



UNIVERSIDAD DEL ISTMO



Campus Tehuantepec

## ***Ingeniería en Computación***

**Tema:** Monitoreo y escaneo de vulnerabilidades con las herramientas nmap y wireshark.

**Materia:** Redes de computadoras II.

**Alumno:** Osorio Ramos Jeremy.

**Docente:** IC. Carlos Mijangos Jiménez.

**Semestre:** Séptimo.

**Grupo:** 704.

*Tehuantepec Oaxaca a 17 de Noviembre de 2025.*

# ÍNDICE

**1. Creación del Sandbox**

**2. Introducción**

**3. Desarrollo de la práctica**

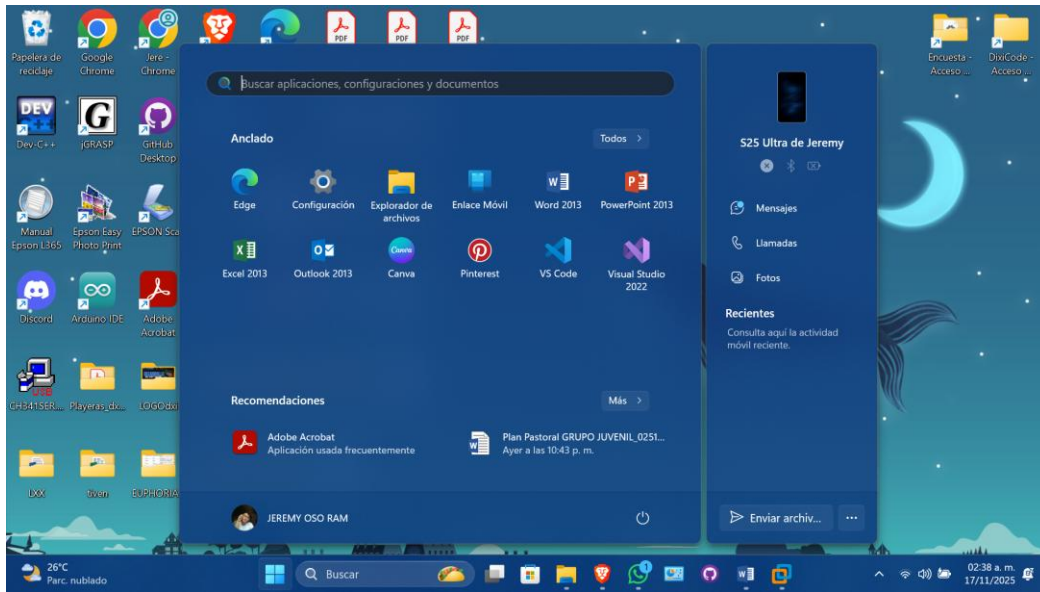
**4. Conclusiones**

**5. Referencias**

## 1. Creación del Sandbox

Para evitar riesgos y realizar pruebas de forma aislada, se creó un sandbox empleando:

1. Host principal: Windows 11.



2. Máquina virtual: VMware Workstation.
3. Sistema operativo invitado: Pop!\_OS Linux (basado en Ubuntu).

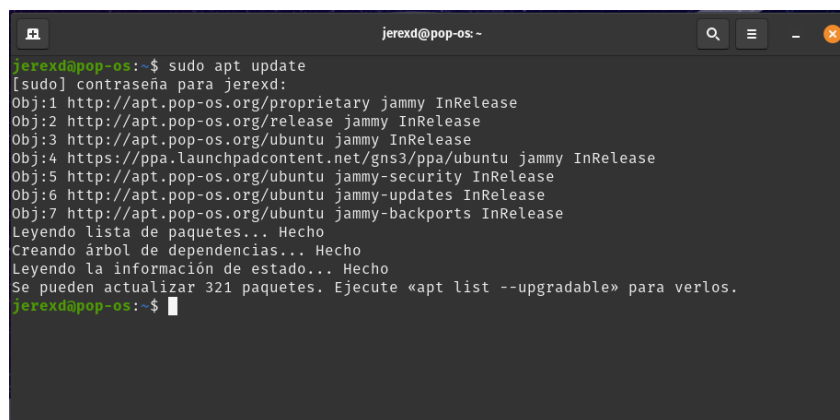


4. Red configurada: NAT o Red Interna (según necesidad del escaneo).

## Configuración empleada

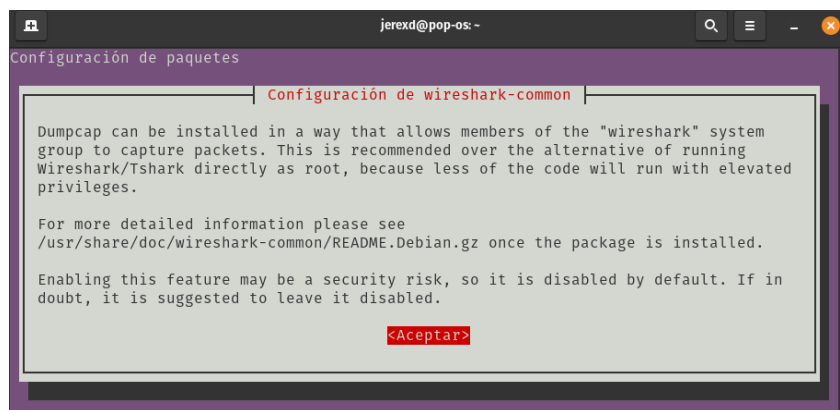
1. Se instaló VMware Workstation en Windows 11.
2. Se creó una máquina virtual con Pop!\_OS.
3. Para garantizar un entorno seguro, se configuró la red del VM en:
  - NAT para permitir tráfico controlado hacia el exterior.
  - (Opcional) Red Interna para realizar escaneos sin afectar otros dispositivos.
4. En Pop!\_OS se instalaron las herramientas:

**sudo apt update**



```
jerexd@pop-os: ~  
jerexd@pop-os:~$ sudo apt update  
[sudo] contraseña para jerexd:  
Obj:1 http://apt.pop-os.org/proprietary jammy InRelease  
Obj:2 http://apt.pop-os.org/release jammy InRelease  
Obj:3 http://apt.pop-os.org/ubuntu jammy InRelease  
Obj:4 https://ppa.launchpadcontent.net/gns3/ppa/ubuntu jammy InRelease  
Obj:5 http://apt.pop-os.org/ubuntu jammy-security InRelease  
Obj:6 http://apt.pop-os.org/ubuntu jammy-updates InRelease  
Obj:7 http://apt.pop-os.org/ubuntu jammy-backports InRelease  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias... Hecho  
Leyendo la información de estado... Hecho  
Se pueden actualizar 321 paquetes. Ejecute «apt list --upgradable» para verlos.  
jerexd@pop-os:~$
```

**sudo apt install nmap wireshark -y**



```
Configuración de paquetes  
Configuración de wireshark-common  
  
Dumpcap can be installed in a way that allows members of the "wireshark" system  
group to capture packets. This is recommended over the alternative of running  
Wireshark/Tshark directly as root, because less of the code will run with elevated  
privileges.  
  
For more detailed information please see  
/usr/share/doc/wireshark-common/README.Debian.gz once the package is installed.  
  
Enabling this feature may be a security risk, so it is disabled by default. If in  
doubt, it is suggested to leave it disabled.  
  
<Aceptar>
```

Este sandbox permite realizar pruebas de escaneo y monitoreo sin comprometer otros equipos de la red.

## 2. Introducción

### 2.1 Conceptos generales

1. **Vulnerabilidad**: fallas o debilidades en un sistema que pueden ser explotadas por un atacante.
2. **Escaneo de red**: proceso de identificar hosts, servicios y posibles fallas en una red.
3. **Monitoreo de tráfico**: análisis de los paquetes que circulan en la red para detectar comportamientos sospechosos.
4. **Auditoría de seguridad**: conjunto de acciones para evaluar la seguridad de un sistema.

### 2.2 Nmap

**Nmap** (Network Mapper) es una herramienta de código abierto usada para:

- Detectar hosts activos
- Identificar puertos abiertos
- Reconocer servicios y versiones
- Detectar vulnerabilidades
- Evaluar firewalls

Es fundamental en auditorías de seguridad por su capacidad para realizar escaneos profundos y sigilosos.

### 2.3 Wireshark

**Wireshark** es un analizador de protocolos que permite:

- Inspeccionar paquetes en tiempo real
- Ver protocolos usados por los equipos
- Identificar conexiones no autorizadas
- Detectar ataques o comportamientos inusuales

Se utiliza tanto para defensa como análisis forense.

## 2.4 Requisitos

1. Sistema operativo Windows 11 como host
2. VMware Workstation
3. Linux Pop!\_OS como VM
4. Instalación correcta de Nmap y Wireshark
5. Permisos de superusuario en Linux
6. Conexión a red NAT o interna

