6.3.3. Практическое задание: реализация инструмента для обнаружения сетевого сниффинга

Цель: написать скрипт на Python, который будет мониторить сеть и обнаруживать подозрительную активность, связанную со сниффингом сетевого трафика.

Шаги выполнения

1. Изучение сетевых протоколов и методов сниффинга

Основы сетевых протоколов:

- Ethernet – основной протокол канального уровня дл я проводных сетей

- IP (Internet Protocol) – протокол сетевого уровня, который обеспечивает маршрутизацию данных между устройствами

- TCP/UDP – протоколы транспортного уровня, обеспечивающие передачу данных между приложениями

Методы сниффинга:

- Пассивный сниффинг – перехват трафика без изменения его содержания (используется в коммутируемых сетях)

- Активный сниффинг – включает атаки, такие как ARP-спуфинг, для перенаправления трафика через устройство злоумышленника

Инструменты для сниффинга:

Wireshark – популярный инструмент для анализа сетевого трафика

Tepdump – командная строка для захвата и анализа сетевого трафика

Ettercap – инструмент для выполнения активного сниффинга

1. Выбор библиотек и инструментов

Scapy – библиотека для создания, отправки, захвата и анализа сетевых пакетов

Psutil – библиотека для получения информации о сети и системе.

1. Написание кода для мониторинга сети

From scapy.all import sniff

import psutil

import logging

logging.basicConfig(level=logging.INFO, format=’%(asctime)s-%(message)s’)

#Функция для анализа пакетов

def packet\_callback(packet):

if packet.haslayer(“IP”):

src\_ip = packet[“IP”].src

dst\_ip = packet[“IP”].dst

logging.info(f”Packet: {src\_ip}-> {dst\_ip}”)

#Функция для мониторинга сети

def monitor\_network(interface):

logging.info(f”Starting network monitoring on {interface}”)

sniff(iface=interface, prn=packet\_callback, store=False)

if \_\_name\_\_ == “\_\_main\_\_”:

network\_interface = psutil.net\_if\_addrs().keys()

# Получение списка сетевых интерфейсов

Monitor\_netwotk(network\_interface[0])

#Используйте нужный интерфейс

1. Обработка и анализ сетевых пакетов.

Анализ пакетов:

- Используйте Scapy для захвата и анализа пакетов

- Изучите заголовки пакетов, чтобы определить типы трафика и выявить аномалии

Пример анализа пакетов:

From scapy.all import ARP, sniff

#Функция для анализа ARP-пакетов

def detect\_arp\_spoof(packet):

if packet.haslayer(ARP) and packet[ARP].op ==2:

try:

real\_mac = ip\_mac\_mapping[packet[ARP].psrc]

response\_mac = packet[ARP].hwsrc

if real\_mac != response\_mac:

logging.warning(f”ARP Spoofing detected:

{packet[ARP}.psrc) is being sppofed. Real MAC: {real\_mac}, Fake MAC: {response\_mac}”)

except KeyError:

ip\_mac\_mapping[packet[ARP].psrc] = packet[ARP].hwsrc

sniff(store=False, prn=detect\_arp\_spoof, iface=interface)

1. Выявление аномалий и создание уведомлений

Механизмы выявления аномалий

- Создайте функции для прокерки частоты отправки пакетов с одного IP-адреса

- Отслеживайте повторяющиеся запросы на определенные порты

Создание уведомлений:

- Логи в файл

- Отправка email-уведомлений при обнаружении подозрительной активности

Пример отправки email-уведомления

Import smtlib

From email.mime.text import MIMEText

def send\_alert(message):

msg = MIMEText(message)

msg[“Subject“] = “Network Alert”

msg[“From”] = [your\_email@xample.com](mailto:your_email@xample.com)

msg[“To”] = [admin@example.com](mailto:admin@example.com)

with smtplib.SMTP(“smtp.example.com”, 587) as server:

server.starttls()

server.login([your\_email@example.com](mailto:your_email@example.com), “ypur\_password”)

server.sendmail([your\_email@example.com](mailto:your_email@example.com), [admin@example.com](mailto:admin@example.com), msg.as\_string())

#Используйте функцию send\_alert в анализе пакетов

If real\_mac != response\_mac:

message = f”ARP Spoofing detected: {packet[ARP].psrc} is neing spoofed. Real MAC: {real\_mac},

Fake Mac: {response\_mac}”

logging.warning(message)

send\_alert(message)

1. Тестирование и отладка

- Используйте инструменты для генерации сниффинга, такие как Ettercap, чтобы проверить, может ли ваш скрипт обнаружить атаки.

-Проведите тесты в различных сетевых средах, чтобы убедиться в надежности скрипта

1. Улучшение и оптимизация

Оптимизация:

- Улучшите алгоритмы обнаружения аномалий

-Добавьте больше типов обнаруживаемых атак, таких как DNS-спуфинг или DoS-атаки

Пример оптимизированного кода:

from scapy.all import sniff, ARP

import logging

import amtplib

from email.mime.text import MIMEText

logging.basicConfig(level=logging.INFO, format=’%(asctime)s-%(message)s’)

ip\_mac\_mapping={}

def send\_alert(message):

msg = MIMEText(message)

msg[“Subject”] = “Network Alert”

msg[“From”] = [your\_email@example.com](mailto:your_email@example.com)

msg[“To”] = [admin@example.com](mailto:admin@example.com)

with smtplib.SMTP(“smtp.example.com”,587) as server:

server.starttls

server.login([your\_email@example.com](mailto:your_email@example.com), “your\_password”)

server.sendmail([your\_email@example.com](mailto:your_email@example.com), [admin@example.com](mailto:admin@example.com), msg.as\_string())

def detect\_arp\_spoof(packet):

if packet.haslayer(ARP) and packet[ARP].op == 2:

real\_mac = ip.mac\_mapping.get(packet[ARP].psrc)

response\_mac = packet[ARP].hwsrc

if real\_mac and ral\_mac != response\_mac:

message = f “ARP Spoofing detected: {packet[ARP].psrc} is being spoofed. Real MAC: {real\_mac}, Fake MAC: {response.\_mac}”

logging.warning(message)

send\_alert(message)

else:

ip\_mac\_mapping[packet[ARP].psrc]] = response\_mac

def monitor\_network(interface):

logging.info(f”Starting ARP Spoofing detection on {interface}”)

sniff(store=False, prn=detect\_arp\_spoof, iface=interface)

if \_\_name\_\_ == “\_\_main\_\_”:

network\_interface = “eth0” #Замените на нужный интерфейс

monitor\_network(network\_interface)