PREVENTION OF ARP SPOOFING USING NMAP

Submitted for

INFORMATION SECURITY MANAGEMENT- CSE3502

WINTER SEMESTER 2021-2022

Submitted by

AKKICHETTY NITHINSAI-19BIT0223
NIKHIL C-19BIT0331
KUNAL KUMAR-19BIT0171

Submitted to

Dr. JOHN SINGH K

BACHELOR'S IN TECHNOLOGY

IN

School of Information Technology and Engineering

Vellore Institute of Technology,Vellore

CONTENTS

ABSTRACT

Generally speaking, the average user doesn't have much visibility over who or what is on their network, so anyone with the password can slip in and start using bandwidth. In other situations, a roommate or family member may be hogging all the bandwidth unfairly to play video games or stream videos. Some routers have web interfaces that allow you to set limits on each connection, but without the password to the router, this option may not be accessible

ARP spoofing makes devices send data to the attacking computer rather than the router, allowing us to send the data on to the router at whatever speed we like. Some routers have web interfaces that allow you to set limits on each connection, but without the password to the router, this option may not be accessible

Our project focus on using Nmap for detecting nodes present on the LAN network and an ARP poisoning tool to control network traffic in order to throttle devices from a wireless network and on later part of the project we implement ideas and provide solution for the prevention of such ARP poisoning attacks.

LITERATURE SURVEY

1. PAPER NAME: An Analysis on the Schemes for Detecting and Preventing ARP Cache Poisoning Attacks

AUTHORS:Cristina L. Abad,Rafael I. Bonilla

PAPER DEFINITION : : In this paper they have analysed each of several schemes and identified their strengths and weakness. They have also proposed guidelines for design of alternative solution to ARP cache poisoning.

2. PAPER NAME :Detection and Prevention of ARP Spoofing using Centralized Server

AUTHORS:D. Srinath,S. Panimalar,A. Jerrin Simla,J. Deepa

PAPER DEFINITION: This paper presents centralized server ARP model that maintains legitimate user list with minimal of intrusion. A secure authentication is added for more secure communication & well established network for centralized networks.

3. PAPER NAME: Mitigating ARP Spoofing Attacks in Software-Defined Networks

AUTHORS:Ahmed M.AbdelSalam, Ashraf B. El-Sisi,Vamshi Reddy K

PAPER DEFINITION: This paper describes a solution for mitigating types of ARP attacks. This extends in SDN controller to include an ARP module, that detect & stop the attack , avoids overloading and provide fast and reliable solution. An openflow is used for communication between controller and switches

4. PAPER NAME: ARP Spoofing: A Comparative Study for Education Purposes

AUTHORS:Zouheir Trabelsi,Wassim El-Hajj

PAPER DEFINITION: In this paper,they conducted an extensive work to know whichSecurity Solutions are able to detect a very dangerous MAC layer attack called ARPSpoofing. The experimental results discussed in their work can be used to assist security instructorsin selecting the appropriate security solutions to be used during the hands-on labs, aswell as for building secure LAN network

5. PAPER NAME: Towards More Sophisticated ARP Spoofing Detection/Prevention Systems in LAN Networks

AUTHORS:Mohamed Al-Hemairy, Saad Amin,Zouheir Trabelsi

PAPER DEFINITION:In this research they evaluate the most famous & expensive detection and prevention [IDS/IPS] systems for detecting all types of ARP spoofing attacks and introduce an algorithm which can be implemented in IDS/IPS systems to enhance it's security.

6. PAPER NAME:A Comprehensive Analysis of Spoofing

AUTHORS:P. Ramesh Babu,D.Lalitha Bhaskari,CH.Satyanarayana

PAPER DEFINITION: They introduce and explain following spoofing attacks in thispaper: IP, ARP, E-Mail, Web, and DNS spoofing. Some of the outcomes might be sport,theft, vindication or some other malicious goal. It also describes about various spoofing types and gives a small view on detection and prevention of spoofing attacks

7. PAPER NAME: A New Approach to Prevent ARP Spoofing

AUTHORS:Divya Sharma, Oves Khan, Kanika Aggarwal, Preeti Vaidya

PAPER DEFINITION:— The proposed mechanism in this Paper is simple which can block attack at the source of attack itself and also it can prevent IP & ARP spoofing based attacks.

8. PAPER DEFINITION :Feasibility analysis of different methods for prevention against ARP spoofing

AUTHORS:Mr Sumit Miglani , Inderjeet Kaur

PAPER DEFINITION : In this paper they tell that there no much reliable and effective technique to prevent from ARP spoofing. So, there still need of a lot of work that could be done.There are many tools available to perform the attack but none to ensure complete security from such attacks. We could purpose some changes in the existing algorithms for ARP Cache poisoning prevention and detection for a hostrunning Linux.

