

NETWORK SCANNER - ARP SPOOFER

MINOR PROJECT REPORT

Submitted by

RAVINDER(1619066)

MOHIT (1619061)

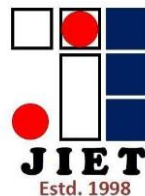
in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

BRANCH COMPUTER SCIENCE AND ENGINEERING



JIND INSTITUTE OF ENGINEERING AND TECHNOLOGY - JIND

KURUKSHETRA UNIVERSITY

(2021-2022)

CERTIFICATE

This is to certify that Minor project entitled “**NETWORK SCANNER – ARP SPOOFER**” is a bonafide work carried out by “**RAVINDER(1619066) & Mohit(1619061)**” under my guidance and supervision and submitted in partial fulfillment of the award of B. Tech degree in Computer science and Engineering. The work embodied in the Minor Project has not been submitted for the award of other degree or diploma to the best of my knowledge

Ms. Sapna (Asst.prof.)
(PROJECT SUPERVISION)

Ms. Sapna Aggrawal
(Head of the Department)

STUDENT'S DECLARATION

I hereby certify that the work which is being presented in the minor project report entitled "NETWORK SCANNER –ARP SPOOFER" in fulfillment of the requirement for the award of the Degree of Bachelor of Technology in Department of Computer Science & Engineering of Jind Institute of Engineering and Technology, Jind, Kurukshetra University, Kurukshetra, Haryana is an authentic record of my own work carried out during 6th semester.

Ravinder
(1619066)

Mohit
(1619061)

ACKNOWLEDGEMENT

We are highly grateful to the Dr. S.K Singh, Principal, **Jind Institute of Engineering and Technology**, Jind, for providing this opportunity.

The constant guidance and encouragement received from Ms Sapna Aggrawal, HOD (CSE/IT, deptt.), JIET, Jind has been of great help in carrying out the project work and is acknowledged with reverential thanks.

We would like to express a deep sense of gratitude and thanks profusely to A.P Sapna project guide, without the wise counsel and able guidance, it would have been impossible to complete the report in this manner.

We express gratitude to other faculty members of CSE department of JIET for their intellectual support throughout the course of this work.

Finally, the authors are indebted to all whosoever have contributed in this report work

RAVINDER	MOHIT
(1619066)	(1619061)

ABSTRACT

The project on “**Arp Spoofer**” is allow you to linking of an attacker’s MAC address with the IP address of a legitimate computer or server on the network.

Networks have become an integral part of today’s world. The ease of deployment, low-cost and high data rates have contributed significantly to their popularity. There are many protocols that are tailored to ease the process of establishing these networks. Nevertheless, security-wise precautions were not taken in some of them. In this paper, we expose some of the vulnerability that exists in a commonly and widely used network protocol, the Address Resolution Protocol (ARP) protocol. Effectively, we will implement a user friendly and an easy-to-use tool that exploits the weaknesses of this protocol to deceive a victim’s machine and a router through creating a sort of Man-in-the-Middle (MITM) attack. In MITM, all of the data going out or to the victim machine will pass first through the attacker’s machine. This enables the attacker to inspect victim’s data packets, extract valuable data (like passwords) that belong to the victim and manipulate these data packets. We suggest and implement a defense mechanism and tool that counters this attack, warns the user, and exposes some information about the attacker to isolate him.

The project allow you to scan all devices in a same local area network and gather MAC Address of the device along with its IP. Also this project allow us to gather all details of the IP address.

Keywords: **Python , Flask, Bootstrap 3, CSS , HTML**

Contents

CONTENTS	PAGE
Certificates	1
Student's Declaration	2
Aknowledgement	3
Abstract	4
Introduction <ul style="list-style-type: none">• Research Methodology• Definition of Terms	6
Analysis <ul style="list-style-type: none">• Packet Inspection Procedure• Arp Working• Limitation of ARP• Potential Vulnerabilities of ARP	8
Methodology	15
Design and Implementation of Network Scanner Algorithm	22
Python Module <ul style="list-style-type: none">• Scapy Module	25
System/Project Requirements	27
Implementation	30
Project Output <ul style="list-style-type: none">• Arp spoofing output• IP-Lookup Output	32
Testing	34
Conclusion and Recommendation <ul style="list-style-type: none">• Structure of the Project Files	35
References	37

INTRODUCTION

Communication is becoming more and more important in today's life, whether it is work-related, or to get connected with family and friends. Computer networks play a major part in this process. In recent days, it is hard to find a computer or a smart phone that is not connected to a network in some sort, whether it is directly connected to the Internet service provider (ISP) or through a local area network (LAN) alongside other devices.

To facilitate this process, network engineers came up with a set of rules called protocols for message exchange between computers. These protocols maintain the connection between the connected devices within a network and allow them to work efficiently and trouble free. Engineers, who tailored some of these protocols, did not take into account the bad intentions of some users who may exploit the vulnerabilities of these protocols. Some network attacks may give access to attackers (hackers) or allow them to alter the behavior of the network by deceiving these vulnerable protocols with false information. By doing so, they might capture valuable information of an institute or an organization or sensitive private data which may harm individuals and infringe their privacy.

In this paper, some of these vulnerabilities are illustrated, mainly in the Address Resolution Protocol (ARP) protocol. An application that exploits these vulnerabilities to perform a "Man-in-the-Middle" attack is implemented. In addition, an application that works as a defense against these types of attacks is designed and implemented.

Man-in-the-Middle (MITM) attack is an active eavesdropping attack, where in a communication session between two devices A and B, the attacker deceives A by pretending to be B. This means whenever A wants to send a message to B, it actually sends it to the attacker who will read/modify the message then forward it to B in order to keep the continuity of the communication. The attacker will be able to read and modify all the contents of the communication before forwarding it to B.

ARP is a protocol used by the data link layer to map IP address to MAC address. Before encapsulating the network layer packet in a data link layer frame, the host sending the packet needs to know the recipient's MAC address. Given the IP address of a host, to find its MAC address, the source node broadcasts an ARP request packet which asks about the MAC address of the owner of the IP address. This request is received by all nodes inside the LAN. The node that owns this IP address replies with its MAC address. It is worth mentioning that the ARP reply is unicast while the ARP request is broadcast. When the host receives the ARP reply, usually the IP/MAC addresses mapping are saved as entries in a table called the "ARP table". This table is used as a cache where the node will send an ARP request only if the ARP table does not contain the IP/MAC mapping. ARP's weakness lies in the fact that it is a stateless protocol, i.e., it accepts ARP replies without having to send an ARP request. ARP spoofing attack exploits this vulnerability by sending ARP reply messages that contain the IP address of a network resource, such as the default gateway or a DNS server, to a victim machine. The attacker replaces the MAC address of the corresponding network resource with his machine's MAC address. The victim's machine that receives the spoofed ARP replies cannot distinguish them from legitimate ones. Moreover, the ARP tables usually use the result of the last ARP reply only.

The attacker then takes the role of man in the middle; any traffic directed to the legitimate resource is sent

through the attacking system. The attacker reads the packet looking for sensitive data; it may modify that data, and then passes it to the designated destination. As this attack occurs on the lower levels of the TCP/IP protocol stack, the end-user is unaware to the attack occurrence. Furthermore, ARP spoofing is also capable of performing denial of service (DoS) attacks if the attacker drops the packets instead of passing them. This causes the victim's machine to be denied from service from network resources. There exist some tools which simplify generating ARP spoofing attack, like Ettercap which is developed by Ornaghi and Valleri and another tool developed by Wagner.

In this paper, ARP spoofing attacks; MITM is implemented. And most importantly, a defense against these attacks is suggested and designed by developing a simple tool that can be deployed on clients' machines to warn them in case of an attack. Moreover, it exposes the IP and MAC addresses of the attacker.

RESEARCH METHODOLOGY

The research method used for this project work is quantitative research reviews the current system, provide its description, identifying the discrepancies and eventually giving a suitable solution. Therefore, the method used in the design and collections of information from various sources are as follows:

- Studying the present system in detail and the organizational style.
- Knowing and understanding the input and output processes of the existing system.
- A qualitative form of interview was conducted in the organization to understand the mode of operation of the old system.

DEFINITION OF TERMS

Association: an association is an organized body of people who have an interest, activity, or purpose in common; a society.

Web based application: a web-based application is a software package that can be accessed through the web browser. the software and database reside on a central server rather than being installed on the desktop system and is accessed over a network.

Web browser: a web browser is a software application used to enable computer users locate and access web pages.

