A Discussion of Security Testing and Penetration Testing in Related to Cyber-Security: Simulated Man in The Middle Attack on a Private Network

|  |  |
| --- | --- |
| Huynh  Computer Sciences Dept  Purdue University Fort Wayne  Fort Wayne, USA  [huyntl02@pfw.edu](mailto:huyntl02@pfw.edu) | Alswairki  *Computer Sciences Dept*  *Purdue University Fort Wayne*  Fort Wayne, USA  [alswmj01@pfw.edu](mailto:alswmj01@pfw.edu) |

***Abstract*—** This paper introduces security testing principles. It also discusses security testing types, security testing roles, security breaches, and the relationship between the Software Testing Life Cycle (STLC)(\*) and the Software Development Life cycle (SDLC) (\*). The paper also discusses popular hacking tools and demonstrates the process of a “Man-In-The-Middle-Attack” (MITM) (\*) on a private network. Introducing the tools that are used in a Man-In-The-Middle Attack and an overview of Penetration Testing Methods (Pen Testing)(\*) and its related work**.**

***Keywords—*** *Man-In-The-Middle*, *Penetration Testing, Software Testing, STLC, SDLC, Software Quality, Secure software, Secure programming, Secure Breaches, and Media Access Control****.***

# I. INTRODUCTION

## A. Software Security Testing Introduction

Security Testing (or Software Testing Security) determines that software protects data and maintains security specifications as given. Security testing uncovers vulnerabilities of the system and determines that the data and resources of the system are well-protected. It ensures that the software system and its applications are free from any threats (or risks) that can cause a loss. To understand and implement good security test plans, we must understand the software quality, the Software Development Life Cycle (SDLC) Software Testing Life Cycle (STLC), and the software requirements. We also need to understand major issues that cause security breaches.

Security-related bugs can differ from traditional bugs in several ways:

* Security testing is often fundamentally different from traditional testing because it emphasizes what an application should not do rather than what it should do (as pointed out in [9]).
* Malicious attackers do intelligently search for vulnerabilities. If they succeed, they cause problems for other users who may be adversely affected. Compounding the problem, malicious hackers are known to script successful attacks and distribute them [9].
* Since most developers are not currently trained in secure programming practices, security analysts carry a greater burden in verifying that secure programming practices are adhered to.[9]
* Many security requirements can be neither refined nor dropped even if they are untestable. E.g.,” an attacker should never be able to take control of the application,” would be regarded as untestable in a traditional software development setting [9]

As a result, secure software development is intrinsically harder than traditional software development. Therefore, testing has an expanded role. Software testing also has other strengths that can be leveraged during secure software development:

* Testing can help confirm that the developers did not overlook insecure programming practices.
* A vulnerability is usually taken more seriously if there is a known exploit for it (developing exploits is the domain of penetration testing).
* Testing can be used to help identify and mitigate risks from third-party components, where development artifacts like source code and architecture diagrams are unavailable
* Testing can be used to provide metrics of software insecurity and help raise the alarm when software is seriously flawed from a security standpoint.
* Every design artifact views the software system at a certain level of abstraction. Attackers like to find the abstractions used by developers and work their way around them. No person or group can view a software system at all possible levels of abstraction, but testing can help by perhaps finding (at least some) flaws that are not visible in the design artifacts [9]

It is often said that security testing is only a small part of secure programming [10]. It is very difficult to find all security-related problems in a software system. Thus, no effective mitigation strategy should be overlooked.

## B. Unintended Error, Intended Action, and Hacker Motivation

Unintended errors can cause by software errors, software faults, or software failures. Software errors are errors made by the programmer. They are two types of errors: Syntax (grammatical) Errors and Logical Errors (e.g, multiply instead of adding two operands). All software errors may not cause software faults since they may not be executed. A software fault becomes a software failure when it is activated. Faults may be found in the software due to the way the software is executed.

Intended Actions are actions that take advantage of unintended errors (security bugs) to gain access to secure systems (that they are not allowed to access). E.g., hackers take advantage of weak security methods to gain their benefits… Intended actions are punished by federal law (maximum is from 10 to 20 years).

Hacker’s motivations include Achieving financial gains (stealing credit card information, misusing data [apply for loans, cars, credit cards…], selling data on dark webs…), carrying out political agendas, performing corporate espionage, providing a point (hacktivist), taking personal revenge, causing harm for personal enjoyment, mitigating cyber threats…

## C. Software Security Testing Goal, Principles, and Focus:

The goals of Software Security Testing are identifying the threats in the system, measuring the potential vulnerabilities of the system, detecting every possible security risk in the system, and helping developers in fixing the security problems through coding.

Principles of Software Security Testing are confidentiality, integrity, authentication, authorization, availability, and non-repudiation.

The major focuses of Software Security are Network Security, Web Application Testing (Client-side and Server-side), and System Security.

## D. Test Case Designing for Security Testing:

Some test case that designed specifc for software security testing [9][10]:

* Test if users can directly access bookmarked web pages without logging in
* Test if the system restricts users to download the file without logging in
* Test if previously accessed pages are not accessible after logging out (i.e. Sign out and then press the Back button to access the page accessed before)
* Test if the industry standard username & password rules are enforced
* Test if sensitive information (passwords, id, credit cards, etc.) is stored as plain text.

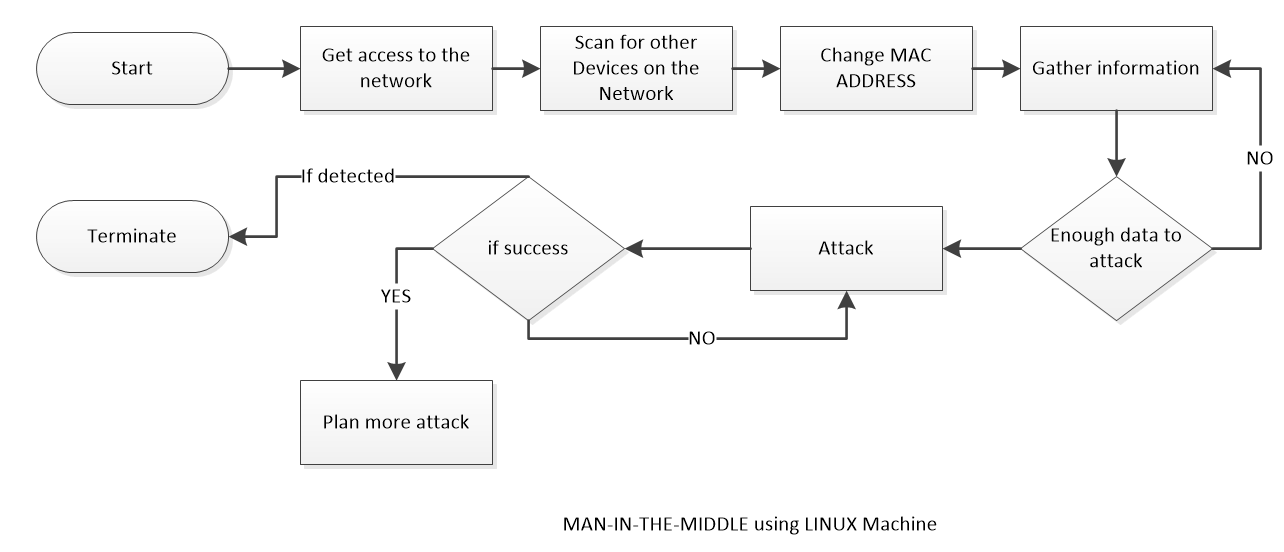


Fig 9. Attack plan diagram

* Test if bookmarking is disabled on secure pages by default
* Test if source code is invisible to users
* Test if older version web browsers can access the app (older version web browsers do not support SSL).
* Test if multiple attempts are being blocked
* Test if the system completely logs out the current user after time out
* Test if the user’s connection is stable and secure
* Verify that relevant information (upload, download, activities) is written to the log files and that information should be traceable
* Test if the SSL encryption is done correctly and verifies the integrity of the information
* Prevent the same username to log in at the same time
* Check if important credentials are updated immediately
* Test if error messages don’t contain important information [8]