## 3. Desarrollo de la práctica

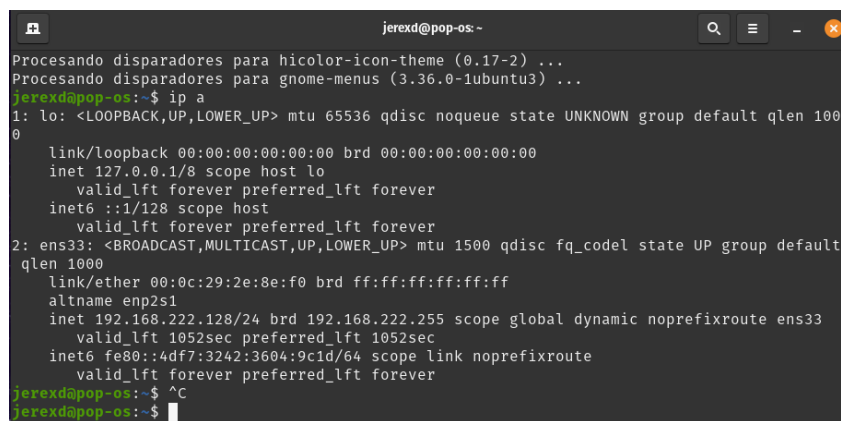
### 3.1 Escaneos con Nmap

#### 3.1.1 Verificación del entorno

Primero, se identificó la dirección IP de la máquina host y la máquina virtual:

En Pop!\_OS:

**ip a**



```
jerexd@pop-os: ~
Procesando disparadores para hicolor-icon-theme (0.17-2) ...
Procesando disparadores para gnome-menus (3.36.0-1ubuntu3) ...
jerexd@pop-os:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:2e:8e:f0 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.222.128/24 brd 192.168.222.255 scope global dynamic noprefixroute ens33
        valid_lft 1052sec preferred_lft 1052sec
    inet6 fe80::4df7:3242:3604:9c1d/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
jerexd@pop-os:~$ ^C
jerexd@pop-os:~$
```

#### 3.1.2 Escaneo básico (Ping Scan)

Se utilizó para verificar hosts activos en la red NAT:

**sudo nmap -sn 192.168.222.0/24**

```
jerexd@pop-os: ~  
[sudo] contraseña para jerexd:  
Starting Nmap 7.80 ( https://nmap.org ) at 2025-11-17 03:06 CST  
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 0.54 seconds  
jerexd@pop-os:~$ ^C  
jerexd@pop-os:~$ sudo nmap -sn 192.168.222.0/24  
Starting Nmap 7.80 ( https://nmap.org ) at 2025-11-17 03:09 CST  
Nmap scan report for 192.168.222.1  
Host is up (0.00060s latency).  
MAC Address: 00:50:56:C0:00:08 (VMware)  
Nmap scan report for _gateway (192.168.222.2)  
Host is up (0.00066s latency).  
MAC Address: 00:50:56:EB:E1:FC (VMware)  
Nmap scan report for 192.168.222.254  
Host is up (0.00038s latency).  
MAC Address: 00:50:56:E5:6C:4F (VMware)  
Nmap scan report for pop-os (192.168.222.128)  
Host is up.  
Nmap done: 256 IP addresses (4 hosts up) scanned in 2.48 seconds  
jerexd@pop-os:~$
```

Resultado: muestra la lista de IP activas dentro de la red asignada por VMware.

### 3.1.3 Escaneo de puertos (TCP Connect Scan)

**sudo nmap -sT 192.168.222.128**

- Identifica puertos abiertos.
- Es más fácil de detectar por firewalls.

```
jerexd@pop-os: ~  
Nmap scan report for 192.168.222.1  
Host is up (0.00060s latency).  
MAC Address: 00:50:56:C0:00:08 (VMware)  
Nmap scan report for _gateway (192.168.222.2)  
Host is up (0.00066s latency).  
MAC Address: 00:50:56:EB:E1:FC (VMware)  
Nmap scan report for 192.168.222.254  
Host is up (0.00038s latency).  
MAC Address: 00:50:56:E5:6C:4F (VMware)  
Nmap scan report for pop-os (192.168.222.128)  
Host is up.  
Nmap done: 256 IP addresses (4 hosts up) scanned in 2.48 seconds  
jerexd@pop-os:~$ sudo nmap -sT 192.168.222.128  
Starting Nmap 7.80 ( https://nmap.org ) at 2025-11-17 03:16 CST  
Nmap scan report for pop-os (192.168.222.128)  
Host is up (0.00012s latency).  
All 1000 scanned ports on pop-os (192.168.222.128) are closed  
Nmap done: 1 IP address (1 host up) scanned in 0.23 seconds  
jerexd@pop-os:~$
```

Resultado: Nmap realizó un escaneo tipo TCP Connect, en el cual intenta establecer una conexión completa TCP (SYN → SYN/ACK → ACK) con los puertos más comunes del sistema objetivo.

