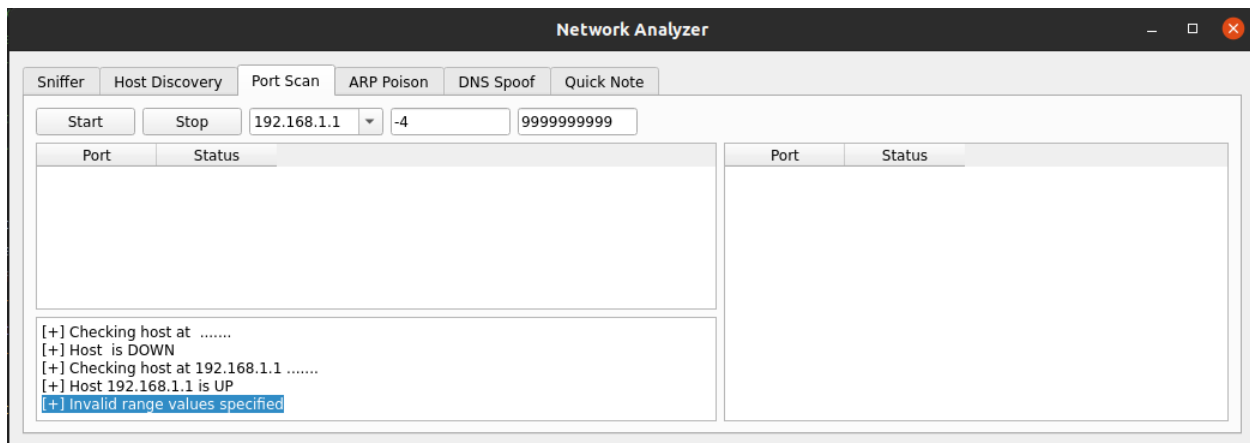
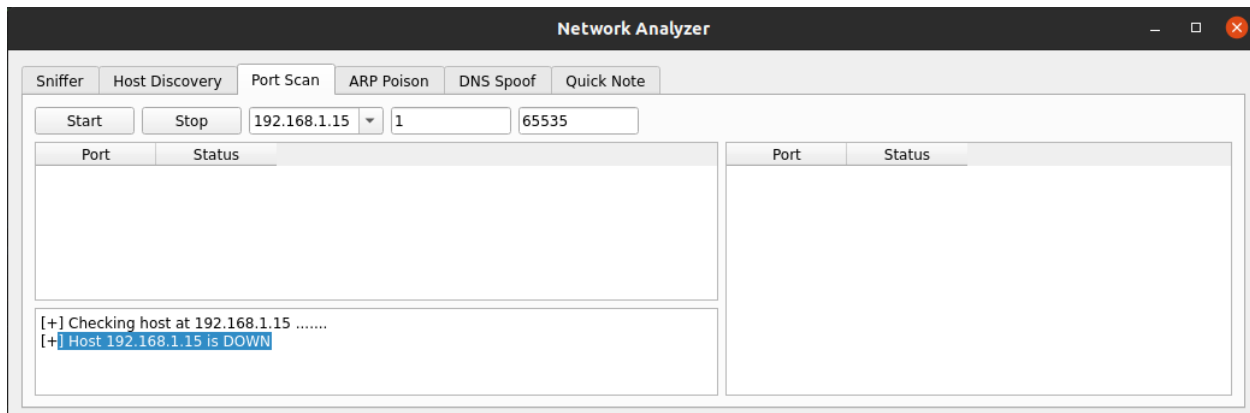
Sniffer Host Discovery Port Scan ARP Poison DNS Spoof Quick Note

Start Stop 192.168.1.1 1 100

Port	Status
1 53	Open
2 80	Open

Port	Status
26 26	Closed
27 27	Closed
28 28	Closed
29 29	Closed
30 30	Closed
31 31	Closed
32 32	Closed
33 33	Closed

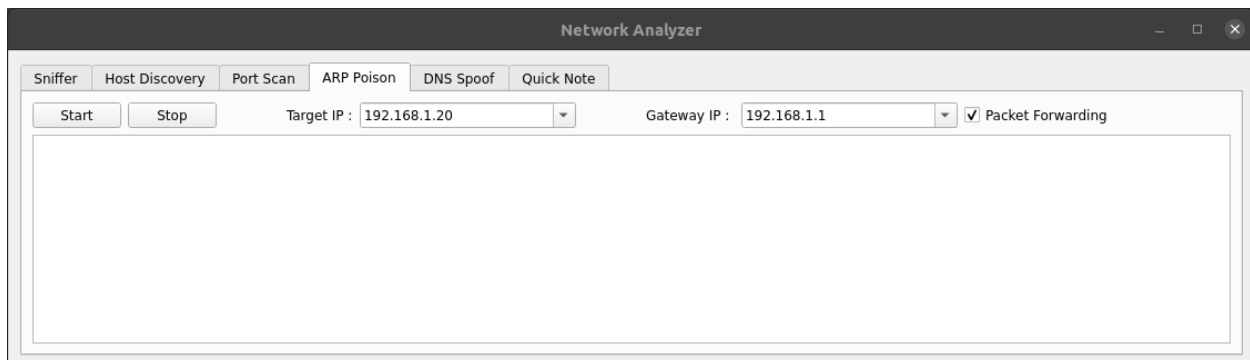
[+] Checking host at 192.168.1.1 .....  
[+] Host 192.168.1.1 is UP



