Penetration testing

1. Experiment: Creating a Malicious Program and Sending a Virus to a Target Machine (For Educational Purposes)

Aim

To create a simple malicious program that simulates the behavior of a virus, then demonstrate how it could potentially be sent to a target machine to raise awareness about how viruses spread and the importance of system security.

Requirements

1. Programming Environment (Python, C, or similar).
2. Virtual Machine (VM): For safe testing, use an isolated environment like a virtual machine (e.g., VirtualBox, VMware).
3. A basic understanding of programming.
4. A controlled network to send the malicious payload (use only within a sandbox or test environment).

Procedure

1. **Create a Simple Malicious Program (Example: Python)**
   Below is a basic Python program that simulates a virus by creating a harmless payload, such as opening a message box repeatedly (just for simulation, without actual harm). This example does not cause harm, but demonstrates how malicious payloads work.

python

Copy code

```
# Simple Python "Virus" Simulation: This will display a pop-up message to simulate malicious behavior.

import time

import ctypes


def create_virus():

  # This function simulates opening pop-ups

  for _ in range(5):

    ctypes.windll.user32.MessageBoxW(0, "Your system is infected!", "Warning", 0x10)
```

```
    time.sleep(1)


if __name__ == '__main__':
  create_virus()
```

- The above code creates a loop where a warning message box pops up five times on the target machine, simulating a harmless effect of a virus.
- Note: In real malware, such programs may alter system files, steal data, or cause system crashes.

2. **Compile and Package the Program**
   To send this virus, compile the Python script into an executable (.exe) file. This can be done using tools like PyInstaller or cx_Freeze for Python.

Example command to compile using PyInstaller:

bash

Copy code

```
pyinstaller --onefile malicious_script.py
```

- This will create a standalone .exe file that can be executed on a target machine.

3. **Sending the Malicious Program**
   There are several ways a malicious program can be sent to a target machine. In a controlled, educational setting, this could be done over a local network for demonstration:
   - File Sharing: Send the executable via a shared folder or file transfer tool (e.g., SCP, SMB).
   - Email (Simulated Phishing): You could simulate sending an email with the .exe file as an attachment.

Example email message:

- Subject: "Important System Update"
- Body: "Click on the attached file to install the latest security update."
- Attachment: The compiled .exe file.

4. **Execution on Target Machine (Simulated):**
   Once the target system (the virtual machine in this case) receives the file, they execute it. In a real-world scenario, the malicious program would run and perform its designed function. Here, it will display a warning message.
5. **Observe the Behavior and Mitigation:**
   - Observation: Watch the program execute and see the virus simulation in action (the pop-up warnings).

- **Mitigation: Discuss how antivirus software, firewalls, and system security policies can prevent the execution of malicious software. Emphasize the importance of not running unknown files, especially from untrusted sources.**

Conclusion

In this experiment, we created a simple virus simulation to raise awareness about how malicious programs function. By observing the virus's behavior (in this case, harmless pop-up messages), we learned how such programs can be deployed, how they can be disguised, and how important it is to have security mechanisms in place to protect against them.

The demonstration also highlights the significance of safe computing practices: avoiding suspicious downloads, verifying email attachments, using antivirus software, and maintaining system security

2. Experiment: Bypassing the Operating System to Obtain Information Using John the Ripper Tool

---

Aim

To use John the Ripper to recover hashed credentials by cracking password hashes, demonstrating how weak passwords can compromise system security.

Requirements

1. John the Ripper tool installed on a Linux or Windows system.
2. A system with hashed passwords (e.g., /etc/shadow file on Linux or SAM file on Windows).
3. Administrative privileges to access the hash files in a controlled environment.
4. Hash types and sample password hashes (ensure ethical and authorized use).

Procedure

1. Install John the Ripper:
    - On Linux:

bash

Copy code

sudo apt-get update

sudo apt-get install john

    - On Windows: Download from the official John the Ripper website.
2. Obtain Hash Files:
    - On Linux:
        - Access the /etc/shadow file (requires root privileges):

bash

Copy code

sudo cat /etc/shadow

        - Extract the hashes for cracking.
    - On Windows:
        - Use tools like pwdump to extract the Security Account Manager (SAM) database hashes.
3. Identify Hash Types:

- Use tools or hash identification websites to determine the type of hash (e.g., MD5, SHA-1, bcrypt).
- Alternatively, John can automatically detect the hash type.