El resultado obtenido fue:

*“All 1000 scanned ports on 192.168.222.128 are closed.”*

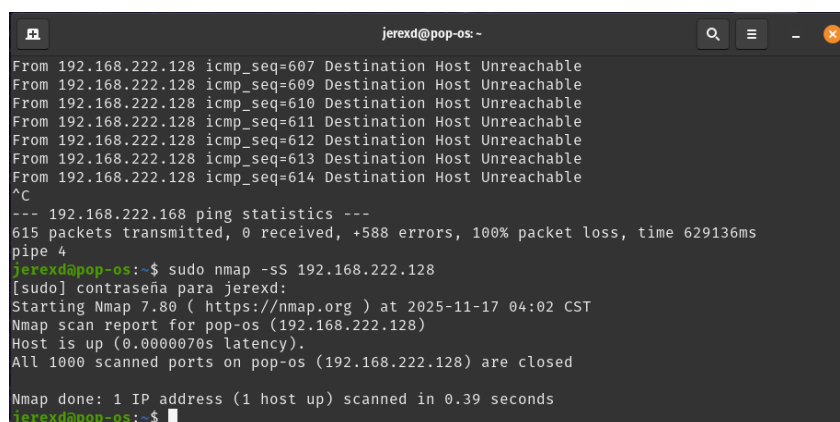
Esto indica que la máquina Pop!\_OS no tiene servicios escuchando en los puertos estándar. En otras palabras, no existen aplicaciones de red activas (como SSH, HTTP, FTP, SMB, etc.).

Este comportamiento es típico de sistemas Linux recién instalados, los cuales manejan una política de seguridad restrictiva por defecto.

#### 3.1.4 Escaneo SYN (Stealth Scan)

**sudo nmap -sS 192.168.222.128**

- Más sigiloso
- Evita completar el handshake TCP
- Reduce la probabilidad de ser detectado



```
jerexd@pop-os: ~  
From 192.168.222.128 icmp_seq=607 Destination Host Unreachable  
From 192.168.222.128 icmp_seq=609 Destination Host Unreachable  
From 192.168.222.128 icmp_seq=610 Destination Host Unreachable  
From 192.168.222.128 icmp_seq=611 Destination Host Unreachable  
From 192.168.222.128 icmp_seq=612 Destination Host Unreachable  
From 192.168.222.128 icmp_seq=613 Destination Host Unreachable  
From 192.168.222.128 icmp_seq=614 Destination Host Unreachable  
^C  
--- 192.168.222.168 ping statistics ---  
615 packets transmitted, 0 received, +588 errors, 100% packet loss, time 629136ms  
pipe 4  
jerexd@pop-os: ~$ sudo nmap -sS 192.168.222.128  
[sudo] contraseña para jerexd:  
Starting Nmap 7.80 ( https://nmap.org ) at 2025-11-17 04:02 CST  
Nmap scan report for pop-os (192.168.222.128)  
Host is up (0.0000070s latency).  
All 1000 scanned ports on pop-os (192.168.222.128) are closed  
Nmap done: 1 IP address (1 host up) scanned in 0.39 seconds  
jerexd@pop-os: ~$
```

Resultado: Esto indica que la máquina no tiene servicios escuchando en los puertos más comunes. Desde un punto de vista de seguridad, esto es positivo, ya que reduce la superficie de ataque y minimiza el riesgo de intrusiones. El host está encendido y responde a las sondas, pero no ofrece servicios accesibles de forma remota.

#### 3.1.5 Detección de versiones (Service Detection)

**sudo nmap -sV 192.168.222.128**

Permite saber qué servicios están corriendo y qué versión tienen.

Para identificar los servicios activos en la máquina virtual Pop!\_OS, se levantó un servidor HTTP temporal mediante el comando:

**sudo python3 -m http.server 8080**

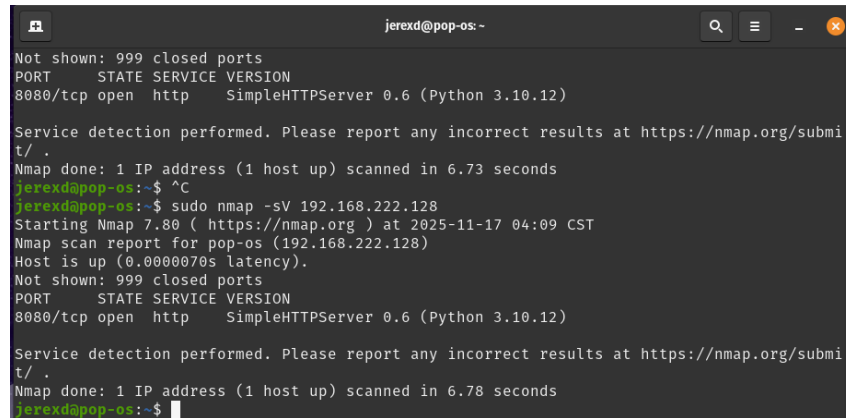
Posteriormente se ejecutó el escaneo:

```
sudo nmap -sV 192.168.222.128
```