## ARP Poison

ARP poisoning attack is a type of man in the middle attack achieved through address resolution protocol (ARP). A bad actor sends ARP message to a target saying that he is a default gateway and at the same time bad actor sends ARP message to default gateway saying that he is a targeted device. Thus, completing these steps all traffic goes through malicious machine. Result of this all traffic can be captured before reaching the internet.

To initiate ARP poisoning attack, first the user must provide target IP address and gateway IP address. If there is intention to prevent target from reaching internet by executing Denial of Service (DoS) attack, user can leave out packet forwarding disabled. If packet forwarding is disabled user will be notified that it will result to DoS attack on a targeted device.



**Start:** Initiates ARP poisoning attack assuming that target and default gateway is online, does checks before starting to ensure that both devices are online. If they are online, the ARP attack begins.

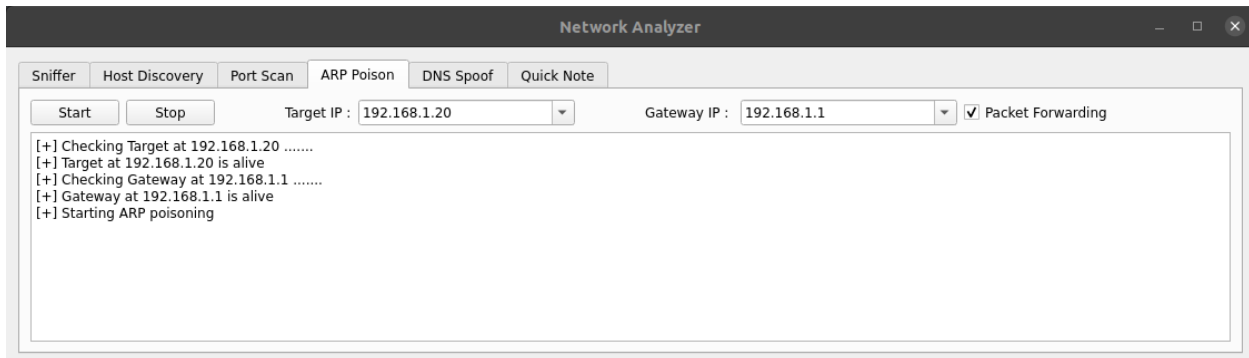
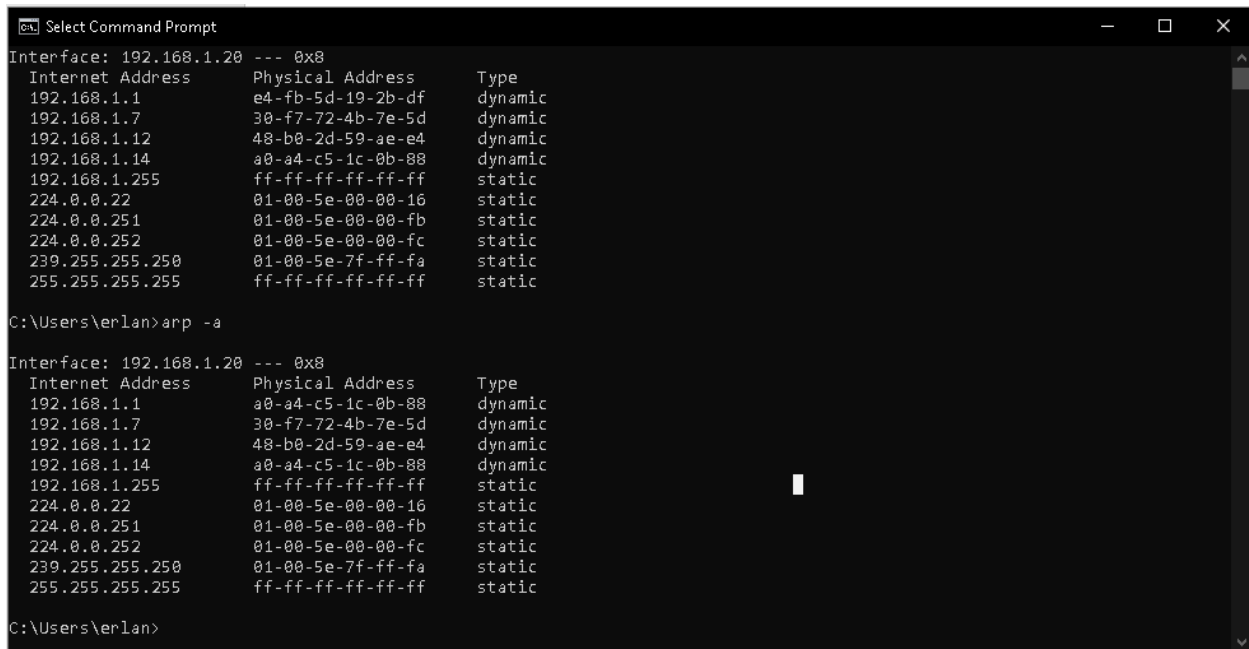
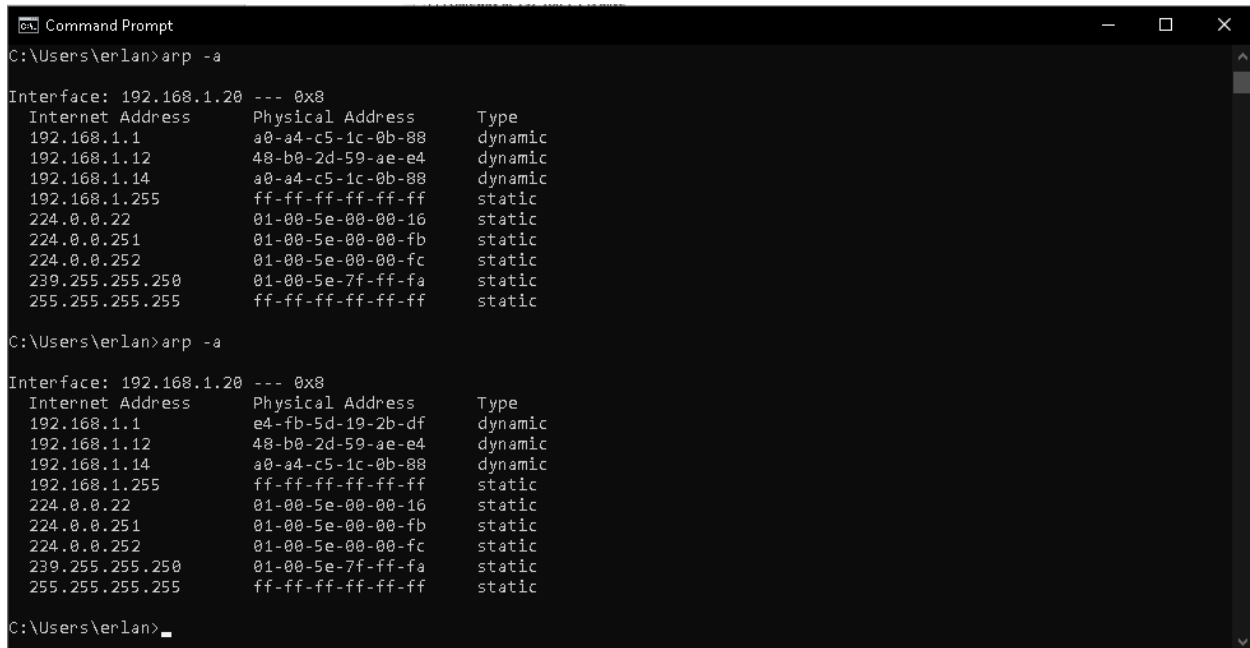


