Q1: 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 Procedure:

1. A detailed explanation of how Scapy can be utilized to capture and dissect network packets. 2. A step-by-step breakdown of the process you followed to capture and analyze the network traffic.
3. Identification and interpretation of any suspicious or anomalous network behavior observed in the captured packets.
4. Recommendations for mitigating the identified security risks and securing the network against similar threats in the future.

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

Ans: Let's dive into the world of network packet analysis using Scapy. As a cybersecurity analyst, you'll find Scapy to be a powerful tool for dissecting, analyzing, and understanding network traffic. Here's a step-by-step guide on how to achieve this:

Analyzing Network Packets with Scapy

1. Introduction to Scapy

Scapy is a Python program that enables you to send, sniff, dissect, and forge network packets. It allows you to construct tools for probing, scanning, or even attacking networks. Scapy's versatility makes it an essential tool for network professionals and security enthusiasts.

2. Installation

First, make sure you have Scapy installed. You can do this using pip:

```
pip install scapy
```

3. Capturing and Analyzing Packets

Let's start by capturing network packets from a pcap file (e.g., "capture.pcap"). We'll loop over the packets and display their protocol layers and values:

```
from scapy.all import *

# Open the pcap file
packets = PcapReader("capture.pcap")
```

```
# Loop over each packet and show its details
for packet in packets:
    print(packet.show())
```

4. Filtering by Protocol

Next, we can filter packets based on the payload protocol. For instance, to process only DNS packets, use:

```
from scapy.all import *

# Open the pcap file
packets = PcapReader("capture.pcap")

# Filter DNS packets
for packet in packets:
    if packet.hasLayer(DNS):
        # Process DNS packets here
        pass
```

5. Detecting Domain Generation Algorithms (DGA)

DGA algorithms are used by malware to generate domain names for command and control servers. We can detect them by counting empty DNS responses received by each IP in the capture:

```
from scapy.all import *

packets = PcapReader("capture.pcap")
counts = {}

for packet in packets:
    if packet.haslayer(DNS) and packet[DNS].qr == 1 and
packet[DNS].ancount == 0:
        # DNS query returned no answer
        # Extract the destination IP (device that sent the query)
        dest_ip = packet[IP].dst
        counts[dest_ip] = counts.get(dest_ip, 0) + 1

# Analyze the counts to identify potential DGA activity
# (e.g., unusually high number of empty DNS responses)
```
6. Recommendations

Based on the analysis, consider the following recommendations:

- Investigate further any suspicious or anomalous behavior observed.
- Implement network segmentation and access controls.
- Monitor and log network traffic continuously.
- Educate users about security best practices.

Remember that Scapy provides a powerful toolkit for network analysis, and with practice, you’ll become adept at identifying threats and securing your network.

Q2: Imagine you are working as a cybersecurity analyst at a prestigious firm. Recently, your company has been experiencing a surge in cyber attacks, particularly through phishing emails and websites. These attacks have not only compromised sensitive information but also tarnished the reputation of the company. In light of these events, your team has been tasked with developing a robust solution to detect and mitigate phishing websites effectively. Leveraging your expertise in Python programming and cybersecurity, your goal is to create a program that can accurately identify phishing websites based on various features and indicators.

Assignment Task: Using the Python programming language, develop a phishing website detection system that analyzes website characteristics and determines the likelihood of it being a phishing site.

Expected Procedure: 1. Accept 2 web URL. One real and another one phishing. 2. Analyze the data from both the websites. 3. Identify the phishing site.

Expected Code: 1. Phishing Website Detection with Python

Ans: Detecting phishing websites is crucial for safeguarding sensitive information and maintaining a company's reputation. Let's create a simple Python program that analyzes website characteristics and identifies potential phishing sites.

We'll use an external API to assess the risk associated with a given URL. Specifically, we'll leverage the **ipqualityscore API** to check if a URL is suspicious or potentially malicious. Here's how you can achieve this:

Phishing Website Detection with Python

1. Sign Up for ipqualityscore API Access

First, sign up for an account on the ipqualityscore website to obtain your API key. This key will allow us to query the API for risk scores related to URLs.

2. Required Packages

We'll use the `requests` library to send HTTP requests to the API. If you haven't installed it yet, you can do so using:

```
pip install requests
```

3. Main Program

Below is a Python script that takes a URL as input, queries the ipqualityscore API, and provides information about the URL's risk level:

```
import requests
import urllib.parse
import json
```

```
def check_phishing_url(url):
    # Encode the URL
    encoded_url = urllib.parse.quote(url, safe='')

    # Replace 'YOUR_KEY' with your actual API key
    api_url = "https://ipqualityscore.com/api/json/url/YOUR_KEY/"
    response = requests.get(api_url + encoded_url)

    # Parse the JSON response
    data = response.json()

    # Print relevant fields
    print("Domain:", data.get("domain"))
    print("IP Address:", data.get("IP_address"))
    print("Risk Score:", data.get("risk_score"))
    print("Suspicious:", data.get("suspicious"))
    print("Phishing:", data.get("phishing"))
    print("Malware:", data.get("malware"))
    print("Spamming:", data.get("spamming"))
    print("Adult Content:", data.get("adult"))

# Example usage
if __name__ == "__main__":
    url_to_check = "https://www.google.com"  # Replace with the URL
you want to analyze
    check_phishing_url(url_to_check)
```

4. Interpretation

- **Risk Score**: A higher score indicates a higher risk level (85+ is high risk).
- **Suspicious**: Indicates whether the URL is suspected of being malicious.
- **Phishing**: Indicates if the URL is associated with phishing behavior.
- **Malware**: Indicates if the URL is associated with malware or viruses.
- **Spamming**: Indicates if the domain is associated with email spam.
- **Adult Content**: Indicates if the URL or domain hosts adult content.

Remember to replace `"YOUR_KEY"` with your actual API key. Adjust the classification policy based on your specific needs.

This program provides a quick assessment, but for a more comprehensive solution, consider incorporating machine learning techniques or additional features. Happy detecting!