PROYECTO FINAL DEL CURSO DE REDES Y TRANSMISIÓN DE DATOS

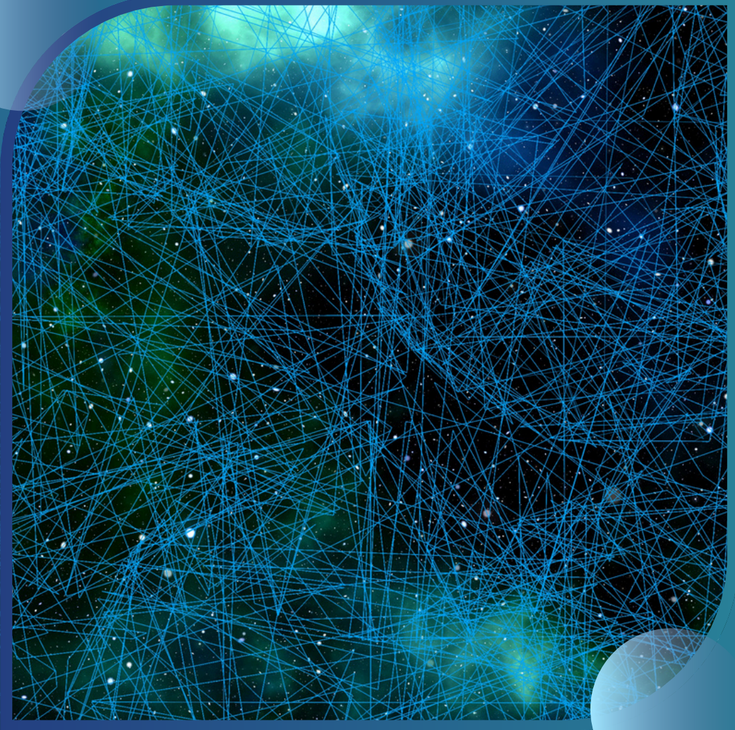
GENERADOR DE PAQUETES CAPAZ DE EVALUAR LOS LÍMITES Y SEGURIDAD DE UNA RED UTILIZANDO GNS3 VM Y SCAPY

GRUPO 3:

- Quispe Cabello, Jose Alessandro
- Sanchez Wong, Jatziry Fernanda

INTRO

El proyecto tiene como objetivo principal diseñar e implementar un generador de paquetes capaz de evaluar la seguridad y el rendimiento de una red. Este sistema permitirá simular distintos escenarios, desde tráfico legítimo de alta carga hasta tráfico malicioso, para identificar vulnerabilidades y analizar el comportamiento de dispositivos de red en condiciones diversas.



ANTECEDENTES

La seguridad en redes es esencial ante amenazas crecientes y la complejidad actual. Este trabajo se basa en antecedentes como el uso de herramientas avanzadas, pruebas de estrés y simuladores como GNS3 para analizar vulnerabilidades. Rincón (2021) destaca el hacking ético y estándares como NIST para evaluar redes IPv4, mientras que Hussain et al. (2021) presentan IoT-Flock, una herramienta para simular tráfico y detectar ataques, enfoques relevantes para fortalecer la seguridad en redes.

METODOLOGÍA PROPUESTA

El proyecto se desarrolló en fases que abarcan desde la planificación hasta la documentación de resultados, garantizando un diseño e implementación sistemáticos del generador de paquetes y el análisis del entorno de red. Se incluye el esquema general, los elementos requeridos, las actividades principales y los scripts del sistema.

- Hardware:
  - Una PC o laptop con Oracle VM VirtualBox configurado para alojar la máquina virtual de GNS3.
  - La máquina virtual de GNS3 actuará como la red simulada.
- Software:
  - GNS3 VM en Oracle VM VirtualBox.
  - Wireshark instalado en tu PC anfitriona (Windows) para la captura y análisis de tráfico.
  - Uso de Python en la terminal de Windows (o un entorno de desarrollo como VSCode) para ejecutar los scripts de Scapy.