9. PAPER DEFINITION: Principle of and Protection of Man-in-the-middle Attack Based on ARP Spoofing

AUTHORS:Guo Hao and Guo Tao

PAPER DEFINITION: In this article,the author describes a method of man-in the-middle attack in detail, and proposes some precautionary measures for preventing such attacks. Man-in-the-middle attack, one of the most important means of network attack for a hacker, consists of various attack modes, including a strong characteristic of concealment, and thus can cause great harm.

INTRODUCTION

If you find yourself with a roommate hogging limited data bandwidth with video games or discover a neighbor has invited themselves into your Wi-Fi network, you can easily take back control of your internet access. Evillimiter is an ARP poisoning tools used in kali linux for throttling the devices present in the network, Evil Limiter uses ARP spoofing to make devices send data to the attacking computer rather than the router, allowing us to send the data on to the router at whatever speed we like

ARP SPOOFING : Address Resolution Protocol (ARP) is a protocol that enables network communication to a specific device in a network. ARP poisoning is a man-in-middle attack that allows intruder to interrupt communication between network devices.

TOOLS USED :

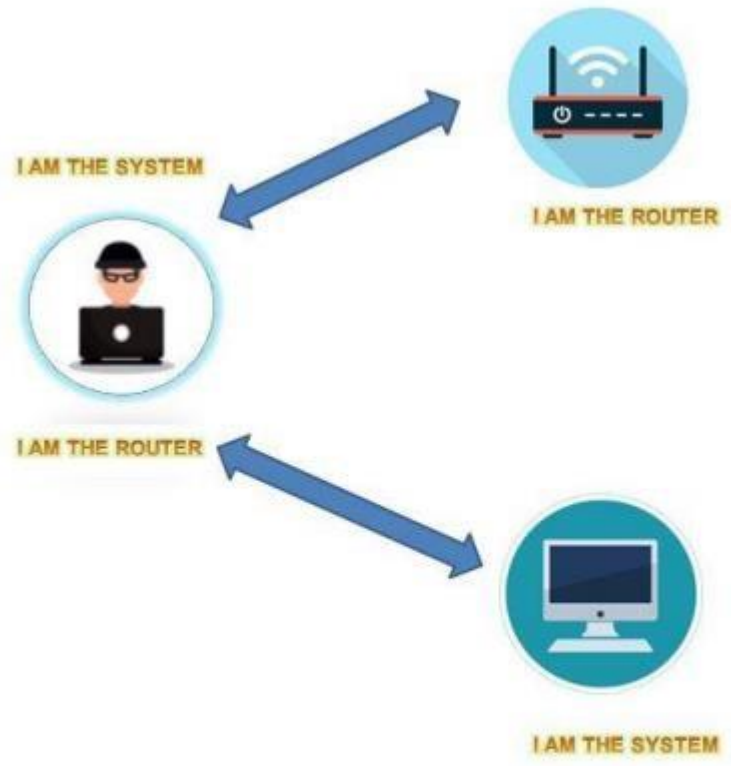
NMAP : Nmap in short is a Network Mapper, is a free, open-source tool for vulnerability scanning and network discovery. Network administrators use Nmap to identify what devices are running on their systems, discovering hosts that are available and the services they offer, finding open ports and detecting security risks.

EVILLIMITER: It is a tool which is developed on the base concept of ARP spoofing or ARP poisoning which is basically used for throttling or totally banning interconnection to a device or node present on the local network i.e WIFI or an hotspot.

WIRESHARK:

It is used for network troubleshooting, analysis, software and communications protocol development, and education. Wireshark is the world's foremost network protocol analyzer. It lets you see what's happening on your network at a microscopic level. It is the de facto (and often de jure) standard across many industries and educational institutions. Wireshark development thrives thanks to the contributions of networking experts across the globe. It is the continuation of a project that started in 1998.

