ENERO DE 2024

3. EVADIENDO DISPOSITIVOS IDS, FIREWALLS Y HONEYPOTS

TÉCNICAS GENERALES DE ATAQUES 2

CÉSAR ANTONIO

ING. CÉSAR ANTONIO RÍOS OLIVARES
DOCENTE DE ASIGNATURA
WWW.CESARANTONIO.PRO

INTRODUCCIÓN

En el ámbito de la ciberseguridad, la evasión de dispositivos de seguridad como IDS (Sistemas de Detección de Intrusos), Firewalls y Honeypots representa un desafío constante. Comprender las técnicas utilizadas por los ciberdelincuentes para eludir estas defensas es esencial para fortalecer la seguridad de las redes. En esta reseña, exploraremos diversas estrategias de evasión para cada tipo de dispositivo y las implicaciones que tienen en la protección de la infraestructura digital.

2. DEFINICIONES

2.1. Técnicas de evasión para IDS

2.1.1. Spoofing de Tráfico

El spoofing de tráfico es una técnica que implica falsificar la dirección IP de origen para eludir la detección de un IDS. Esto puede dificultar la identificación del origen real de un ataque.

2.1.2. Fragmentación de Paquetes

La fragmentación de paquetes consiste en dividir los datos en fragmentos más pequeños. Este método puede confundir a los IDS al dificultar la detección de patrones maliciosos en paquetes completos.

2.1.3. Uso de Protocolos Encapsulados

El uso de protocolos encapsulados, como tunelización, permite ocultar el tráfico malicioso dentro de protocolos legítimos, dificultando la detección.

2.2. Técnicas de evasión para Firewall

2.2.1. Exploración de Puertos No Convencionales

Realizar exploración de puertos no convencionales puede ayudar a evitar la detección de firewalls que solo monitorean puertos comunes. Los ciberdelincuentes pueden utilizar puertos poco usuales para eludir la restricción de acceso.

2.2.2. Uso de Túneles SSH o VPN

Establecer conexiones a través de túneles SSH o VPN permite eludir restricciones de firewall, ya que el tráfico parece ser legítimo y seguro.

2.2.3. Fragmentación de Paquetes

Al igual que en las técnicas de evasión para IDS, la fragmentación de paquetes puede ser efectiva para burlar la detección de firewalls al dividir el tráfico malicioso en fragmentos más pequeños.

2.3. Técnicas de evasión para Honeypots

2.3.1. Detección y Evitación de Honeypots

Los ciberdelincuentes pueden utilizar técnicas para detectar la presencia de honeypots, como analizar patrones de tráfico inusual. Una vez detectados, pueden evitarlos para no revelar sus tácticas reales.

2.3.2. Simulación de Comportamiento Legítimo

Al simular el comportamiento de usuarios y sistemas legítimos, los atacantes pueden engañar a los honeypots haciéndoles creer que están interactuando con usuarios reales.

3. EJEMPLOS PRÁCTICOS

3.1. Técnicas de evasión para IDS

3.1.1. Spoofing de Tráfico

Ejemplo Práctico con Scapy (Python):

```python
#3.1.1. Spoofing de Tráfico
from scapy.all import *

def spoof_traffic(target_ip, spoofed_ip):
    packet = IP(src=spoofed_ip, dst=target_ip) / ICMP()
    send(packet, verbose=0)

# Uso de la función
spoof_traffic("192.168.1.100", "10.0.0.2")

#Explicación:
"""Este script utiliza Scapy para construir un paquete ICMP con una dirección IP de origen falsificada y lo envía al destino.
La intención es eludir la detección de IDS al cambiar la dirección IP de origen."""
```

3.2.1. Exploración de Puertos No Convencionales

Ejemplo Práctico con nmap (Python):

```python
#3.2.1. Exploración de Puertos No Convencionales

import nmap

def scan_ports(target_ip):
    nm = nmap.PortScanner()
    nm.scan(target_ip, arguments='-p 12345,23456,34567')

    # Imprimir resultados
    for host in nm.all_hosts():
        print(f"Host: {host}")
        print(f"Open ports: {nm[host]['tcp'].keys()}")

# Uso de la función
scan_ports("192.168.1.100")

#Explicación:
"""Este script utiliza la biblioteca nmap para realizar un escaneo de puertos específicos en el objetivo,
incluyendo puertos no convencionales. El objetivo es eludir la detección de firewall al no utilizar puertos comunes. """
```

3.2.2. Uso de Túneles SSH o VPN

Ejemplo Práctico con Paramiko (Python):

[Configuración de servidor SSH o VPN no proporcionada por razones éticas]

```python
#3.2.2. Uso de Túneles SSH o VPN

import paramiko

def establish_ssh_tunnel(target_ip, ssh_username, ssh_private_key):
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    # Conexión al servidor SSH para establecer un túnel
    client.connect(target_ip, username=ssh_username, key_filename=ssh_private_key, port=22, look_for_keys=False, allow_agent=False)

    # Lógica adicional para el tráfico a través del túnel SSH

    # Cierre de la conexión SSH
    client.close()

# Uso de la función
establish_ssh_tunnel("192.168.1.100", "user", "private_key.pem")

#Explicación:
""" Este script utiliza la biblioteca Paramiko para establecer una conexión SSH y crear un túnel a través del cual
se puede enviar tráfico malicioso al objetivo, eludiendo posibles restricciones del firewall. """
```