Catalogue: a product catalogue is a file that contains a list of all the products you want to advertise. Each line of the product catalogue contains a description of each product, including an id, name, category, availability, product URL, image URL and other attribute

ANALYSIS

ARP is a protocol that works inside the LAN when a node has an IP address of a local destination and wants to know the MAC address of that destination. If the destination is outside the LAN, then there will be an intermediate node called the default gateway (router) which forwards the data to outside the LAN. In this case, the ARP is used to map the IP address of the gateway to its related MAC address. The gateway IP address is either statically stored inside the nodes or dynamically collected from the DHCP server. In the MITM attack, the attacker modifies the ARP tables to either pretend being the gateway or the target device.

For example, in a communication between two computers A and C with IPs 10.0.1.2 and 10.0.2.2 respectively, as shown in Figure 1. Computer A sends a packet to computer C which is outside the LAN with the source IP of A and destination IP of C. Then, computer A tries to find out whether the destination IP is in the same network or outside it. A will discover that 10.0.2.2 is outside the network. Therefore, A should forward the packet to the gateway which is Router 1 with IP address 10.0.1.1.

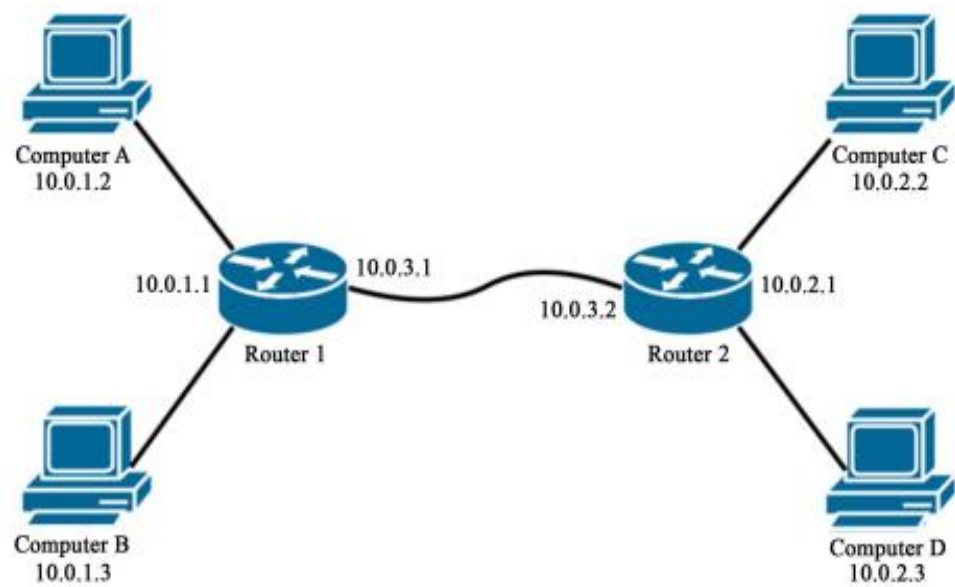
The IP packet is sent to the lower layer of the OSI model (data link layer). The source MAC address is set to the MAC address of computer A, and the destination MAC address is the one that matches Router 1 interface which has IP address 10.0.1.1. If A's ARP table does not contain the MAC address of the 10.0.1.1 interface, then A broadcasts an ARP request packet to figure out the MAC address of the gateway.

The attacker receives the broadcasted ARP request, and replies by a spoofed ARP reply packet to the victim's machine (i.e., machine A). Even if the gateway sends a legitimate ARP reply, the attacker may send multiple spoofed ARP reply packets to overwrite the legitimate ARP reply packet sent by the gateway. In ARP protocol, the newest reply overwrites the oldest ones. Therefore, the ARP cache on the victim machine will save the spoofed IP/MAC mapping.

Now, all traffic going out of the victim's machine to the gateway will be sent to the attacker's machine. In order to manipulate the gateway's ARP table to force all the packets destined to the victim from outside the LAN, to be received by the attacker, the same above procedure will be repeated by sending spoofed ARP reply packets to the gateway. Additionally, to maintain the victim's connection to the gateway running, such that the attack will be un-detectable, the attacker must keep forwarding the victim's traffic to the gateway and vice versa.

Figure 2 shows a network topology with one attacker, one victim machine where an MITM attack will be applied against it, and one router which plays the role of a gateway. Table 1 shows the contents of the spoofed ARP reply packets that should be sent by the attacker. Both packets should hold a proper op-

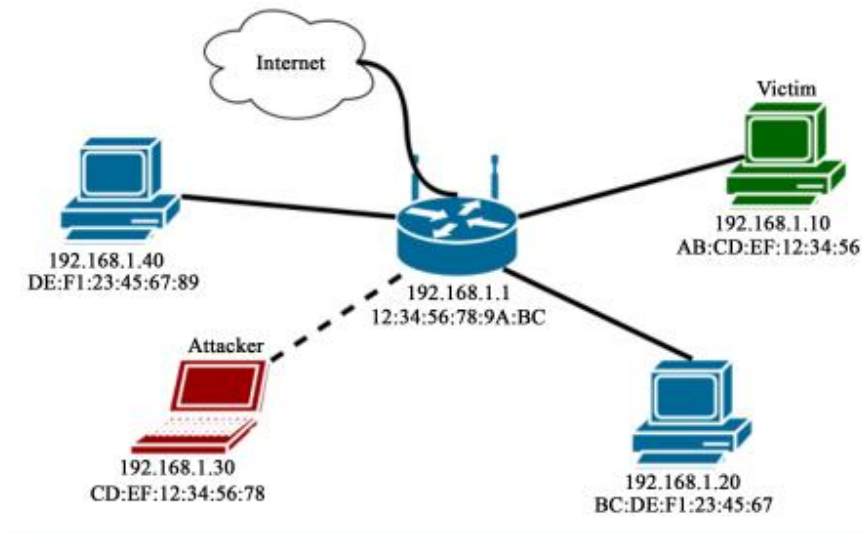
code denoting them as ARP reply packets. Table 2 shows the IP/MAC cache entries of the victim’s machine and the router; before and after the attack.



(Example of network topology)

Table 1. Required ARP Messages fields to be sent

ARP Message Fields	To victim	To router
Sender Mac address	CD:EF:12:34:56:78	CD:EF:12:34:56:78
Sender IP address	192.168.1.1	192.168.1.10
Target Mac ddress	AB:CD:EF:12:34:56	12:34:56:78:9A:BC
Target IP address	192.168.1.10	192.168.1.1



(A victim and an attacker in a LAN.)

Table 2. ARP cache entries before and after ARP spoofing attack.

	Victim's Machine		Router	
	IP	MAC	IP	MAC
Before	192.168.1.1	12:34:56:78:9A:BC	192.168.1.10	AB:CD:EF:12:34:56
After	192.168.1.1	CD:EF:12:34:56:78	192.168.1.10	CD:EF:12:34:56:78

Subsequently, any traffic from the victim to outside the LAN will pass through the attacker machine. For example, suppose that the victim opens a web page on an HTTP server with some external IP address (A.B.C.D) that is outside the LAN. The operating system running on the victim's machine will choose the router as the next hop. To encapsulate the network layer packet inside a data link frame, the victim should know the MAC address of the router. It will check the ARP cash table first. The ARP table has the poisoned mapping of router IP to the attacker's MAC address, which is the result of the ARP spoofed reply packets. Hence, that packet will be sent to the attacker's machine instead of the real HTTP server. If the attacker drops the packets, then this will results in a DoS attack where the victim is precluded from browsing the Internet. On the other hand, if the attacker forwards the traffic to its real destination, the attack is classified as a MITM attack.

Some settings should be performed at the attacker machine in order to allow it to forward the packet even if the IP is not destined to it. This is because the network layer at the attacker machine will drop the packet if it is not destined to it. To change that, a feature called the forwarding feature in the attacking machine's operating system must be enabled.

When forwarding is enabled on a machine, it reads the destination IP address in the packet, and forwards

it to the machine with MAC address bounded to that IP address based on the machine's ARP table (which contains the legitimate information). Packet forwarding is implemented in most operating systems; including Microsoft Windows, Linux and Mac OS. However, packet forwarding is disabled by default. It can be enabled easily as will see later. If packet forwarding is enabled, the victim connections will keep working without any suspensions. The attacker can still inspect the packets and extract any desired information

Packet Inspection procedure: After succeeding in attracting the traffic to the attacker's machine, the attacker needs to filter it and extract the information from the packets. Since there are millions of packets going in or out the network, it is impossible to manually check every single packet for the needed information. Therefore the attacker will filter the packets to extract the ones of interest. The most convenient way of filtering the packets is by determining which packets are valuable for the attacker. First thing that comes to mind are the ones that contain sensitive information like passwords. Moreover, the websites that the victim browses might be of interest for the attacker. Each of these types of packets has different fingerprints

The packet inspection process is done by disassembling each packet, searching for the protocols that are of interest and the fingerprint of these packets to extract the information that are needed. Figure 3 shows a packet dump as an example.