4. Prepare the Hashes for Cracking:
    - Save the extracted hashes to a text file (e.g., hashes.txt).
5. Run John the Ripper:
    - Basic Usage: Crack the password hashes:

bash

Copy code

john hashes.txt

    - Specify a Wordlist: Use a predefined wordlist for cracking (e.g., rockyou.txt):

bash

Copy code

john --wordlist=/path/to/rockyou.txt hashes.txt

    - Incremental Mode: Brute-force all possible combinations:

bash

Copy code

john --incremental hashes.txt

6. Monitor Progress:
    - View John's progress and statistics:

bash

Copy code

john --status

7. Recover Cracked Passwords:
    - Once cracking is complete, view the cracked passwords:

bash

Copy code

john --show hashes.txt

8. Analyze Results:
    - Compare the cracked passwords with security best practices.
    - Identify weak passwords and recommend improvements.

Conclusion

Using John the Ripper, we successfully demonstrated how weak password hashes can be cracked, revealing the importance of strong passwords and proper hashing techniques. This exercise highlights the critical need for robust password policies and secure hashing algorithms.

3.Experiment: Performing Information Gathering and Port Scanning in a Network Using Nmap

---

Aim

To gather information about a target system and perform a port scan to identify open ports and services using the Nmap tool.

Requirements

1. **Nmap** tool installed on a Linux or Windows system.
2. A target system or network (ensure you have permission to scan).
3. A controlled environment such as a virtual lab (e.g., Metasploitable2).

Procedure

1. **Install Nmap (if not already installed):**
    - On Linux:

bash

Copy code

sudo apt-get install nmap

    - On Windows: Download from the official Nmap website and install it.
2. **Set Up the Target Environment:**
    - Use a known target like Metasploitable2 or a test server in a virtualized setup.
    - Ensure that the target system is running and reachable over the network.
3. **Perform Information Gathering:**
    - **Ping Scan:** Check if the target is alive:

bash

Copy code

nmap -sn <Target_IP>

    - **OS and Service Detection:** Identify the operating system and running services on the target:

bash

Copy code

nmap -A <Target_IP>

    - **DNS and Host Information:** Gather DNS information:

bash

Copy code

nmap -sL <Target_IP_Range>

4. **Perform Port Scanning:**
    - **Basic Port Scan:** Identify open ports on the target:

bash

Copy code

nmap <Target_IP>

    - **Scan for Specific Ports:** Check specific ports (e.g., 22 and 80):

bash

Copy code

nmap -p 22,80 <Target_IP>

    - **Aggressive Scan:** Perform a detailed scan with service and version detection:

bash

Copy code

nmap -T4 -A <Target_IP>

5. **Save Scan Results:**
    - To save the scan output in a text file:

bash

Copy code

nmap -oN scan_results.txt <Target_IP>

    - For a more detailed XML report:

bash

Copy code

nmap -oX scan_results.xml <Target_IP>

6. **Analyze the Results:**
    - Review the output to identify:
        - Open ports and services.
        - Potential vulnerabilities based on detected software versions.
        - Hostnames and network topology if available.

Conclusion

Using the Nmap tool, we successfully performed information gathering and port scanning to identify active systems, open ports, and running services in the target network. These insights can be used to understand the network structure and improve security measures.

4. **Experiment: Implementing Various Social Engineering Techniques Using the Social Engineering Toolkit (SET)**

Aim

To use the Social Engineering Toolkit (SET) to simulate various social engineering attacks and understand the techniques attackers use to exploit human vulnerabilities.

Requirements

1. **Kali Linux** or any Linux distribution with SET installed.
2. A target system for testing (with explicit permission).
3. Administrative privileges to run SET.

Procedure

1. **Install SET (if not already installed):**
    - SET is pre-installed in Kali Linux. If not available, install it using:

bash

Copy code

```
git clone https://github.com/trustedsec/social-engineer-toolkit.git
cd social-engineer-toolkit
sudo python3 setup.py
```

2. **Launch SET:**
    - Open the terminal and start SET by typing:

bash

Copy code

```
sudo setoolkit
```

    - Accept the license agreement, and the main menu will appear.
3. **Select an Attack Vector:**
    - SET provides multiple attack vectors. Choose one based on the scenario:
        1. **Social-Engineering Attacks:** Choose this option for techniques like phishing and credential harvesting.
        2. **Website Attack Vectors:** To clone websites and inject malicious code.
        3. **Infectious Media Generator:** To create infected USB drives or files.