PROTOCOLO TCP E ICMP: DETALLES Y CONFIGURACIÓN

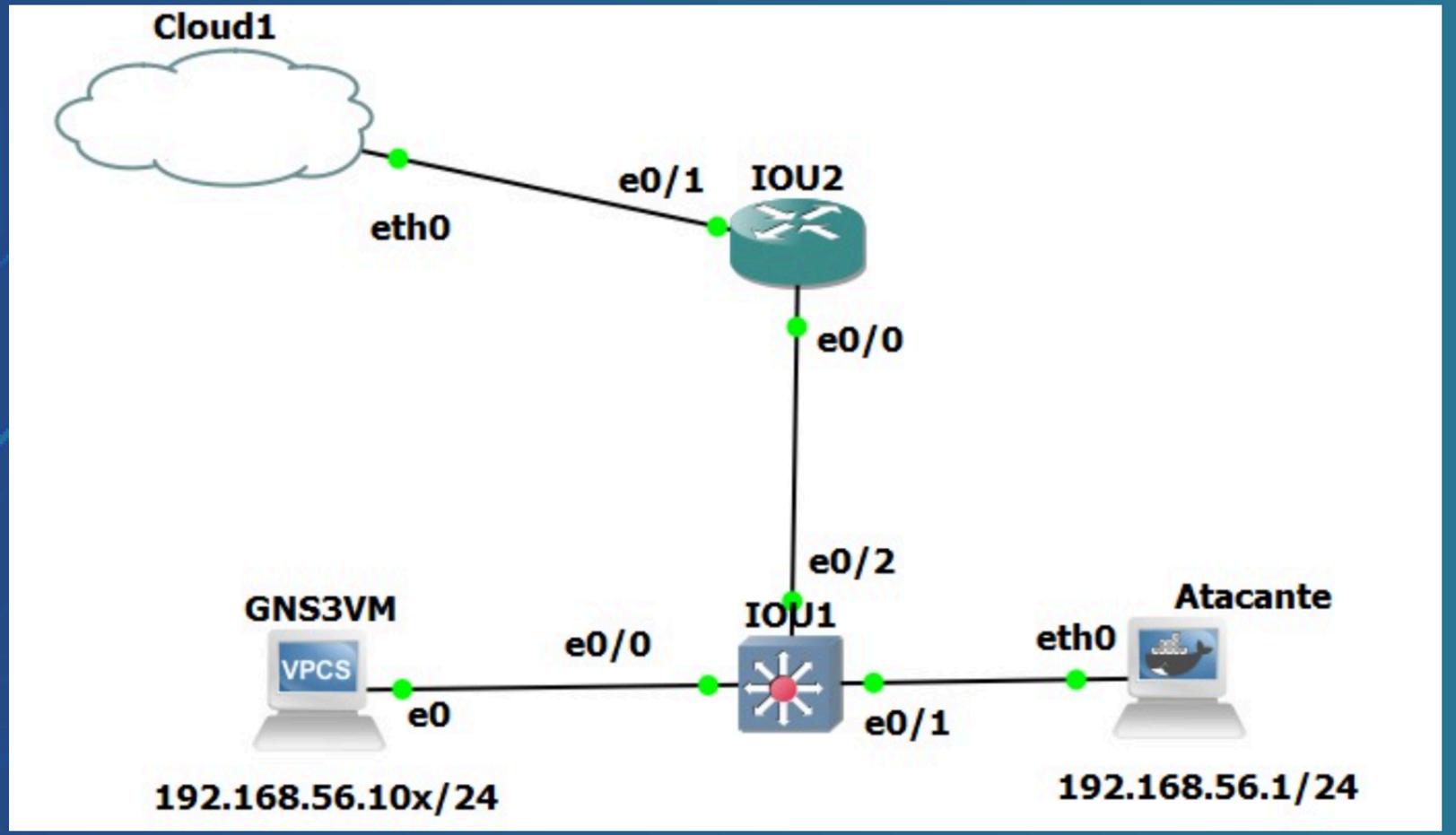
Protocolo ICMP

- Ping básico
  - Tipo: Echo Request (Tipo 8, Código 0)
  - Tamaño: 64-1500 bytes
  - Intervalo: 1 paquete/0.1 s (configurable)
  - Objetivo: Verificar conectividad básica.
- Ping Flood
  - Tipo: Echo Request (512-1024 bytes)
  - Tasa: Hasta 1000 paquetes/s
  - Objetivo: Evaluar tolerancia a alta carga ICMP.

Protocolo TCP

- Tráfico legítimo
  - Tipo: Conexiones estándar (SYN, SYN-ACK, ACK)
  - Puertos: 80 (HTTP), 443 (HTTPS), 22 (SSH)
  - Tasa: 10-100 conexiones/s
  - Objetivo: Medir respuesta de firewalls e IDS/IPS.
- Ataque SYN Flood
  - Tipo: TCP SYN sin completar handshake
  - IP de origen: Aleatoria (simula ataques distribuidos)
  - Tasa: Alta, con intervalos breves
  - Objetivo: Probar mitigación de ataques DoS.

