CYBER SECURITY ASSIGNMENT-8

user

R. AISHWARYA REDDY 2406CYS121

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

ANSWER - 1

IF A SECURITY BREACH, THE STEPS BELOW MIGHT BE FOLLOWED BY A TEAM

Initial Actions:

Assemble the Team: Convene a team with expertise in security analysis, network infrastructure, and forensics.

Gather Information: Collect details from the IT department about the unusual network activity.

This includes:

Timestamps of the activity

Specific nature of the unusual traffic (e.g., data transfer, port scans)

Originating IP address

Secure and Isolate: Isolate the compromised system (if identified) to prevent further lateral movement within the network.

Threat Analysis:

Identify the IP Address: Use geolocation tools to determine the origin of the suspicious IP address. Is it a known malicious source or a suspicious location?

Investigate Network Logs: Analyze firewall logs, intrusion detection system (IDS) alerts, and system logs for suspicious activity around the reported timeframe.

Check for Malware: Run anti-malware scans on potentially compromised

systems to identify any malicious software.

Search for Exfiltration: Look for signs of data exfiltration, such as unauthorized file transfers or unusual outbound traffic.

Containment and Eradication:

Change Credentials: Reset passwords for potentially compromised accounts with strong, unique credentials.

Patch Systems: Ensure all systems are up-to-date with the latest security patches to address any known vulnerabilities.

Review Security Policies: Analyze existing security policies to identify any weaknesses that might have been exploited.

Evidence Collection and Reporting:

Document Everything: Maintain a detailed log of all actions taken, findings, and remediation steps.

Collect Forensics: Secure and preserve potentially compromised systems for further forensic analysis by specialists (if necessary).

Prepare a Report: Document the investigation process, findings, and recommendations for next steps. This report should be presented to relevant stakeholders within the organization.

Post-Incident Activities:

Identify Root Cause: Conduct a root cause analysis to determine how the breach

occurred and what vulnerabilities were exploited.

Improve Security Posture: Implement additional security measures based on the findings, such as enhanced access controls, network segmentation, or user training programs.

Lessons Learned: Review the incident response plan and identify areas for improvement to ensure a more efficient response for future incidents.

Absolutely! Let's break down the investigation of the suspicious network activity using Scapy in Google Colab, just like we'd do in a real-world cybersecurity scenario. Remember, capturing live

traffic on a production network is risky, so we'll focus on analyzing a captured file (pcap) containing the suspicious traffic.

1. How Scapy Helps Us Analyze Network Traffic

Imagine Scapy as a powerful Swiss Army knife for network traffic analysis. Here's what it can do:

Capture Live Traffic (Not Used Today): While this feature is handy, we won't use it today to avoid disrupting the actual network. We'll focus on analyzing pre-captured traffic (pcap file).

Read and Write pcap Files: Scapy's rdpcap function lets us read packets from a pcap file, and wrpcap can be used to write captured packets to a new file (not used here).

Dissect Network Packets: Scapy allows us to peel back the layers of a network packet like an onion. We can examine individual layers (Ethernet, IP, TCP, UDP) and extract information like source/destination IP addresses, ports, protocols, and even packet payloads (with caution).

Expected Code:

1. Write a Python code for Network Packet Analysis with Scapy

Code:

1. Import Libraries:

```
!pip install scapy
```

!pip install scapy: This line installs the Scapy library if it's not already available in the Colab environment. To run this only once at the beginning.

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

from scapy.all import sniff, rdpcap, wrpcap, IP, TCP, UDP: This line imports

the necessary functions from the Scapy library.

sniff (not used in this example) is used to capture live network traffic.

rdpcap is used to read packets from a captured pcap file.

wrpcap is used to write captured packets to a pcap file (not used in this example).

IP, TCP, and UDP are used to access specific layers of the network packets for analysis.

2. Load Captured Traffic:

Uploading your captured pcap file to Google Colab:

Using the rdpcap function to read the packets from the file.

Uploading the pcap file:

There are a couple of ways to upload your captured pcap file to your Google Colab environment:

Upload from local machine:

Go to the "Files" tab in the left sidebar of your Colab notebook.

Click the upload button (arrow pointing upwards icon) and select your pcap file from your local machine.

Upload from Google Drive:

If your pcap file is stored in your Google Drive, you can mount your Drive to Colab

Once mounted, you can access your Drive files within Colab's file system. You can then reference the file path within the rdpcap function.

2. Using rdpcap function:

Once your pcap file is uploaded and accessible within Colab, you can use the rdpcap function from the Scapy library to read the packets.

The code snippet in the previous explanation assumes the file is located in your current working directory within Colab. Here's a breakdown of the code:

filename = "suspicious_traffic.pcap": This line defines a variable filename and assigns the name of your pcap file (replace "suspicious_traffic.pcap" with your actual filename).

packets = rdpcap(filename): This line uses the rdpcap function. It takes the filename as an argument and reads the packets from the specified file. The function returns a list containing all the captured packets. We store this list in the variable packets for further analysis.

```
filename = "data/suspicious_traffic.pcap"
packets = rdpcap(filename)
```

Load Captured Packets:

Python

filename = "suspicious_traffic.pcap"

packets = rdpcap(filename)

Use code with caution.

Filter Packets by Suspicious IP:

Replace "10.0.0.10" with the actual suspicious IP address you're investigating:

Python

suspicious_ip_address = "10.0.0.10"

suspicious_packets = [packet for packet in packets if packet[IP].src == suspicious_ip_address]

Use code with caution.