# II. RELATED WORK

## Related work on software testing industry

* The North American security testing market is registering a CAGR of 22.6% over the forecast period 2021-2026 as well as it is a highly regulated region globally with numerous regulations and compliances [16].
* There are a lot of companies that specialize in security testing for validating application/software security defenses, and they provide security testing services as well as vulnerability management solutions for their client’s applications and systems.
* The Global Security Testing Market is segmented based on:
  1. Security testing market by Tools:
* Automated testing tools
* Penetration testing tools
* Web testing tools
* Code Review tools
  1. Security Testing Market by Organization Size:
* Large companies
* Small & medium business
  1. Security Testing Market by Type [16]:
* Application Security Testing
* Network Security Testing
* Device Security Testing
* Social Engineering
  1. Security Testing Market by Deployment Model [16]:
* Cloud-Based
* On-Premises
  1. e. Security Testing Market, By Vertical [16]
* Government
* Telecom
* Banking, Financial Services
* Retail
* Healthcare
* Energy And Utilities
* Others
  1. Security Testing Market, By Geography [16]
* North America
* Europe
* Asia Pacific
* Rest of the World

## Type of Security Testing

There are seven types of security testing [6]:

* Vulnerability Scanning: Vulnerability scanning is performed with the help of automated software to scan a system to detect the known vulnerability patterns
* Security Scanning: Security scanning is the identification of network and system weaknesses. Later, it provides solutions for reducing these defects or risks. Security scanning can be carried out in both manual and automated ways
* Penetration Testing: Penetration testing is the simulation of an attack from a malicious hacker. It includes an analysis of a particular system to examine for potential vulnerabilities from a malicious hacker that attempts to hack the system
* Risk Assessment: In risk assessment testing security risks observed in the organization are analyzed. Risks are classified into three categories (low, medium, and high). This testing endorses controls and measures to minimize the risk.
* Security Auditing: Security auditing is an internal inspection of applications and operating systems for security defects. An audit can also be carried out via line-by-line checking of code.
* Ethical Hacking: Ethical hacking is different from malicious hacking. Ethical hacking aims to expose security flaws in the organization’s system.
* Posture Assessment: It combines security scanning, ethical hacking, and risk assessments to provide an overall security posture of an organization.

## C. Pen Testing: Simulate Man-In-The-Middle Attack on a system.

To successfully design the penetration testing plan in MITM Attack on a private network. There is some Pre-requirements knowledge that hackers need to achieve:

* Python Scapy Package: Scapy is a powerful Python-based interactive packet manipulation program and library. It can forge or decode packets of a wide number of protocols, send them on the wire, capture them, store or read them using pcap files, match requests and replies, and much more. It is designed to allow fast packet prototyping by using default values that work.

Text

Description automatically generated

Fig 4. ARP Table Network Scanner by Truc Huynh

* Kali Linux: is an open-source, Debian-based Linux distribution geared toward various information security tasks, such as Penetration Testing, Security Research, Computer Forensics, and Reverse Engineering [11]. Over 600 penetration testing tools are pre-installed [11]
* Cyber Security principles: Applied Cyber Security Principle to find a weakness in the system (bypass filter if using same MAC address), network topology
* Networking Principles: Understand MAC Address, Access Point, Various Networking Devices, Address Resolution Protocol (ARP), Domain Name Server (DNS)

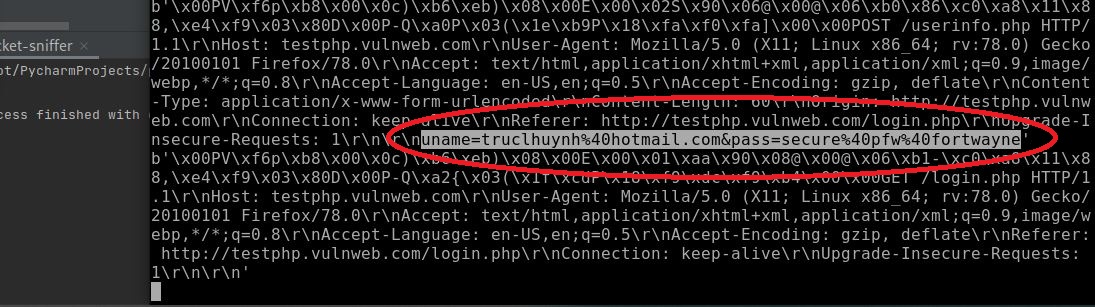


Fig 5. Original data is captured without string extraction by Truc Huynh

* OSI Model and TCP/IP Model: Understand the Layers Architecture of OSI Model and TCP/IP Models, Understand IPV4 and IPV6.
* Web Application Structures: (Client-Server Model), HTML, CSS, PHP (or other serverside scripting languages such as Python, Node JS).
* Python Programming: String Manipulation, Parsing HTML, Sending & receiving HTTP requests, Netfilterqueue, Socket Programming, Data Structures, OOP…
* PHP Web Application Programming.
* Unix/Linux Command Line Usage.

Tools are used in the Man-In-The-Middle Attack:

* Network Scanner: The screenshot of our Network Scanner application. The results from Fig 4 are used along with MAC Address Changer to hack into the victim’s PC. The script can be found in Fig 11.
* MAC Address Changer: Basic script for MAC Address Changer can be found in Fig 12.
* ARP Spoofer (ARP Cache Poisoning): Scripts for ARP Spoofer can be found in fig 13.
* Packet Sniffer: Fig 5 and 6 show the return result of Packet Sniffer. While Fig 5 shows the raw data only, Fig 6 is an enhancement version with HTTP extraction. Return results from Fig 6 can be used for data collection or directly automation attacks. Scripts for Packet Sniffer can be found in Fig 14.