ESQUEMA DE IMPLEMENTACIÓN

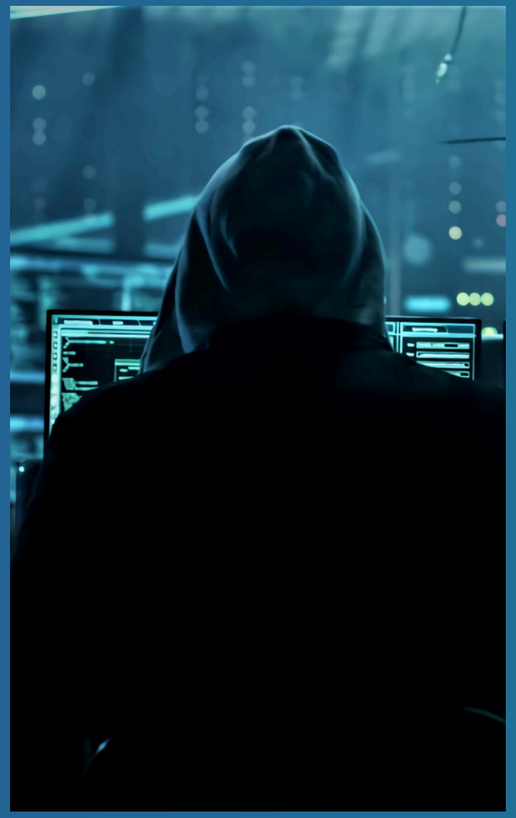


CONFIGURACIÓN DE GRUPOS DE PAQUETES

- **Tráfico ICMP Básico: D**estino: IP del router principal, Tamaño: 64 bytes, Tasa: 1 paquete/segundo.
- **Tráfico ICMP Malicioso:** Destino: Servidor simulado, Tamaño: 1024 bytes, Tasa: 1000 paquetes/segundo.
- **Tráfico TCP Legítimo:** Destino: Servidor simulado, Puertos: 80 y 443, Frecuencia: 10 conexiones/segundo.
- **Ataque SYN Flood:** Destino: Firewall, Puertos: Aleatorios, Frecuencia: 1000 paquetes/segundo.

HERRAMIENTAS Y CONFIGURACIÓN ADICIONAL

- Generación de Tráfico: Scripts en Python utilizando Scapy para personalizar los paquetes.
- Análisis de Tráfico: Wireshark para capturar y analizar la comunicación en tiempo real dentro de la red simulada.
- Sistema Operativo: Las máquinas virtuales utilizan sistemas Linux por su flexibilidad y compatibilidad con herramientas como Scapy.



FASES Y ACTIVIDADES DEL PROYECTO

Fase	Actividades
Planificación	• Identificación de las limitaciones del entorno y ajustar las herramientas necesarias. • }Configuración de la conexión entre la máquina anfitriona (Windows) y las máquinas virtuales (GNS3 VM o Linux VM).
Diseño del sistema	• Creación de scripts personalizados en Python utilizando Scapy • .Diseño de una red dentro de GNS3 o VirtualBox, asegurando que las máquinas puedan comunicarse.
Implementación	• Configuración de Red: Asignación de IP estáticas y configuración de servicios en máquinas objetivo (e.g., servidor web en Linux). • Generación de Tráfico: Ejecución de scripts Scapy para probar ICMP, Ping Flood, TCP legítimo y SYN Flood. • Captura de Tráfico: Uso de Wireshark o tcpdump para capturar paquetes en tiempo real.
Documentación	• Registro de los resultados obtenidos •).Proposición de recomendaciones basadas en los datos obtenidos.

