

Packet Sniffing using Scapy

by
Venkat Nithin Atturu
Manisha Bobba
Sulakshana Mucheli
Nagavarshini Surapaneni
Malluvalasa Kiran Mai
Alekhya Bugata

Abstract

Network engineers and information security specialists encounter a number of issues daily where the current technologies are largely inadequate in solving the issue. The available technologies can certainly produce the necessary outcomes, but occasionally they also deliver massive amounts of data that require time and effort to sort through. Therefore, having a basic understanding of scripting is always beneficial for problem-solving, research, and automation, all of which result in time, money, and effort savings.

The Python programming language and Scapy framework are used to construct a rapid packet sniffer, which is elaborated on in this article. In this paper, the use of two sniffing techniques, Raw sockets, and Scapy, to achieve better performance in terms of maximum capture packet rate are analyzed and compared. A piece of software called a packet sniffer is used to eavesdrop on, record, and examine data and network activity. Writing a packet sniffer helps us to grasp packet layers, network packet components, construction, sniffing, and dissection, and it also enables us to get our hands dirty.

Introduction

Computer networks are expanding in size incredibly quickly today, as is evident. In recent years, both its number of users and network traffic have grown. In order to maintain a smooth and effective network, it is crucial to monitor both network traffic

and user activity. Due to the large number of data accessible, maintaining and monitoring a complex network is an overwhelming task. Packet sniffing is used for this goal. In order to monitor network activity and assist network managers in detecting issues, packet sniffing is vital. Packet sniffing is the practice of gathering, collecting, and logging some or all packets that pass through a computer network, regardless of how the packet is addressed. In this way, every packet, or a defined subset of packets, may be gathered for further analysis. You as a network administrator can use the collected data for a wide variety of purposes like monitoring bandwidth and traffic.

A packet sniffer, sometimes called a packet analyzer, is composed of two main parts. First, a network adapter connects the sniffer to the existing network. Second, software that provides a way to log, see or analyze the data collected by the device. Scapy is a powerful and versatile packet manipulation tool written in python. Using Scapy, a user will be able to send, sniff, dissect and forge network packets. Scapy also has the capability to store the sniffed packets in a PCAP file. Using scapy, we will be able to handle tasks like trace routing, probing, scanning, unit tests, and network discovery with ease. All of these properties make scapy useful for network-based attacks.

As mentioned before scapy performs a wide range of networking tasks and one such task is packet sniffing. Packet sniffing is the process of capturing all the packets flowing across a computer network. The sniffed packets give away a lot of

information like what website a user visits, what contents the user sees, what the user downloads, and almost everything. The captured packets are usually stored for future analysis.

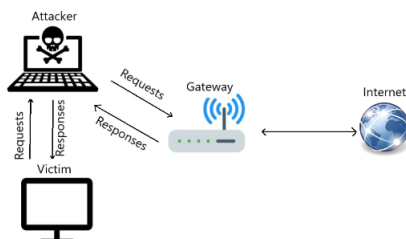
Proposed System

For the Analysis of the packets that are being sent and received in the network we are using a library called Scapy that is available in python. Scapy is a network manipulation library that is used to interact with the Network, as it is a powerful packet manipulation tool. It can transmit packets over the wire, collect them, match requests and responses, forge or decode packets of many different protocols, and do a lot more. The majority of traditional operations, including scanning, trace routing, probing, unit testing, attacks, and network discovery, can be handled with ease. We also used the sockets that are network programming

Sockets and the socket API is used to send messages across a network. They provide a form of inter-process communication (IPC). The network can be a logical, local network to the computer or one that's physically connected to an external network, with its own connections to other networks. The main code of the project is implemented in the Linux Operating system later to transfer the data from Linux to Windows where we have developed a front-end application using Flutter Framework which uses the Dart programming language we have used the REST API's of the sockets.

Methodology

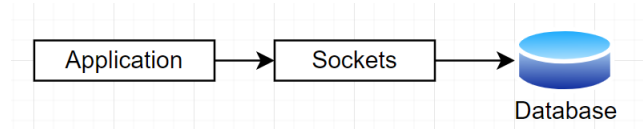
1 ARP Spoofing



ARP spoofing, also referred to as ARP poisoning, is a Man in the Middle technique that enables attackers to eavesdrop on network device communication. This is how the attack operates:

1. The attacker must obtain network access. The IP addresses of at least two devices—say let's a workstation and a router—are found after they perform a network search.
2. To send out fake ARP responses, the attacker uses a spoofing tool like ARP spoof or Driftnet.
3. The fake responses claim that the attacker's MAC address is the proper MAC address for both the workstation's and router's IP addresses. By doing this, the router and workstation are tricked into connecting to the attacker's system rather than to each other.
4. The two devices then start communicating with the attacker instead of one another directly after updating their ARP cache entries. The attacker is now illegally intercepting all conversations.