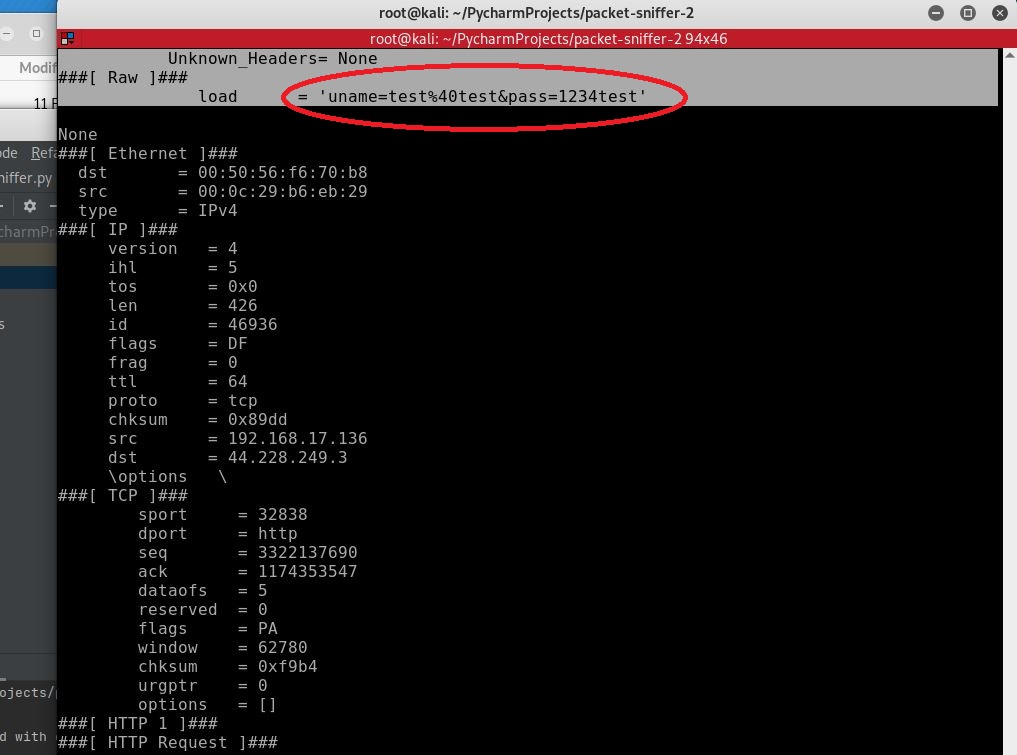


Fig 6. Data is captured using extraction of HTML layer by Truc Huynh

* DNS Spoofer (DNS Cache Poisoning): Scripts for DNS Spoofer can be found in fig 15.
* File Interceptor.
* Machine Learning Model for data collection and extraction.
* Decision-Making Model which is built from the collected data.

# III. SOLUTION

In this use case, I will demonstrate how to implement a MITM attack on a private network (a private system that we don’t have permission to get access to). A private Network can only access by devices within its network. All tools will be written from scratch and the source code can be found at [13]. Please look at the Pre-requirement section if you are not sure about any topics that I mention in the next sections.

The benefit of writing our hacking tools is that we can automate the whole hacking process, implement machine learning & artificial intelligence, and provide efficient source code control. Also, we will not reply in any tools in case an update is occurring.

Private networks usually exist on a physical building with access within that building. However, it now had been extended to mobile technology with performance-critical data transfer. Thus, Cyber Security engineer has more work to do. Private Networks are more secure than public networks, however, they won’t be secure if an adversary gets access to one of the computers in the network. Man-In-The-Middle can be established just by one computer being hacked and spread out to other devices within the network. Depending on the size of the attack can cause millions of dollars in damage.

## A. System under normal operation

Under normal operation, each client is connected to an access point within the organization (inside its building). Please notice that access to the private network only can be granted within the building (wired and wireless). Please refer to Fig 1.

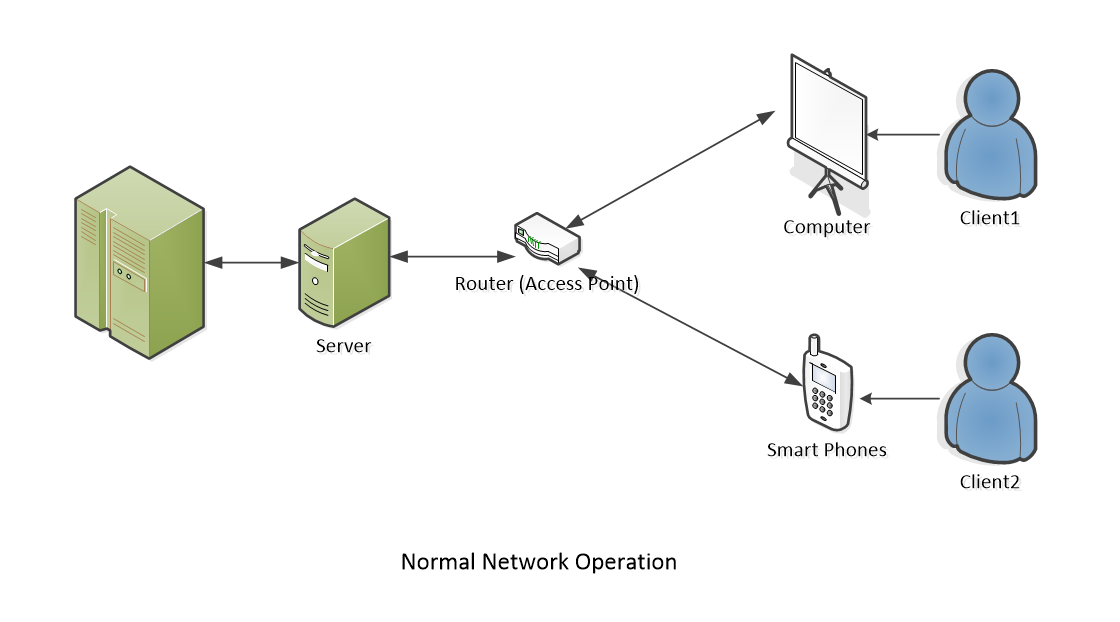


Fig 1. Normal Operation by Truc Huynh

## B. Hackers Gain Access to the system

Access can be gained in many ways insider attack, malware backdoor, code Injector, or malware package. I will not focus on how the hackers gain access to the system. My focus is on simulating the strategy that spread out the virus after gaining access to the system. Please refer to Fig 2.

Hackers can use remote devices that are set up within the building or gain control of one of the devices within the organization to perform the task. They start with one device and then spread the attack to all other devices. Each device that gets accessed by the hacker can become a bot and send out information or spread the virus to other devices within the network. Some viruses can contain themselves, create a backdoor, and pass security scanners by changing their MAC address or IP address. Depending on how many devices hackers want to control, they may need a supercomputer to handle the task or just a USB-size Linux machine.