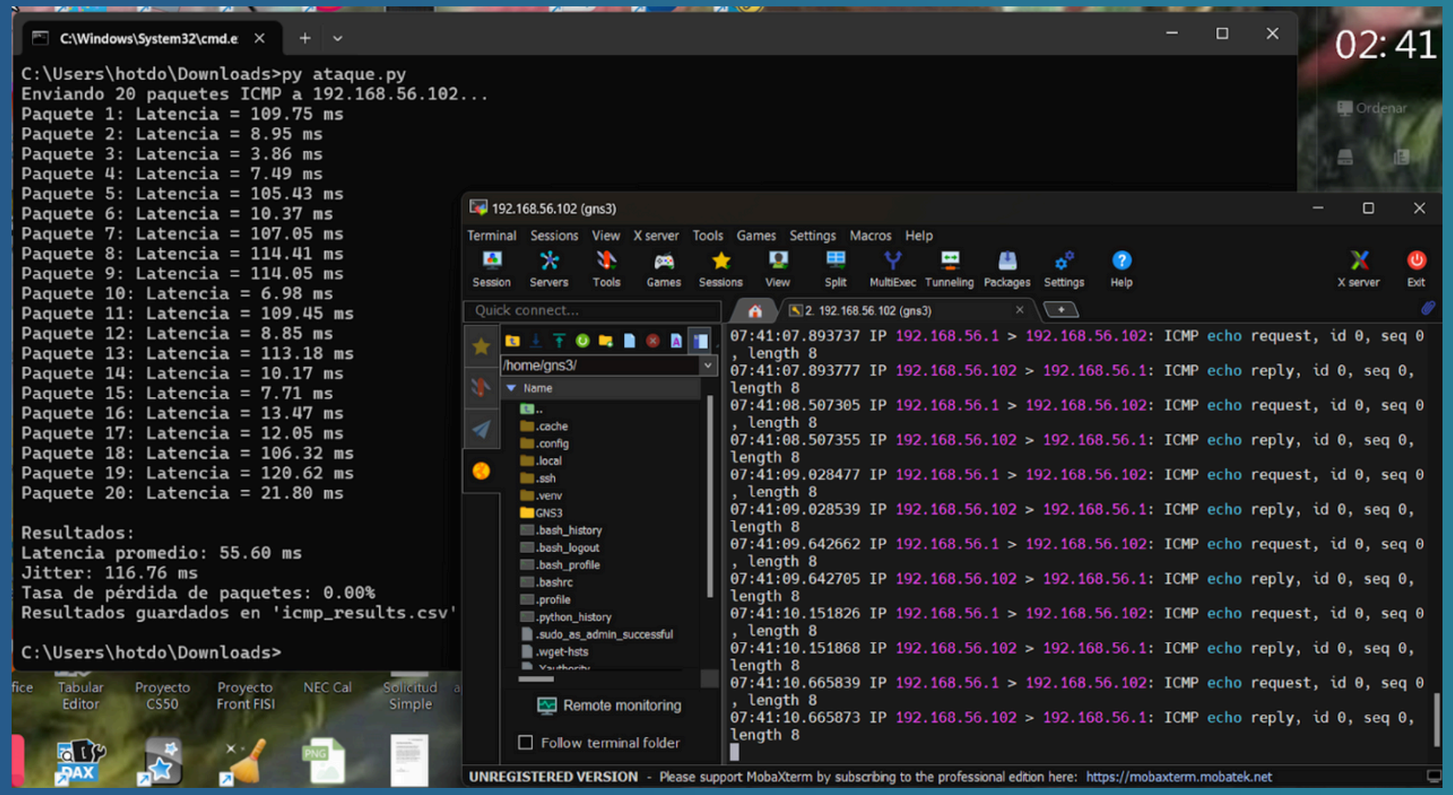
RESULTADOS

PRUEBA 1: GENERADOR ICMP CON MEDICIÓN DE RENDIMIENTO1: GENERADOR ICMP CON MEDICIÓN DE RENDIMIENTOTURE

```
import time
from scapy.all import *
import pandas as pd

def generate_and_measure_icmp(target_ip, count=10, interval=1):
    latencies = []
    lost_packets = 0

    print(f"Enviando {count} paquetes ICMP a {target_ip}...")

    for i in range(count):
        send_time = time.time()
        packet = IP(dst=target_ip) / ICMP()
        reply = sr1(packet, timeout=2, verbose=False)
        receive_time = time.time()

        if reply:
            latency = (receive_time - send_time) * 1000  # Convertir a ms
            latencies.append(latency)
            print(f"Paquete {i+1}: Latencia = {latency:.2f} ms")
        else:
            lost_packets += 1
            print(f"Paquete {i+1}: Perdido")

        time.sleep(interval)

    # Calcular métricas
    avg_latency = sum(latencies) / len(latencies) if latencies else 0
    jitter = max(latencies) - min(latencies) if len(latencies) > 1 else 0
    loss_rate = (lost_packets / count) * 100

    print("\nResultados:")
    print(f"Latencia promedio: {avg_latency:.2f} ms")
    print(f"Jitter: {jitter:.2f} ms")
    print(f"Tasa de pérdida de paquetes: {loss_rate:.2f}%")

    # Guardar datos en CSV
    data = {"Packet": list(range(1, count+1)), "Latency (ms)": latencies + [None]*lost_packets}
    df = pd.DataFrame(data)
    df.to_csv("icmp_results.csv", index=False)
    print("Resultados guardados en 'icmp_results.csv'")

# Uso
target_ip = "192.168.56.102"
generate_and_measure_icmp(target_ip, count=20, interval=0.5)
```

RESULTADOS

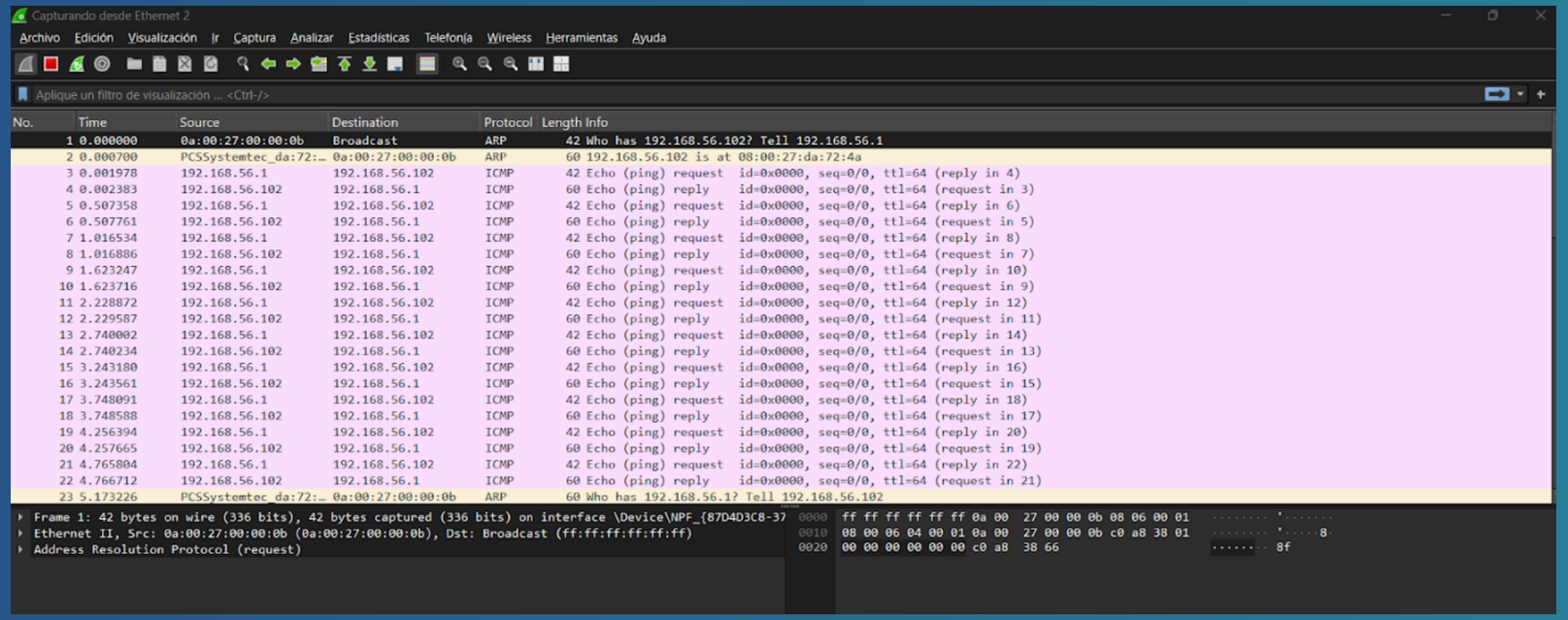
PRUEBA 1: GENERADOR ICMP CON MEDICIÓN DE RENDIMIENTO1: GENERADOR ICMP CON MEDICIÓN DE RENDIMIENTOTURE



