Assignment - 8

Question - 1

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 analyse the network

traffic.

3. Identification and interpretation of any suspicious or anomalous network behaviour 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

Solution:

Using Scapy for Network Packet Analysis

1. How Scapy Analyzes Network Packets:

Scapy is a powerful Python library that allows you to craft, send, sniff, and dissect network packets. It provides a user-friendly interface for manipulating network traffic on various layers of the OSI model. Here's how Scapy helps analyze network packets:

- **Packet Sniffing:** Scapy can capture packets flowing through your network interface. By specifying the interface and filters, you can target specific traffic related to the suspicious IP address.
- **Packet Dissection:** Scapy allows you to deconstruct captured packets, revealing information from each layer (e.g., Ethernet header, IP header, TCP/UDP header, application data).
- **Traffic Analysis:** By examining captured packets, you can identify patterns, suspicious content, and potential vulnerabilities.

2. Analyzing Network Traffic with Scapy:

Step 1: Import Libraries and Define Interface

Python

```
from scapy.all import sniff, get_if_list

# Get available network interfaces

interfaces = get_if_list()

# Choose the interface for capturing traffic (replace with your interface name)

interface = interfaces[0].name


# Define suspicious IP address

suspicious_ip = "10.0.0.123"
```

Step 2: Capture Packets with Filter

Python

```
def analyze_packets(packet):
    # Filter packets based on source or destination IP address
    if packet[IP].src == suspicious_ip or packet[IP].dst == suspicious_ip:
        # Print captured packet information (source, destination, protocol, etc.)
        print(packet.summary())


# Capture packets for a specific duration (adjust capture time as needed)

sniff(iface=interface, prn=analyze_packets, filter="ip", timeout=60)
```

Step 3: Analyze Captured Packets

- **Review packet summaries:** The analyze_packets function will print summaries of captured packets containing the suspicious IP address.
- **Identify Anomalies:** Look for unusual patterns like:
    - High volume of traffic from the suspicious IP.
    - Unusual ports being used (e.g., port scanning attempts).
    - Specific protocols not typically used by your organization (e.g., known exploit tools).
    - Suspicious content within the captured data payload (if applicable).

4. Recommendations and Mitigation Strategies:

Based on the identified anomalies, you can recommend mitigation strategies like:

- **Blocking the suspicious IP:** If the activity is clearly malicious, block the suspicious IP address at the firewall.
- **Deep Packet Inspection (DPI):** Implement DPI to analyze the content of packets for malicious payloads.
- **Vulnerability Scanning:** Conduct a vulnerability scan on your network to identify potential weaknesses exploited by the attacker.
- **User Awareness Training:** Educate users about phishing attempts and social engineering tactics to prevent them from falling victim.

Note: This is a basic example. Scapy offers various functionalities for deeper analysis. Refer to the Scapy documentation for advanced filtering, specific protocol dissection, and crafting custom filters based on your needs.

Disclaimer: This code snippet is for educational purposes only. Modifying network configurations or blocking IP addresses might require administrative privileges and should be done with caution in a production environment.

Question – 2

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

Solution:

Phishing Website Detection System with Python (Basic Approach)

This Python program demonstrates a basic approach to identifying potential phishing websites based on several features.

Disclaimer: This is a simplified example for educational purposes. Real-world phishing detection requires more sophisticated techniques and machine learning models.

Features Analyzed:

1. **URL Features:**
    - Presence of unusual subdomains or typos in the URL.
    - Matching the domain name with the website's legitimate name.
2. **Website Content:**
    - Presence of suspicious elements like excessive grammatical errors or urgency cues.
    - Mismatched website content with the advertised purpose.

1. Libraries and Functions:

Python

```
import requests
from bs4 import BeautifulSoup

def check_url(url):
  """
  Analyzes a URL for suspicious characteristics.

  Args:
    url: The URL to be analyzed.

  Returns:
    A dictionary containing analysis results.
  """
  results = {}
  try:
    response = requests.get(url)
    response.raise_for_status()  # Raise exception for bad status codes
    soup = BeautifulSoup(response.content, "lxml")

    # Check URL features
    results["domain_match"] = url.split("//")[1].split(".")[0] in url.lower()
    results["unusual_subdomains"] = any(subdomain.count("-") >= 2 for subdomain in
url.split(".")[1:-1])

    # Check website content (basic checks)
    results["grammatical_errors"] = len(soup.find_all(text=lambda text: text and any(error in
text for error in ["teh", "liek", "2 much"]))) > 0
    results["urgency_cues"] = len(soup.find_all(text=lambda text: text and any(cue in
text.lower() for cue in ["immediate action", "limited time"]))) > 0
    results["content_mismatch"] = "bank" not in url.lower() and "bank" in
soup.find_all(text=True)[0].lower()
```

```
  except requests.exceptions.RequestException as e:
    results["error"] = f"Error fetching website: {e}"

  return results
```

2. Analyzing Real and Phishing URLs:

```
Python
# Replace with a legitimate website URL
legitimate_url = "https://www.google.com/"

# Replace with a suspected phishing website URL
phishing_url = "https://phishingsite.example/"

legitimate_results = check_url(legitimate_url)
phishing_results = check_url(phishing_url)

print("Analysis of Legitimate URL:")
print(legitimate_results)

print("\nAnalysis of Suspected Phishing URL:")
print(phishing_results)
```

3. Identifying Phishing Site:

Based on the analysis results, identify inconsistencies that might indicate a phishing website. For example:

- Presence of unusual subdomains or typos in the URL.
- Mismatch between the domain name and the website's content.
- Grammatical errors, urgency cues, or content mismatch not present in the legitimate website.

Note: This is a basic example. Real-world phishing detection involves more features, advanced checks, and machine learning models trained on phishing website datasets.