Parsing the first 14 bytes, yields a relation to layer 2, the information that can be extracted from this layer are the MAC addresses of the source and the destination. MAC addresses are 48 bit or 6 bytes long. The first 6 bytes represent the destination address (c8:3a:35:1f:1b:c8) while the second 6 bytes represent the source address (a4:17:31:0c:9a:65). Bytes offset 14 through 33 belongs to layer 3 which contains IP addresses and the network protocol.

offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	c8	3a	35	1f	1b	c8	a4	17	31	0c	9a	65	08	00	45	00
16	00	46	88	01	00	00	40	11	81	70	c0	a8	01	6b	d0	43
32	de	de	df	81	00	35	00	32	8d	31	d6	ee	01	20	00	01
48	00	00	00	00	00	01	03	77	77	77	02	6a	75	03	65	64
64	75	02	6a	6f	00	00	01	00	01	00	00	29	10	00	00	00
80	00	00	00	00												

Table 3 shows some decoded parts of the packet and the information extracted from these parts. It is clear from the table that the packet is a standard DNS query, issued by a machine with IP address of 192.168.1.107 asking a DNS server at IP address 208.67.222.222 about the domain www.ju.edu.jo.

Table 3. Inspection of the packet shown in Figure 3in Hexadecimal.

Byte offset	Hex data	Decoded data	Description
23	11	17	UDP protocol
26:29	C0 A8 01 6B	192.168.1.107	Source IP address
30:33	D0 43 DE	208.67.222.222	Destination IP address
34:35	DF 81	57217	Source port
36:37	00 35	53	Destination port
44:45	01 20	0120	DNS parameters
54:68	03 77...6F	www.ju.edu.jo	DNS query domain

How does ARP work

ARP table provides a mapping between IP addresses and MAC addresses. This mapping is stored on a machine in the form of a table where the key is some IP address and the value is the MAC address associated with the IP address. This mapping is often referred to as ARP cache or ARP table. This table is extremely helpful to improve the performance because network host consults this table to get the MAC address for a given IP if the table contains the mapping and broadcasting to resolve an IP address whenever you want to send a packet to a network host in the local area network is extremely expensive. It could easily cripple your LAN. Besides the mapping, each entry in the table has a Time-To-Live. After TTL, ARP table will delete the entry. This is helpful for avoiding table to be full

Initially this mapping (will now be referred to as ARP-Table) is empty when the host first comes on the network. In the case where a host has an empty table or simply no entry for some desired IP address, the host broadcasts to the broadcast MAC address (ff:ff:ff:ff:ff:ff) with a query for the unknown IP. This query is in the form of an ARP message where the op field, specifying the message type, is set to ‘who-has’. In a ‘who-has’ query, the message contains some pdest information; this field specifies the IP we want resolved. Finally, since the sender must have a way to receive a response, the query message also contains a source MAC address (hwsrc field). Since this message is broadcasted, all hosts on the network receive this message, however the ARP protocol states that every host that does not have the matching IP should simply drop the packet. The one host which does have this destination IP will respond to the sender with an ARP message encoded with the ‘is-at’ op. Also, it learns the MAC address of the sender IP and adds the entry to its ARP table. This preemptive caching is efficient because usually the responder

host will eventually send messages to the sender host. In this response message, it sends back its own MAC address within the 'hwsrc' field of the response. It uses the requests src field to then send a unicast message to the sender of the query. When the sender gets this unicast message, it caches the entry in its ARP table to avoid next broadcast. The figure below shows a sample ARP request and response as seen in our final presentation test.

###[ARP]###	
hwtype	= 0x1
ptype	= 0x800
hwlen	= 6
plen	= 4
op	= who-has
hwsrc	= cc:cc:cc:cc:cc:cc
psrc	= 192.168.1.64
hwdst	= ff:ff:ff:ff:ff:ff
pdst	= 192.168.1.128

###[ARP]###	
hwtype	= 0x1
ptype	= 0x800
hwlen	= 6
plen	= 4
op	= is-at
hwsrc	= 2e:bf:ac:ef:5a:b5
psrc	= 192.168.1.128
hwdst	= cc:cc:cc:cc:cc:cc
pdst	= 192.168.1.64

(Figure 1: Broadcast ARP query (left), Unicast ARP response (right). Important Fields Highlighted in White.)

Limitations of ARP

The limitations of ARP are directly related to its purpose. The purpose of ARP is to allow two hosts on the same network to communicate with one another. There is a fundamental assumption made within this paradigm and that is the fact that these hosts are all assumed to be trusted. Therefore, it doesn't have authentication step. While this assumption may have been well-founded when this protocol was invented (in 1982), it is no longer well founded. There are many cases in which a malicious user may be able to join a network. One such example is public networks, such as the ones seen in starbucks and some hotels. In this case any malicious user can join the local network along with their potential victims. Another such instance is an otherwise benign user being compromised on the network and coming under the control of a malicious user. In both these cases, all hosts on the network blindly trust the ARP responses of these malicious invaders.

Potential vulnerabilities of ARP

ARP is is often considered as 2.5 layer in networks because it is widely used to resolve IP addresses into MAC addresses. When a packet needs to be transferred from one host to another in a local area network, the destination IP must be translated into MAC address since any layer below network layer only understands MAC address. To resolve the mapping, an ARP request is sent from the host. The request is a broadcast message that is sent out on the entire LAN. Every host in the LAN will receive the message and is expected to reply only if the IP matches its host. This is known as ARP reply. To avoid sending a

broadcast message every time you want to send a message to another host in the LAN, every hosts caches ARP replies. Next time, when the host wants to send a message, it will consult the cache first. If cache has the mapping, the sender uses the entry in the cache. The problem in here is that hosts cache the ARP reply regardless of who sent it. That is, no verification of the identity of the sender is done at all. This lack of verification can expose a large attack surface to attackers on the network and allow them to spoof ARP messages that can enable them to gain substantial power over the data flow within the network.

METHODOLOGY

Introduction

This Section describes the methodology applied during the development of Shoppingly X. A methodology is a model, which project managers employ for the design, planning, implementation and achievement of their project objectives. Effective project management is essential in absolutely any organization, regardless of the nature of the business and the scale of the organization. From choosing a project to right through to the end, it is important that the project is carefully and closely managed. Based on the nature of my project solution, it was essential to use incremental Software development life cycle (SDLC). The project typically has a number of Phases and the level of control required over each phase are primarily defined by the nature of the Project, the complexity of the same and the industry to which the Project has to cater to. An Incremental (SDLC) model consists of a number of dependent increments that are completed in a prescribed sequence. Each increment includes a Launching, Monitoring and Controlling, and Closing Process Group for the functions and features in that increment only. Each increment integrates additional parts of the solution until the final increment, where the remaining parts of the solution are integrated.

Justification for the Methodology

This model can be used when the requirements of the complete system are clearly defined and understood, like the case of this project where;

- Major requirements were evidently defined; however, some details evolved with time.
- There was a need to complete the project within a short time schedule.
- A new technology is being used or the resources with needed skill set are not available. I was learning Flask and Django and could iterate from one technology to another to ensure I effectively implement all the functionalities.
- The project had some high-risk features and goals.