2 Socket's



Sockets are the bidirectional endpoints of a communication channel. Sockets may communicate within the process, between processes on the same machine, or between processes on the same machine or between processes on different continents. We use the socket module in python to create and use sockets.

In this project, we have implemented the concepts of python sockets so that the data that is being generated in one OS can be transferred to the database in the other OS.

3 Scapy

With the help of the Python tool Scapy, one may send, sniff, analyze, and forge network packets. This capability enables the creation of tools that can probe, scan, or attack networks. By importing

it into Python programs, it can be used as a library or interactively through the command line interface. It is compatible with Windows, Mac OS X, and Linux operating systems. The majority of classical tasks, such as scanning, tracerouting, probing, unit testing, assaults, or network discovery, can be handled with ease using Scapy. Hping, Arpspoof, arp-sk, arping, p0f, and even some parts of Nmap, tcpdump, and tshark can be replaced by it. Scapy's primary benefit is that, in contrast to other tools, it enables us to modify network packages at a low level, enabling us to use existing network protocols and parameterize them to meet our needs.

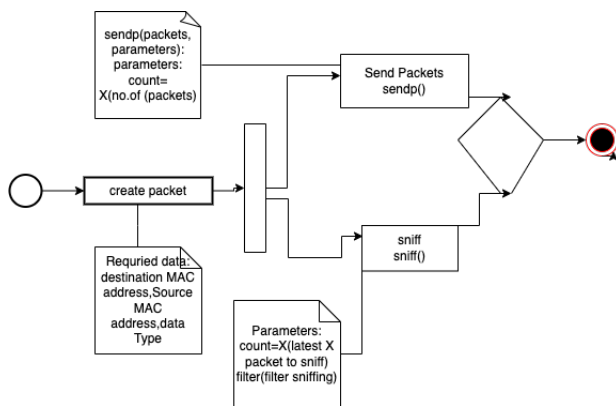


Fig 1: Working of scapy as an interactive packets manipulation tool

Strengths and weakness of the scapy tool are:

Strengths:

1. Interactive language.
2. Multi-tasking.
3. Modularity.
4. Extensibility.
5. Simple packet forging and analyzing.
6. Bypass local firewall
7. High-level functions already implemented.

Weakness:

1. Link layers are not managed well.
2. The DNS packets received/sent are not reassembled exactly as the original.
3. Can't handle a large number of packets simultaneously.
4. Partial support for certain complex protocols.

Scapy supports the following protocols:

Ethernet, TCP/IP, UDP, IPv4 and IPv6, ARP.

WORK FLOW

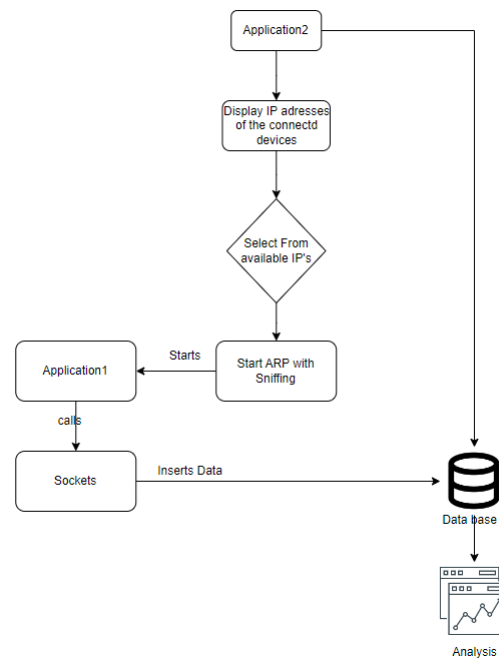


Fig 2: Work flow diagram

System Architecture

The project has been done in a different method such that both applications 1 and 2 have a common database in which the data is stored. We used the socket programming and REST API's of the Flask Architecture to store the data in the database and to perform some analysis after the data was stored in the database.

The project is primarily implemented in Linux OS, and all of the data produced by the program will be later kept in a common database, where we have performed all the analysis using graphs that the Flutter framework supports.

Sockets are used in this way to efficiently transfer data between Operating Systems. The main reason for the project to be implemented in this way is that Windows doesn't support EtherCap.