4. **Simulate a Phishing Attack (Example):**
    - From the menu, select:

bash

Copy code

Social-Engineering Attacks → Website Attack Vectors → Credential Harvester Attack Method

    - Choose **Site Cloner** and enter the URL of the website to be cloned (e.g., a login page).
    - SET will create a cloned version of the site and host it locally or on the specified IP.
5. **Test the Attack:**
    - Share the link (generated by SET) with the test target (ensure it is a controlled environment).
    - When the target enters credentials on the cloned site, they are captured and displayed in the terminal.
6. **Simulate Email Spoofing:**
    - Select:

bash

Copy code

Social-Engineering Attacks → Email Attack Vectors → Email Spoofing Attack

    - Configure the email sender, recipient, and message content.
    - Use a payload or attachment to simulate phishing emails.
7. **Simulate USB-Based Attacks:**
    - Choose:

bash

Copy code

Infectious Media Generator

    - Create a malicious payload and save it to a USB drive.
    - When the USB is used on the target system, it simulates an exploit or data theft.
8. **Analyze Results:**
    - Record the captured information or successful execution of the social engineering attack.
    - Review the logs generated by SET for insights into the attack's effectiveness.

Conclusion

Using the Social Engineering Toolkit (SET), we simulated various social engineering attacks, including phishing, email spoofing, and malicious USB payloads. These exercises emphasize the importance of user awareness and robust security measures to prevent real-world exploitation.

5.Aim

To perform vulnerability assessment on a target system using a Vulnerability Assessment and Penetration Testing (VAPT) tool to identify potential security weaknesses.

Requirements

1. A VAPT tool (e.g., Nessus, OpenVAS, or Acunetix).
2. A controlled test environment with a vulnerable target system (e.g., Metasploitable2, DVWA).
3. A computer with administrative access to install and run the VAPT tool.

Procedure

1. **Set Up the Environment:**
    - Install the chosen VAPT tool:
        - For **Nessus**, download it from the Tenable Nessus website and follow the installation steps.
        - For **OpenVAS**, install it using:

bash

Copy code

sudo apt-get install openvas

openvas-setup

    - Configure a vulnerable system (e.g., Metasploitable2) in a virtualized environment to act as the target.
2. **Launch the VAPT Tool:**
    - Open the tool's interface (web or application-based).
    - Create a new scan profile in the VAPT tool.
3. **Define Target Scope:**
    - Add the IP address or domain of the target system to the scan profile.
    - Specify the type of scan (e.g., full system scan, web application scan, or specific port scan).
4. **Configure Scan Settings:**
    - Adjust scan settings, such as authentication details (if required), to simulate a more realistic assessment.
    - Select the vulnerability database to be used (most tools update automatically).

5. **Run the Scan:**
    - Start the scan and monitor its progress.
    - The tool will enumerate vulnerabilities, misconfigurations, and outdated software in the target system.
6. **Analyze Results:**
    - Review the scan report, which will categorize vulnerabilities based on severity (e.g., Critical, High, Medium, Low).
    - Identify details such as affected components, CVE references, and remediation steps.
7. **Save and Document Findings:**
    - Export the scan results in a preferred format (e.g., PDF, HTML).
    - Document key findings, including vulnerabilities detected and suggested mitigations.

Conclusion

The VAPT tool successfully identified vulnerabilities in the target system. The assessment highlights areas that need immediate attention to enhance the security posture, such as updating software, patching known vulnerabilities, and correcting misconfigurations. Regular assessments are critical for maintaining secure systems.

6. **Experiment: Uncovering Vulnerabilities in the Network Using Metasploit**

Aim

To identify and exploit vulnerabilities in a network using the Metasploit Framework.

Requirements

1. Metasploit Framework installed on a Linux or Windows system (e.g., Kali Linux).
2. A controlled test environment (e.g., Metasploitable2, DVWA, or other vulnerable virtual machines).
3. Administrative privileges on the testing system.