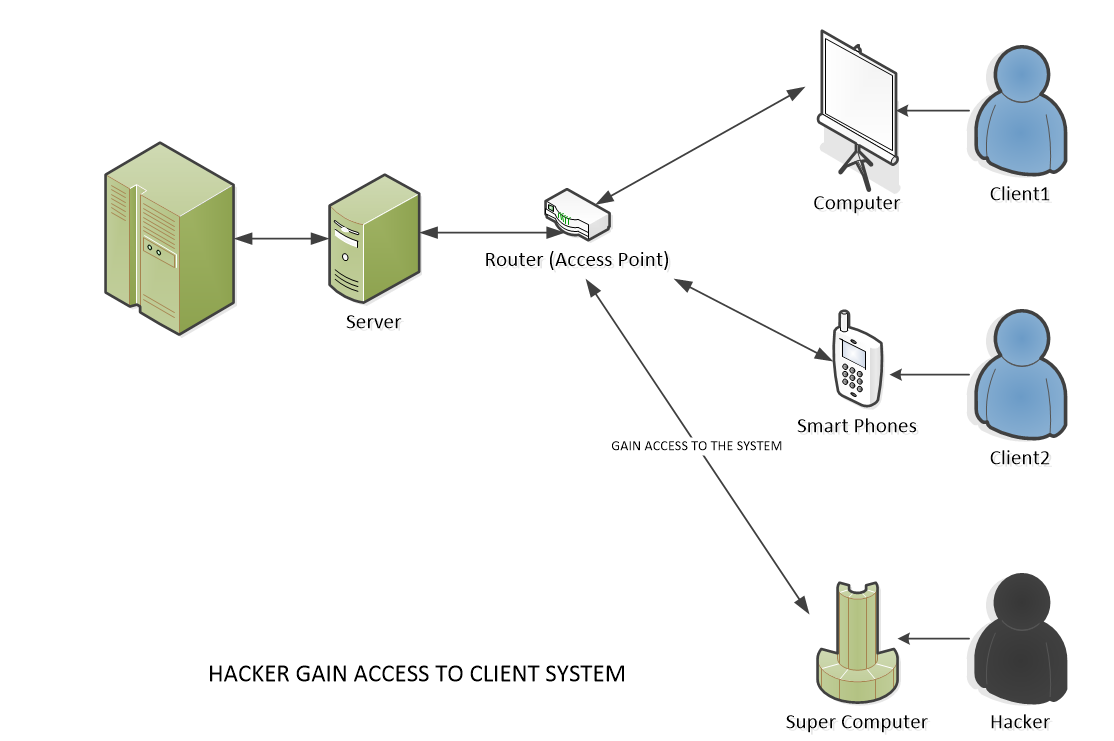


Fig 2. Hackers gain access to the system by Truc Huynh

The plan of attack is using multiple tools that are written in Python. Please note that all the tools are written from scratch using the Python Scapy package, Python Scapy.HTTP, and Kali Linux. All the tools are contained within one script file (less than 200 Kilobyte (KB)) and stored in the Kali Linux (custom USB). I was amazed how lightweight the virus was with much power and technology embedded: Machine Learning, Automation, packet modification, Socket Programming, Netfilterqueue, String Manipulation… The required libraries (Pandas, OS, Scapy…) are pre-installed and ready to launch on the Kali Linux USB. The total size of the USB is less than 2.5 GB. I only need a Linux kernel, python 3, and some Python libraries to perform the attack. The plan is designed by Truc Huynh, with the idea from [14]:

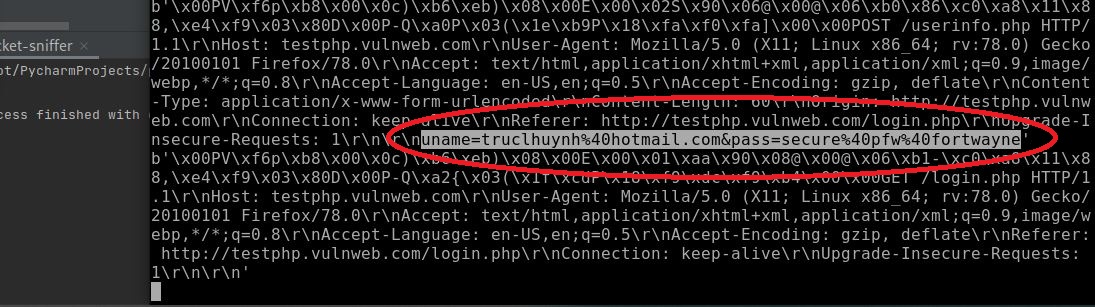


Fig 5. Original data is captured without string extraction

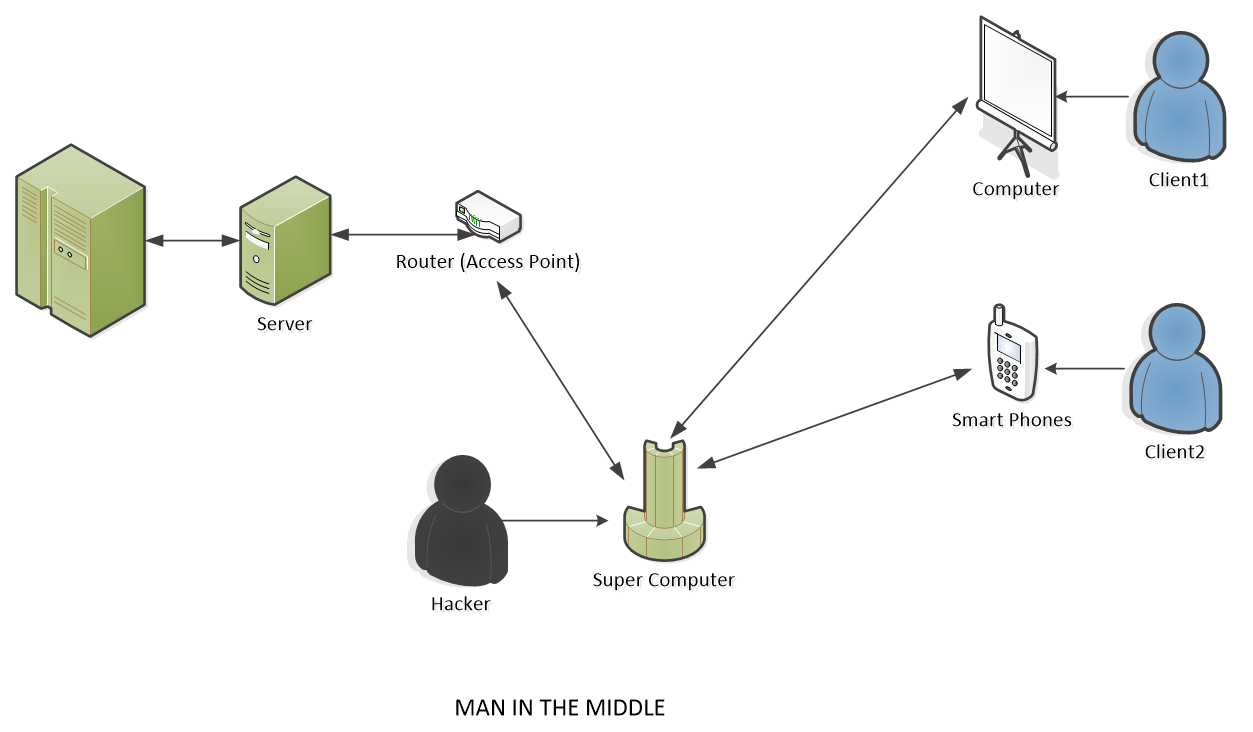


Fig 3. Man-In-The-Middle is established by Truc Huynh

* 1. Step 1:

Hackers get access to one computer. Hackers can get access to a local computer within a private network through a USB stick equipped with a custom Linux version. Hackers will have to physically plug that USB into one machine. Another way to gain access is that they can enable a backdoor on the user’s computer (by trapping users installing our MITM software on their machine).

While this is the hardest task to complete, let's just assume that we have already completed it. The IT Team will enforce policies such as ‘no USBs are allowed to plugin or restriction on employee’s activity (no social media access)’. However, access can still be gained with an internal attack (someone who works there or know the system well) [14].