System Architecture

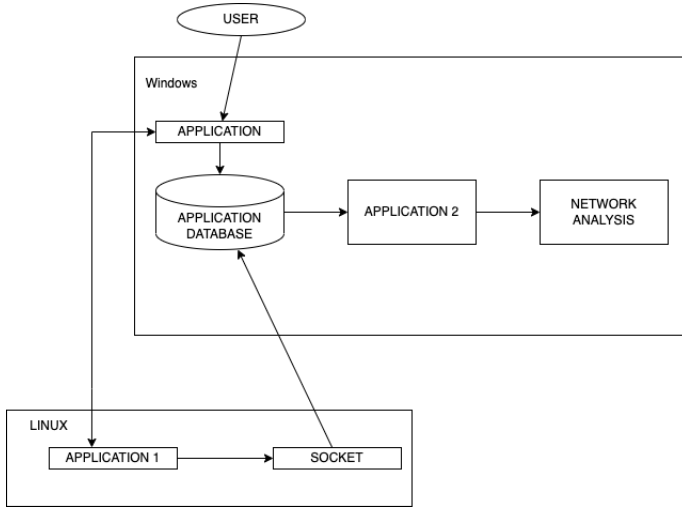


Fig 3: System Architecture diagram

Implementation

The project is implemented in a unique way, with the front end of the application being developed using the Flutter framework and the back end of the project being implemented using the Python libraries scrapy and ethertcap on the Linux operating system. Additionally, we have implemented APIs of the socket programming to insert data that is generated by the program that runs on the Linux operating system. Once the network traffic is captured and we are ready to do the analysis, we move on to the next step.

A. Dashboard 1:

We have a total of 2 Dashboards in the implemented project. Dashboard 1 shows the data of the number of devices connected to the network in different interfaces, and also after the packet sniffing is carried out we implemented the maps by using the IP address of the destination server. Then by using the location coordinates, lines showing how the source IP is connected to the destination IP have been drawn. The Destination locations are marked in red color and the Source locations are marked in blue which shows the geographical location

of various servers that the source is interacting with.

The Pie-Chart implemented shows the percentage of packets that are responsible for the domain they belong to out of the total captured packets in an interactive way. We have another feature - Top Remote Destinations, which shows the domains in the order of the number of packets that are responsible for each domain in decreasing order with the IP address it is associated with and the country flag that the IP address belongs to.

B. Dashboard 2:

In this dashboard, we have plotted the bar graphs for the captured packets in two different ways.

1. In the first graph, we plotted the Number of packets that are exchanged between the local machine by the country. This will show the number of packets that are flowed from the country by the country name and the count of packets.

2. In the Second bar graph, we plotted the graph using the data that is saved in the process of packet sniffing, by using flutter. This graph shows the number of servers that the local machine has accessed by the country, So it shows the number of servers in the country they are located in.

C. Top Traffic Hosts:

In this tab, we are showing all the information that we have regarding the top accessed hosts during the sniffing with the IP address and the Organisation name along with the URL with which the local machine is interacting and the city in which the server is located and the country to which the server belongs to and also it has the option to show the RAW-DATA.

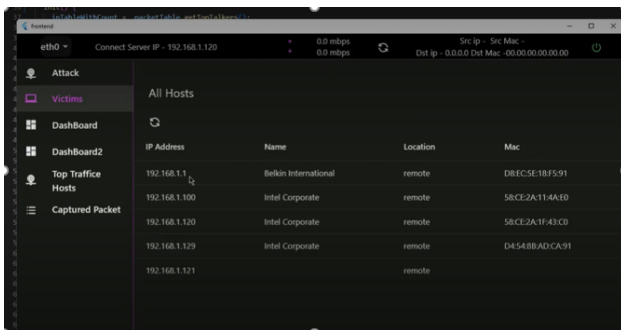
The following are some of the parameters that are shown in the raw data:

1. SOURCE IPV4
2. DESTINATION IPV4
3. URL
4. Protocol
5. HEAD
6. BODY

Results and Analysis

Victims:

The victim's tab is designed to show all the devices that are connected to the network that the local machine is connected to it has all the values like The IP address of the devices, the name of the devices that the IP address is associated with and the location and the MAC address of the device that is associated with.