Procedure

1. **Install and Launch Metasploit Framework:**
    - On **Kali Linux**, Metasploit is pre-installed. Start it by typing:

bash

Copy code

msfconsole

    - On other Linux systems, install Metasploit using:

bash

Copy code

sudo apt install metasploit-framework

msfconsole

2. **Set Up the Target System:**
    - Use a vulnerable virtual machine (e.g., Metasploitable2) to act as the target.
    - Ensure that the target is running and accessible on the same network as the Metasploit system.
3. **Perform Network Scanning (Optional):**
    - Use Nmap or Metasploit's integrated scanner to identify potential targets:

bash

Copy code

nmap -A <Target_IP>

    - Alternatively, in Metasploit:

bash

Copy code

use auxiliary/scanner/portscan/tcp

set RHOSTS <Target_IP>

run

4. **Search for Vulnerabilities:**
    - Identify potential vulnerabilities by searching for known exploits:

bash

Copy code

search <Service_Name>

Example:

bash

Copy code

search vsftpd

5. **Select an Exploit Module:**
    - Choose a specific exploit from the search results:

bash

Copy code

use exploit/<exploit_path>

Example:

bash

Copy code

use exploit/unix/ftp/vsftpd_234_backdoor

6. **Set Exploit Options:**
    - Configure the target settings for the exploit:

bash

Copy code

set RHOST <Target_IP>

set RPORT <Port_Number> # If required

7. **Execute the Exploit:**
    - Run the exploit to test for the vulnerability:

bash

Copy code

exploit

- If successful, you may gain access to the target system or receive confirmation of the vulnerability.

8. **Generate a Report (Optional):**
    - Use Metasploit to document findings:

bash

Copy code

db_export -f xml /path/to/report.xml

9. **Mitigation Recommendations:**
    - Review the exploited vulnerabilities and recommend solutions, such as patching, updating, or configuring firewalls to block specific traffic.

Conclusion

The Metasploit Framework successfully identified and exploited vulnerabilities in the network. This experiment demonstrated how attackers could exploit misconfigurations or outdated software. The findings emphasize the importance of regular vulnerability assessments and timely patching.

7. **Experiment: Performing ARP Poisoning and MAC Flooding to Perform Denial of Service Using Cain & Abel Tool**

Important Disclaimer

ARP poisoning, MAC flooding, and Denial of Service (DoS) attacks can disrupt network communications and are **illegal and unethical** if used outside a controlled, authorized environment. This experiment must be performed **only** in a **test environment** such as a virtual machine or a dedicated lab network, with **explicit permission** from the network administrator.

This experiment is for educational purposes to raise awareness about network vulnerabilities and the importance of network security.

Aim

To demonstrate **ARP poisoning** and **MAC flooding** attacks using **Cain & Abel**, a tool for performing various types of network attacks. These techniques are used to disrupt communication between devices on the network and cause **Denial of Service (DoS)**.

Requirements

1. **Cain & Abel** tool installed on a Windows machine.
2. A network with at least two devices or virtual machines (one target and one attacker).
3. Administrator privileges on the attacker machine.
4. **Target machine**: The machine on which ARP poisoning or MAC flooding will be performed.

Procedure: ARP Poisoning (Man-in-the-Middle Attack)

1. **Install and Launch Cain & Abel:**
    - Download and install Cain & Abel from the official website.
    - Launch Cain & Abel with administrative privileges.
2. **Configure Cain & Abel for ARP Poisoning:**
    - Go to the **"ARP Poisoning"** tab under the **"Sniffer"** section.
    - **Enable Sniffer** by clicking the "Start Sniffer" button to capture packets on the network.
    - Click **"Target"** to add the IP addresses of the **Gateway** (router) and **Target machine** (victim machine).
    - Cain & Abel will automatically detect the devices connected to your network. Select the **Gateway** and **Target machine** from the list of available devices.

3. **Start ARP Poisoning:**
    - With both the **Gateway** and **Target machine** selected, click the **"Poison"** button.
    - Cain & Abel will now send **malicious ARP packets** to both the target and gateway, causing the network traffic between them to pass through the attacker machine. This creates a **Man-in-the-Middle (MitM)** attack.
    - **Effect:** All network traffic between the target and gateway is intercepted by the attacker. This could allow the attacker to eavesdrop on communications, modify data, or redirect traffic.
4. **Observe the Effect:**
    - You can view the captured packets under the **"Sniffer"** tab. The attacker can inspect all traffic passing through the network.
    - You can also inject fake packets or redirect traffic to simulate further malicious activity.