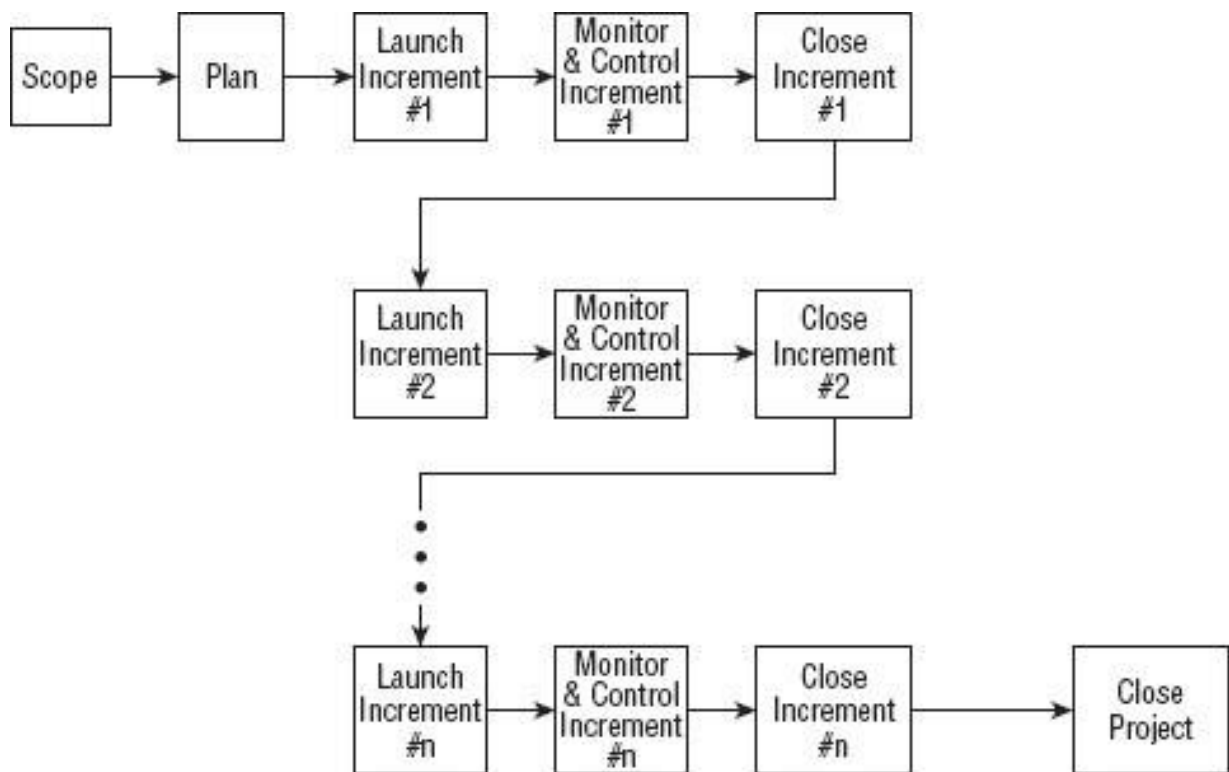


Figure 1: Incremental Project Management Life Cycle

The Incremental model is much better equipped to handle change. Each incremental functionality is verified by the customer and hence the relative risk in managing large and complex projects is substantially reduced. On the downside, there is a possibility of gold plating, wherein the functionalities not really required end up being built into the Product or Deliverable. In a nutshell, Incremental SDLC provide plethora of advantages inducing;

- Generates working software quickly and early during the software life cycle.
- This model is more flexible and less costly to change scope and requirements.

- It is easier to test and debug during a smaller iteration.
- In this model customer can respond to each built.
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during it'd iteration.

SYSTEM ANALYSIS

Analysis is an important part of any project; if analysis is not done properly then whole project move in the wrong direction. It also provides a schedule for proper project work. Analysis task divided into 3 areas: -

- ✓ Problem Recognition.
- ✓ Feasibility Study.
- ✓ Requirement Analysis.

Feasibility Study

Feasibility study of the system is a very important stage during system design. Feasibility study is a test of a system proposal according to its workability impact on the organization, ability to meet user needs, and effective use of resources. Feasibility study decides whether the system is properly developed or not. There are five types of feasibility as mentioned below:

1. Technical Feasibility
2. Time Schedule feasibility
3. Operational feasibility

4. Implementation feasibility
5. Economic Feasibility
1. Technical Feasibility

Technical feasibility corresponds to determination of whether it is technically feasible to develop the software. Here those tools are considered, which will be required for developing the project. The tools, which are available, and tools, which will be required, are taken into account. Considering all above points and aspects it is observed that the cost incurred in developing this project from a technical perspective would not be too high. Thus, it is feasible for company as well as for me to develop this system.

2. Time Feasibility

Time feasibility corresponds to whether sufficient time is available to complete the project.

Parameters considered:

- Schedule of the project.
- Time by which the project has to be completed.
- Reporting period

Considering all the above factors it was decided that the allotted time that is 3 months was sufficient to complete the project.

3. **Operational Feasibility**

Operational feasibility corresponds to whether users are aware of interface environment and sufficient resources are available or not.

Parameters considered:

- People with a basic knowledge of computers would be able to use our system very effectively and easily, as the system would have an intuitive **GUI**. The director and employees of Arp spoofer project have a basic operating knowledge of computers, so understanding the working of the system and using it would be easy from the decision maker's point of view.
- All the relevant necessary resources for implementing and operating this system are already present in office.

Bearing in mind the above factor, it was observed that the cost would be incurred in developing this project from an operational standpoint would be low. Thus, it would be operational feasible for the company.

4. Implementation Feasibility

Implementation Feasibility is about basic infrastructure required to develop the system. Considering all below points, it is feasible to develop system.

Factors considered:

- All the minimum infrastructure facility required like PC, books, technical manuals are provided.
- Proper guidance is provided.
- All necessary data and files are provided.

5. Economic Feasibility

Economic Feasibility is about total cost incurred for the system. The software resource requirement of the proposed system is flask for functional and backend development and HTML, CSS, JS for the frontend UI.

Designing and Implementing an Algorithm To Discover clients To The Same Networks

Network Scanner Algorithm : Discover clients on network.

Steps:

1. Create arp request directed to the broadcast MAC asking for an IP.
 - Use ARP to ask who has target ip.
 - Set destination MAC to broadcast MAC.
2. Send Packet and receive response .
3. Parse the response.

Implementation of Network Scanner Algorithm:

Step 1:

1. `#!/usr/bin/env python`
2. `import scapy.all as scapy`
3. `def scan(ip):`
4. `arp_request =scapy.ARP(pdst=ip)`
5. `broadcast = scapy.Ether(dst= 'ff:ff:ff:ff:ff:ff')`
6. `broadcast.show()`
7. `arp_request_broadcast =broadcast/arp_request`
8. `arp_request_broadcast.show()`
9. `scan('10.0.2.1/24')`

Output:

```
root@kali:~/PycharmProjects/network_scanner# python network_scanner.py
##[ ARP ]##
hwtype = 0x1
ptype = 0x800
hwlen = 6
plen = 4
op = who-has
hwsrc = 08:00:27:be:0c:78
psrc = 10.0.2.8
hwdst = 00:00:00:00:00:00
pdst = Net('10.0.2.1/24')
```

```

###[ Ethernet ]###
dst      = ff:ff:ff:ff:ff:ff
src      I = 08:00:27:be:0c:78
type     = 0x9000

###[ Ethernet ]###
dst      = ff:ff:ff:ff:ff:ff
src      = 08:00:27:be:0c:78
type     = 0x806

###[ ARP ]###
hwtype   = 0x1
ptype    = 0x800
hwlen    = 6
plen     = 4
op       = who-has
hwsrc    = 08:00:27:be:0c:78
psrc     = 10.0.2.8
hwdst    = 00:00:00:00:00:00
pdst     = Net('10.0.2.1/24')

root@kali:~/PycharmProjects/network_scanner#

```

Step 2:

1. `#!/usr/bin/env python`
2. `import scapy.all as scapy`
3. `def scan(ip):`
4. `arp_request = scapy.ARP(pdst=ip)`
5. `broadcast = scapy.Ether(dst= 'ff:ff:ff:ff:ff:ff')`
6. `arp_request_broadcast = broadcast/arp_request`
7. `answered, unanswered = scapy.srp(arp_request_broadcast, timeout =1)`
8. `print(answered.summary())`
9. `scan('10.0.2.1/24')`

Output:

```

root@kali:~/PycharmProjects/network_scanner# python network_scanner.py
Begin emission:
****Finished to send 256 packets.

Received 4 packets, got 4 answers, remaining 252 packets
Ether / ARP who has 10.0.2.1 says 10.0.2.8 ==> Ether / ARP is at 52:54:00:12:35:00 says 10.0.2.1 / Padding
Ether / ARP who has 10.0.2.2 says 10.0.2.8 ==> Ether / ARP is at 52:54:00:12:35:00 says 10.0.2.2 / Padding
Ether / ARP who has 10.0.2.3 says 10.0.2.8 ==> Ether / ARP is at 08:00:27:32:f7:07 says 10.0.2.3 / Padding
Ether / ARP who has 10.0.2.7 says 10.0.2.8 ==> Ether / ARP is at 08:00:27:08:af:07 says 10.0.2.7 / Padding
None

```

Step 3:

1. `#!/usr/bin/env python`
2. `import scapy.all as scapy`
3. `def scan(ip):`

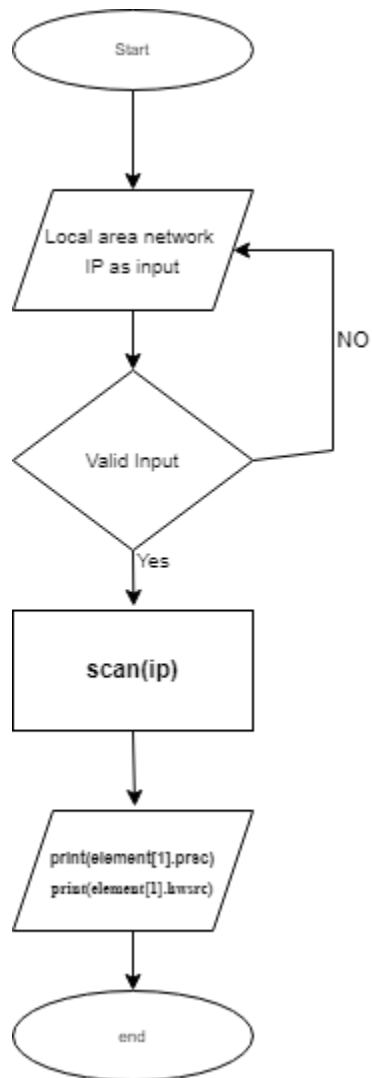
4. `arp_request =scapy.ARP(pdst=ip)`
5. `broadcast = scapy.ether(dst= 'ff:ff:ff:ff:ff:ff')`
6. `arp_request_broadcast =broadcast/arp_request`
7. `answered_list = scapy.srp(arp_request_broadcast,
 timeout =1)[0]`
- 8.
9. `for element in answered_list:`
10. `print(element[1].prsc)`
11. `print(element[1].hwsrc)`
12. `scan('10.0.2.1/24')`

Output:

```
root@kali:~/PycharmProjects/network_scanner# python network_scanner.py
Begin emission:
****Finished to send 256 packets.

Received 4 packets, got 4 answers, remaining 252 packets
10.0.2.1I
52:54:00:12:35:00
-----
10.0.2.2
52:54:00:12:35:00
-----
10.0.2.3
08:00:27:32:f7:07
-----
10.0.2.7
08:00:27:08:af:07
-----
```

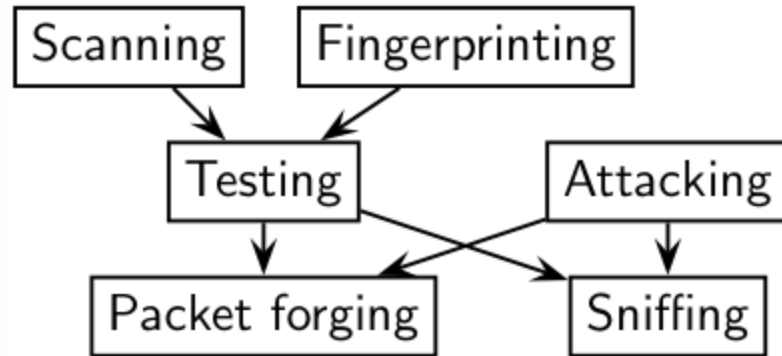
Flowchart:



Python Module : A Python module is a file containing Python definitions and statements. A module can define functions, classes, and variables. A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use. It also makes the code logically organized.

- **Scapy Module :** Scapy is a Python program that enables the user to send, sniff and dissect and forge network packets. This capability allows construction of tools that can probe, scan or attack networks. In other words, Scapy is a powerful interactive packet manipulation program. It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much

more. Scapy can easily handle most classical tasks like scanning, tracerouting, probing, unit tests, attacks or network discovery. It can replace hping, arpspoof, arp-sk, arping, p0f and even some parts of Nmap, tcpdump, and tshark.



Scapy also performs very well on a lot of other specific tasks that most other tools can't handle, like sending invalid frames, injecting your own 802.11 frames, combining techniques (VLAN hopping+ARP cache poisoning, VOIP decoding on WEP encrypted channel, ...), etc.

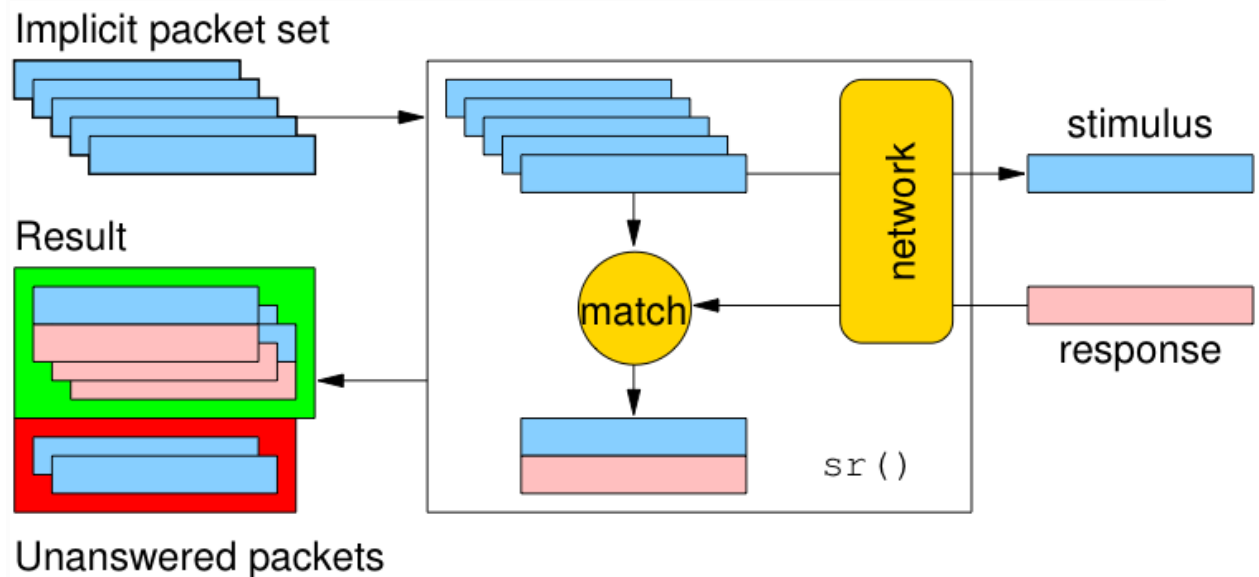
The idea is simple. Scapy mainly does two things: sending packets and receiving answers. You define a set of packets, it sends them, receives answers, matches requests with answers and returns a list of packet couples (request, answer) and a list of unmatched packets. This has the big advantage over tools like Nmap or hping that an answer is not reduced to (open/closed/filtered), but is the whole packet.

On top of this can be build more high level functions, for example, one that does traceroutes and give as a result only the start TTL of the request and the source IP of the answer. One that pings a whole network and gives the list of machines answering. One that does a portscan and returns a LaTeX report.

Probe once, interpret many

Network discovery is blackbox testing. When probing a network, many stimuli are sent while only a few of them are answered. If the right stimuli are chosen, the desired information may be obtained by the responses or the lack of responses. Unlike many tools, Scapy gives all the information, i.e. all the stimuli sent and all the responses received. Examination of this data will give the user the desired information. When the

dataset is small, the user can just dig for it. In other cases, the interpretation of the data will depend on the point of view taken. Most tools choose the viewpoint and discard all the data not related to that point of view. Because Scapy gives the complete raw data, that data may be used many times allowing the viewpoint to evolve during analysis. For example, a TCP port scan may be probed and the data visualized as the result of the port scan. The data could then also be visualized with respect to the TTL of response packet. A new probe need not be initiated to adjust the viewpoint of the data.



System requirements: The following packages should be installed globally, as the superuser, for all users on the system to access.

- Python 2.7.x or 3.4 and above.
- Python development libraries (i.e. header files for compiling C code)
- pip
- virtualenv
- virtualenvwrapper

The following sections describe the process for installing these requirements on various systems. In each of the following examples, it is assumed you will be using a root account (or some other privileged account).