Image below shows ARP table on a targeted device before initiating attack and after. By observing changes from first output address (gateway) 192.168.1.1 has a MAC address that ends in “df” and the device that ARP attack initiated from has address of 192.168.1.14 and corresponding MAC address that ends in “88”. After attack is initiated from the output we can see that addresses 192.168.1.1 and 192.168.1.14 corresponds to the same MAC address that ends in “88”.





**Stop:** Stops currently active attack and sends reset packets to the targeted device and the gateway. Thus, hiding steps of an attack. Image below shows ARP table straight after stop was pressed.



```
C:\Users\erlan>arp -a

Interface: 192.168.1.20 --- 0x8
Internet Address      Physical Address      Type
192.168.1.1           a0-a4-c5-1c-0b-88    dynamic
192.168.1.12          48-b0-2d-59-ae-e4    dynamic
192.168.1.14          a0-a4-c5-1c-0b-88    dynamic
192.168.1.255        ff-ff-ff-ff-ff-ff    static
224.0.0.22            01-00-5e-00-00-16    static
224.0.0.251          01-00-5e-00-00-fb    static
224.0.0.252          01-00-5e-00-00-fc    static
239.255.255.250      01-00-5e-7f-ff-fa    static
255.255.255.255      ff-ff-ff-ff-ff-ff    static

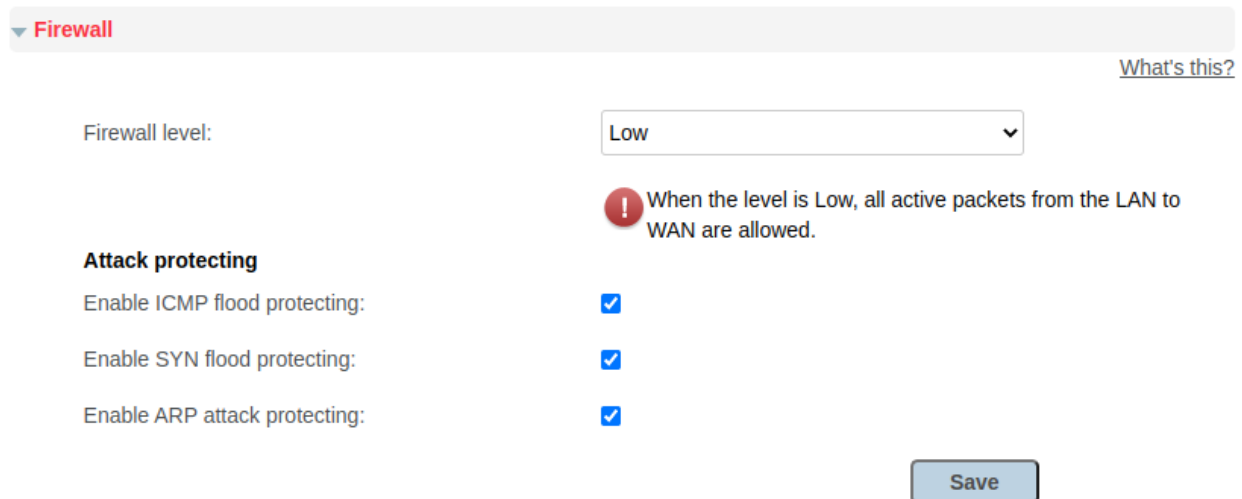
C:\Users\erlan>arp -a

Interface: 192.168.1.20 --- 0x8
Internet Address      Physical Address      Type
192.168.1.1           e4-fb-5d-19-2b-df    dynamic
192.168.1.12          48-b0-2d-59-ae-e4    dynamic
192.168.1.14          a0-a4-c5-1c-0b-88    dynamic
192.168.1.255        ff-ff-ff-ff-ff-ff    static
224.0.0.22            01-00-5e-00-00-16    static
224.0.0.251          01-00-5e-00-00-fb    static
224.0.0.252          01-00-5e-00-00-fc    static
239.255.255.250      01-00-5e-7f-ff-fa    static
255.255.255.255      ff-ff-ff-ff-ff-ff    static

C:\Users\erlan>
```

**Target IP:** Takes in targeted device IP address that attack will be performed on.

**Gateway IP:** Takes in gateway IP address that device that attack is coming from will pretend to be default gateway. Image below shows that gateway settings have ARP attack protection enabled. This tool avoids detection of ARP attack, as per gateway settings it is supposed to protect against such attacks

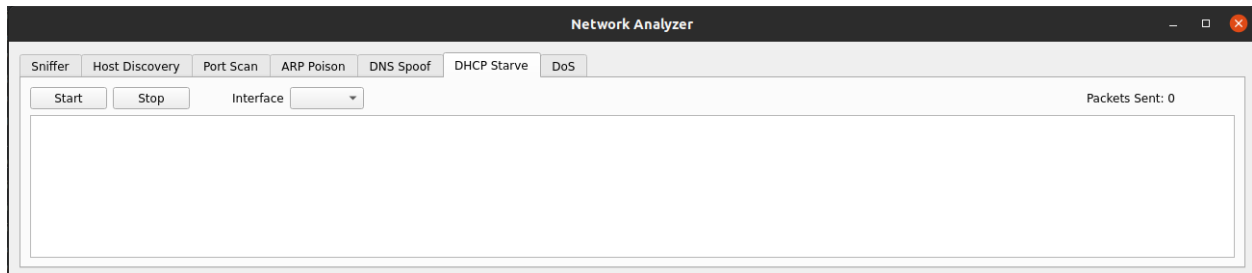


**Packet Forwarding:** Enables packet forwarding on running device, works on MacOS, Linux and Windows devices. If packet forwarding is not enabled it is resulting to DDoS/DoS attack on a targeted device.

## DHCP Starvation

DHCP starvation tab gives the ability for the application to perform DHCP starvation attack. This attack targets DHCP servers whereby forged DHCP requests are created by an application with the intent of exhausting all available IP addresses that can be allocated by the DHCP server. By performing this attack, legitimate network users can be denied service.

To initiate DHCP starvation attack, start button must be pressed and application will start crafting forged DHCP packets and broadcast them to the broadcast domain.



**Start:** starts DHCP starvation attack by forging DHCP packets and broadcasting them to the broadcast domain.

**Stop:** stops DHCP starvation attack

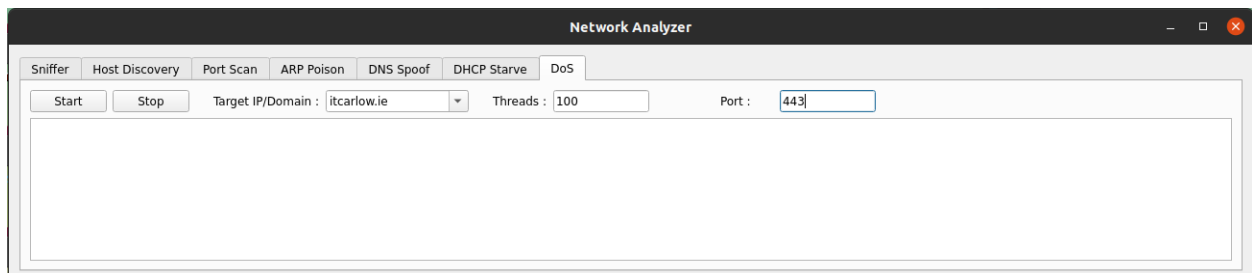
**Interface:** provides selection of interfaces that are available on a computer that the application is running. If nothing is selected default active interface will be chosen.