El resultado fue:

```
PORT      STATE SERVICE VERSION
```

```
8080/tcp open  http    SimpleHTTPServer 0.6 (Python 3.10.12)
```



```
jerexd@pop-os: ~  
Not shown: 999 closed ports  
PORT      STATE SERVICE VERSION  
8080/tcp open  http    SimpleHTTPServer 0.6 (Python 3.10.12)  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submi  
t/ .  
Nmap done: 1 IP address (1 host up) scanned in 6.73 seconds  
jerexd@pop-os:~$ ^C  
jerexd@pop-os:~$ sudo nmap -sV 192.168.222.128  
Starting Nmap 7.80 ( https://nmap.org ) at 2025-11-17 04:09 CST  
Nmap scan report for pop-os (192.168.222.128)  
Host is up (0.0000070s latency).  
Not shown: 999 closed ports  
PORT      STATE SERVICE VERSION  
8080/tcp open  http    SimpleHTTPServer 0.6 (Python 3.10.12)  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submi  
t/ .  
Nmap done: 1 IP address (1 host up) scanned in 6.78 seconds  
jerexd@pop-os:~$
```

Esto demuestra que Nmap no solo detecta que un puerto está abierto, sino que es capaz de identificar con precisión:

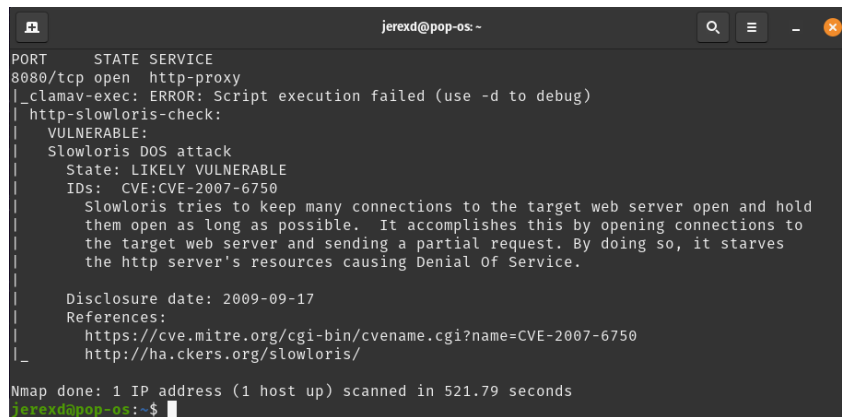
- El protocolo (HTTP)
- El tipo de servidor (SimpleHTTPServer)
- La versión del software (0.6)
- El lenguaje de ejecución (Python 3.10.12)

Esta capacidad convierte a Nmap en una herramienta clave para auditorías de seguridad, ya que permite detectar servicios vulnerables o configurados incorrectamente.

### 3.1.6 Escaneo de vulnerabilidades (script NSE)

```
sudo nmap --script=vuln 192.168.222.128
```

Evalúa vulnerabilidades conocidas en los servicios detectados.

A terminal window titled 'jerexd@pop-os: ~' showing the output of an Nmap scan. The output identifies a Slowloris DoS attack vulnerability on port 8080/tcp. It states the state is 'LIKELY VULNERABLE' and provides details about the attack, including its disclosure date (2009-09-17) and references to CVE-2007-6750. The scan was completed in 521.79 seconds.

```
jerexd@pop-os: ~  
PORT      STATE SERVICE  
8080/tcp  open  http-proxy  
|_clamav-exec: ERROR: Script execution failed (use -d to debug)  
|_http-slowloris-check:  
|_  VULNERABLE:  
|_    Slowloris DOS attack  
|_    State: LIKELY VULNERABLE  
|_    IDs: CVE:CVE-2007-6750  
|_    Slowloris tries to keep many connections to the target web server open and hold  
|_    them open as long as possible. It accomplishes this by opening connections to  
|_    the target web server and sending a partial request. By doing so, it starves  
|_    the http server's resources causing Denial Of Service.  
|_    Disclosure date: 2009-09-17  
|_    References:  
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6750  
|_      http://ha.ckers.org/slowloris/  
Nmap done: 1 IP address (1 host up) scanned in 521.79 seconds  
jerexd@pop-os: ~$
```

Este comando utiliza los scripts NSE de la categoría *vuln*, cuyo objetivo es detectar vulnerabilidades conocidas, configuraciones inseguras y servicios que puedan ser explotados.

El resultado del análisis reveló la siguiente vulnerabilidad:

- CVE-2007-6750 – Slowloris DoS Attack
- Estado: *Likely Vulnerable*
- Servicio afectado: Servidor HTTP en el puerto 8080
- Descripción:

Slowloris es una técnica de Denial of Service (DoS) que consiste en abrir múltiples conexiones al servidor web y enviar peticiones HTTP incompletas de forma muy lenta. Al no cerrar las conexiones y mantenerlas ocupadas, el servidor agota sus recursos y deja de responder a nuevas solicitudes legítimas.