* 1. Step 2:

Established Man-In-The-Middle by redirecting the flow of the packet by running ‘ARP Spoofer’. The ‘ARP Spoofer’ will run ‘Network Scanner’ to find all the IP and MAC addresses on the network. Our virus then stores the finding result and runs ‘MAC Address Changer to change our MAC address on the hacking devices (USB stick or remote computer) to any MAC address of any local computer (within the private network). Depending on the role and security of the devices on the network. May take extra time to wait for the right opportunities to establish Man-In-The-Middle (security patches, updates…)[14].

* 1. Step 3:

Gather information by using ‘Packet Sniffer’. ‘Packet Sniffer’ is used to read the packet and data flow through the hacker interface. Use the information that ‘Packet Sniffer’ collects to create a suitable plan for spreading the virus to another machine within the network. Please note that the data collected from the victim’s computer can be used to create an attack automatically.

For example, a user who uses their work computer to access iTunes can be traped to download a backdoor on their PC. The hacker can make the installed application fails and motivate the victim to reinstall the app (simply delete some scripts from the iTunes sourcecode). Once the users attempt to reinstall iTunes, the hackers can easily route them to a fake iTunes and install the virus into their system.

In most scenarios, data gathering is the hardest task. After data is collected on the victim machine, the hacker will send the data to themselves. Another approach is that the virus will generate the attack plan by itself (which is the ideal design) [14]. In a large organization where network scanners and firewall applications are enforced, hackers may not want to make a lot of data transfers to prevent detection.

Building a machine learning model or deploying a machine learning model to the virus is the best solution. However, this required extended knowledge of Python Programing, Network Architecture, and a large user behavior database. I will continue working on philosophy when I have a chance.

* 1. Step 4:

Modifying data and spreading the viruses. Using the plan that creates in step 3 to attack other computers. Depending on the security structure of the network using ‘DNS Spoofer’ or ‘File Interceptor’ (or both). Using ‘File Interceptor’ to modify HTTP data sent over HTTP replaces a user’s download request with a completely different file (virus, backdoor…). Using ‘DNS Spoofer’ (modify data in DNS Layer) to redirect the destination on the computer on the network (e.g. to a fake website) so that the hacker can install a backdoor on another local computer. Then the virus will slowly spread and contain itself.

def network\_scan(ip):  
 arp\_request = scapy.ARP(pdst=ip)

# use custom Ether to capture MAC of the broadcast

broadcast = scapy.Ether(dst='ff:ff:ff:ff:ff:ff')  
 arp\_request\_broadcast = broadcast / arp\_request

# srp return 2 lists answered and unanswered list

answered\_list = scapy.srp(arp\_request\_broadcast, timeout=1, verbose=False)[0]  
   
client\_list = []

for answer in answered\_list:  
 client\_list.append({"ip": answer[1].psrc, "mac": answer[1].hwsrc})

return client\_list

Fig 12. Network Scanner Script

#!/usr/bin/env python3

import scapy.all as scapy

import time

# get\_mac ip take the ip address and convert it to answered list

def get\_mac(ip):

arp\_request = scapy.ARP(pdst=ip)

# use custom Ether to capture MAC of the broadcast

broadcast = scapy.Ether(dst='ff:ff:ff:ff:ff:ff')

arp\_request\_broadcast = broadcast / arp\_request

# srp return 2 list answered and unanswered list

answered\_list = scapy.srp(arp\_request\_broadcast, timeout=1, verbose=False)[0]

return answered\_list[0][1].hwsrc

def spoof(target\_ip\_sp, spoof\_ip):  
 target\_mac = get\_mac(target\_ip\_sp)  
 packet = scapy.ARP(op=2, pdst=target\_ip\_sp,   
 hwdst=target\_mac,

psrc=spoof\_ip)  
 scapy.send(packet, verbose=False)

Fig 13. ARP Spoofer Script

The ideal is that they can avoid detection by the network administrator, or any security system on the network[14]. In fact, I have not accomplished this state on my applications.

* 1. Step 5:

The virus decides if the attack is a success or not. Make sure the attack doesn’t create any evidence that leads to the hacker (protocol tracing, IP Address tracing).

# IV. TOOLS OF THE MAN-IN-THE-MIDDLE ATTACK

## Developed Environment

To test the MITM attack, I need a development environment. My environment is a private network that contains a couple of computers. All computers are virtual machines. Required software: VM Ware, Kali Linux, Microsft Windows. Required hardware: network interfaces, a USB, and a host computer (minimum requirements are 64 bits, Intel Core i7 10th Gen, 36 GB of ram).

All the codes below are not fully functional by themselves. Due to the limited of pages, I did not include the whole script. Supported functions are needed to run the defined scripts. Please refer to [13] for all functional scripts.

* USB: Kali Linux must be installed on the USB along all with the required packages.
* A remote computer for the hacker to communicate with the Linux USB

Tools used in the MITM attack are executed in Fig 10. Please refer to Fig 10 for a better understanding of execution structures.

* MAC Address Changer: Basic script for MAC Address Changer in Fig 11
* Network Scanner: The screenshot of our Network Scanner application. The results from Fig 4 are used along with MAC Address Changer to hack into the victim’s PC. The script can be found in Fig 12.
* ARP Spoofer (ARP Cache Poisoning): Scripts for ARP Spoofer can be found in Fig 13
* Packet Sniffer: Fig 5 and 6 show the return result of Packet Sniffer. While Fig 5 shows the raw data only, Fig 6 is an enhancement version with HTTP extraction. Return results from Fig 6 can be used for data collection or directly automation attacks. Scripts for Packet Sniffer can be found in Fig 14

def mac\_changer(interface, new\_mac):  
 print("[+] Changing MAC Address for "  
 + interface + " to " + new\_mac)  
   
 # disable the MAC address  
 subprocess.call(["ifconfig", interface, "down"])   
   
 subprocess.call(["ifconfig", interface, "hw",   
 "ether", new\_mac])  
 # enable the mac Address  
 subprocess.call(["ifconfig", interface, "up"])