## Denial of Service (DoS, DDoS)

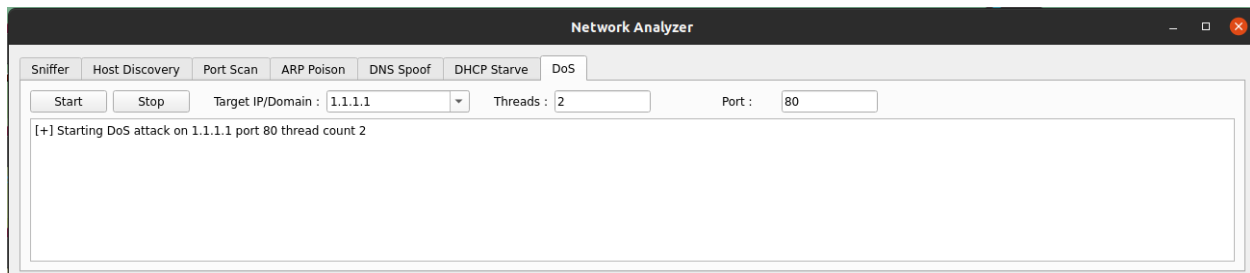
A Denial of Service (DoS) is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with huge amounts of traffic. The DoS attack deprives legitimate users from accessing network or web pages.

This application tries to hide its steps by randomizing source IP and source port for each of the threads. If one thread was executed application won't randomize IP address or port. Thus, making this application perform with multiple threads as DDoS attack by randomizing source ports and IP addresses. If anyone would inspect traffic, it would look like packet flow comes from many different sources.

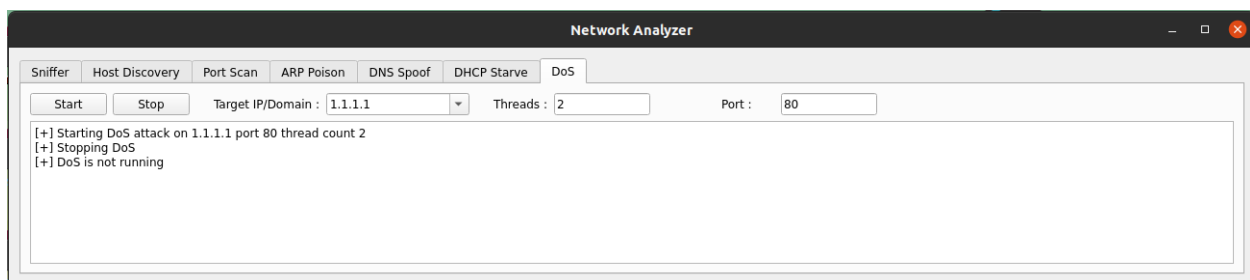
To initiate DoS attack using network analyzer application target IP address or domain name must be provided, together with number of threads and targeted port number. If threads are provided more than one application will start sending packets from multiple threads thus initiating Distributed Denial of Service (DDoS) attack.



**Start:** starts Denial of Service (DoS) attack, assuming IP/Domain, thread count and port is provided



**Stop:** stops currently running DoS attack, if attack is not currently running provides feedback allowing user to know this. Below screengrab when “Stop” was pressed twice, first time it stopped DoS attack, second time informed in a feedback field that attack is not running now and there is nothing to stop.



**Target IP/Domain:** takes in parameter that holds IP address or Domain name of a target.

**Threads:** takes parameter that defines how many threads must be executed while performing DoS attack.

**Port:** takes parameter that tells application what port to target.

## Use Case Examples

### ARP Attack Capturing Traffic from Target

Example below uses targets device IP address of 192.168.1.20 and default gateways IP address of 192.168.1.1

1. Turn on “Network Analyzer” and navigate to “Sniffer” tab.
2. Provide filter as host address of a targeted device “host 192.168.1.20” and press “Set Filter”
3. Navigate to “ARP Poison” tab
4. Provide Target IP as “192.168.1.20” and default gateway IP of “192.168.1.1”
5. Enable packet forwarding
6. Press “Start”
7. Navigate back to “Sniffer” tab and press “Run”

Below screengrab shows that only packets that has host address of 192.168.1.20 are captured using steps provided above.