Debian/Ubuntu

Flask-Diamond installs cleanly on Debian and Ubuntu systems released after 2011.

```
apt-get install python python-dev python-pip build-essential
apt-get install sqlite-dev
pip install --upgrade pip
pip install --upgrade virtualenv
pip install virtualenvwrapper
```

Project requirements :

```
Jinja2==3.1.2
MarkupSafe==2.1.1
nose==1.3.7
oauthlib==3.2.0
platformdirs==2.5.2
pyparser==2.21
PyJWT==2.4.0
python3-openid==3.2.0
pytz==2022.1
Random-Word==1.0.7
requests==2.27.1
requests-oauthlib==1.3.1
scapy==2.4.5
scapy-python3==0.26
six==1.16.0
social-auth-app-django==4.0.0
social-auth-core==4.3.0
soupsieve==2.3.2.post1
SQLAlchemy==1.4.39
sqlparse==0.4.2
uritemplate==4.1.1
urllib3==1.26.9
virtualenv==20.14.1
Werkzeug==2.1.2
asgiref==3.5.2
beautifulsoup4==4.11.1
certifi==2021.10.8
cffi==1.15.1
charset-normalizer==2.0.12
click==8.1.3
coreapi==2.3.3
coreschema==0.0.4
cryptography==37.0.2
defusedxml==0.7.1
distlib==0.3.4
Django==4.0.5
django-bootstrap4==22.1
django-templated-mail==1.1.1
django-rest-framework==3.13.1
django-rest-framework-simplejwt==4.8.0
djoser==2.1.0
filelock==3.7.1
Flask==2.1.2
Flask-Login==0.6.1
Flask-SQLAlchemy==2.5.1
idna==3.3
itsdangerous==2.1.2
itypes==1.2.0
```

Table: 1 Hardware Requirements

SL	Hardware	Minimum System Requirement
01	Processor	2.4 GHz Processor speed
02	Memory	2 GB RA
03	Disk Space	128 GB

Table: 2 Software Requirements

SL	Software	Minimum System Requirement
01	Operating System	Windows 8, Windows 10 or MAC Ox 10.8,10.9, or 10.11,LINUX
02	Runtime Environment	PyCharm or Visual Studio Code

IMPLEMENTATION

The ARP class from Scapy library is used to forge ARP reply packets. Considering the network topology shown in Figure 2, the following code block illustrates how to create a packet object of the ARP packet that should be sent to the victim's machine.

```
fake_vic = ARP(op=2,  
hwsrc="CD:EF:12:34:56:78",  
psrc="192.168.1.1",  
hwdst="AB:CD:EF:12:34:56",  
pdst="192.168.1.10")
```

The first parameter is the operation code of the ARP packet. The value 2 denotes that it is a reply packet. The second parameter is the MAC address of the source machine. The third parameter defines the IP address of the source machine. The IP address of the router is passed to this parameter to deceive the victim's machine. The fourth parameter is the destination IP address which is of the victim's machine. And the fifth parameter is the MAC address of the destination which is the victim's machine MAC address.

To send a packet, a function called send() provided by Scapy library that is capable of sending a packet object passed to it. The following code is a call to send() function to send the previously created ARP packet. *send(fake_vic)*

After applying this command, the ARP table will have poisoned entries. Now, any packet sent from the victim's machine and directed at the router will be sent to the attacking machine. Another packet object should be created to deceive the router, with same parameter but with different values that correspond to the attacking situation shown in Figure 2. The following code block shows the creation of this packet.

```
fake_gw = ARP(op=2,  
hwsrc="CD:EF:12:34:56:78",  
psrc="192.168.1.10",  
hwdst="12:34:56:78:9A:BC",  
pdst="192.168.1.1")
```

Most routers feature a web based interface to manage their settings and configurations, which can be used to show the ARP table entries of the router to verify the success of this step. A problem that could arise now is that after awhile the router may send an ARP reply telling its correct MAC address. This means the victim's machine will not remain deceived. A solution to this problem is to keep sniffing ARP replies sent by the router and resend the defined forged ARP replies once an ARP reply is detected. Scapy library provides the sniff() function to sniff specific data packets matching a defined filter. The ARP packet intended to be detected is either

directed at or issued from the router or the victim's machine. Therefore, any ARP packet containing the IP address of the router or the victim's machine should be followed by forged ARP packets; to keep the attack functioning. The following call to sniff() function blocks the loop till any ARP packet matching the filter is detected.

```
while True:
    sniff(filter="arp and host 192.168.1.1 or host
192.168.1.10", count=1)
    send(fake_vic)
    send(fake_gw)
```

The remaining last step to accomplish a complete MITM attack is to forward the IP packets to its real destination based on the attacker's machine ARP table. Linux systems provide this capability at the kernel level out of the box. The system flag "net.ipv4.ip_forward" holds the state of IP forwarding which is 0 by default in most Linux distributions. Once it is set to 1, the kernel starts the IP forwarding process. Sysctl is a user space interface to change system kernel parameters at run-time. The following command is to enable IP forwarding:

Now, the attacker is in position of man in the middle. The victim's traffic should pass through the attacker's machine and then forwarded to its proper destination. It is clear from the details above that the attacker now is able to filter packets, modify the exchanged packets, and send forged packets. This is the source to many other attacks like DNS spoofing attack and phishing attacks. Due to space limitations, the implementation details of packet inspection and those attacks will not be shown here.

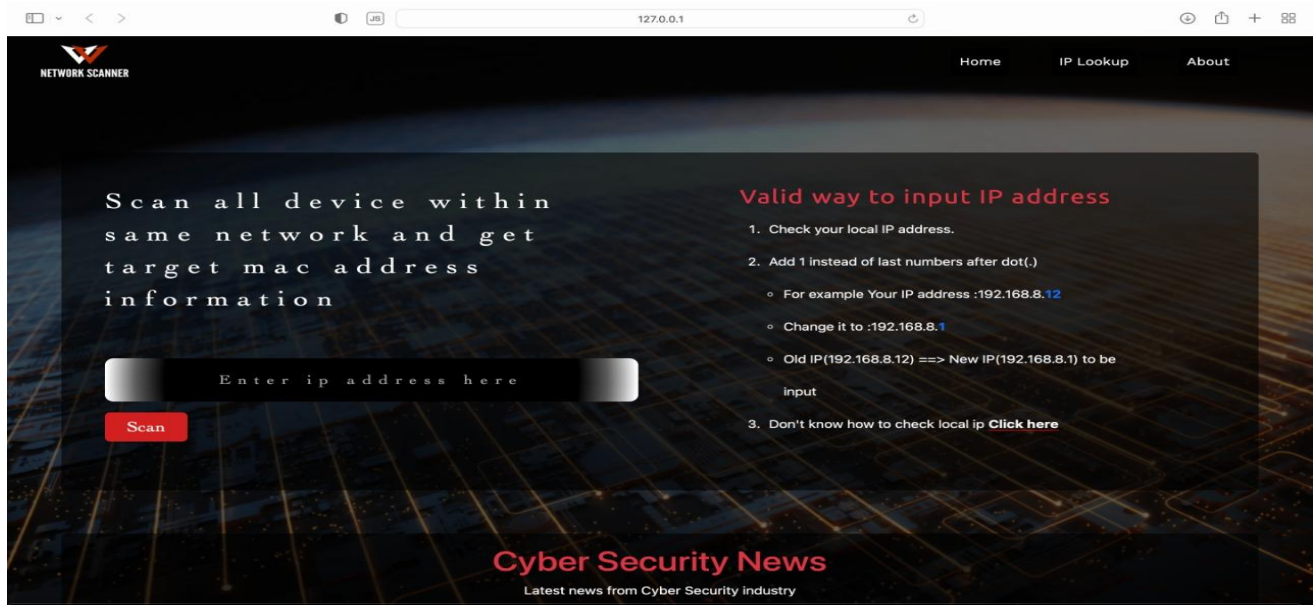
Project Output

At the starting screen of the application user encounters a home page. And here user see input bar where user enter the local ip address for perform arp attack.

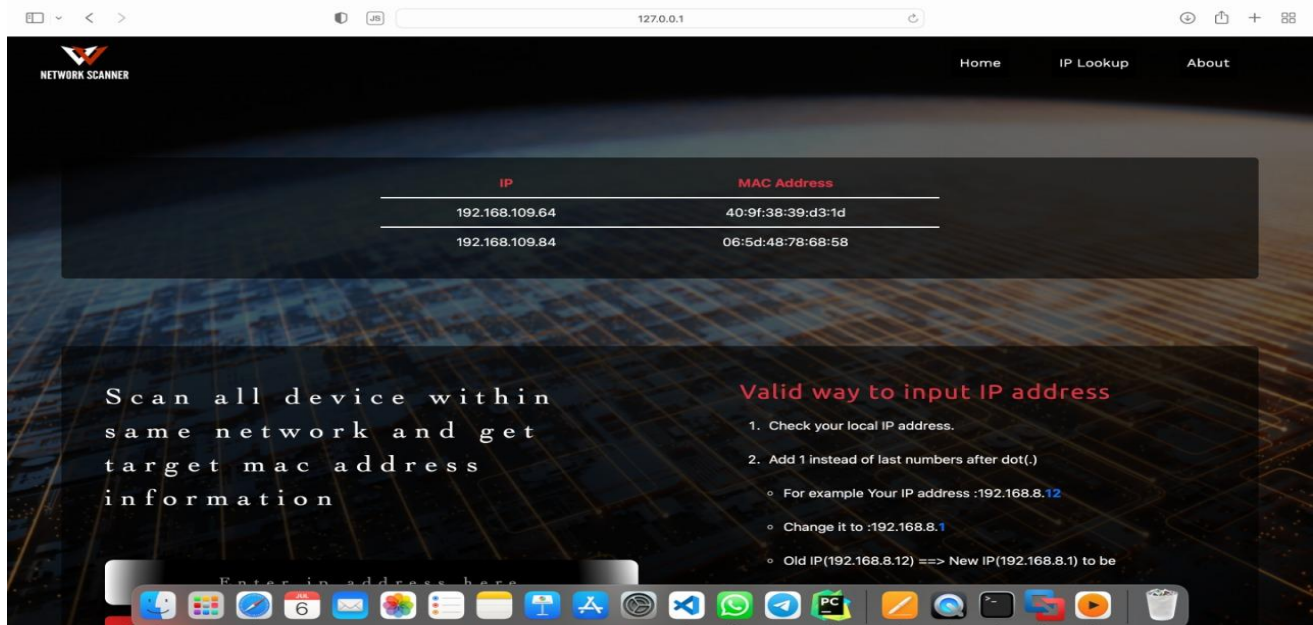
The other main key points or innovation in our web application is that user also gather information about ip address by going to IP Lookup page.