Analyze Packet Details:

This code iterates through the filtered packets and prints relevant information:

Python

for packet in suspicious_packets:

```
        print(f"Source: {packet[IP].src}, Destination: {packet[IP].dst}")
        print(f"Protocol: {packet[IP].proto}")
        if packet.haslayer(TCP):
            print(f" - Port: {packet[TCP].sport} > {packet[TCP].dport}")
        elif packet.haslayer(UDP):
            print(f" - Port: {packet[UDP].sport} > {packet[UDP].dport}")
        print("")  # Line break for readability
```

Use code with caution.

Identify Suspicious Activity:

Analyze the printed output for patterns that might indicate a threat, such as:

Unusual ports: Ports not commonly used for standard services (e.g., port 22 for SSH, port 80 for HTTP) could be suspicious.

Frequent scans or probes: A lot of packets from the suspicious IP scanning your network for open ports might be a red flag.

Large data exfiltration: Packets with large payloads going outwards could suggest data theft.

```
# Load captured traffic
filename = "suspicious_traffic.pcap"
packets = rdpcap(filename)

# Filter packets by suspicious IP address
suspicious_ip_address = "10.0.0.10"
suspicious_packets = [packet for packet in packets if packet[IP].src ==
suspicious_ip_address]

# Analyze packet details
for packet in suspicious_packets:
    print(f"Source: {packet[IP].src}, Destination: {packet[IP].dst}")
    print(f"Protocol: {packet[IP].proto}")
    if packet.haslayer(TCP):
        print(f"  - Port: {packet[TCP].sport} > {packet[TCP].dport}")
    elif packet.haslayer(UDP):
        print(f"  - Port: {packet[UDP].sport} > {packet[UDP].dport}")
    print("")  # Line break for readability
```

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

Answer-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

Website Analysis Simulation:

By giving two URLs as input—one legitimate and the other phishing—we will simulate website analysis. Next, we'll examine attributes such as:

The SSL certificate is present: Secure communication on legitimate websites is usually achieved through HTTPS.

Features of URLs: Phishing URLs may have odd subdomains, typos, or suspicious characters.

Content of the website: Phishing websites frequently use urgency techniques, grammatical errors, or call on users to take immediate action.

Code:

```
def is_phishing_website(url):
  """
```

```
  Analyzes website characteristics and assigns a score based on phishing
indicators.

  Args:
      url: The URL of the website to be analyzed.

  Returns:
      A score between 0 (likely legitimate) and 10 (highly suspicious).
  """
  score = 0

  # Check for HTTPS (encrypted communication)
  if not url.startswith("https://"):
    score += 3  # Higher score for non-HTTPS

  # Analyze URL for suspicious characters, typos, or subdomains
  if any(char in url for char in ["!", "%", "^", "&"]):
    score += 2
  if "/" in url and url.split("/")[0] != url.split("/")[-1]:  # Check for
unusual subdomains (heuristic)
    score += 1

  # Simulate content analysis
  # This part is for educational purposes only and doesn't access real
websites
  content = "This is an example website. It is secure and legitimate."  #
Replace with simulated content
  if "urgent" in content.lower() or "click now" in content.lower():
    score += 4  # Penalize urgency or pressure tactics

  return score

# Example usage (replace URLs with yours)
real_url = "https://www.google.com/"
phishing_url = "http://not-google.com/login"  # Simulate a phishing URL

real_score = is_phishing_website(real_url)
phishing_score = is_phishing_website(phishing_url)

print(f"Real website score: {real_score} (low score indicates likely
legitimate)")
print(f"Phishing website score: {phishing_score} (high score indicates
suspicion)")

if phishing_score > real_score:
```

```
  print("The second website is more likely to be a phishing attempt!")
else:
  print("The results are inconclusive. More analysis might be needed.")
```

Verifying the authenticity of a website is essential. One way to do this is by using the website verification tool provided at https://www.getsafeonline.org/checkawebsite/. This tool allows users to assess the credibility of a website by scrutinizing various factors such as its security features, domain registration details, and reputation.

Identifying a website's IP address can be crucial for security purposes. Utilize the Website to IP Lookup service available at nslookup.io to determine the specific IP address associated with a website. Understanding the IP address provides insights into the website's hosting environment and aids in detecting potential threats or suspicious activities.

The NATIONAL CYBERCRIME REPORTING PORTAL [NCRP] at

https://i4c.mha.gov.in/ncrp.aspx serves as a centralized platform for reporting cybercrimes. By accessing this portal, individuals can submit complaints related to various cyber offenses, contributing to the efforts in combating cybercrime and ensuring a safer online environment for all users.

Safeguarding against fraudulent payments during online transactions is paramount. The Fraud Payment Check website, accessible at https://www.lookout.com/life/fr

ee-online-shopping-checker, offers a valuable resource to verify the legitimacy of payment platforms. By utilizing this service, users can mitigate the risks associated with online shopping and protect their financial information from potential scams or unauthorized transactions.

Assessing the reputation of a website is integral to ensuring a secure browsing experience. The Website Reputation Checker provided at https://www.urlvoid.com

enables users to evaluate the trustworthiness of a website by analyzing its reputation across multiple sources. This comprehensive assessment aids in making informed decisions regarding website interactions and mitigating the likelihood of encountering malicious content or fraudulent activities.

Prioritizing website security is imperative for safeguarding sensitive information. Explore the SSL Tools available at https://www.ssltrust.in/ssl-tools/website-security-check to

assess and enhance the security measures implemented on a website. These tools facilitate the evaluation of SSL certificates, encryption protocols, and vulnerability scans, empowering website administrators to bolster their defenses against cyber threats and maintain a secure online environment for visitors.