Network Analyzer

Sniffer Host Discovery Port Scan ARP Poison DNS Spoof Quick Note

Run Stop Save Open Clear Interface Filter Set Filter Host Names Packets: 810

	Source	Destination	Protocol	Type	Summary
1	192.168.1.20	192.168.1.12	TCP	IPv4	Ether / IP / TCP 192.168.1.20:49738 > 192.168.1.12:8009 PA
2	192.168.1.20	192.168.1.1	ARP	ARP	Ether / ARP is at a0:a4:c5:1c:0b:88 says 192.168.1.20
3	192.168.1.12	192.168.1.20	TCP	IPv4	Ether / IP / TCP 192.168.1.12:8009 > 192.168.1.20:49738 PA
4	192.168.1.12	192.168.1.20	TCP	IPv4	Ether / IP / TCP 192.168.1.12:8009 > 192.168.1.20:49738 PA
5	192.168.1.20	192.168.1.12	TCP	IPv4	Ether / IP / TCP 192.168.1.20:49738 > 192.168.1.12:8009 A
6	192.168.1.14	192.168.1.20	ARP	ARP	Ether / ARP who has 192.168.1.20 says 192.168.1.14
7	192.168.1.20	192.168.1.14	ARP	ARP	Ether / ARP is at 08:00:27:c2:7c:ad says 192.168.1.20

```

###[ Ethernet ]###
dst = 48:b0:2d:59:ae:e4
src = a0:a4:c5:1c:0b:88
type = IPv4
###[ IP ]###
version = 4
ihl = 5
tos = 0x0
len = 150
id = 52712
flags = DF
frag = 0
ttl = 128
proto = tcp
chksum = 0xa908
src = 192.168.1.20
dst = 192.168.1.12
options \
###[ TCP ]###
sport = 49738
dport = 8009
seq = 1873682463
ack = 2073868658
dataofs = 5
reserved = 0
flags = PA
window = 1023
chksum = 0xe493
urgptr = 0

```

[+] Filter set to 'host 192.168.1.20 '  
 [+] Sniffing started on 'Default' interface  
 [+] Results selected from row 1  
 [+] Inspect packets when sniffing is stopped  
 [+] Sniffing stopped  
 [+] Filter reset  
 [+] Results selected from row 2  
 [+] Results selected from row 1



## Inspect packet flow during DDoS attack

1. Turn on application
2. Navigate to "DoS" tab
3. Provide targeted IP address, example used "1.1.1.1"
4. Provide threads "100" so application will generate 100 threads with random source IP addresses and random source ports attached to the random IP's
5. Provide port number to target, as an example used port "80"
6. Press "Start"
7. Navigate to sniffer tab
8. Provide filter "host 1.1.1.1"
9. Press "Set filter"
10. Inspect packet flow
11. Pie chart shows how DDoS attack looks from visual perspective below

Sniffer Host Discovery Port Scan ARP Poison DNS Spoof DHCP Starve DoS

Run Stop Save Open Clear Bar Chart Pie Chart Interface

Filter: host 1.1.1.1 | Set Filter | Host Names | Packets: 2173

	Source	Destination	Protocol	Type	Summary
1	174.204.218.84	1.1.1.1	TCP	IPv4	Ether / IP / TCP 174.204.218.84:56948 > 1.1.1.1:8192 S
2	32.253.111.98	1.1.1.1	TCP	IPv4	Ether / IP / TCP 32.253.111.98:60704 > 1.1.1.1:8192 S
3	231.218.121.32	1.1.1.1	TCP	IPv4	Ether / IP / TCP 231.218.121.32:49221 > 1.1.1.1:8192 S
4	126.12.98.44	1.1.1.1	TCP	IPv4	Ether / IP / TCP 126.12.98.44:6047 > 1.1.1.1:8192 S
5	153.24.126.55	1.1.1.1	TCP	IPv4	Ether / IP / TCP 153.24.126.55:4750 > 1.1.1.1:8192 S
6	41.171.230.206	1.1.1.1	TCP	IPv4	Ether / IP / TCP 41.171.230.206:47656 > 1.1.1.1:8192 S
7	47.166.45.137	1.1.1.1	TCP	IPv4	Ether / IP / TCP 47.166.45.137:244 > 1.1.1.1:8192 S
8	220.87.228.41	1.1.1.1	TCP	IPv4	Ether / IP / TCP 220.87.228.41:9091 > 1.1.1.1:8192 S
9	74.132.167.253	1.1.1.1	TCP	IPv4	Ether / IP / TCP 74.132.167.253:6476 > 1.1.1.1:8192 S
10	230.105.240.56	1.1.1.1	TCP	IPv4	Ether / IP / TCP 230.105.240.56:1040 > 1.1.1.1:8192 S
11	142.108.36.48	1.1.1.1	TCP	IPv4	Ether / IP / TCP 142.108.36.48:51049 > 1.1.1.1:8192 S
12	70.248.115.17	1.1.1.1	TCP	IPv4	Ether / IP / TCP 70.248.115.17:15218 > 1.1.1.1:8192 S
13	105.194.209.111	1.1.1.1	TCP	IPv4	Ether / IP / TCP 105.194.209.111:19395 > 1.1.1.1:8192 S
14	29.112.62.161	1.1.1.1	TCP	IPv4	Ether / IP / TCP 29.112.62.161:51185 > 1.1.1.1:8192 S
15	91.189.238.7	1.1.1.1	TCP	IPv4	Ether / IP / TCP 91.189.238.7:16157 > 1.1.1.1:8192 S
16	215.45.138.42	1.1.1.1	TCP	IPv4	Ether / IP / TCP 215.45.138.42:6420 > 1.1.1.1:8192 S
17	111.49.89.105	1.1.1.1	TCP	IPv4	Ether / IP / TCP 111.49.89.105:18904 > 1.1.1.1:8192 S
18	40.112.203.7	1.1.1.1	TCP	IPv4	Ether / IP / TCP 40.112.203.7:34163 > 1.1.1.1:8192 S
19	244.125.208.175	1.1.1.1	TCP	IPv4	Ether / IP / TCP 244.125.208.175:39747 > 1.1.1.1:8192 S
20	217.41.98.148	1.1.1.1	TCP	IPv4	Ether / IP / TCP 217.41.98.148:10554 > 1.1.1.1:8192 S
21	71.38.99.36	1.1.1.1	TCP	IPv4	Ether / IP / TCP 71.38.99.36:21620 > 1.1.1.1:8192 S
22	183.14.31.165	1.1.1.1	TCP	IPv4	Ether / IP / TCP 183.14.31.165:3766 > 1.1.1.1:8192 S
23	106.41.127.178	1.1.1.1	TCP	IPv4	Ether / IP / TCP 106.41.127.178:41080 > 1.1.1.1:8192 S