La vulnerabilidad se detectó porque el servidor utilizado (SimpleHTTPServer / http.server de Python) carece de mecanismos de protección contra conexiones lentas, no implementa límites de sesión y no gestiona adecuadamente peticiones incompletas.

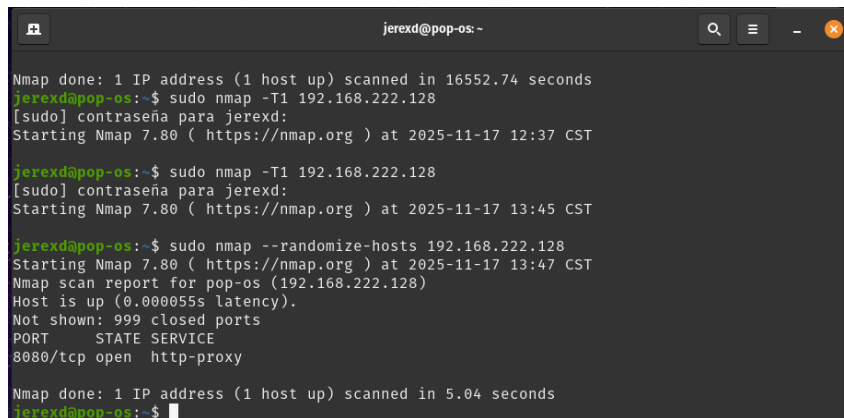
Este resultado demuestra la importancia de utilizar servidores robustos o configuraciones adicionales de seguridad, especialmente en entornos de producción, para evitar ataques de denegación de servicio.

### 3.1.7 Medidas para no ser detectado

Nmap tiene escaneos “evasivos”:

- Aleatorizar el orden de hosts:

**sudo nmap --randomize-hosts 192.168.222.128**

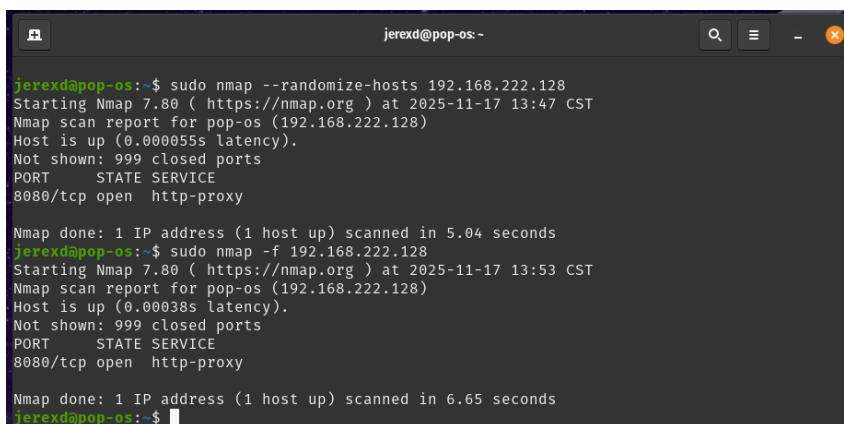


```
jerexd@pop-os: ~  
Nmap done: 1 IP address (1 host up) scanned in 16552.74 seconds  
jerexd@pop-os:~$ sudo nmap -T1 192.168.222.128  
[sudo] contraseña para jerexd:  
Starting Nmap 7.80 ( https://nmap.org ) at 2025-11-17 12:37 CST  
  
jerexd@pop-os:~$ sudo nmap -T1 192.168.222.128  
[sudo] contraseña para jerexd:  
Starting Nmap 7.80 ( https://nmap.org ) at 2025-11-17 13:45 CST  
  
jerexd@pop-os:~$ sudo nmap --randomize-hosts 192.168.222.128  
Starting Nmap 7.80 ( https://nmap.org ) at 2025-11-17 13:47 CST  
Nmap scan report for pop-os (192.168.222.128)  
Host is up (0.000055s latency).  
Not shown: 999 closed ports  
PORT      STATE SERVICE  
8080/tcp  open  http-proxy  
  
Nmap done: 1 IP address (1 host up) scanned in 5.04 seconds  
jerexd@pop-os:~$
```

Se probó el parámetro *--randomize-hosts*, cuyo propósito es alterar el orden de escaneo para evitar patrones detectables por sistemas de seguridad. Sin embargo, debido a que en esta práctica solo se trabajó con una única dirección IP, la opción no tuvo efecto visible, pero el comando se ejecutó correctamente y permitió comprobar el funcionamiento del escaneo básico.

