Assignment 8

Imagine you are a cybersecurity analyst working for a large multinational corporation. One morning, your team receives an urgent report about a potential security breach in the company's network. The IT department has noticed unusual network activity originating from a particular IP address. Your team has been tasked with investigating this incident to determine if it poses a threat to the organization's network security.

Assignment Question:

1. Using the Python library Scapy, analyze the network packets associated with the suspicious IP address provided.

Expected Code:

1. Write a python code to Network Packet Analysis with Scapy

Scapy is a powerful Python library used for network packet manipulation. It can be used to capture, dissect, generate, and manipulate network packets. Scapy is particularly useful for cybersecurity professionals for network analysis and penetration testing. It supports various protocols and provides tools to decode and display packet contents.

Step-by-Step Breakdown for Capturing and Analyzing Network Traffic

Step 1: Install Scapy

First, install Scapy using pip if it isn't already installed:

pip install scapy

Step 2: Capture Network Packets

To capture packets from the network, you need to use Scapy's sniffing function. You can specify the IP address to filter the packets related to the suspicious activity.

```
from scapy.all import sniff

# Define the suspicious IP address

suspicious_ip = "192.168.1.100"

# Function to filter packets based on the suspicious IP

def packet_filter(packet):

    return suspicious_ip in packet[IP].src or suspicious_ip in packet[IP].dst

# Capture packets
```

packets = sniff(filter="ip", prn=lambda x: x.summary(), store=True, count=100, lfilter=packet_filter)

Step 3: Analyze Captured Packets

Scapy provides several methods to analyze the captured packets. You can inspect each packet to look for anomalies.

```
from scapy.all import IP, TCP, UDP

# Function to analyze packets

def analyze_packets(packets):

  for packet in packets:

    if IP in packet:

      ip_src = packet[IP].src

      ip_dst = packet[IP].dst

      print(f"Packet: {packet.summary()}")

      print(f"Source IP: {ip_src}")

      print(f"Destination IP: {ip_dst}")


      if TCP in packet:

        print(f"TCP Packet: {packet[TCP].summary()}")

        # Add additional TCP-specific analysis if needed


      if UDP in packet:

        print(f"UDP Packet: {packet[UDP].summary()}")

        # Add additional UDP-specific analysis if needed


      print("\n")
```

 # Call the function to analyze captured packets

analyze_packets(packets)

Identification and Interpretation of Suspicious Network Behavior

During the analysis phase, you look for unusual patterns such as:

Unusual ports being accessed.

High number of packets sent or received in a short time.

Unexpected protocol usage.

Payload content that might indicate malicious activity (e.g., suspicious commands or scripts).

Recommendations for Mitigating Identified Security Risks

Implement Intrusion Detection Systems (IDS): Use IDS to monitor network traffic for suspicious activities.

Firewall Configuration: Ensure that the firewall is properly configured to block unauthorized access.

Network Segmentation: Segment the network to limit the spread of potential attacks.

Regular Updates and Patching: Keep all systems and software up to date with the latest security patches.

User Training: Educate employees on recognizing phishing attempts and other common attack vectors.

Access Controls: Implement strict access controls and ensure that only authorized personnel have access to sensitive information.

Expected Python Code for Network Packet Analysis with Scapy

```
from scapy.all import sniff, IP, TCP, UDP


# Define the suspicious IP address

suspicious_ip = "192.168.1.100"


# Function to filter packets based on the suspicious IP

def packet_filter(packet):

  return IP in packet and (suspicious_ip == packet[IP].src or suspicious_ip == packet[IP].dst)


# Capture packets

packets = sniff(filter="ip", prn=lambda x: x.summary(), store=True, count=100, lfilter=packet_filter)


# Function to analyze packets

def analyze_packets(packets):

  for packet in packets:

    if IP in packet:

      ip_src = packet[IP].src

      ip_dst = packet[IP].dst

      print(f"Packet: {packet.summary()}")

      print(f"Source IP: {ip_src}")

      print(f"Destination IP: {ip_dst}")


      if TCP in packet:

        print(f"TCP Packet: {packet[TCP].summary()}")
```

```
        # Add additional TCP-specific analysis if needed

    if UDP in packet:
        print(f"UDP Packet: {packet[UDP].summary()}")
        # Add additional UDP-specific analysis if needed

    print("\n")

# Call the function to analyze captured packets
analyze_packets(packets)
```

This code captures 100 packets related to the specified IP address and provides a summary of each packet, including the source and destination IPs and details about TCP and UDP packets. Adjust the count as needed to capture more packets for a thorough analysis.