The Main activity:

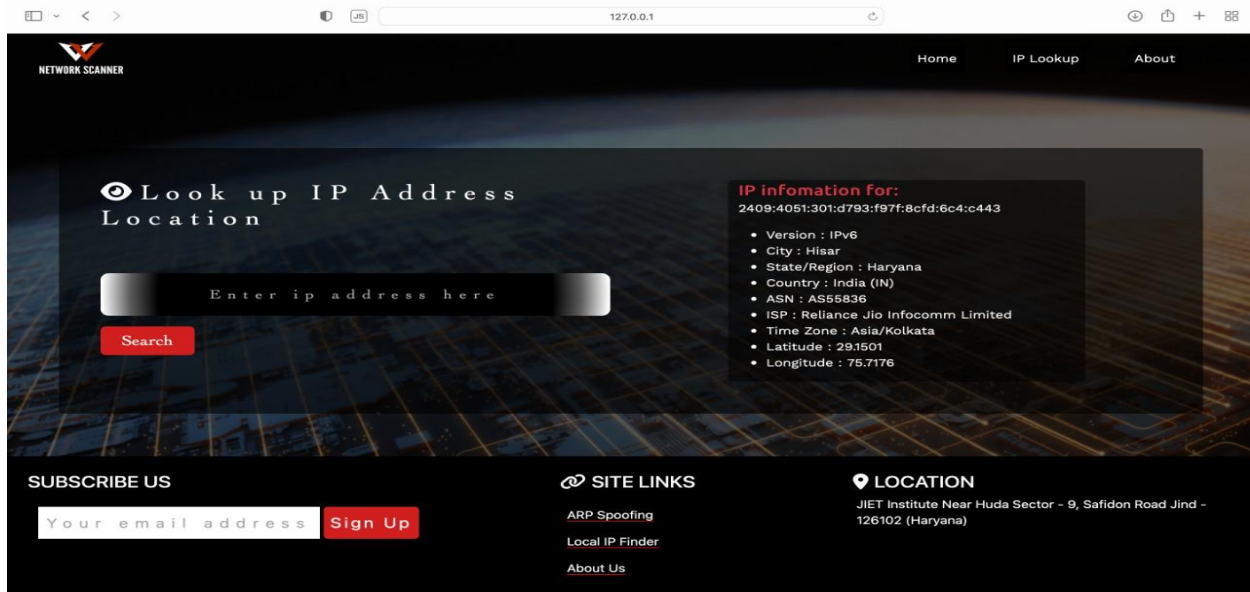
The following is the GUI of the Main activity



Output



IP-Lookup: An IP address lookup **determines the location of any IP address**. The results provide quite a bit of information. Details include city, state/region, postal/zip code, country name, ISP, and time zone. As a result, this data is used by various agencies to find the owner of an IPv4 or IPv6 address.



The screenshot shows a web application titled "NETWORK SCANNER" with a navigation bar containing "Home", "IP Lookup", and "About". The main content area features a large heading "Look up IP Address Location" and a search input field with the placeholder text "Enter ip address here". A red "Search" button is positioned below the input field. To the right of the search area, a box displays "IP information for:" followed by the IP address "2409:4051:301:d793:f97f:8cfd:6c4:c443". Below the IP address, a list of details is shown:

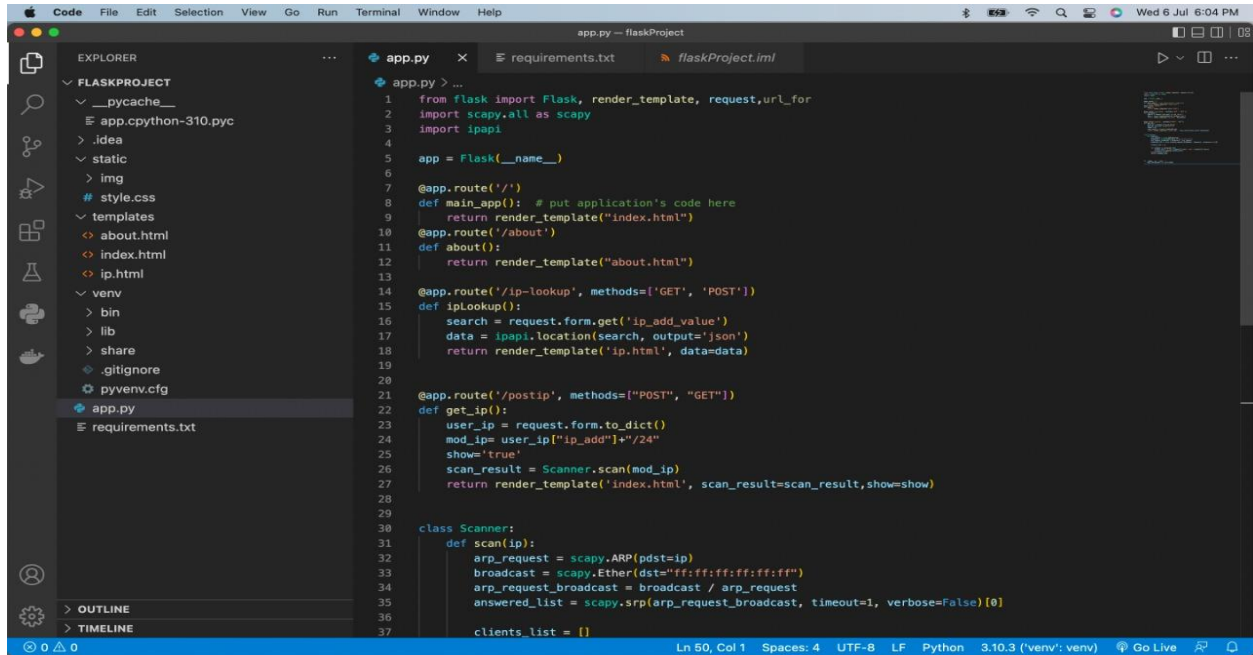
- Version : IPv6
- City : Hisar
- State/Region : Haryana
- Country : India (IN)
- ASN : AS55836
- ISP : Reliance Jio Infocomm Limited
- Time Zone : Asia/Kolkata
- Latitude : 29.1501
- Longitude : 75.7176

The footer of the application is divided into three sections: "SUBSCRIBE US" with an email input field and a "Sign Up" button; "SITE LINKS" with links to "ARP Spoofing", "Local IP Finder", and "About Us"; and "LOCATION" with the address "JIET Institute Near Huda Sector - 9, Safidon Road Jind - 126102 (Haryana)".

