Assignment – 8

Gumma V L Prasad
(H.T.No: 2406CYS107)

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 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

Answer :

1. Explanation of Using Scapy to Capture and Dissect Network Packets

Scapy is a powerful Python library used for network packet manipulation, including crafting, sending, sniffing, and dissecting network packets. Here's how it can be utilized:

- Packet Sniffing: Capturing packets on the network interface for analysis.
- Packet Dissection: Breaking down the captured packets to analyze each layer and its respective fields.
- Protocol Support: Understanding and dissecting various network protocols (TCP, UDP, ICMP, etc.).

2. Step-by-Step Breakdown of Capturing and Analyzing Network Traffic

Step 1: Install Scapy

```
pip install scapy
```

Step 2: Import Scapy and Start Capturing Packets

```
from scapy.all import sniff, IP

# Define the suspicious IP address
suspicious_ip = '192.168.1.100'

# Function to handle each captured packet
def packet_callback(packet):
    if IP in packet:
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst
        if ip_src == suspicious_ip or ip_dst == suspicious_ip:
            print(packet.summary())

# Capture packets (e.g., for 60 seconds)
packets = sniff(prn=packet_callback, timeout=60)
```

Step 3: Analyze the Captured Packets

```
from scapy.all import rdpcap

# Read the captured packets from a file (if saved)
packets = rdpcap('captured_packets.pcap')

for packet in packets:
    if IP in packet:
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst
        if ip_src == suspicious_ip or ip_dst == suspicious_ip:
            print(packet.show())
```

3. Identification and Interpretation of Suspicious Behavior

While inspecting the packet details:

- Unusual Traffic Patterns: High volume of traffic from/to the suspicious IP.
- Unusual Ports: Communication over uncommon ports that might be used for malicious purposes.
- Payload Analysis: Anomalous payload content indicating potential exploitation or data exfiltration.

4. Recommendations for Mitigating Identified Risks

- Implement Network Segmentation: Restrict traffic flow between sensitive parts of the network.

- Intrusion Detection and Prevention Systems (IDPS): Deploy systems to detect and block suspicious activities.
- Regular Network Audits: Conduct periodic reviews of network traffic and configurations.
- Access Controls: Ensure strict access controls and monitor user activities.
- Patch Management: Keep systems updated to mitigate vulnerabilities.

Code: Network Packet Analysis with Scapy

```
from scapy.all import sniff, IP

# Define the suspicious IP address
suspicious_ip = '192.168.1.100'

# Function to handle each captured packet
def packet_callback(packet):
    if IP in packet:
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst
        if ip_src == suspicious_ip or ip_dst == suspicious_ip:
            print(packet.summary())
            # Optionally save to a file for later analysis
            with open('captured_packets.pcap', 'ab') as f:
                f.write(bytes(packet))

# Capture packets (e.g., for 60 seconds)
packets = sniff(prn=packet_callback, timeout=60)

# Optionally, read captured packets from a file for further analysis
from scapy.all import rdpcap

packets = rdpcap('captured_packets.pcap')

for packet in packets:
    if IP in packet:
        ip_src = packet[IP].src
        ip_dst = packet[IP].dst
        if ip_src == suspicious_ip or ip_dst == suspicious_ip:
            packet.show()
```

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:

To develop a phishing website detection system using Python, we'll follow these steps:

1. Collect Data: Accept two URLs, one legitimate and one phishing.
2. Extract Features: Analyze various characteristics of the websites.
3. Build a Model: Use the extracted features to determine the likelihood of a website being a phishing site.

Procedure

1. Accept URLs

We will take two URLs from the user, one real and one phishing.

2. Analyze Data from Websites

Extract features that are indicative of phishing websites. Common features include:

- URL length
- Presence of special characters
- Use of HTTPS
- Domain age
- IP address usage
- Number of external links

3. Identify the Phishing Site

Use a rule-based approach to identify phishing characteristics or employ a simple machine learning model for classification.

Expected Code

Here's a Python script to perform phishing website detection:

Step 1: Import Required Libraries

```
import requests
from urllib.parse import urlparse
import whois
from datetime import datetime
from sklearn.ensemble import RandomForestClassifier
import numpy as np
```

Step 2: Define Feature Extraction Functions

```
def get_domain_age(domain_name):
    try:
        whois_info = whois.whois(domain_name)
        creation_date = whois_info.creation_date
        if isinstance(creation_date, list):
            creation_date = creation_date[0]
        age = (datetime.now() - creation_date).days
        return age
    except:
        return -1

def extract_features(url):
    features = []

    # URL Length
    features.append(len(url))

    # Presence of special characters
    special_characters = ['@', '-', '_', '.', '~', '%']
    features.append(sum([1 for char in url if char in special_characters]))

    # Use of HTTPS
    features.append(1 if urlparse(url).scheme == 'https' else 0)

    # Domain age
    domain = urlparse(url).netloc
    features.append(get_domain_age(domain))

    # Number of external links (dummy feature for example)
    try:
        response = requests.get(url)
        features.append(response.text.count('href="http'))
    except:
        features.append(-1)

    return features
```

Step 3: Accept URLs and Extract Features

```
legit_url = input("Enter a legitimate URL: ")
phishing_url = input("Enter a phishing URL: ")

legit_features = extract_features(legit_url)
```

```
phishing_features = extract_features(phishing_url)

print("Legit URL Features:", legit_features)
print("Phishing URL Features:", phishing_features)
```

Step 4: Train a Simple Model (Optional)

For a more robust solution, we can use a pre-trained model. Here, we demonstrate a simple training setup using the extracted features:

```
# Example data - In practice, this would be a larger, pre-labeled dataset
X_train = [
    legit_features,
    phishing_features
]
y_train = [0, 1]  # 0 = Legitimate, 1 = Phishing

# Train a simple Random Forest Classifier
clf = RandomForestClassifier()
clf.fit(X_train, y_train)

# Predict on new data
new_url = input("Enter a URL to check: ")
new_features = extract_features(new_url)
prediction = clf.predict([new_features])[0]

if prediction == 1:
    print("The URL is likely a phishing site.")
else:
    print("The URL is likely legitimate.")
```

Conclusion

This script provides a basic framework for detecting phishing websites using URL features. For real-world applications, you should expand the feature set, use a larger labeled dataset for training, and consider more sophisticated models and techniques.