SCHOOL OF CONTINUING AND DISTANCE EDUCATION

JAWAHARLAL NEHRU TECHNOLOGICALUNIVERSITY - HYDERABAD

Kukatpally, Hyderabad – 500 085, Telangana, India.

SIX MONTH ONLINE CERTIFICATE COURSES – 2023

CYBER SECURITY - ASSIGNMENT - 08

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

Ans:

```
from scapy.all import *
# Define a function to analyze packets
def analyze_packets(packet):
    if IP in packet:
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst
        protocol = packet[IP].proto
        print(f"Source IP: {ip_src}, Destination IP: {ip_dst}, Protocol: {protocol}")
# Sniff packets and apply the analyze_packets function
sniff(prn=analyze_packets, count=10)  # Adjust count as needed
```

Scapy is a powerful Python library that allows for the manipulation of network packets at a low level. It can be used to create, manipulate, decode, and analyze network packets. Here's a detailed explanation of how Scapy can be utilized to capture and dissect network packets:

Capturing Packets:

Scapy provides a function called sniff() to capture packets from the network interface. This function allows you to specify various parameters such as the number of packets to capture, a filter to apply to captured packets, and a callback function to process each captured packet.

Dissecting Packets:

Once packets are captured, Scapy provides a wide range of protocols and layers that can be dissected and analyzed. For example, you can access and extract

information from the Ethernet, IP, TCP, UDP, ICMP, DNS, and other layers of the packet.

Crafting Packets:

Scapy allows you to create custom packets by specifying the values for various fields in different protocol layers. This is useful for testing network protocols, creating custom network tools, or even crafting network attacks (be responsible!).

Manipulating Packets:

Scapy enables you to modify existing packets by changing their fields or adding new layers. This can be useful for testing network devices or analyzing the behavior of network protocols.

Analyzing Packets:

Scapy provides various methods for analyzing packet contents, such as summarizing packets, extracting specific fields, calculating checksums, and even decoding packet payloads.

Some common signs of suspicious or anomalous network behavior that can be identified by analyzing captured packets:

Unusual Source or Destination IP Addresses:

Packets originating from or sent to unexpected IP addresses, especially those associated with known malicious actors or blacklisted addresses, could indicate suspicious activity.

Unusual Protocol Usage:

Detection of unusual protocols or unexpected protocol combinations in the network traffic may indicate malicious activity or misconfiguration.

Large Amounts of Outbound Traffic:

Abnormally high volumes of outbound traffic from a particular host may suggest that it has been compromised and is being used for malicious purposes such as participating in a botnet or launching a DDoS attack.

Unusual Port Activity:

Communication on non-standard or uncommon ports, especially when associated with well-known services or protocols, could be indicative of suspicious behavior.

Unexpected Protocol Behavior:

Detection of protocol anomalies, such as unusual packet sizes, unexpected sequence numbers, or abnormal handshake patterns, may indicate attempted exploitation or evasion techniques.

Repeated Access Attempts:

Multiple failed login attempts, repeated connection requests, or excessive scanning activity directed at a particular host or network segment may suggest a reconnaissance or intrusion attempt.

Unexpected Payload Content:

Examination of packet payloads for suspicious content, such as unexpected commands, encoded data, or known malware signatures, can help identify malicious activity.

Signs of Data Exfiltration:

Detection of large, uncharacteristic transfers of sensitive data, unusual file types being transmitted, or encrypted communication channels established with suspicious endpoints may indicate data exfiltration attempts.

Spoofed or Altered Packet Headers:

Analysis of packet headers for signs of IP address spoofing, forged source or destination addresses, or manipulated packet attributes can reveal attempts to disguise the origin or nature of network traffic.

Unusual Timing Patterns:

Examination of packet arrival times or frequency of network activity for irregular patterns, such as sudden spikes in traffic or prolonged periods of inactivity, may indicate automated scanning or coordinated attacks.

To mitigate the identified security risks and secure the network against similar threats in the future, here are some recommendations:

Implement Access Controls:

Enforce strong access controls by using firewalls, access control lists (ACLs), and network segmentation to restrict unauthorized access to critical network resources.

Update and Patch Systems Regularly:

Keep all systems, including servers, routers, switches, and endpoints, up to date with the latest security patches and software updates to address known vulnerabilities and minimize the risk of exploitation.

Deploy Intrusion Detection and Prevention Systems (IDPS):
Implement IDPS solutions to monitor network traffic in real-time, detect suspicious or malicious activity, and automatically block or mitigate potential threats before they can cause harm.

Enable Network Logging and Monitoring:
Enable comprehensive logging and monitoring of network traffic, system events, and security-related activities to facilitate timely detection, analysis, and response to security incidents.

Educate and Train Employees:
Provide regular cybersecurity awareness training to employees to raise awareness of common security threats, best practices for safe computing, and procedures for reporting suspicious activity or incidents.

Use Strong Authentication Mechanisms:
Implement multi-factor authentication (MFA) and strong password policies to enhance the security of user accounts and prevent unauthorized access to sensitive information.

Encrypt Network Traffic:
Use encryption protocols such as SSL/TLS for securing sensitive data in transit and implementing virtual private networks (VPNs) to create secure communication channels for remote access and data exchange.

Perform Regular Security Audits and Assessments:
Conduct periodic security audits and vulnerability assessments to identify and remediate security weaknesses, misconfigurations, and compliance gaps within the network infrastructure.

Establish Incident Response Procedures:
Develop and document incident response procedures to guide the response to security incidents, including the containment, eradication, and recovery processes, and ensure that all relevant stakeholders are aware of their roles and responsibilities.

Engage with Security Experts:
Consider engaging with cybersecurity experts, consultants, or managed security service providers (MSSPs) to perform security assessments, provide guidance on best practices, and assist in implementing advanced security controls.

2Q) 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.

Ans;

```
import requests
from bs4 import BeautifulSoup
import re
def fetch_html(url):
    try:
        response = requests.get(url)
        if response.status_code == 200:
            return response.text
        else:
            return None
    except:
        return None
def extract_features(html):
    features = {}
    if html is None:
        return features

    soup = BeautifulSoup(html, 'html.parser')

    # Count number of links
    links = soup.find_all('a')
    features['num_links'] = len(links)

    # Count number of forms
    forms = soup.find_all('form')
    features['num_forms'] = len(forms)

    # Check for presence of JavaScript
```

```
    scripts = soup.find_all('script')
    features['has_js'] = len(scripts) > 0

    # Check for presence of 'submit' button in forms
    submit_buttons = soup.find_all('input', {'type': 'submit'})
    features['has_submit_button'] = len(submit_buttons) > 0

    # Check for presence of URL in action attribute of forms
    for form in forms:
        action = form.get('action')
        if action and re.match(r'^https?://', action):
            features['form_has_https_action'] = True
            break
    else:
        features['form_has_https_action'] = False
  return features
def analyze_phishing(features):
    # Phishing detection rules can be implemented here
    # For demonstration, we'll just use a simple heuristic
    if features['has_js'] or features['num_links'] > 10:
        return True
    else:
        return False
def main():
    url = input("Enter the URL to analyze: ")
    html = fetch_html(url)
    if html:
        features = extract_features(html)
        is_phishing = analyze_phishing(features)
        if is_phishing:
            print("Warning: The website is likely a phishing site!")
        else:
            print("The website seems safe.")
    else:
```

```
        print("Failed to fetch HTML from the provided URL.")

if __name__ == "__main__":
    main()
```