RESULTADOS

```
Resultados:
Latencia promedio: 58.61 ms
Jitter: 129.83 ms
Tasa de pérdida de paquetes: 0.00%
Resultados guardados en 'icmp_results.csv'
```

PRUEBA 1: GENERADOR ICMP CON MEDICIÓN DE RENDIMIENTO1: GENERADOR ICMP CON MEDICIÓN DE RENDIMIENTOTURE



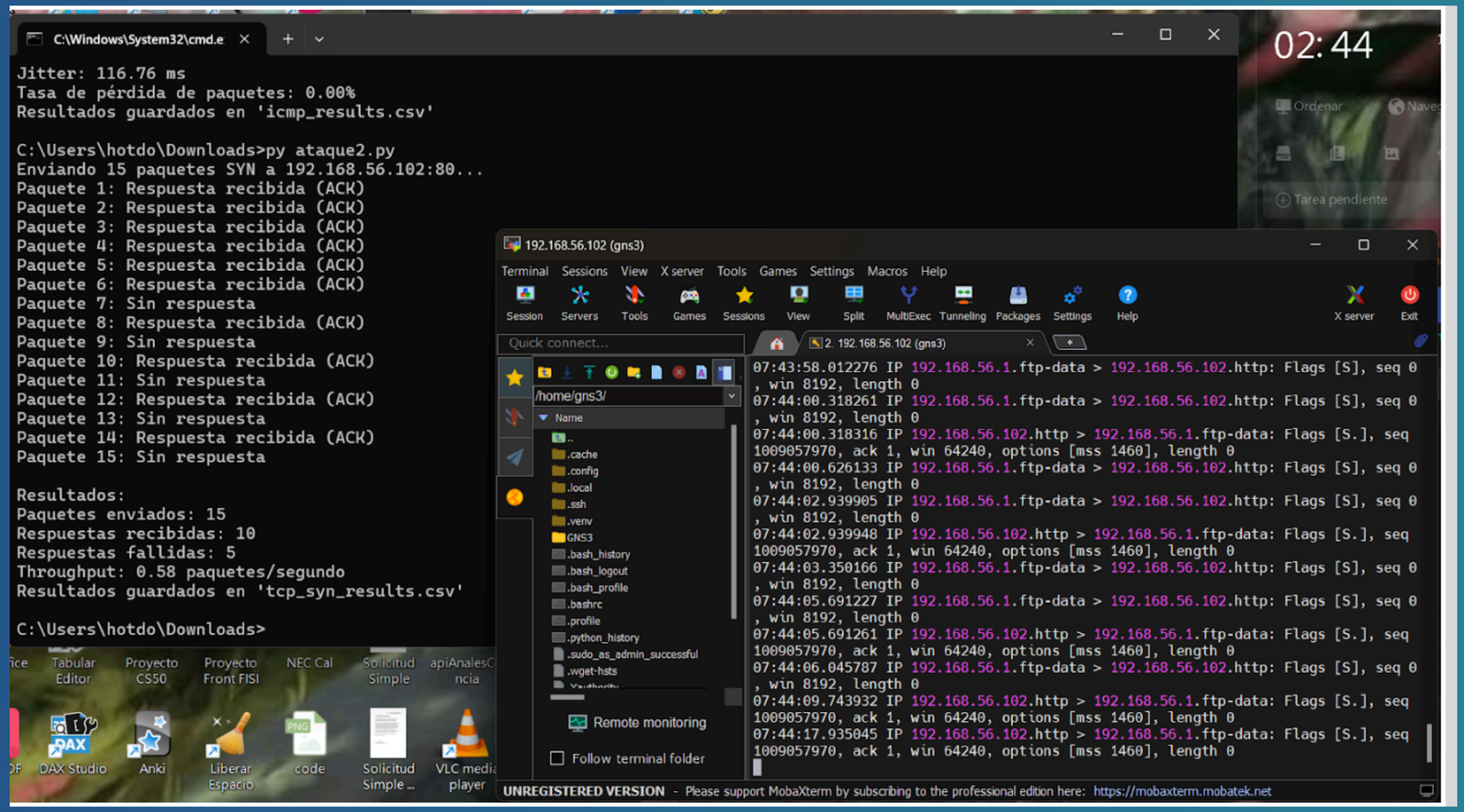
RESULTADOS

PRUEBA 2: GENERADOR TCP SYN

```
import time
from scapy.all import *
import pandas as pd

def generate_and_measure_syn(target_ip, target_port, count=10, interval=0.5):
    success_count = 0
    failed_count = 0
    start_time = time.time()

    print(f"Enviando {count} paquetes SYN a {target_ip}:{target_port}...")

    for i in range(count):
        packet = IP(dst=target_ip) / TCP(dport=target_port, flags="S")
        response = sr1(packet, timeout=2, verbose=False)

        if response and response.haslayer(TCP) and response[TCP].flags == "SA":
            success_count += 1
            print(f"Paquete {i+1}: Respuesta recibida (ACK)")
        else:
            failed_count += 1
            print(f"Paquete {i+1}: Sin respuesta")