Fig 11. Mac Address Changer Script

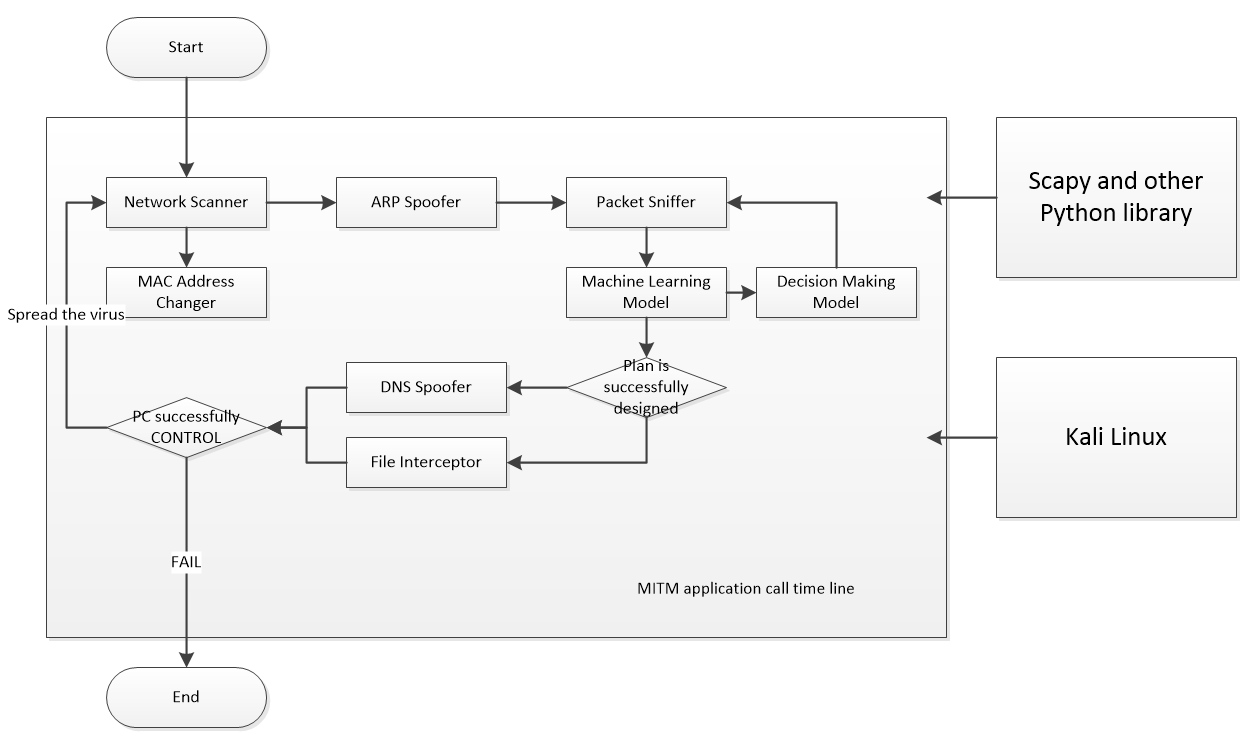


Fig 10. App execution diagram

* DNS Spoofer (DNS Cache Poisoning) script can be found in Fig 15
* File Interceptor
* Machine Learning Model for data collection and extraction
* Decision-Making Model which is built from the collected data

## MAC Address Changer

Media Access Control (or MAC) address is a permanent physical and unique address. MAC address is assigned to a network interface by the device manufacturer. Whether the target has a wireless card or an Ethernet card, each of these network cards comes with a unique address to this card. There are no two devices within the victim’s network that would have the same MAC address. The MAC address will always be the same for this device even if someone unplugs it from one computer and connects it to another.

While the MAC address is used within the network to identify devices and transfer data between devices, the IP address is used on the Internet to identify computers and communicate between devices on the Internet. Therefore, each piece of data (or packet) that is sent within the network contains a source MAC and a destination MAC. This packet would flow from the source MAC to the destination MAC because this is how it identifies devices within its network.

def process\_packet(packet):  
 scapy\_packet = scapy.IP(packet.get\_payload())  
 if scapy\_packet.haslayer(scapy.DNSR):  
 print(scapy\_packet.show())  
 packet.accept()  
  
  
queue = netfilterqueue.NetfilterQueue()  
# call back function used in every sing queue  
queue.bind(0, process\_packet)  
queue.run()

Fig 15. DBS Spoofer Script

def sniff(interface):  
 scapy.sniff(iface=interface, store=False, prn=process\_sniffed\_packet)  
  
def get\_url(packet):  
 return packet[http.HTTPRequest].Host + packet[http.HTTPRequest].Path

Fig 14. Packet Sniffer Script

MAC address is often used by filters (firewalls) to prevent or allow devices to connect to networks. Connected devices can do specific tasks on that network. Changing the hacker’s MAC address to another device's MAC address will allow hackers to impersonate this device and do things they are not allowed to do. Also, they'll be able to bypass filters or connect to networks that only specific devices with specific MAC addresses can connect to or hide their identity.

A basic MAC Address Changer is created by using Python. Depending on the hacking tools and interface that the hacker is going to use, the required packages are different. Since I am using Kali Linux and Python, the required Python packages are” subprocess”, ”optparse”, and “re”. Full Source code and test can be retrieved at [13].

## Network Scanner

Information gathering is one important step when it comes to Pen testing. Hackers can't gain access to a system if they don't have enough information about it. For my attack, the plan is to hack into a private network and control one or more devices on this network. The first step is discovering all the connected clients to this network. Then I get their MAC and IP addresses. Finally, we try to gather more information and run some attacks to gain access to the victim.

Several programs (Net Discover, Nmap) can handle this task. Net Discover and Nmap are scanners that come with Kali Linux. They can show us all the connected devices within the same network (plus their IP addresses and their Mac address). However, I decided to write my network scanner to automate my attack later. My Network Scanner is very similar to Net Discover. Fig 4 shows the result after running the Network Scanner script (Fig 12).

Using ARP to discover all the connected clients on the same networks. As we receive the returned data (IP addresses and Mac addresses) from the scan, we can use this to gather information on the next attack [14]. Measuring the security level of each device gives us a better success rate.

A basic Network Scanner can be created by Python and Linux commands. Depending on the hacking tools and interface that the hacker is going to use, the required packages are different. Since I am using Kali Linux and Python, the required Python packages are ”scapy”, and “argparse”.

## ARP Spoofer (ARP Cache Poisoning)

ARP can be used to discover all the connected clients on the same network. An ARP Spoofing program allows us to redirect the flow of the packets. Instead of them flowing through the internet, they would flow through the hacker’s computer first. Any requests sent and any responses received by the target computer will have to flow through the hacker’s computer. This means any messages, websites, images, usernames, and passwords entered by the victim will have to flow through the hacker’s computer. This allows hackers to read this information, modify it, or drop it. Therefore, ARP Spoofing is a very serious and very powerful attack. The reason why the attack is possible is that ARP is not very secure. Therefore, we need to understand network architecture as well as the OSI model to design this type of attack.

To understand how this works we also need to have a basic understanding of ARP functionality. In networking, ARP is used to help network clients to identify other connected clients on the same network. ARP gets other clients’ Mac addresses that connect on the same network using the Mac address table (or ARP table). Each computer has an ARP table that links IP addresses on the same network to their Mac addresses [15].

The value in the ARP table can be easily modified by exploiting the ARP protocol. Normally when a device connects to the network and wants to send requests, it will send them to the router. Then, the router will go and send that request to the Internet, wait for the response, and then forward the response to the device that requested it [15].

If the hacker’s machine or the victim’s machine or any other computer on the network wants to send a request, they will send that request directly to the router. To interfere with the connections, we can exploit the ARP protocol and send two ARP responses: “one to the gateway and one to the victim”. I’m gonna tell the gateway that I am at the IP of the victim. Then, the gateway (access point) will update its ARP table. Moreover, it will associate the IP of the victim with my Mac address (I must change the hacking computer’s MAC address to the victim's MAC address). I'll do the same with the victim: I'll send it an ARP response. I’m going to tell the victim that I am at the router IP. Therefore, it's going to update its ARP table and associate the IP of the router with my own Mac address. As a result of this, the victim is gonna think that I am the router, and the router is gonna think that I am the victim [15].