- Fragmentación de paquetes:

**sudo nmap -f 192.168.222.128**



```
jerexd@pop-os: ~  
jerexd@pop-os:~$ sudo nmap --randomize-hosts 192.168.222.128  
Starting Nmap 7.80 ( https://nmap.org ) at 2025-11-17 13:47 CST  
Nmap scan report for pop-os (192.168.222.128)  
Host is up (0.000055s latency).  
Not shown: 999 closed ports  
PORT      STATE SERVICE  
8080/tcp  open  http-proxy  
  
Nmap done: 1 IP address (1 host up) scanned in 5.04 seconds  
jerexd@pop-os:~$ sudo nmap -f 192.168.222.128  
Starting Nmap 7.80 ( https://nmap.org ) at 2025-11-17 13:53 CST  
Nmap scan report for pop-os (192.168.222.128)  
Host is up (0.00038s latency).  
Not shown: 999 closed ports  
PORT      STATE SERVICE  
8080/tcp  open  http-proxy  
  
Nmap done: 1 IP address (1 host up) scanned in 6.65 seconds  
jerexd@pop-os:~$
```

Se ejecutó el escaneo con el parámetro *-f*, el cual fragmenta los paquetes enviados por Nmap en múltiples piezas pequeñas.

Esta técnica se utiliza para evadir firewalls o sistemas de detección (IDS/IPS) que analizan la estructura normal de los paquetes.

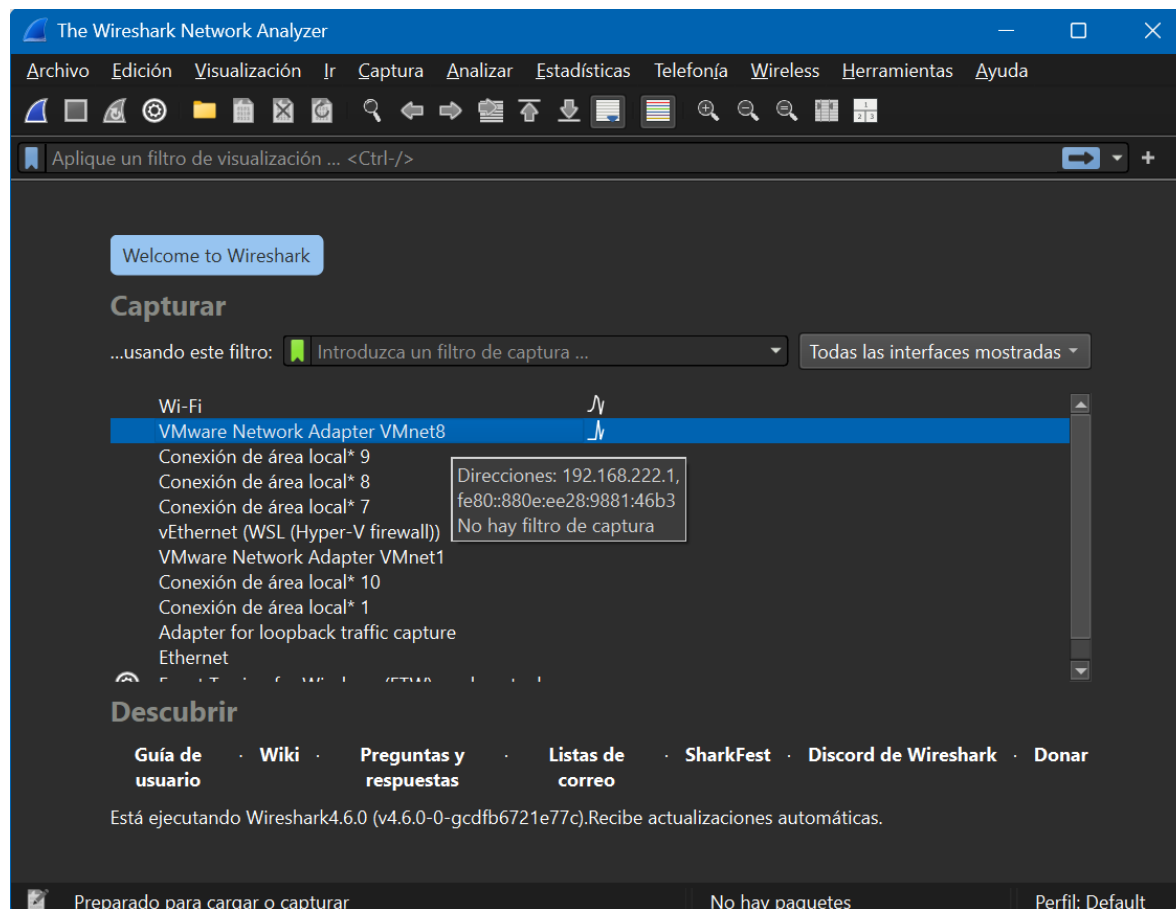
En el entorno de prueba (sandbox), la respuesta fue idéntica al escaneo convencional debido a que no existe un firewall avanzado que filtre o bloquee tráfico fragmentado. El host respondió correctamente y se detectó nuevamente el puerto 8080/tcp como abierto, demostrando que la fragmentación no altera la visibilidad del servicio, sino únicamente la forma en la que se transmite el paquete.

