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 and discusses types of security testing, why we need security testing, the root cause of security breaches, and the relationship between Software Testing Life Cycle (STLC), Software Development Life cycle (SDLC), and security testing. in this paper, we are going to use the case of simulating a Man-In-The-Middle Attack on a system, and the tools that can be used in a Man-In-The-Middle Attack (Python, Kali Linux, Web Application Security Testing) and give readers an overview of Penetration Testing Methods (Pen Testing) and its related work**.**

***Keywords—****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. Another word to say: “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 application are free from any threats or risks that can cause a loss. To understand and implement good security test plans we must understand software quality, software development framework (SDLC) software testing life cycle (STLC), and software requirements and 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]

The result is that secure software development is intrinsically harder than traditional software development. Therefore, testing also 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 some insecure programming practices
* A vulnerability is usually taken more seriously if there is a known exploit for it, but 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 (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 / if it is activated. Faults may be found in the software due to the way the software is executed. Other constraints on the software‘s execution such as execution options.

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). Intended Actions include: Hackers take advantage of weak security to gain their benefits; hackers create opportunities so that they can hack into the applications or systems… Intended actions are punished by federal law (10 to 20 years).

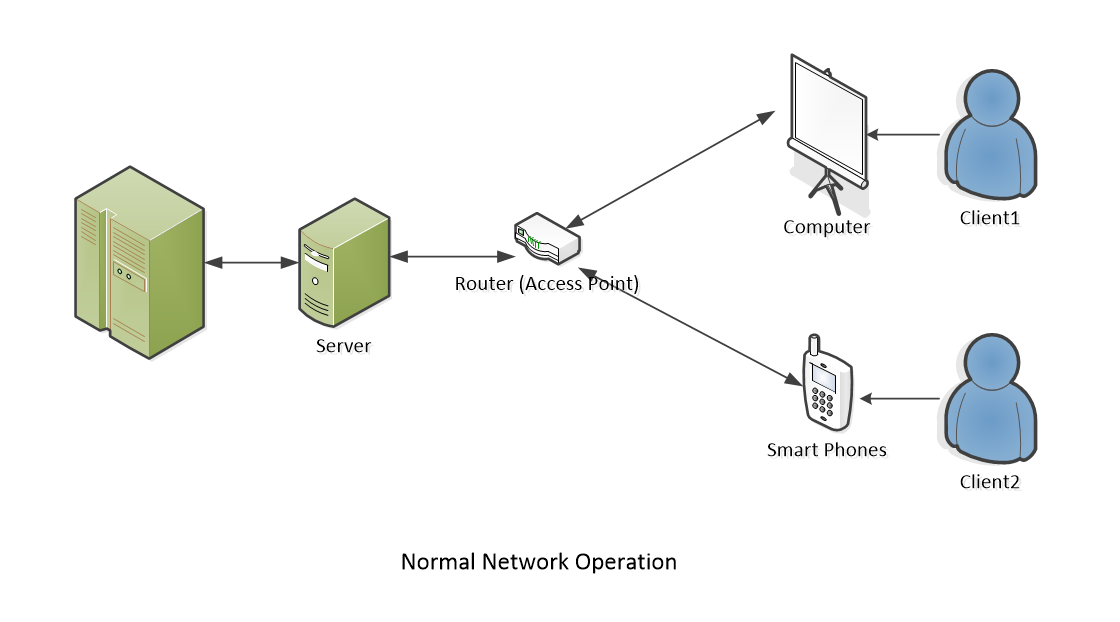
Hacker’s motivations include: Achieving financial gains (stealing credit card information, misusing data [apply for loans, cars, credit cards…], selling data on dark webs…), carrying out political agendas, performing corporate espionage, providing a point (hacktivist), taking personal revenge, causing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

Fig 1. Normal Operation by Truc Huynh

## D. Test Case Designing for Security Testing:

* 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 should not be 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 numbers, credit card numbers, etc.) is stored as plain text. They should be encrypted and in Asterix format.
* 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) are 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

# II. RELATED WORK

## A. Type of Security Testing

There are seven types of security testing:

* 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

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

To successfully launch the attack. 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