Procedure: MAC Flooding (DoS Attack)

1. **Configure Cain & Abel for MAC Flooding:**
    - Go to the **"MITM"** (Man-in-the-Middle) tab.
    - Select **"MAC Flooding"** from the attack options available.
2. **Start MAC Flooding Attack:**
    - Select the **Target Switch** (or device).
    - Click **"Start Flooding"** to begin sending random MAC addresses to the switch. This will overwhelm the switch's MAC address table.
3. **Effect of MAC Flooding:**
    - A **MAC Flooding** attack attempts to flood a network switch with a large number of fake MAC addresses, forcing it to act as if it has no memory of MAC addresses.
    - The switch will enter **fail-open mode**, where it floods all incoming packets to all ports, essentially **disrupting communication** within the network.
    - As a result, the **target machine** will not be able to communicate with other devices effectively, leading to a Denial of Service (DoS) scenario.
4. **Stop the Attack:**
    - Once the flooding is complete and you have observed the effect, click **"Stop Flooding"** to end the attack and restore normal operation.

Conclusion

In this experiment, we demonstrated how **ARP poisoning** and **MAC flooding** can be used to perform **Denial of Service (DoS)** attacks on a network.

- **ARP Poisoning** can lead to **Man-in-the-Middle** attacks, where the attacker intercepts, modifies, or eavesdrops on network traffic.
- **MAC Flooding** can **disrupt network communication** by overwhelming the switch with fake MAC addresses, causing it to flood all ports.

This experiment highlights the vulnerabilities in networking protocols and the importance of securing networks against these types of attacks by using methods like **static ARP entries**, **port security**, **MAC address filtering**, and **strong network monitoring**.

8. **Experiment: Capturing and Analyzing Live Traffic Using Wireshark Tool**

Aim

To capture and analyze live network traffic using Wireshark to identify patterns, protocols, and potential anomalies in the network.

Requirements

1. Wireshark installed on your system.
2. A system connected to a network.
3. Administrative privileges to capture live traffic.

Procedure

1. **Install Wireshark (if not already installed):**
    - On Linux:

bash

Copy code

sudo apt-get install wireshark

    - On Windows or macOS: Download Wireshark from the official website and follow the installation instructions.
2. **Launch Wireshark:**
    - Open Wireshark with administrative privileges.
    - Select the network interface you want to monitor from the list of available interfaces.
3. **Start Capturing Traffic:**
    - Click on the interface name (e.g., eth0, wlan0) to start capturing live traffic.
    - Monitor the traffic in real-time as packets are captured.
4. **Apply Filters:**
    - To focus on specific traffic, use filters in the **Filter bar**:
        - Capture HTTP traffic:

http

Copy code

http

        - Capture ICMP (ping) traffic:
          icmp
        - Capture traffic to/from a specific IP:
          ip.addr == <Target_IP>
    - Press **Enter** to apply the filter.
5. **Inspect Individual Packets:**
    - Select any packet from the captured list.
    - Use the details pane to analyze the packet:
        - **Frame Layer:** General information like capture time and packet size.
        - **Ethernet Layer:** MAC addresses and type of traffic.
        - **IP Layer:** Source and destination IP addresses.
        - **Transport Layer (TCP/UDP):** Ports and sequence numbers.
        - **Application Layer:** Protocol-specific details like HTTP, DNS, or FTP.
6. **Save the Capture (Optional):**
    - Save the captured traffic for later analysis:
      File → Save As and choose a .pcap file format.
7. **Analyze Captured Traffic:**
    - Identify patterns such as high-volume traffic (potential DDoS attacks) or anomalies like unauthorized connections.
    - Use the **Statistics menu** to generate visualizations:
        - Protocol Hierarchy
        - Endpoints
        - IO Graphs
8. **Stop the Capture:**
    - Once sufficient data is captured, click **Stop** to end the session.

Conclusion

Using Wireshark, we successfully captured and analyzed live network traffic, identifying different protocols and inspecting packet-level details. This experiment highlights the importance of monitoring for maintaining network security and identifying potential issues.