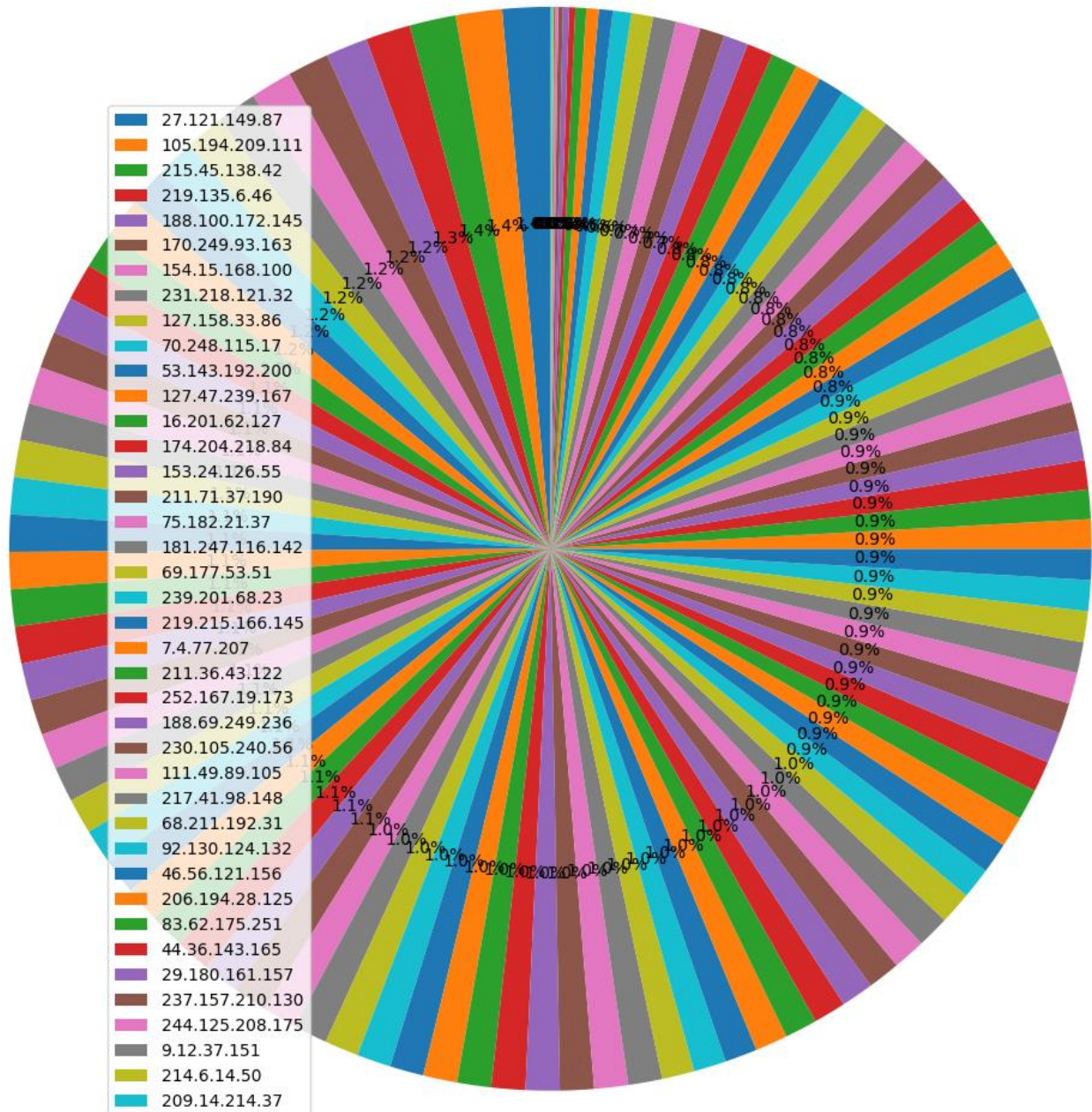
```

### Ethernet ###
dst = 00:90:0b:32:ee:a5
src = a0:a4:c5:1c:0b:88
type = IPv4
### IP ###
version = 4
ihl = 5
tos = 0x0
len = 40
id = 1
flags = 0
frag = 0
ttl = 64
proto = tcp
chksum = 0xefac
src = 174.204.218.84
dst = 1.1.1.1
options =
### TCP ###
sport = 56948
dport = http
seq = 0
ack = 0
dataofs = 5
reserved = 0
flags = 5
window = 8192
chksum = 0x25fb
urgptr = 0
options =