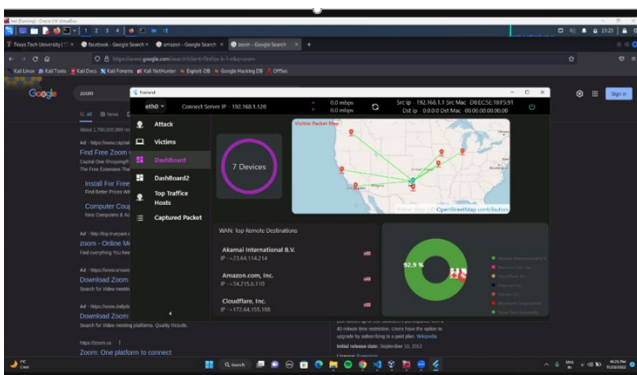


IP Address	Name	Location	Mac
192.168.1.1	Belkin International	remote	D8EC5E18F391
192.168.1.100	Intel Corporate	remote	58CE2A114A4B
192.168.1.130	Intel Corporate	remote	58CE2A1F43C0
192.168.1.129	Intel Corporate	remote	D454BBADCA91
192.168.1.131		remote	

Dashboard 1:

Once the attack is carried out, we need to analysis the packets to see when the attack is carried out. We can see that 7 devices are connected to the network and have also marked the geographical location of each servers that the local machine has accessed and the pie chart showing the domain name with the percentage of packets they are responsible for.

We also have implemented a feature to show the domain names and the flags of the country for which the number of packets they belong in the descending order of the count of packets.



Dashboard 2:

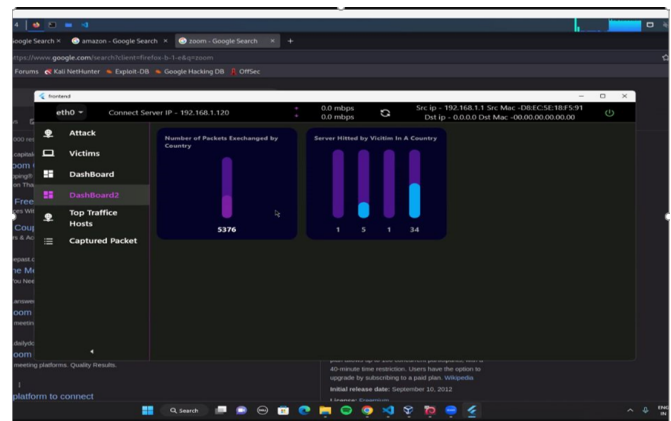
In this dashboard, we have plotted 2 different graphs

1. Number of Packets Exchanged by the country.:

This graph shows us the total number of packets that have been exchanged between all the servers and the local machine by the name of the Country from which the packets are flowed in between.

2. The Other graph is plotted for the number of servers hit by the local machine by the country.:

This Graph shows the number of servers that are located in each country that the local machine has accessed during the network sniffing.



Top Traffic Hosts:

In this dashboard all the details that are related to the packets are displayed - the IP address of the destination, the name of the organization, the URL by which we are accessing the server, the city where the server is located, and the country to which the server belongs to. Moreover, we also developed a Show RAW DATA button, where all the captured data is shown in JSON format when it is clicked.

IP	hostname	city	region	country
88.212.201.198	United Network LLC	host198.rax.ru	Moscow	Moscow
95.213.133.91	OOO Network of	St-Petersburg	Saint Petersburg	St-Petersburg
142.250.138.95	Google LLC	rw-in-f95.1e100.net	Richardson	Texas
157.240.19.35	Facebook, Inc.	edge-star-mini-shv-01-dfw5.facebook.com	Dallas	Texas
104.22.34.226	Cloudflare Inc.	null	San Francisco	California
142.251.33.4	Google LLC	dfw25644-in-f4.1e100.net	Dallas	Texas
129.118.241.25	Texas Tech University	ccib02-vip25.ttu.edu	Lubbock	Texas
108.156.245.83	Amazon.com	server-108-156-245-83.dfw56.r.cloudfront.net	Eufess	Texas
88.212.201.204	United Network LLC	host204.rax.ru	Moscow	Moscow
88.212.202.52	United Network LLC	host152.rax.ru	Moscow	Moscow
95.213.133.93	OOO Network of	St-Petersburg	Saint Petersburg	St-Petersburg
20.189.173.2	Microsoft Corporation	null	San Jose	California

Results:

All the Data captured in the Sniffing is stored in two tables of a database.