        time.sleep(interval)

    end_time = time.time()
    total_time = end_time - start_time
    throughput = success_count / total_time if total_time > 0 else 0

    print("\nResultados:")
    print(f"Paquetes enviados: {count}")
    print(f"Respuestas recibidas: {success_count}")
    print(f"Respuestas fallidas: {failed_count}")
    print(f"Throughput: {throughput:.2f} paquetes/segundo")

    # Guardar datos en CSV
    data = {"Packet": list(range(1, count+1)), "Status": ["ACK" if i < success_count else "No Response" for i in range(count)]}
    df = pd.DataFrame(data)
    df.to_csv("tcp_syn_results.csv", index=False)
    print("Resultados guardados en 'tcp_syn_results.csv'")

# Uso
target_ip = "192.168.56.102"
target_port = 80
generate_and_measure_syn(target_ip, target_port, count=15, interval=0.3)
```

RESULTADOS

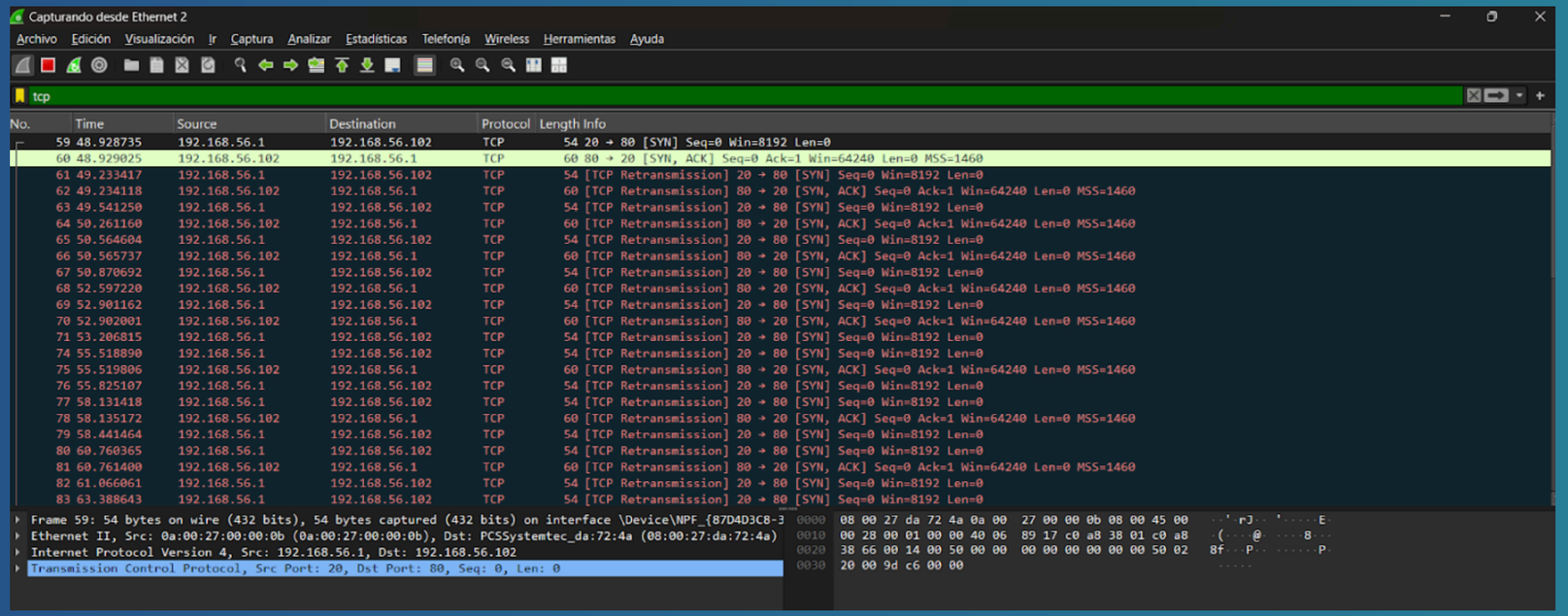
PRUEBA 2: GENERADOR TCP SYN



RESULTADOS

```
Resultados:
Paquetes enviados: 15
Respuestas recibidas: 10
Respuestas fallidas: 5
Throughput: 0.58 paquetes/segundo
Resultados guardados en 'tcp_syn_results.csv'
```