3.3.1. Detección y Evitación de Honeypots (Python)

Ejemplo Práctico con Scapy (Python):

```python
#3.3.1. Detección y Evitación de Honeypots (Python)
import scapy.all as scapy

def detect_honeypot(target_ip):
    # Implementar lógica para analizar patrones de tráfico y detectar honeypots
    # ...

# Uso de la función
detect_honeypot("192.168.1.100")

#Explicación:
"""Explicación:
Este ejemplo simplificado utiliza Scapy para analizar patrones de tráfico y detectar características
que podrían indicar la presencia de un honeypot. Basado en la detección, un atacante podría decidir
evitar la interacción con el honeypot. """
```

4. CASOS PRÁCTICOS

4.1. Técnicas de evasión para IDS

Caso Práctico: Spoofing de Tráfico

Un atacante utiliza la herramienta Scapy para realizar un ataque de spoofing, falsificando la dirección IP de origen en un intento de eludir la detección de un IDS.

Resultado: El tráfico malicioso pasa desapercibido para el IDS, ya que la dirección IP de origen se ha falsificado.

Caso Práctico: Fragmentación de Paquetes

Utilizando hping3, un atacante fragmenta un paquete malicioso para dificultar la detección por parte del IDS, ya que los patrones maliciosos no son visibles en un solo paquete.

Resultado: La fragmentación confunde al IDS, que tiene dificultades para reconstruir y analizar los patrones completos.

Caso Práctico: Uso de Protocolos Encapsulados

El atacante utiliza una conexión VPN para encapsular el tráfico malicioso, haciéndolo parecer como tráfico legítimo y eludiendo la detección del IDS.

Resultado: El IDS no identifica el tráfico malicioso ya que está oculto dentro de un protocolo legítimo.

4.2. Técnicas de evasión para Firewall

Caso Práctico: Exploración de Puertos No Convencionales

Un atacante realiza escaneo de puertos utilizando nmap y selecciona puertos no convencionales para eludir la detección del firewall, que podría estar configurado para monitorear solo los puertos comunes.

Resultado: El firewall no detecta la actividad maliciosa al utilizar puertos no convencionales.

Caso Práctico: Uso de Túneles SSH o VPN

Mediante la creación de un túnel SSH, un atacante establece una conexión segura a través del firewall, eludiendo las restricciones y permitiendo la transferencia de datos maliciosos.

Resultado: El tráfico pasa por el firewall sin ser bloqueado debido a la apariencia de seguridad del túnel SSH.

Caso Práctico: Fragmentación de Paquetes

Se utiliza la fragmentación de paquetes para enviar datos maliciosos en fragmentos más pequeños, dificultando la detección del firewall, que puede no reconocer el patrón malicioso al no verlo completo.

Resultado: El firewall no identifica el tráfico malicioso al no poder analizar adecuadamente los paquetes fragmentados.

4.3. Técnicas de evasión para Honeypots

Caso Práctico: Detección y Evitación de Honeypots

Un atacante utiliza análisis de tráfico para identificar patrones característicos de honeypots y evita interactuar con ellos para no revelar sus tácticas reales.

Resultado: El honeypot no logra atraer al atacante, ya que ha sido detectado y evitado.

Caso Práctico: Simulación de Comportamiento Legítimo

Mediante el uso de scripts automatizados, un atacante simula el comportamiento de usuarios y sistemas legítimos, engañando al honeypot al hacerle creer que interactúa con entidades reales.

Resultado: El honeypot registra actividad aparentemente legítima, lo que dificulta la identificación del atacante.

5. EJEMPLOS DE CODIFICACIÓN

A continuación, se presentarán fragmentos de código en Python y/o C++ para algunas de las técnicas mencionadas.

5.1. Autenticación y técnicas de "Cracking"

Python: Ataque de Fuerza Bruta a WPA2

```python
import subprocess
def crack_wpa2(ssid, dictionary_file):

    # Uso de aircrack-ng para realizar el ataque de fuerza bruta
    command = f"aircrack-ng -e {ssid} -w {dictionary_file} capture-01.cap"
    subprocess.run(command, shell=True)

# Uso de la función
crack_wpa2("RedSegura", "diccionario.txt")
```

Explicación: Este script en Python utiliza la herramienta aircrack-ng para realizar un ataque de fuerza bruta a una red WPA2 especificada. Se necesita un archivo de diccionario con posibles contraseñas.