Anytime the victim wants to send any requests the requests will have to flow through my computer. I'm gonna forward them to the router. Anytime the access point or the router wants to send responses, they're going to go to my machine because it thinks I am the victim. Then I'm going to forward it to the victim. This puts me in the middle of the connection and gives me so much power once I become the man in the middle

The main reason why I can do all of this is that ARP is not secure. First, clients can accept responses even if they did not send the request.

I am gonna send the response to the access point and the response to the victim telling them that I am at a specific IP. They will accept that response anyway without asking who I am or asking where this IP’s original destination is. Furthermore, they're also not going to verify who I am.

When I say that I am at the victim IP, I am not at that IP. because this computer is at this IP, the access point will trust, and it will update its ARP table based on the information that I sent. The same goes for the victim. Thus, these are the two main weaknesses of ARP protocol that allow us to run ARP spoofing attacks

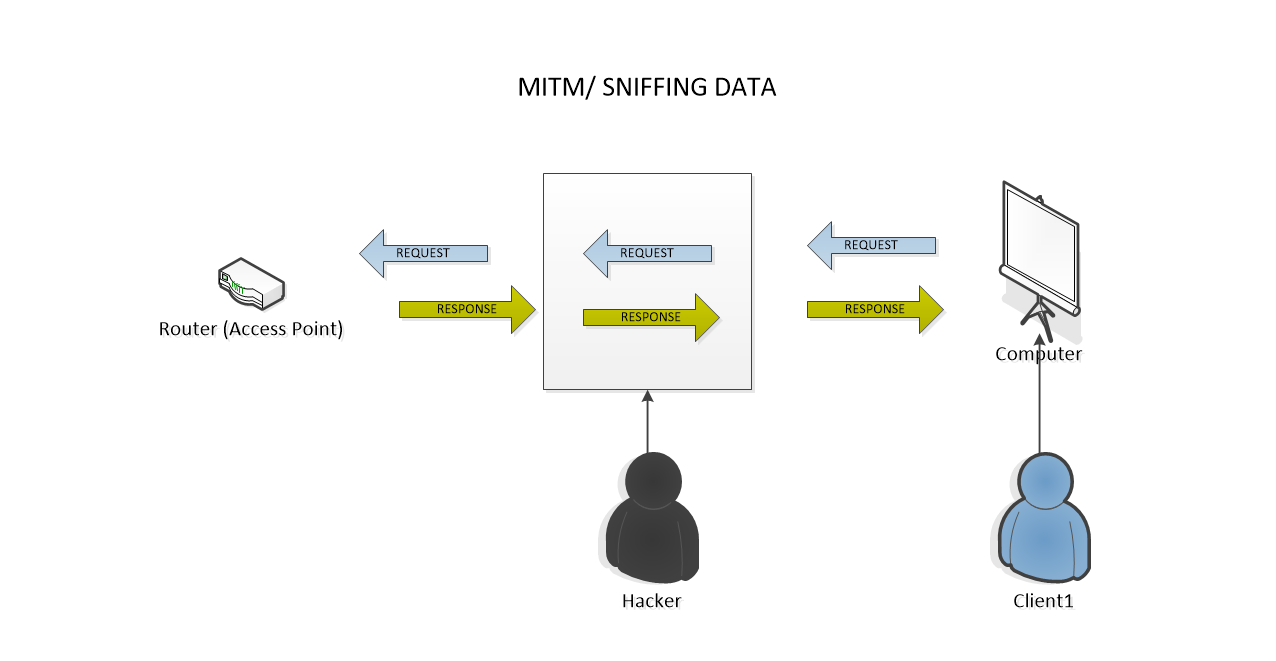


Fig 7. Packet Sniffing Diagram by Truc Huynh

The required Python packages for this application are” scapy”, and “time”. The source code to run this program can be found at [13]. Since “scapy. sniff” does all the hard work for us, we do not need to create anything new here.

## Packet Sniffer

ARP Spoofer can put me in the middle of the connection. However, hackers can’t read the information flowing through their computers automatically. Since data is transferred in packets and unreadable, we'll need a packet sniffer to read this information. The Packet Sniffer is a program that reads packets or data that flow through an interface (Fig 7).

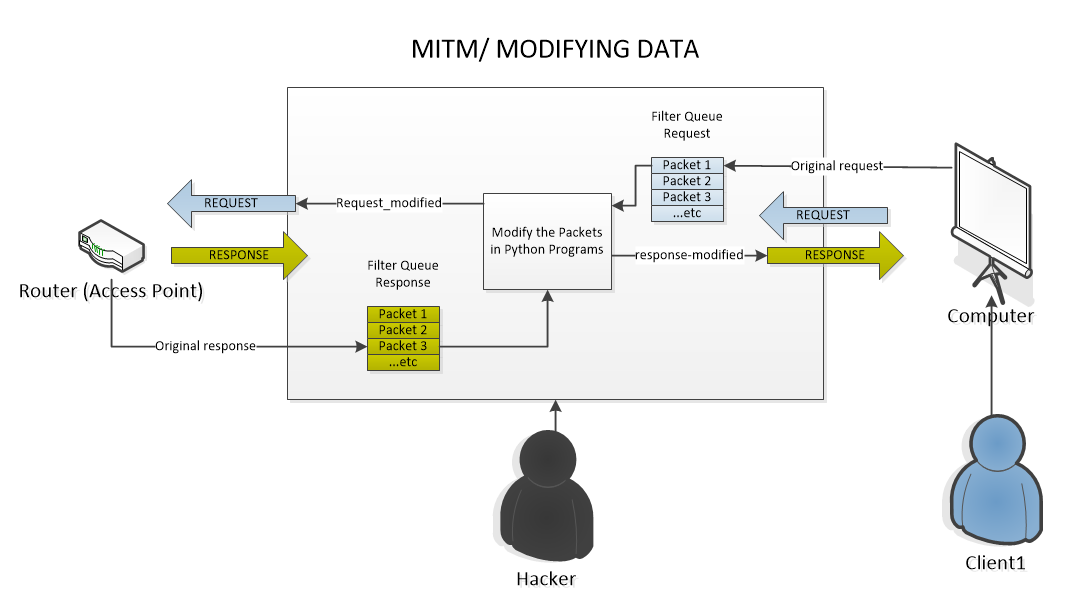


Fig 8. Packet Modifying (DNS Spoofer)

Once we use ARP Spoofer to become the man in the middle, the data will flow through our interface by default. we'll be able to read all the information that flows through my computer that any target computer sends or receives. The gathered information can also help the virus to understand the user behavior as such more complicated attacks can be created on the go.

The data that hackers collected from the victim won’t be able to be read immediately. Please refer to Fig 5 for the sample of data that I extract from a hacking computer. Further manipulation needs to occur here to transfer them into meaningful data (username, passwords, app use, website visit times…). In fig 6, I use the Python string manipulation, regular expression in combination with ‘scapy.HTTP’ (HTTP layer extraction) to extract exact the username and password that the victim enters through a real website that he/she often visits.

In fig 5, hackers can extract the information that the user enters, but it will create a lot of work and data communication between the hackers and their remote computers. This may lead to viruses will be discovered by Network Scanner or other Firewall apps. In fig 6, the extracted data is exactly what the virus needs to enable the MITM attack. Therefore, an automatic attack can happen without sending any data to the hackers.