1. Packet Store table:

This table has all the details that are captured from the details of a packet, later we use these data from the tables to perform the analysis.

srcip	dstip	srcmac	dstmac	rawdata	createddate
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:35.493048
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:35.503993
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:37.493278
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:36.487323
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:39.495778
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:41.493688
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:41.506528
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:41.510508
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:43.513186
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:43.522514
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:43.529971
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:43.540887
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:43.550925
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:43.559894
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:43.567977
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:43.576789
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:45.542826
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:45.551123
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:45.558095
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:45.570866
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:45.579881
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:45.588319
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:45.594193
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:45.601442
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:45.607027
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:45.613476
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:45.622024
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:45.627672
192.168.1.121	52.182.143.210	08:00:27:bac33:11	db:ec5e:18:f5:91	["Ethernet",{"dst":"db:ec5e:18:f5:91",...	2022-11-20 22:21:45.633329

2. IP Look-UP table:

This table is used to plot the graphs for the Geographical location. By using this graph we can use the IP address to get the geographical location of the Server.

ip	hostname	city	region	country
13.107.5.83	Redmond	Washington	US	47.6745, 122.1215
52.182.143.208	Des Moines	Iowa	US	41.5876, 93.6274
20.48.18.123	Des Moines	Iowa	US	41.5876, 93.6274
104.46.162.226	Melbourne	Victoria	AU	-37.8140, 144.9633
17.248.193.31	Richardson	Texas	US	32.9482, -96.7297
108.156.211.70	server-108-156-211-70.dfw56.r.cloudfront.net	Eufess	Texas	32.8371, -97.0820
88.212.201.198	host198.rax.ru	Moscow	RU	55.7522, 37.6156
95.213.133.91	St-Petersburg	St-Petersburg	RU	59.9386, 30.3141
142.250.138.95	rw-in-f95.1e100.net	Richardson	Texas	32.9482, -96.7297
157.240.19.35	edge-star-mini-shv-01-dfw5.facebook.com	Dallas	Texas	32.7831, -96.8067
104.22.34.226	San Francisco	California	US	37.7621, -122.3971
142.251.33.4	dfw25644-in-f4.1e100.net	Dallas	Texas	32.7831, -96.8067
129.118.241.25	ccib02-vip25.ttu.edu	Lubbock	Texas	33.5776, 101.6032
108.156.245.83	server-108-156-245-83.dfw56.r.cloudfront.net	Eufess	Texas	32.8371, -97.0820
88.212.201.204	host204.rax.ru	Moscow	RU	55.7522, 37.6156
88.212.202.52	host152.rax.ru	Moscow	RU	55.7522, 37.6156
95.213.133.93	St-Petersburg	St-Petersburg	RU	59.9386, 30.3141
20.189.173.2	San Jose	California	US	37.3476, 121.8870
142.250.133.94	rw-in-f95.1e100.net	Dallas	Texas	32.7831, -96.8067
108.156.211.7	server-108-156-211-7.dfw56.r.cloudfront.net	Eufess	Texas	32.8371, -97.0820
142.251.33.227	dfw25644-in-f4.1e100.net	Arlington	Texas	32.7357, -97.1081
142.251.45.67	dfw25648-in-f4.1e100.net	Dallas	Texas	32.7831, -96.8067
20.189.42.124	124-42-188-30-3c.googleusercontent.com	Kansas City	Missouri	39.0997, -94.5786
142.250.133.309	rw-in-f95.1e100.net	Dallas	Texas	32.7831, -96.8067
142.251.35.195	dfw25648-in-f4.1e100.net	Dallas	Texas	32.7831, -96.8067
52.7.204.76	ec2-52-77-204-76.ap...	Singapore	Singapore	1.2897, 103.8501
23.64.114.20	623-64-114-20.deploy.static.akamaitechnologies...	Chicago	Illinois	41.8500, -87.6500
52.182.143.210	Des Moines	Iowa	US	41.5876, 93.6274
34.117.237.239	238-337-117-34.bc.googleusercontent.com	Kansas City	Missouri	39.0997, 94.5786
23.64.114.214	623-64-114-214.deploy.static.akamaitechnologies...	Chicago	Illinois	41.8500, -87.6500

Conclusion

By using the ARP Spoofing Man in the Middle attack, packet sniffing is successfully accomplished. We can conclude from all of the above-mentioned screenshots and the analysis that packet sniffing was successfully implemented and that we have been able to access all of the victim's network traffic. Furthermore, we are aware of the physical locations of the servers the victim is accessing along with the raw data that is being transmitted over the network.

Future Work

The project is currently being developed and implemented on devices that are connected to the network to which the attacker is logged in, but in the future, it should be possible to sniff packets from networks in which the attacker is not logged in without using any hacking tools.

References:

[1] S. Ansari, Rajeev S.G. and Chandrasekhar H.S., "Packet Sniffing: A Brief Introduction", IEEE Potentials, 2003, Volume: 21

[2] Rohit Raj S, Shobha G, and Rohit R "SCAPY- A powerful interactive packet manipulation program ", 2022 from IEEE Xplore