Documento Oficial de Instalación e Implementación

Sistema de Detección y Prevención de Intrusiones (IDPS) con Telegram Bot & nftables

SISTEMA OPERATIVO	SERVICIO	MÓDULO BLOQUEO	NOTIFICACIONES
Debian 13 (Trixie)	IDPS Automatizado	nftables Framework	Telegram Bot API

1. Resumen Ejecutivo del Proyecto

El presente documento certifica el despliegue del sistema de seguridad defensivo automatizado (IDPS) sobre el servidor Linux Debian 13. El sistema monitorea activamente el tráfico de red en la interfaz de red principal (`enp0s3`), analizando patrones de paquetes TCP mediante la librería **Scapy**.

Al detectar escaneos de puertos no autorizados (p. ej., Nmap SYN Stealth Scans) que superan un umbral predefinido, el motor aplica de forma dinámica reglas de bloqueo mediante **nftables** y emite una alerta crítica en tiempo real al canal de administración en **Telegram**.

> **Criterio de Prevención de Falsos Positivos:** Se implementó un algoritmo basado en ventana temporal y umbral de paquetes (`5 SYNs en 3 segundos`) para asegurar que el tráfico legítimo (conexiones SSH o solicitudes web únicas) y las peticiones ICMP (Ping) no sean bloqueadas de manera improcedente.

2. Arquitectura de Red e Interfaces

Interfaz	Modo de Red	Configuración IP	Propósito Técnico
`lo`	Loopback	`127.0.0.1 /8`	Tráfico interno del sistema local. Excluido del bloqueo.
`enp0s3`	Adaptador Puente / DHCP	Dinámica (Router)	Salida a Internet para la API de Telegram y servicio SSH.
`enp0s8`	Red Interna / Estática	`192.168.1.145 /24`	Interfaz de pruebas / Monitoreo de segmento local.

3. Configuración de Red en Debian (/etc/network/interfaces)

Para asegurar persistencia y doble interfaz (salida a Internet + red estática local), la configuración fue declarada formalmente de la siguiente manera:

```
# /etc/network/interfaces
auto lo
iface lo inet loopback

# Interfaz 1: Internet / Telegram API
auto enp0s3
iface enp0s3 inet dhcp

# Interfaz 2: Red Interna para monitoreo
auto enp0s8
iface enp0s8 inet static
    address 192.168.1.145
    netmask 255.255.255.0
```

4. Configuración del Firewall (nftables)

5. Código Fuente del Detector Automatizado (/usr/local/bin/detector_nmap.py)

Script Python3 de producción con control de umbrales, integración con Telegram API y ejecución de reglas en Kernel mediante `nftables`.

```
import time
from collections import defaultdict
from scapy.all import sniff, TCP, IP
import requests
import subprocess

# --- PARAMETROS DE CONFIGURACION ---
BOT_TOKEN = "TU_BOT_TOKEN_AQUI"        # Reemplazado por seguridad
CHAT_ID = "TU_CHAT_ID_AQUI"            # Reemplazado por seguridad
INTERFACE = "enp0s3"                   # Interfaz monitoreada

UMBRAL_PAQUETES = 5                    # Minimo de paquetes SYN para bloquear
VENTANA_TIEMPO = 3                     # Segundos de la ventana de analisis

historial_syn = defaultdict(list)
ips_bloqueadas = set()

def enviar_telegram(mensaje):
    url = f"https://api.telegram.org/bot{BOT_TOKEN}/sendMessage"
    payload = {"chat_id": CHAT_ID, "text": mensaje, "parse_mode": "Markdown"}
    try:
        requests.post(url, json=payload, timeout=5)
    except Exception as e:
        print(f"[-] Error notificando a Telegram: {e}")

def bloquear_ip(ip):
    if ip in ips_bloqueadas:
        return

    ips_bloqueadas.add(ip)

    # Aplicar regla dinamicamente en nftables
    cmd = f"sudo nft add element ip defensa ips_bloqueadas {{ {ip} }}"
    subprocess.run(cmd, shell=True)

    msg = (f"🚨 *ALERTA DE SEGURIDAD - IDPS*\n\n"
           f"🔍 *Evento:* Escaneo de puertos detectado (Nmap)\n"
           f"🛑 *Accion:* IP Bloqueada automaticamente\n"
           f"🌐 *IP Atacante:* `{ip}`")

    print(f"[!] IP {ip} bloqueada por superar el umbral de escaneo.")
    enviar_telegram(msg)

def procesar_paquete(pkt):
    if not (pkt.haslayer(IP) and pkt.haslayer(TCP)):
        return

    # Analizar unicamente paquetes con Flag TCP SYN activado
    if pkt[TCP].flags == 'S':
        ip_origen = pkt[IP].src
        ahora = time.time()

        # Limpieza de registros fuera de la ventana de tiempo
        historial_syn[ip_origen] = [t for t in historial_syn[ip_origen] if ahora - t < VENTANA_TIEM
PO]
        historial_syn[ip_origen].append(ahora)

        # Verificacion de umbral
```