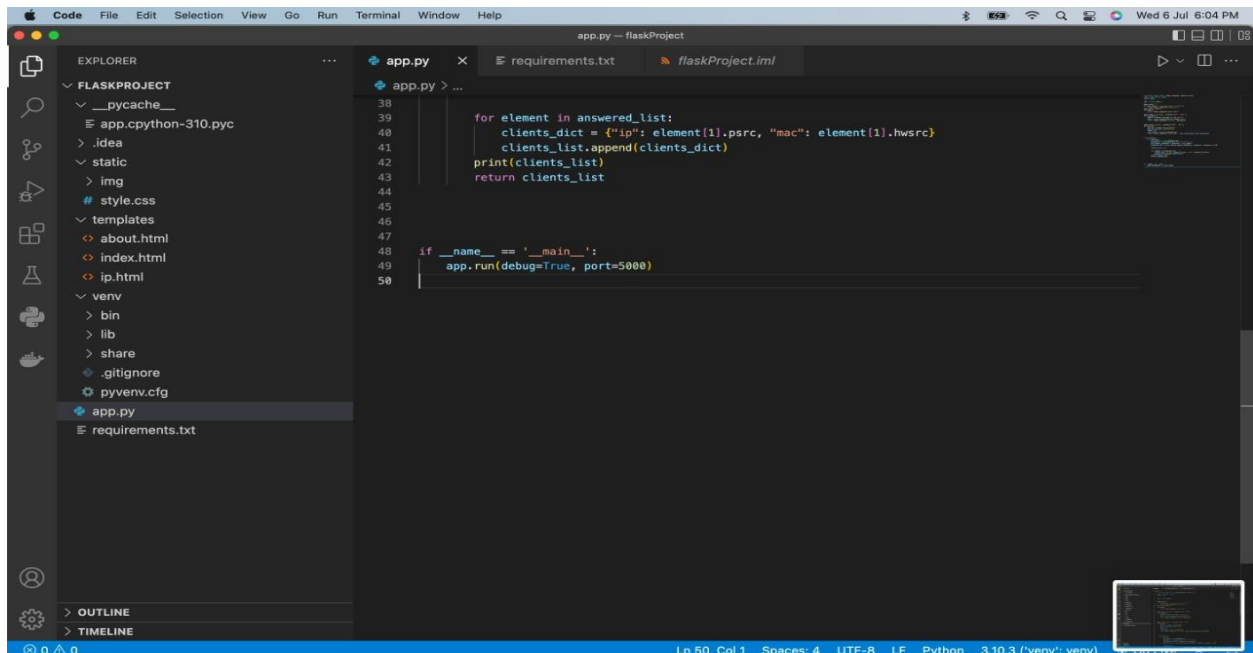
TESTING

Testing was done throughout the development of the project. In fact, I had an android phone connected with the computer for responsive design. Whenever I add a feature or a functionality, I test it right away on both devices Laptop as well as in Mobile.

This helped me find errors right away and have each and every functionality of the web application tested.



```
1 from flask import Flask, render_template, request, url_for
2 import scapy.all as scapy
3 import ipapi
4
5 app = Flask(__name__)
6
7 @app.route('/')
8 def main_app(): # put application's code here
9     return render_template("index.html")
10 @app.route('/about')
11 def about():
12     return render_template("about.html")
13
14 @app.route('/ip-lookup', methods=['GET', 'POST'])
15 def ipLookup():
16     search = request.form.get('ip_add_value')
17     data = ipapi.location(search, output='json')
18     return render_template('ip.html', data=data)
19
20 @app.route('/postip', methods=["POST", "GET"])
21 def get_ip():
22     user_ip = request.form.to_dict()
23     mod_ip = user_ip["ip_add"]+"/24"
24     show = "true"
25     scan_result = Scanner.scan(mod_ip)
26     return render_template('index.html', scan_result=scan_result, show=show)
27
28
29
30 class Scanner:
31     def scan(ip):
32         arp_request = scapy.ARP(pdst=ip)
33         broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
34         arp_request_broadcast = broadcast / arp_request
35         answered_list = scapy.Srp(arp_request_broadcast, timeout=1, verbose=False)[0]
36
37         clients_list = []
```



```
38
39     for element in answered_list:
40         clients_dict = {"ip": element[1].psrc, "mac": element[1].hwsrc}
41         clients_list.append(clients_dict)
42     print(clients_list)
43     return clients_list
44
45
46
47
48 if __name__ == '__main__':
49     app.run(debug=True, port=5000)
50
```

CONCLUSION AND RECOMMENDATION

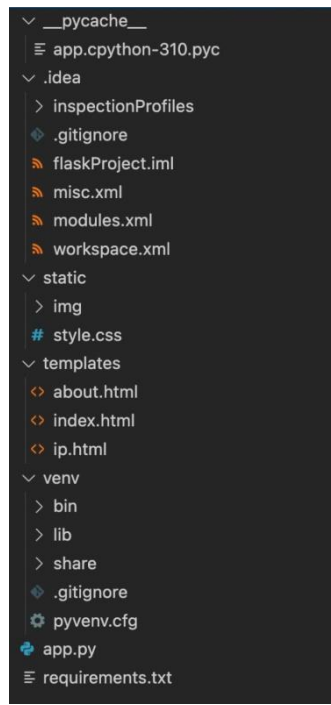
CONCLUSION

The project entitled Network Scanner –Arp Spoofer system was completed successfully. The system has been developed with much care and free of errors and at the same time it is efficient and less time consuming. The purpose of this project was to develop a web application for performing and detecting ARP Attack. This project enabled me gain valuable information and practical knowledge on several topics like designing web pages using html & CSS, usage of responsive templates, designing of full stack flask application,. The entire system is secured. Also, the project helped me understanding about the development phases of a project and software development life cycle. I learned how to test different features of a project. This project has given me great satisfaction in having designed an application which can be implemented to any nearby shops or branded shops selling various kinds of products by simple modifications. However, it was very challenging learning and developing an application using a new technology.

The fact that the ARP protocol is a stateless protocol has very dangerous consequences. And attacker could manipulate the connection of the users, steal their data, and even redirect their traffic to different websites than the ones they requested. Network administrators should become more aware of these attacks and take countermeasures against them. Users should also become more aware of them and use security solutions to prevent getting their data stolen.

In this paper, after explaining the vulnerabilities of the ARP protocol and their different types and categories, we have designed and implemented an application that performs ARP spoofing and MITM attack. After that, we have designed and implemented a simple and distributed defense mechanism and tool that works on end user devices to keep him protected from these types of attacks. This mechanism helps in protecting the user from such attacks as shown by applying and running final product on the user machine.

Structure of the Project Files



Flask is a micro web framework written in Python.

- **app.py** : This file is used basically as a command-line utility and for deploying, debugging, or running our web application. It contains code for run-server, or make migrations or migrations, and code for performing all operation of web application etc. that we use in the shell.
- **Static** folder contains all the static files like CSS and images.
- **Env** folder is the project specific development environment. Its created through a command
`'virtualenv Env'`
- **templates** folder contain all the html files of the web application

RECOMMENDATION

There is a scope for further development in our project to a great extent. A number of features can be added to this system in future like providing. The feature like adding an Arp detection, Packet Sniffer, DNS Spoofer. Another feature we wished to implement was providing all cyber security and ethical hacking tool for education purpose only.. These features could have been Implemented if time and skills did not limit me

Reference:

1. Plummer, D.C. (1982) An Ethernet Address Resolution Protocol. RFC 826
2. Ornaghi, A. and Valleri, M. (2004) A Multipurpose Sniffer for Switched LANs. <http://ettercap.sf.net>
3. Wagner, R. (2001) Address Resolution Protocol Spoofing and Man in the Middle Attacks. SANS Institute. <https://www.sans.org/reading-room/whitepapers/threats/address-resolution-protocol-spoofing-man-in-the-middle-attacks-474>
4. Bellovin, S.M. (2004) A Look Back at “Security Problems in the TCP/IP Protocol Suite”. Proceedings of the 20th Annual Computer Security Application Conference (ACSAC), Tucson, 6-10 December 2004, 229-249. <http://dx.doi.org/10.1109/CSAC.2004.3>
5. Bruschi, D., Ornaghi, A. and Rosti, E. (2003) S-ARP: A Secure Address Resolution Protocol. Proceedings of the 19th Annual Computer Security Applications Conference, Las Vegas, 8-12 December 2003, 66-74. <http://dx.doi.org/10.1109/csac.2003.1254311>
6. Gouda, M.G. and Huang, C-T. (2003) A Secure Address Resolution Protocol. Computer Networks, 41, 57-71. [http://dx.doi.org/10.1016/S1389-1286\(02\)00326-2](http://dx.doi.org/10.1016/S1389-1286(02)00326-2)
7. Issac, B. (2009) Secure ARP and Secure DHCP Protocols to Mitigate Security Attacks. International Journal of Network Security, 8, 107-118.
8. Pansa, D. and Chomsiri, T. (2008) Architecture and Protocols for Secure Land by Using a Software-Level Certificate and Cancellation of ARP Protocol. ICCIT'08 3rd International Conference on Convergence and Hybrid Information Technology, 2, 21-26.
9. Jinhua, G. and Kejian, X. (2013) ARP Spoofing Detection Algorithm Using ICMP Protocol. International Conference on Computer Communication and Informatics, Coimbatore, 4-6

January 2013, 1-6. <http://dx.doi.org/10.1109/iccci.2013.6466290>

- 10.** Hong, S., Oh, M. and Lee, S. (2013) Design and Implementation of an Efficient Defense Mechanism against ARP Spoofing Attacks Using AES and RSA. Mathematical and Computer Modelling, 58, 254-260. <http://dx.doi.org/10.1016/j.mcm.2012.08.008>
- 11.** Gouda, M.G. and Huang, C.T. (2003) A Secure Address Resolution Protocol. Computer Networks, 41, 57-71. [http://dx.doi.org/10.1016/S1389-1286\(02\)00326-2](http://dx.doi.org/10.1016/S1389-1286(02)00326-2)