```

[+] Sniffing stopped  
[+] Filter reset  
[+] Results selected from row 1  
[+] Results selected from row 1  
[+] Results selected from row 2  
[+] Results selected from row 1  
[+] Clearing out previous results  
[+] Sniffing started on 'Default' interface  
[+] Sniffing stopped  
[+] Filter reset  
[+] Results selected from row 5  
[+] Results selected from row 1

IP	Count
27.121.149.87	31
105.194.209.111	30
215.45.138.42	30
219.135.6.46	29
188.100.172.145	27
170.249.93.163	27
154.15.168.100	27
231.218.121.32	26
127.158.33.86	26



### Original Design and Specifications Vs. Current Design and Specifications

Not a lot was changed from the start to finish of my project only some functionality added as I felt I had time to do so. Original design had login component, after consulting my supervisor we have decided to remove this component as application has to be run with administrator/root privileges so user will be logged in as root/administrator user, removing need of registration component. The notes tab was removed as I don't see need of it. Every computer has some text editor even terminal has vim. As it was suggested by a few supervisors I have added DHCP starvation attack and also DoS/DDoS attack. I was also advised to include visual representation of a network capture using graphs, therefore these suggestions were also implemented.



## Learning Outcomes

Below I will outline the learning outcomes that I have achieved during the course of completing this project, both from a technological and personal perspective.

### Technology

This project provided me with the opportunity to gain better understanding of the networking concepts and OSI layers. I was exposed to how technologies can be used to perform network attacks in the real world. I was motivated to learn Python as part of my project, and from research I gained the understanding that this programming language is widely used in the workplace and is very important in my chosen work and study field. Qt framework was something that I also learned during the course of this project. Thread programming was something that at first I have struggled with, but now I feel confident in it. I have also strengthened my understanding and knowledge of portable database technologies SQLite.

### Personal

From a personal perspective I felt that I learned how to better pace myself during this project. Having a project that had many months before the deadline meant that I had to ensure that I work on the project at least a few days a week, while also very importantly, not taking too much time away from other subjects and projects. Time management was crucial skill that I have gained as a result of this project and managing more than one project at a time will be a skill I will be able to carry with me into the workplace.

I also gained better understanding of how the project life cycle works, starting from the initial phase which is research and continuing with the functional specifications and ending with the final result while also presenting my progress on a weekly basis to my supervisor. This also enhanced my presentation skills as well as online communication skills which will most definitely be useful in the workplace.

## Achievement Review

Network Sniffer (Analyzer) had successfully reached all functionality that was outlined in a project brief also as additional functionality that is outlined in a table below.

No.	Description	Outcome
1.	Operational network sniffer	Achieved
2.	Simple point and click interface	Achieved
3.	Ability to ARP poison chosen target	Achieved
4.	Identify hosts on a network	Achieved
5.	Filter options on a sniffer	Achieved
6.	Application works on multiple platforms (OSX, Linux, Windows)	Achieved
7.	Sniffer translates to domain names	Achieved
8.	Local database to retain queries	Achieved
9.	Scan for opened ports on a given target	Achieved
10.	Export capture files in pcap format	Achieved
11.	Import capture files in pcap format	Achieved
12.	Displays packet information including binary format	Achieved
13.	Ability to perform Denial of Service attack (DoS)	Achieved
14.	Identify if host is active	Achieved
15.	Perform DHCP starvation attack	Achieved
16.	Visual representation of packet captures (Bar and Pie charts)	Achieved

## Plagiarism Detection

I declare that all material in this submission is entirely my own work except where duly acknowledged. I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material; including software and other electronic media in which intellectual property rights may reside. I have provided a complete bibliography of all works and sources used in the preparation of this submission. I understand that failure to comply with the Institute's regulations governing plagiarism constitute a serious offence.

Student Name: Erlandas Bacauskas

Student Number: C00242521

Date: 06/04/2022