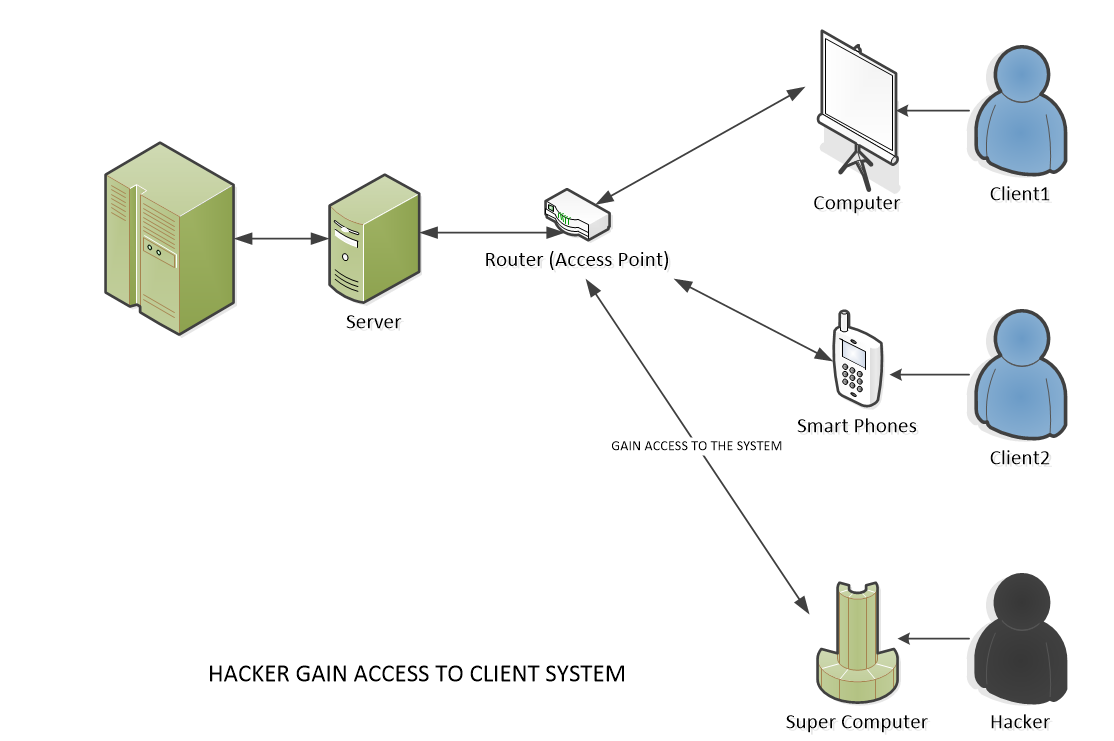


Fig 2. Hackers gain access to the system by Truc Huynh

* 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 were 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)
* 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.

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

* MAC Address Changer
* Network Scanner
* ARP Spoofer (ARP Cache Poisoning)
* Packet Sniffer
* DNS Spoofer (DNS Cache Poisoning)
* File Interceptor

# III. PROBLEM

In this use case, I will demonstrate how to implement a man-in-the-middle-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 the topics that I mention in the next sections. The benefit of writing our program is that hackers can automate the whole hacking process, implement machine learning & artificial intelligence, and provide efficient source code control

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. 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 many other devices within the network. Depending on the size of the attack, some can cause millions of dollars lost

## A. System under normal operation

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

## B. Hackers Gain Access to the system

Access can be gained in many ways insider attack, malware backdoor, code Injector, malware package… I will not focus on how the hackers gain access to the system. However, my focus is to simulate the strategy that hackers to spread out the virus after gaining access to and controlling the system