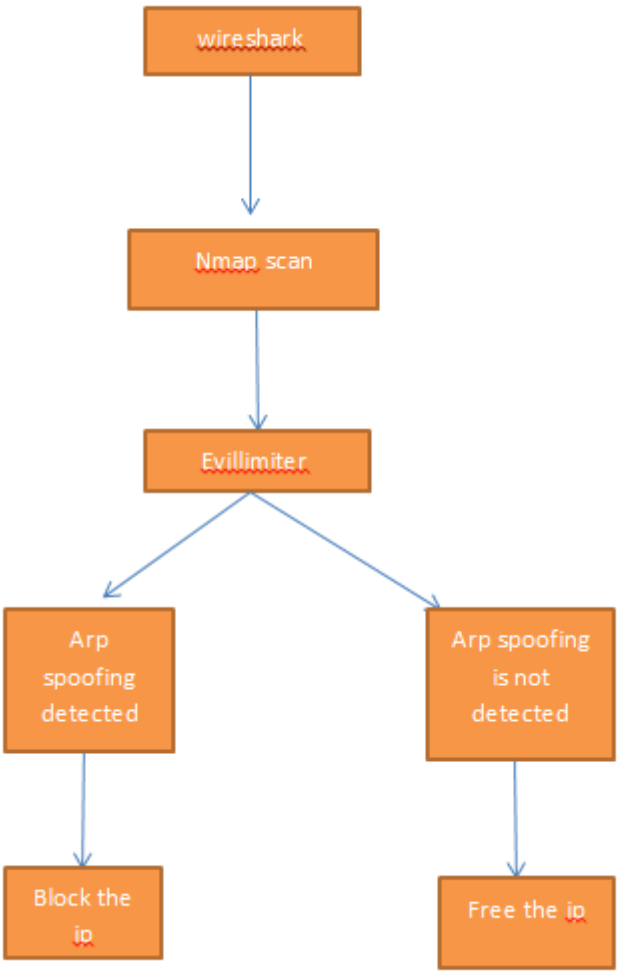
ARCHITECTURE:



The ARP spoofing attacker pretends to be both sides of a network communication channel

The attacker forge responses and advertise that the correct MAC address for both IP addresses, belonging to the router and workstation, is the attacker's MAC address. This makes both router and workstation to connect to the attacker's machine, instead of to each other. The two devices update their ARP cache entries and from that point onwards, communicate with the attacker instead of directly with each other. The attacker is now secretly in the middle of all communications.

ARCHITECTURE(BLOCK DIAGRAM):



MOTIVATION

Although it's difficult to prevent an ARP spoofing attack, encrypting your internet traffic helps to protect your information from being stolen or modified. So, any traffic sent over an HTTPS connection is encrypted. However, manually checking to ensure that every URL you visit uses HTTPS is tedious so by using the information security tools nmap,wireshark and evillimiter(arp poisoning tool)so the main motive is by using the basic security tools we can able to prevent the arp spoofing on the network easily

PROBLEM STATEMENT

ARP spoofing is a type of attack in which a malicious actor sends falsified ARP (Address Resolution Protocol) messages over a local area network. This results in the linking of an attacker's MAC address with the IP address of a legitimate computer or server on the network. Once the attacker's MAC address is connected to an authentic IP address, the attacker will begin receiving any data that is intended for that IP address. ARP spoofing can enable malicious parties to intercept, modify or even stop data in-transit. ARP spoofing attacks can only occur on local area networks that utilize the Address Resolution Protocol.

Existing system

Generally In order to prevent ARP poisoning over centralized system, voting process is used to elect legitimate CS. Validating and Correcting <; IP, MAC > pair entries residing in hosts cache tables, CS successfully prevents ARP poisoning while maintaining performance of the system. And also people generally use the vpn's to hide their ip address to prevent the arp spoofing

Proposed method

Here we simply use the three information security tools to stop the arp spoofing the tools are

- Nmap-open source tool
- Wireshark-packet capturing tool
- Evillimiter- arp poisoning tool

1 first turning on the wireshark for packet capturing the data flowing through the network

2. now performing the nmap –sP scan like This option tells Nmap not to do a port scan after host discovery, and only print out the available hosts that responded to the scan

3.now with the help of the evillimiter arp poising tool will be checking with the active hosts and performing the action of limiting and blocking the bandwidth of duplicate ip address and if there is no duplicate ip address there then freeing all the authenicate or valid ip address

IMPLEMENTATION:

System: kali linux

Tools : nmap ,wireshark

Evillimiter.py:

```
import re
import os
import os.path
import argparse
import platform
import collections
import pkg_resources

import evillimiter.networking.utils as netutils
from evillimiter.menus.main_menu import MainMenu
from evillimiter.console.banner import get_main_banner
from evillimiter.console.io import IO


InitialArguments = collections.namedtuple('InitialArguments',
'interface, gateway_ip, netmask, gateway_mac')


def get_init_content():
```

```
        with
    open(os.path.join(os.path.abspath(os.path.dirname(__file__)),
    '__init__.py'), 'r') as f:
            return f.read()


    def get_version():
        version_match = re.search(r'^_version_=
    [\'"](\d\.\d\.\d)[\'"]', get_init_content(), re.M)
        if version_match:
            return version_match.group(1)

        raise RuntimeError('Unable to locate version string.')


    def get_description():
        desc_match = re.search(r'^_description_= [\'"]((.)*)[\'"]',
    get_init_content(), re.M)
        if desc_match:
            return desc_match.group(1)

        raise RuntimeError('Unable to locate description string.')


    def is_privileged():
        return os.geteuid() == 0
def is_linux():
        return platform.system() == 'Linux'


    def parse_arguments():
        """
        Parses the main command-line arguments (sys.argv)
        using argparse
        """
        parser = argparse.ArgumentParser(description=get_description())
        parser.add_argument('-i', '--interface', help='network interface
    connected to the target network. automatically resolved if not
    specified.')
        parser.add_argument('-g', '--gateway-ip', dest='gateway_ip',
    help='default gateway ip address. automatically resolved if not
    specified.')
        parser.add_argument('-m', '--gateway-mac', dest='gateway_mac',
    help='gateway mac address. automatically resolved if not specified.')
        parser.add_argument('-n', '--netmask', help='netmask for the
    network. automatically resolved if not specified.')
        parser.add_argument('-f', '--flush', action='store_true',
    help='flush current iptables (firewall) and tc (traffic control)
    settings.')
        parser.add_argument('--colorless', action='store_true',
    help='disable colored output.')

        return parser.parse_args()


    def process_arguments(args):
        """
        Processes the specified command-line arguments, adds them to a
    named tuple
        and returns.
        Executes actions specified in the command line, e.g. flush
    network settings
```

```
        """
        if args.interface is None:
            interface = netutils.get_default_interface()
            if interface is None:
                IO.error('default interface could not be resolved.
  specify manually (-i).')
                return
        else:
            interface = args.interface
            if not netutils.exists_interface(interface):
                IO.error('interface {}{}{} does not
  exist.'.format(IO.Fore.LIGHTYELLOW_EX, interface,
  IO.Style.RESET_ALL))
                return

        IO.ok('interface: {}{}{}'.format(IO.Fore.LIGHTYELLOW_EX,
  interface, IO.Style.RESET_ALL))

        if args.gateway_ip is None:
            gateway_ip = netutils.get_default_gateway()
            if gateway_ip is None:
                IO.error('default gateway address could not be resolved.
  specify manually (-g).')
                Return
    else:
            gateway_ip = args.gateway_ip

        IO.ok('gateway ip: {}{}{}'.format(IO.Fore.LIGHTYELLOW_EX,
  gateway_ip, IO.Style.RESET_ALL))

        if args.gateway_mac is None:
            gateway_mac = netutils.get_mac_by_ip(interface, gateway_ip)
            if gateway_mac is None:
                IO.error('gateway mac address could not be resolved.')
                return
        else:
            if netutils.validate_mac_address(args.gateway_mac):
                gateway_mac = args.gateway_mac.lower()
            else:
                IO.error('gateway mac is invalid.')
                return
        IO.ok('gateway mac: {}{}{}'.format(IO.Fore.LIGHTYELLOW_EX,
  gateway_mac, IO.Style.RESET_ALL))

        if args.netmask is None:
            netmask = netutils.get_default_netmask(interface)
            if netmask is None:
                IO.error('netmask could not be resolved. specify manually
  (-n).')
                return
        else:

    netmask=ars
    .netmask
```

```
        IO.ok('netmask: {}{}{}'.format(IO.Fore.LIGHTYELLOW_EX, netmask,
    IO.Style.RESET_ALL))

        if args.flush:
            netutils.flush_network_settings(interface)
            IO.spacer()
            IO.ok('flushed network settings')

        return InitialArguments(interface=interface,
    gateway_ip=gateway_ip, gateway_mac=gateway_mac, netmask=netmask)


    def initialize(interface):
        """
        Sets up requirements, e.g. IP-Forwarding, 3rd party applications
        """
        if not netutils.create_qdisc_root(interface):
            IO.spacer()
            IO.error('qdisc root handle could not be created. maybe flush
    network settings (--flush).')
            return False

        if not netutils.enable_ip_forwarding():
            IO.spacer()
            IO.error('ip forwarding could not be enabled.')
            return False

        return True


    def cleanup(interface):
"""
        Resets what has been initialized
        """
        netutils.delete_qdisc_root(interface)
        netutils.disable_ip_forwarding()


    def run():
        """
        Main entry point of the application
        """
        version = get_version()
        args = parse_arguments()

        IO.initialize(args.colorless)
        IO.print(get_main_banner(version))

        if not is_linux():
            IO.error('run under linux.')
            return

        if not is_privileged():
            IO.error('run as root.')
            return
```

```
    args = process_arguments(args)

    if args is None:
        return

    if initialize(args.interface):
        IO.spacer()
        menu = MainMenu(version, args.interface, args.gateway_ip,
args.gateway_mac, args.netmask)
        menu.start()
        cleanup(args.interface)


if __name__ == '__main__':
    run()
```

SCREENSHOTS OF IMPLEMENTATION:

Step 1:

Sudo install nmap

Step 2:

Sudo install wireshark

Step 3:

Clone evillimiter

https://github.com/bitbrute/evillimiter.git

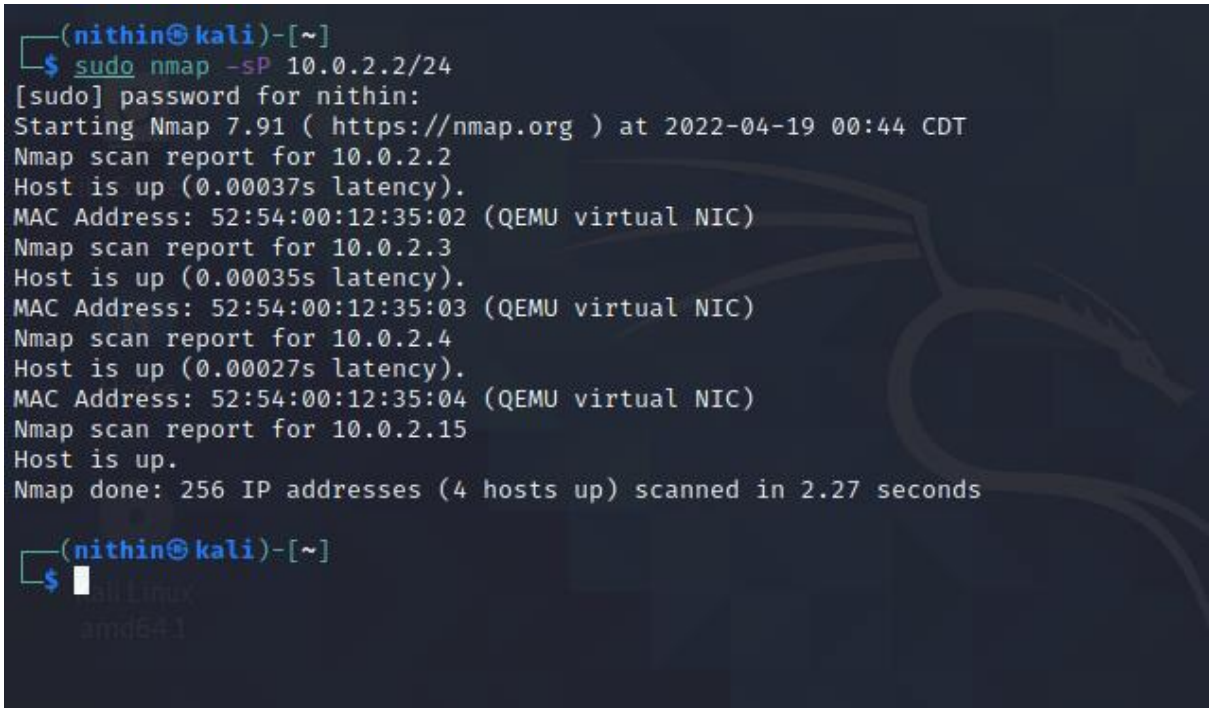
step 4:

evillimiter

scan

Step 5:

Sudo nmap -sP 10.0.2.2/24



Step 6:

Sudo wireshark

Step 7:

Hosts

```
(Main) >>> hosts

Hosts
```

ID	IP address	MAC address	Hostname	Status
0	10.0.2.2	52:54:00:12:35:02		Free
1	10.0.2.3	52:54:00:12:35:03		Free
2	10.0.2.4	52:54:00:12:35:04		Free

Step 8:

Spoofing and blocking and limiting bandwidth of the hosts

Limit 0 1kbit

Block 1

Limit 2 10kbit

```
(Main) >>> hosts

Hosts
```

ID	IP address	MAC address	Hostname	Status
0	10.0.2.2	52:54:00:12:35:02		Free
1	10.0.2.3	52:54:00:12:35:03		Free
2	10.0.2.4	52:54:00:12:35:04		Free

```
(Main) >>> limit 0 1kbit
OK   10.0.2.2 upload / download limited to 1kbit.
(Main) >>> block 1
OK   10.0.2.3 upload / download blocked.
(Main) >>> limit 2 10kbit
OK   10.0.2.4 upload / download limited to 10kbit.
(Main) >>> hosts

Hosts
```