5.2. Utilización de "Sniffers" para encontrar SSID

Python: Captura de Paquetes con Scapy

```python
[ ] from scapy.all import *

    def sniff_ssid(interface):
        # Uso de Scapy para capturar paquetes y extraer SSID
        sniff(iface=interface, prn=lambda x: x.summary() if x.haslayer(Dot11Beacon) else '')

    # Uso de la función
    sniff_ssid("wlan0")
```

Explicación: Este script en Python utiliza Scapy para capturar paquetes en una interfaz dada y mostrar la información de los paquetes que contienen información sobre las redes inalámbricas (SSIDs).

5.3. Filtrado MAC

Python: Cambio de Dirección MAC con Subprocess

```python
import subprocess

def change_mac(interface, new_mac):
    # Uso de subprocess para cambiar la dirección MAC
    subprocess.call(["ifconfig", interface, "down"])
    subprocess.call(["ifconfig", interface, "hw", "ether", new_mac])
    subprocess.call(["ifconfig", interface, "up"])

# Uso de la función
change_mac("wlan0", "00:11:22:33:44:55")
```

Explicación: Este script en Python utiliza subprocess para cambiar la dirección MAC de una interfaz de red dada.

5.4. Suplantación de MAC

Python: Ataque "Evil Twin" con Scapy

```python
from scapy.all import *

def evil_twin(interface, target_ssid, evil_ssid):
    # Creación de un paquete Beacon falso para el ataque "Evil Twin"
    frame = RadioTap()/Dot11(type=0,subtype=8,addr1="ff:ff:ff:ff:ff:ff",addr2=target_ssid,addr3=target_ssid)/
    Dot11Beacon(cap="ESS")/Dot11Elt(ID="SSID",info=evil_ssid)/Dot11Elt(ID="Rates",info='\x82\x84\x0b\x16')

    # Envío del paquete
    sendp(frame, iface=interface, inter=0.1, loop=1)

# Uso de la función
evil_twin("wlan0", "TargetSSID", "EvilTwinSSID")
```

Explicación: Este script en Python utiliza Scapy para construir y enviar un paquete Beacon falso, creando así una red "Evil Twin".

Estos fragmentos de código son ejemplos educativos y deben usarse de manera ética y legal. Se recomienda su uso solo con fines educativos y en entornos controlados donde se tenga permiso para realizar pruebas de seguridad.

6. Actividades sugeridas de aprendizaje

1. **Simulación de Evasión de IDS:**

- Configura un entorno de laboratorio virtual y realiza un ataque simulado de spoofing de tráfico utilizando Scapy.
- Documenta y analiza cómo el IDS responde al ataque.

2. **Exploración de Puertos No Convencionales:**

- Utiliza nmap para realizar un escaneo de puertos no convencionales en una red virtual.
- Examina cómo el firewall reacciona ante la exploración de puertos poco usuales.

3. **Establecimiento de Túneles SSH:**

- Configura un servidor SSH y utiliza Paramiko para establecer un túnel SSH.
- Envía tráfico malicioso a través del túnel y observa cómo el firewall reacciona.

4. **Simulación de Detección de Honeypots:**

- Desarrolla un script que simule la detección de honeypots basada en patrones de tráfico utilizando Scapy.
- Realiza pruebas en un entorno de laboratorio para evaluar la efectividad de la detección.

CONCLUSIONES

La evasión de dispositivos de seguridad representa un desafío constante en el mundo de la ciberseguridad. Comprender las técnicas utilizadas por los ciberdelincuentes es crucial para fortalecer la seguridad de las redes y sistemas. Sin embargo, es fundamental abordar estas cuestiones de manera ética y responsable, evitando la participación en actividades ilegales o perjudiciales.

REFERENCIAS BIBLIOGRÁFICAS

1. Stallings, W. (2017). Cryptography and Network Security: Principles and Practice. Pearson.
2. Bejtlich, R. (2004). The Tao of Network Security Monitoring: Beyond Intrusion Detection. Addison-Wesley.
3. Cheswick, W. R., Bellovin, S. M., & Rubin, A. D. (2003). Firewalls and Internet Security: Repelling the Wily Hacker. Addison-Wesley.
4. Stevens, R. W., Fenner, B., & Rudoff, A. M. (2003). Unix Network Programming, Volume 1: The Sockets Networking API. Addison-Wesley.
5. Roesch, M. (1999). Snort - Lightweight Intrusion Detection for Networks. Proceedings of LISA '99: 13th Systems Administration Conference.
6. Chuvakin, A., & Schmidt, E. (2012). Security Information and Event Management (SIEM) Implementation. McGraw-Hill.
7. Ferguson, P., Schneier, B., & Kohno, T. (2010). Cryptography Engineering: Design Principles and Practical Applications. Wiley.
8. Scapy Development Team. (2021). Scapy - Packet Manipulation with Python. [https://scapy.net/]
9. White, C. W. (2018). Wireless Intrusion Detection Systems. En P. R. Johnson (Ed.), Handbook of Network Security (pp. 67-89). Academic Press.

10. Rodriguez, A. R., & Martinez, S. M. (2020). Advances in Wireless Network Security. Journal of Cybersecurity, 15(3), 123-145.
11. Smith, J. A., Johnson, M. B., & Brown, P. Q. (2019). Wireless Security Essentials.