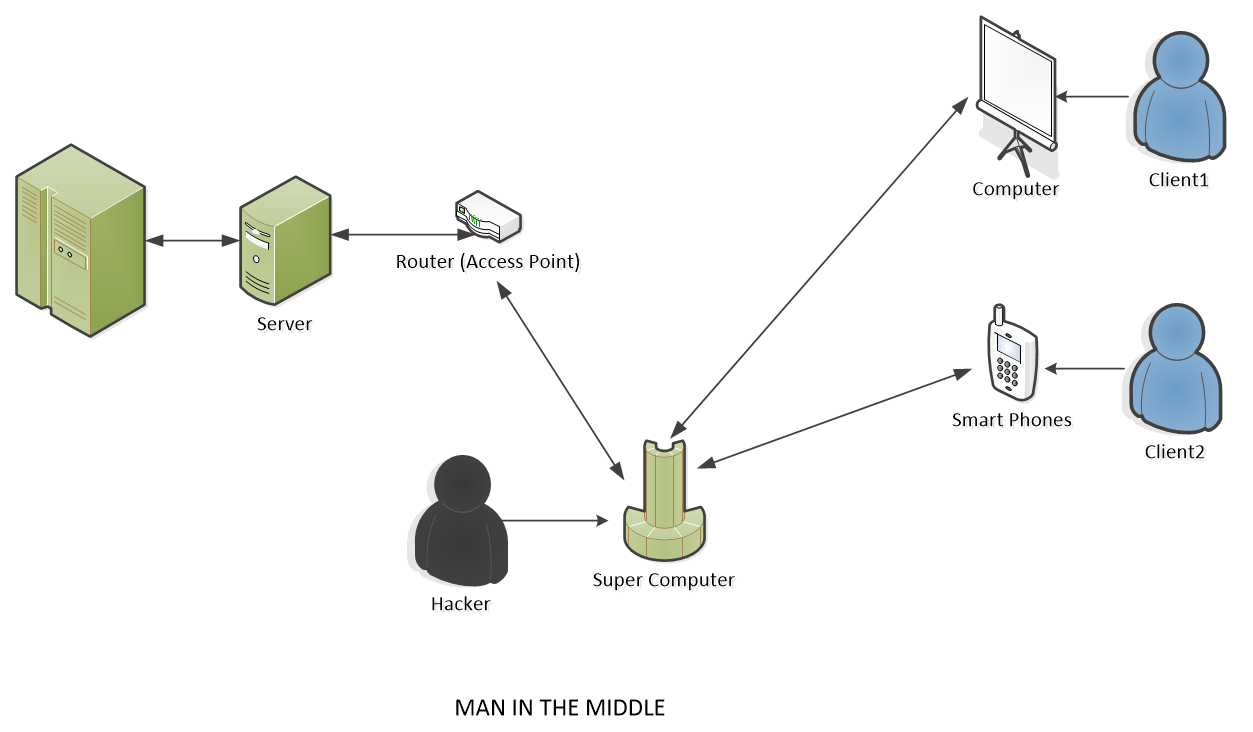


Fig 3. Man-In-The-Middle Etablished

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 then spread the attack to all other devices. Each of the devices gets accessed by the hacker can become a bot and send out information or spread out 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 usually need a supercomputer to handle the task.

The plan of attack is using multiple tools that are written in Python. Please note all the tool is written from scratch using the Python Scapy package, Python Scapy.HTTP, and Kali Linux as a remote computer. All the tools are contained within one script file (less than 200 Kilobyte (KB)) and stored within 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… Of course, the required libraries (Pandas, OS, Scapy) are pre-installed and ready to launch on Kali Linux. The total size of the USB is about 2.5 GB (with all tools and libraries). I only need a Linux kernel, python 3, and a couple of Python libraries to perform the attack. The plan is designed by Truc Huynh, with the idea from [14]:

* 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. Another way to gain access is that they can enable a backdoor on the user’s computer (by trapping users install viruses or malware). Let's just assume that we have already completed this task. While in reality, this is the hardest task to complete
* Step 2: Established Man-In-The-Middle by redirecting the flow of the packet by running ‘ARP Spoofer’. ‘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 a 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…).
* 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.
* 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. The ideal is that they can avoid detection by the network administrator, or any security system on the network (I have not accomplished this on my applications)
* 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. SECURITY PRACTICES

There are security practices that should be implemented 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
* Working with the developer team to implement secure programing principles that are industry standards if possible.
* Develop security methods to prevent insider attacks and outsider attacks (AWS Share Model, AWS Elastic Beanstalk).

# V. 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 …
* **Media Access Control (MAC Address)**: is a permanent, physical, unique address assigned to the network interface by the manufacturer.

# 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