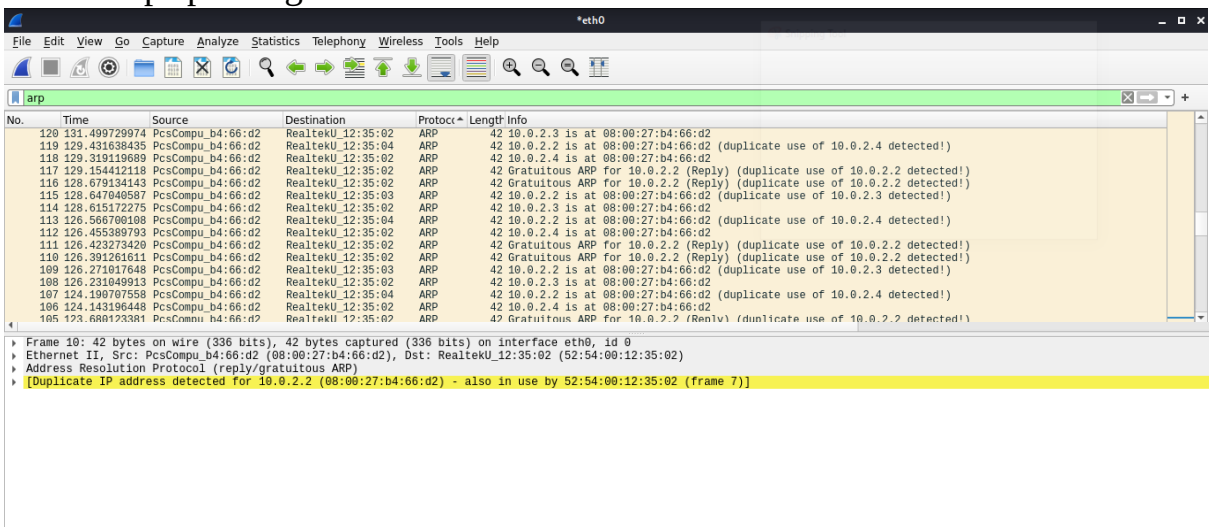
ID	IP address	MAC address	Hostname	Status
0	10.0.2.2	52:54:00:12:35:02		Limited
1	10.0.2.3	52:54:00:12:35:03		Blocked
2	10.0.2.4	52:54:00:12:35:04		Limited

Step 9

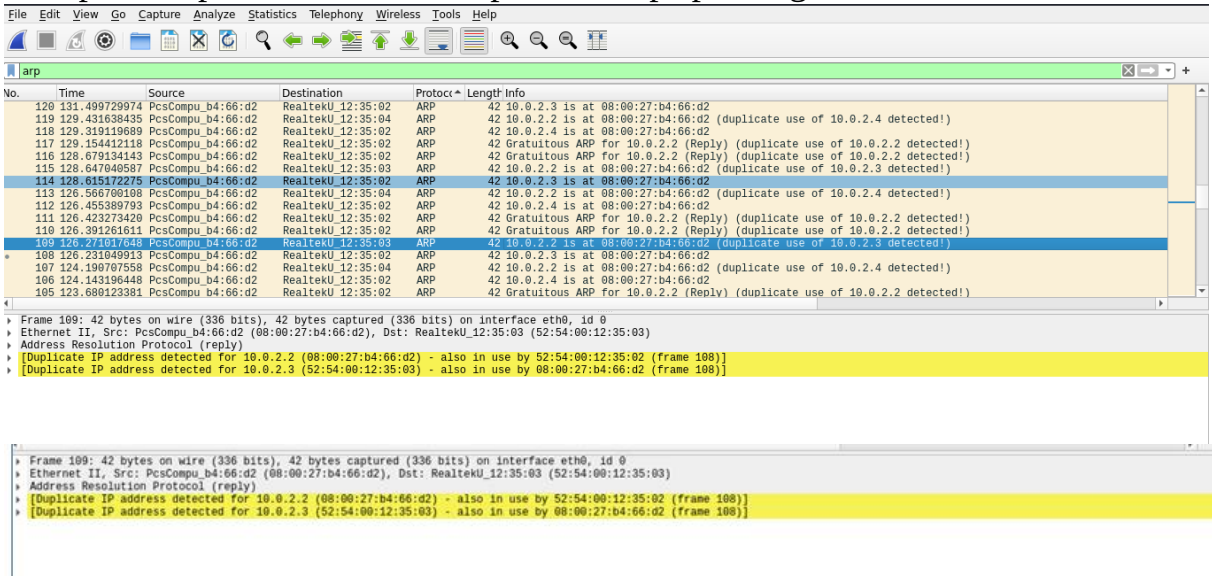
Check wireshark:

Filter : arp

Info: arp spoofing detected



Duplicate ip detected---shows proof of arp spoofing

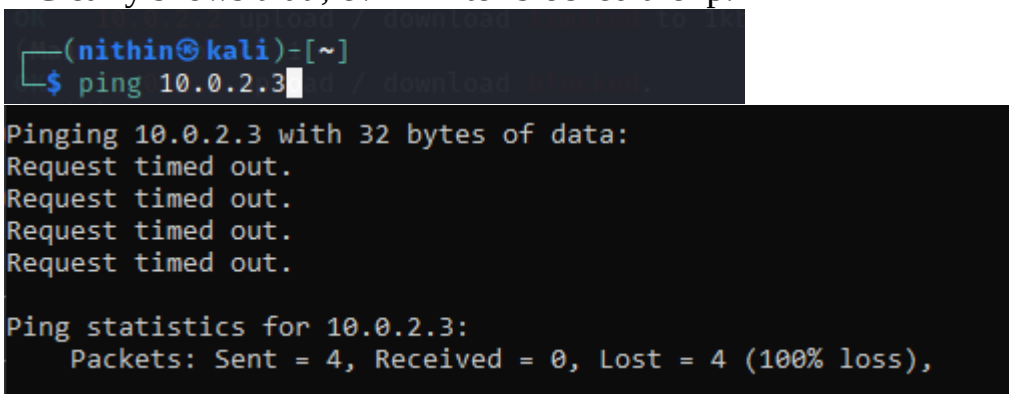


The before picture shows , that the duplicate ip has been founded it is clearly stating the arp poising is performed

Step 10:

Ping ip 1 [blocked ip]

Clearly shows that , evililimiter blocked the ip.



Step 11:

Speed check

As u can see above the speed is in kbps while bandwidth is throttled.

Step 12:

Free all

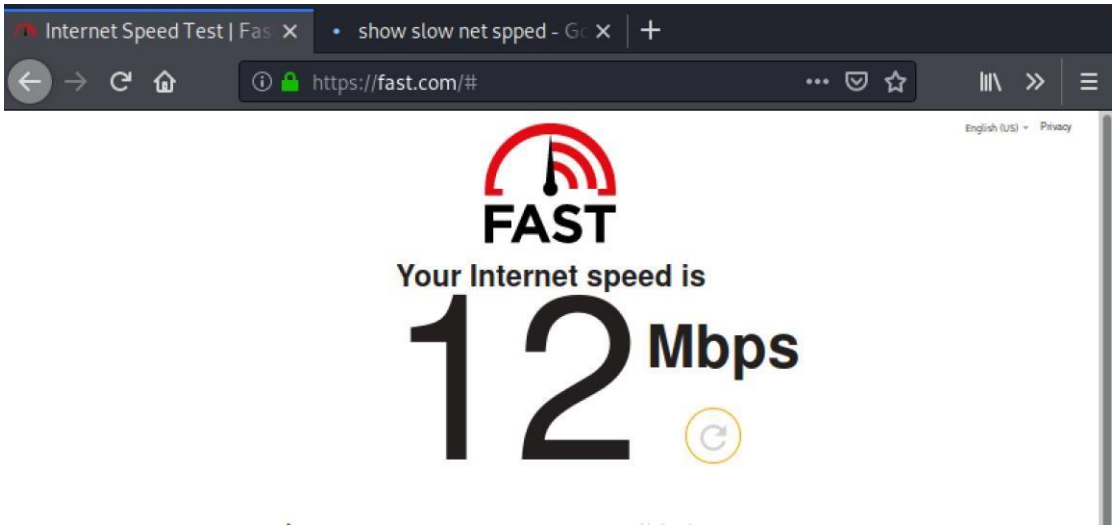
```
(Main) >>> free all
(Main) >>> hosts

Hosts
```

ID	IP address	MAC address	Hostname	Status
0	10.0.2.2	52:54:00:12:35:02		Free
1	10.0.2.3	52:54:00:12:35:03		Free
2	10.0.2.4	52:54:00:12:35:04		Free

Step 13:

Speed check:

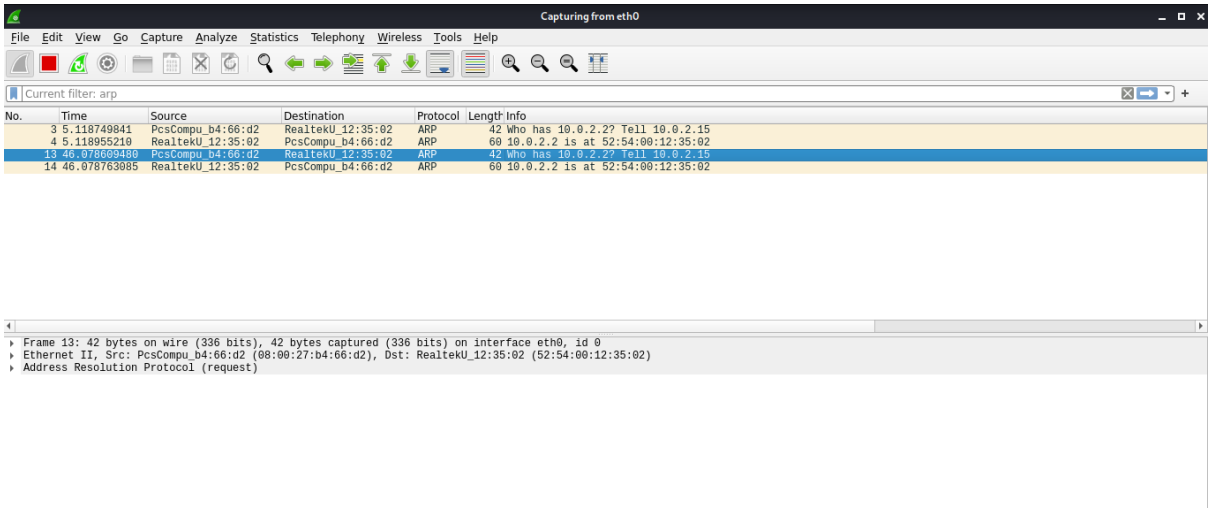
This shows , after freeing the limit speed is in mbps.