Technology:

The required Python packages are ”scapy. all”, and “scapy. layers”. The source code to run this program can be found at [13].

## DNS Spoofer (DNS Cache Poisoning)

DNS Spoofer redirects any request made by the victim in the hacker’s favor. For example, we want to redirect requests made by our victim to some other place (a fake website, download a backdoor…). we can accomplish that task using Python scapy.

The first approach is that create a modified packet and send it to the destination. The main problem with this implementation is that Scapy cannot intercept or drop packets. When we are receiving a request, we also create a modified version of that request. Then, we will send both requests to the target machine. The result is that the target machine is going to receive two requests: the original one and the modified one. The target machine will decide which one it'll execute. Since computers execute the request that they receive first and modifying a request might take some time, the target will never execute the modified request.

A better implementation is creating a queue in the hacker machine and trapping packets inside that queue. Whenever we get a request, we'll put it in my Queue (Request Queue) and never send it to its target. Then I'll access my Queue from my Python Program and modify the packets as I want. After I finish modifying the packet, I will send the request. As a result, the target will only receive one packet which is the modified packet (modified request) (Fig 8).

The same way can be used to modify responses. Whenever I trap responses in my Queue, I access my Queue from my Python program. After modifying the packet, my program (virus) will forward them to the destination (Fig 8).

Python ‘filterqueue’ and ‘scapy’ can be used to complete this task.

# 

# IV. SECURITY PRACTICES

Security practices implement in SDLC to prevent security breaches:

* Build security into Software Engineering Process before the software design. Implement security testing on every step of SDLC. E.g., Security Development Microsoft Lifecycle (SDL)
* Applied all testing methods to make sure we are secure at every stage of the development life cycle.
* Working with the developer team to implement security programming principles that are industry standards.
* Develop security methods to prevent insider attacks and outsider attacks (AWS Share Model, AWS Elastic Beanstalk).

# V. FUTURE WORK & CONCLUSION

## Future Work

Building a machine learning model or deploying a machine learning model into the virus is ideal. However, I haven’t had enough time and resources to test this philosophy. I will continue working on this when I have a chance.

A well-trained machine learning model should be able to self deploy the attack plan (DNS Spoofer or File Interceptor) based on the user’s behavior. Moreover, my ambition is that my model can further navigate the structure or policies, or patterns of the victim network’s security. This needs extensive knowledge of Network Architecture, OSI Models, and HTTP Models. Unfortunately, I will not be able to handle this at this time.

Further study and research will be conducted when I have a chance. Software Security, Network Security, and Security Testing are disciplines that I am passioned and want to learn more about soon.

## Conclusion

To successfully plan a Man-In-The-Middle attack we need both Programming Knowledge and Cyber Security knowledge. We also need a tester mind and be able to find the weakness of the system. Sometimes the weakness is not only a software fault or technical issues but also the user’s behavior. Understanding and controlling users’ activities and behaviors go beyond software security testing principles and belong to cyber-security principles.

I want to mention that even though the software works perfectly fine to prevent threads and protect the data. Users who wholly use the system for their purpose and do not follow the organization’s policies may end up harming the system badly. Therefore, training must be conducted to prevent these incidents.

Some possible regulations such as employees being required to log in with identify cards (ID cards), and their unique usernames and passwords. The cards are also being scanned daily at the entrance to make sure the right person is identified. Along with that physical security policies, the software policies are also enforced by the IT team. Still, incidents happen daily due to the lack of self-discipline in human behavior.

The more I read about security testing methods, the more I realize what steps should be taken to enhance cyber security at workplaces is that we need both constantly update software testing and staff training. When I was working for DOD, we conduct meeting every two weeks or monthly to ensure our policies is up to date and our metrics are still in good standing. Necessary adjustments will be made if bad measures come back. Of course, the bigger the size of one organization the harder it is to make the change. Sometimes it takes months to completely changes. Therefore, implementing security practices in SDLC to prevent security breaches is the ideal solution.

# VI. KEYWORDS

* **Software Testing:** security testing uncovers vulnerabilities of the system and determines that the data and resources of the system are well protected
* **Software Testing Life Cycle (STLC):** Software Testing Life Cycle (STLC) is a sequence of specific activities conducted during the testing process to ensure software quality goals are met. STLC involves both verification and validation activities. There are 6 steps of STLC: Requirement Analysis, Test Planning, Test Cases Designing, Test Environment Setup, Test Execution, Test Closure.
* **Software Quality:** Computer programs, procedures, and possibly associated documentation and data about the operations of the system (IEEE). Four components that need to assure the quality of the software development: code, procedures, documentation, data to operate the software system (from [2])
* **Secure Software Testing:** testing methods that determine software products protect data and maintain security specifications as given.
* **Secure Programming Practice:** is the practice of developing computer software in such a way that guards against the accidental introduction of security vulnerabilities
* **Security Breaches:** A security breach happens when a system fails in its security processes and causes unauthorized party access to the system. A security breach can happen by software problems (error, fault, failures), outdated security technology, ransomware, malware …

# REFERENCES

1. IEEE: Advanced Technology for Humanity, https://www.ieee.org/
2. ISO 9000-3: Quality management and quality assurance standards, https://www.iso.org/standard/26364.html
3. SWEBOK V3.0: Guide to the Software Engineering Body of Knowledge, https://ieeecs-media.computer.org/media/education/swebok/swebok-v3.pdf
4. Tutorial Point: STLC Tutorial, https://www.tutorialspoint.com/stlc/index.htm
5. Software Testing | Security Testing, https://www.geeksforgeeks.org/software-testing-security-testing/?ref=lbp
6. Security Testing: Types, Tools, and Best Practices, https://brightsec.com/blog/security-testing/
7. API Security: The Complete Guide, https://brightsec.com/blog/api-security/
8. STLC (Software Testing Life Cycle) Phases, Entry, Exit Criteria (guru99.com), https://www.guru99.com/software-testing-life-cycle.html
9. Fink, G. & Bishop, M. “Property-Based Testing: A New Approach to Testing for Assurance.” ACM SIGSOFT Software Engineering Notes 22, 4 (July 1997): 74-80
10. McGraw, Gary & Potter, Bruce. “Software Security Testing.” IEEE Security and Privacy 2, 5 (Sept.-Oct. 2004): 81-85.
11. McGraw, Gary. “Application Security Testing Tools: Worth the Money?” Network Magazine, November 1, 2004. (2004).
12. Kali Linux, https://www.kali.org/?msclkid=ccbd3c3faa2511ecbe541363c15a4582
13. Python Scapy Package, https://pypi.org/project/scapy/?msclkid=33343ba2aa2611eca8b9c3abfd8b35c1
14. Truc Huynh: Ethical Hacking Using Python, https://github.com/jackyhuynh/ethical-hacking-using-python
15. Udemy: Learn Python and Ethical Hacking from Scratch, <https://www.udemy.com/join/login-popup/?next=/course/learn-python-and-ethical-hacking-from-scratch/learn/lecture/10800892#overview>
16. <https://www.verifiedmarketresearch.com/product/global-security-testing-market-size-and-forecast-to-2025/>