PRUEBA 2: GENERADOR TCP SYN



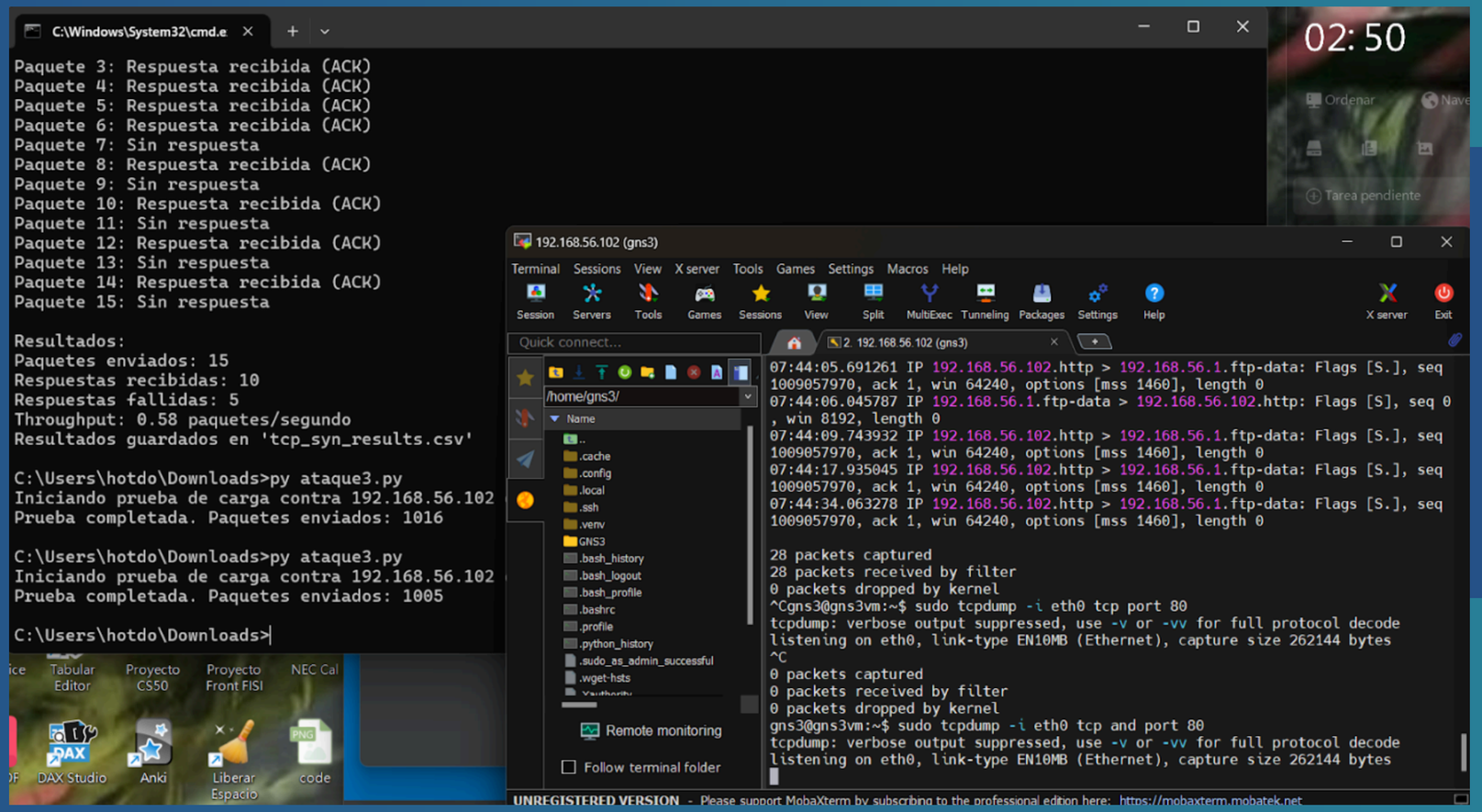
RESULTADOS

PRUEBA 3: GENERADOR DE CARGA CON EVALUACIÓN AVANZADA

```
from scapy.all import *
import random
import time

def load_test(target_ip, duration=30):
    start_time = time.time()
    packet_count = 0

    print(f"Iniciando prueba de carga contra {target_ip} durante {duration}
segundos...")

    while time.time() - start_time < duration:
        payload_size = random.randint(64, 1500)  # Tamaño del paquete
        packet = IP(dst=target_ip) / UDP() / ("X" * payload_size)
        send(packet, verbose=False)
        packet_count += 1
        time.sleep(random.uniform(0.01, 0.1))  # Intervalo aleatorio

    print(f"Prueba completada. Paquetes enviados: {packet_count}")

# Uso
target_ip = "192.168.56.102"
load_test(target_ip, duration=60)
```