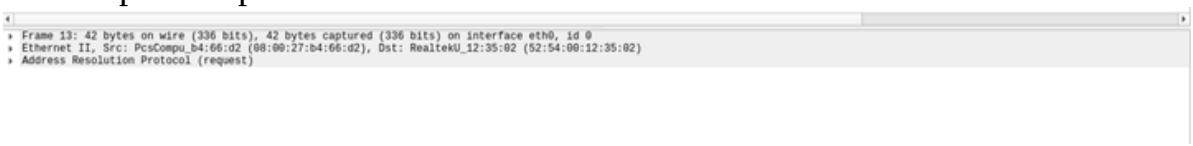
Step 14:

Re-arping in wireshark

After we free all the hosts. The packets that are sent over arp protocol gets re-arped .

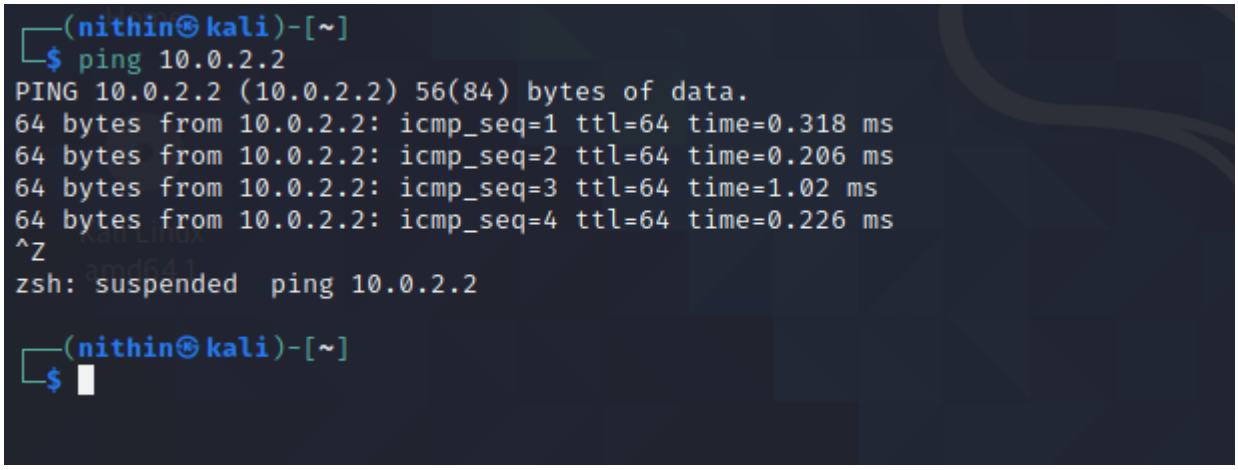


No duplicate ip detected after free hosts:



Step 15:

Now checking the ip address of the free packets wheather the packets is flowing are not



CONCLUSION/FUTURE WORK:

The prevention methods tend to have flaws in certain situations, so even the most secure environment may find itself under attack. If active detection tools are in place as well, then you will know about ARP poisoning as soon as it begins. As long as your network administrator is quick to act once alerted, you can generally shut down these attacks before much damage is done. Hence, it becomes mandatory for everyone to know about technology before using them.

REFERENCES

1. T. Bradley, C. Brown, and A. Malis. Inverse address resolution protocol, Sept. 1998. RFC 2390

2. D. Bruschi, A. Ornaghi, and E. Rosti. S-ARP: A secure address resolution protocol. In Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC '03), Dec. 2003.

3. M. Carnut and J. Gondim. ARP spoofing detection on switched Ethernet networks: A feasibility study. In Proceedings of the 5th Simpósio Segurança em Informática, Nov. 2003

4. isco Systems. Configuring Dynamic ARP Inspection, chapter 39, pages 39:1-39:22. 2006. Catalyst 6500 Series Switch Cisco IOS Sofware Configuration Guide, Release 12.2SX.

5. T. Demuth and A. Leitner. ARP spoofing and poisoning: Traffic tricks. Linux Magazine, 56:26-31, July 2005

6.D. Schuymer. ebtables: Ethernet bridge tables, Mar. 2006. 〈http://ebtables.sourceforge.net〉. (Last accessed April 17, 2006)