9. aim:To scan the web servers using the Nikto tool.

Procedure

1. **Set Up the Environment:**
    - Ensure Nikto is installed on your system. If not, install it using:

bash

Copy code

sudo apt-get install nikto

    - Use a controlled environment with a vulnerable web server such as DVWA or Metasploitable2.
2. **Perform a Basic Scan:**
    - Open the terminal and run the command:

bash

Copy code

nikto -h <Target_IP_or_Domain>

    - Replace <Target_IP_or_Domain> with the IP address or domain name of the target web server.
3. **Perform SSL/TLS Scanning (if applicable):**
    - To check for SSL-related vulnerabilities, run:

bash

Copy code

nikto -h <Target_IP_or_Domain> -ssl

4. **Save the Results:**
    - To save the scan report in a text file, use:

bash

Copy code

nikto -h <Target_IP_or_Domain> -o scan_report.txt

5. **Advanced Options (Optional):**
    - To scan a specific port, use:

bash

Copy code

nikto -h <Target_IP_or_Domain> -p <Port_Number>

- To use a proxy during scanning, run:

bash

Copy code

nikto -h <Target_IP_or_Domain> -useproxy <Proxy_Address>

6. **Analyze Results:**
   - Review the output for:
     - Outdated software.
     - Misconfigured HTTP headers.
     - Known vulnerabilities with corresponding CVE details.

Conclusion

The Nikto tool successfully identified vulnerabilities and misconfigurations in the web server. These findings can be used to improve the security posture of the server by addressing the reported issues, such as updating software and correcting insecure configurations.

10.Experiment: Implementing Intrusion Detection and Prevention System Using Snort Tool

---

Aim

To configure and implement Snort as an Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) to monitor network traffic and detect potential threats.

Requirements

1. A system with Snort installed (Linux or Windows).
2. Test network or virtual environment with traffic generation tools (e.g., Metasploitable2, Kali Linux, or Wireshark).
3. Administrative privileges for configuration.

Procedure

1. **Install Snort:**
    - On Debian-based Linux (e.g., Ubuntu):

bash

Copy code

sudo apt-get update

sudo apt-get install snort

    - During installation, specify the network interface to monitor and set up Snort in IDS mode.
    - On other systems, download Snort from the official website and follow the installation instructions.
2. **Verify Installation:**
    - Confirm Snort is installed by checking the version:

bash

Copy code

snort --version

3. **Configure Snort:**
    - Edit the Snort configuration file, typically located at /etc/snort/snort.conf:
        - Set up the **HOME_NET** variable to match your network:

bash

Copy code

var HOME_NET 192.168.1.0/24

- Configure logging directories:

bash

Copy code

output alert_fast: /var/log/snort/alerts

4. **Add or Customize Rules:**
    - Snort uses rule files to define detection patterns. You can add rules to /etc/snort/rules/local.rules:
      Example rule to detect ICMP (ping) packets:

bash

Copy code

alert icmp any any -> $HOME_NET any (msg:"ICMP Packet Detected"; sid:1000001; rev:1;)

5. **Run Snort in IDS Mode:**
    - Use the following command to start Snort and monitor traffic:

bash

Copy code

sudo snort -A console -q -c /etc/snort/snort.conf -i eth0

Replace eth0 with your network interface.

6. **Generate Traffic for Testing:**
    - Simulate attacks or use a traffic generation tool (e.g., perform a ping or Nmap scan).
    - Monitor the Snort console or log files for alerts:

bash

Copy code

cat /var/log/snort/alerts

7. **Run Snort in IPS Mode (Optional):**
    - Configure Snort to drop malicious packets using inline mode:
        - Install a packet manipulation library like libnetfilter_queue:

bash

Copy code

sudo apt-get install libnetfilter-queue-dev

- Start Snort in IPS mode:

bash

Copy code

sudo snort -Q --daq nfq -c /etc/snort/snort.conf -i eth0

8. **Analyze Alerts and Logs:**
    - Review the generated alerts and take corrective actions based on detected threats.

Conclusion

Snort was successfully configured as an IDS/IPS to monitor network traffic and detect potential threats. Alerts generated by Snort help in identifying malicious activity, while IPS mode actively blocks identified threats. Regular updates to Snort rules are crucial to maintain effectiveness.