RESULTADOS

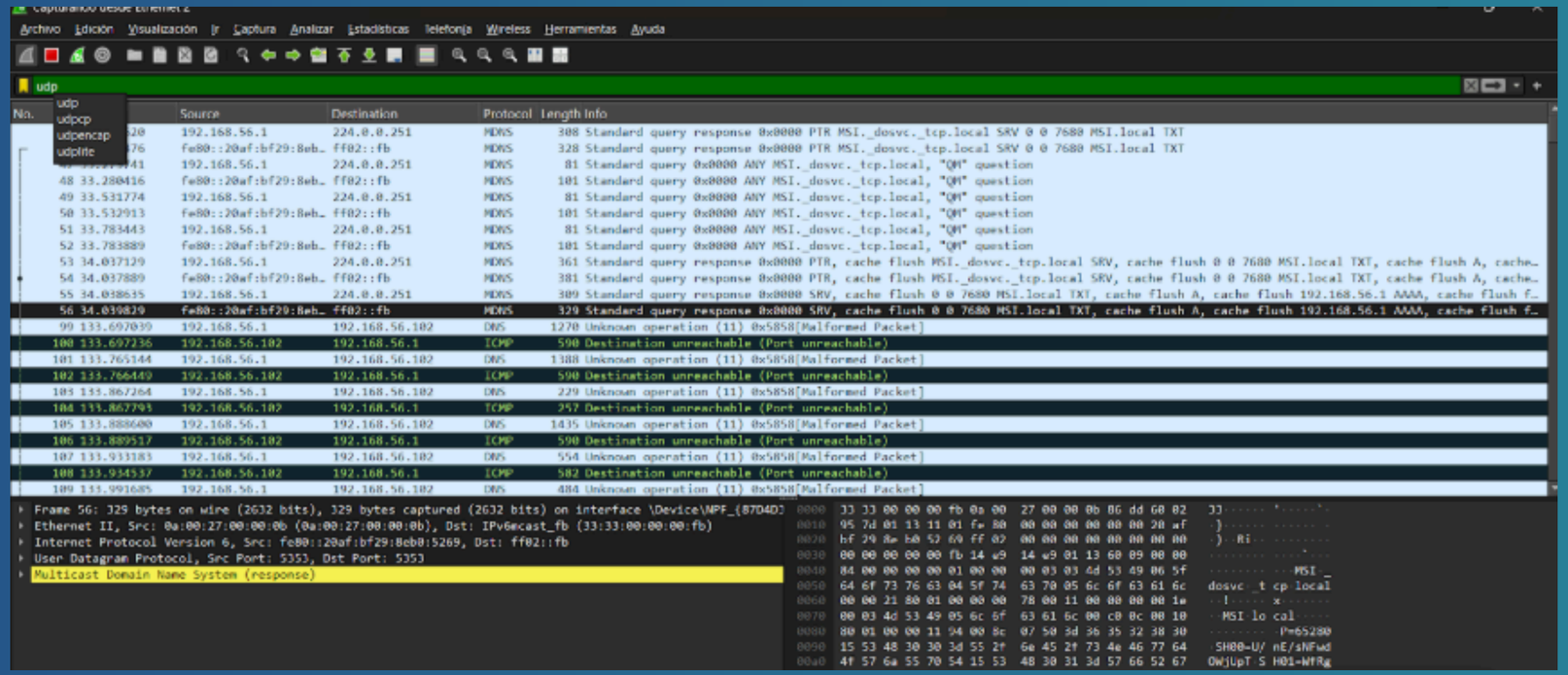
PRUEBA 3: GENERADOR DE CARGA CON EVALUACIÓN AVANZADA



RESULTADOS

PRUEBA 3: GENERADOR DE CARGA CON EVALUACIÓN AVANZADA

```
Iniciando prueba de carga contra 192.168.56.102 durante 60 segundos...
Prueba completada. Paquetes enviados: 992
```



CONCLUSIONES

Resultados clave:

- Tráfico ICMP: Latencia promedio de 58.61 ms (ligeramente elevada), jitter alto de 129.83 ms y tasa de pérdida de paquetes del 0.00%. Picos de latencia (>100 ms) sugieren congestión transitoria.
- Conexiones TCP al puerto 80: Tasa de respuesta del 66.67%, con 33.33% de fallos atribuidos a firewall, congestión o limitaciones del servicio. Throughput moderado de 0.58 paquetes/segundo.
- Cargas intensivas: Identificaron límites y puntos débiles en la capacidad de manejo de tráfico.

Recomendaciones:

1. Monitoreo continuo de latencia y jitter para detectar patrones de congestión.
2. Uso de traceroute para mitigar cuellos de botella.
3. Optimización del firewall y asignación de más recursos al servicio en el puerto objetivo.
4. Repetir pruebas periódicamente para evaluar la estabilidad bajo alta demanda.

Bibliografía

Hussain, F., Abbas, S. G., Shah, G. A., Pires, I. M., Fayyaz, U. U., Shahzad, F., García, N. M., & Zdravevski, E. (2021). A framework for malicious traffic detection in IoT healthcare environment. Sensors, 21(9), 3025. https://doi.org/10.3390/s21093025

Rincón, L. (2021). Test de penetración para el estudio de vulnerabilidades a los ciberataques mediante técnicas de hacking ético en redes IPv4. Revista Electrónica de Estudios Telemáticos, 20(2).

GRACIAS!