Estas opciones permiten evitar IDS/IPS o firewalls simples.

## 3.2 Monitoreo con Wireshark

### 3.2.1 Inicio del monitoreo

Se abrió Wireshark y se seleccionó la interfaz de red activa.

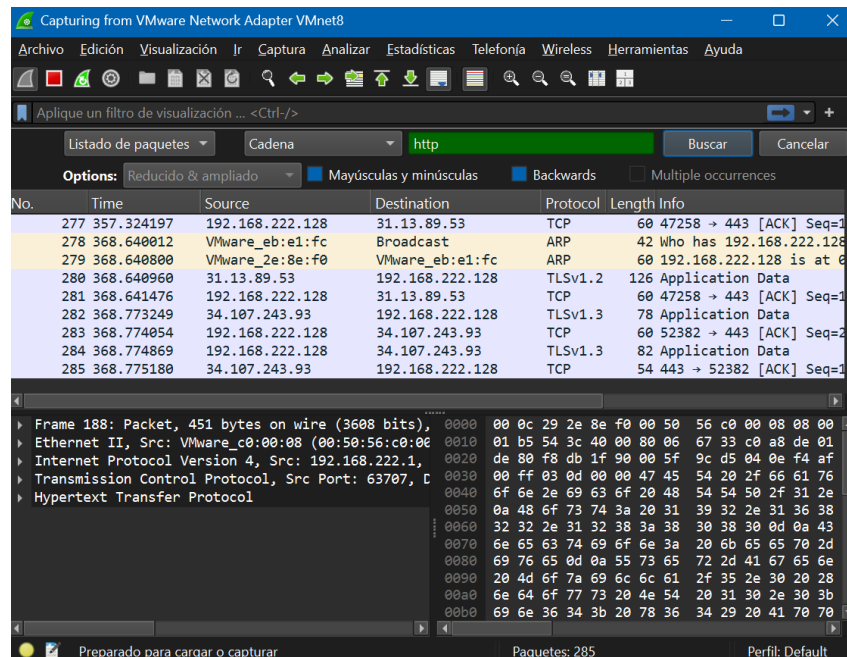


### 3.2.2 Filtros de visualización

Ejemplos usados:

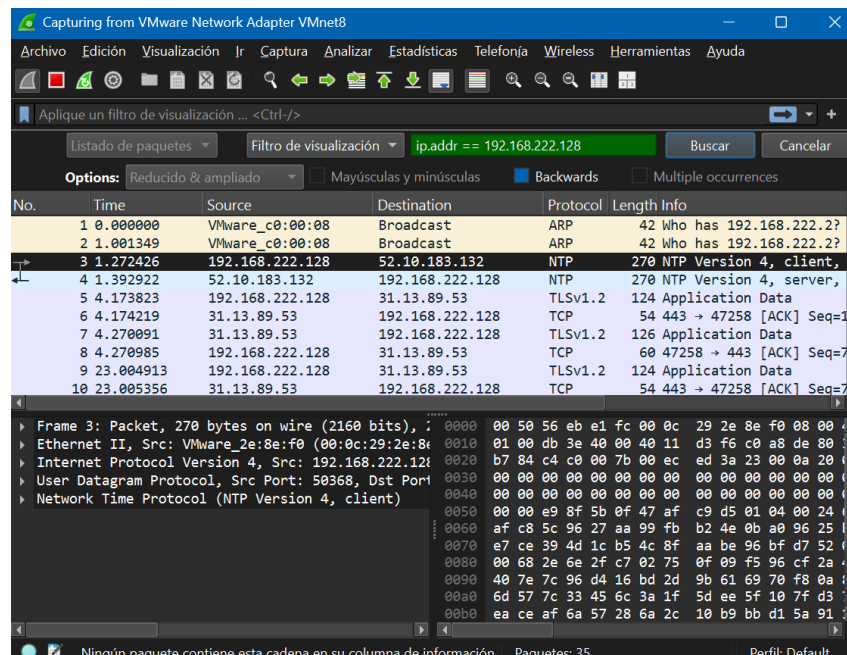
Ver solo tráfico HTTP:

Para esto es necesario hacer uso del servidor previamente abierto en Linux con Python, usando <http://192.168.222.128:8080> estaremos generando tráfico HTTPS, y al monitorearlo con Wireshark podemos hacer uso del filtro **http**.



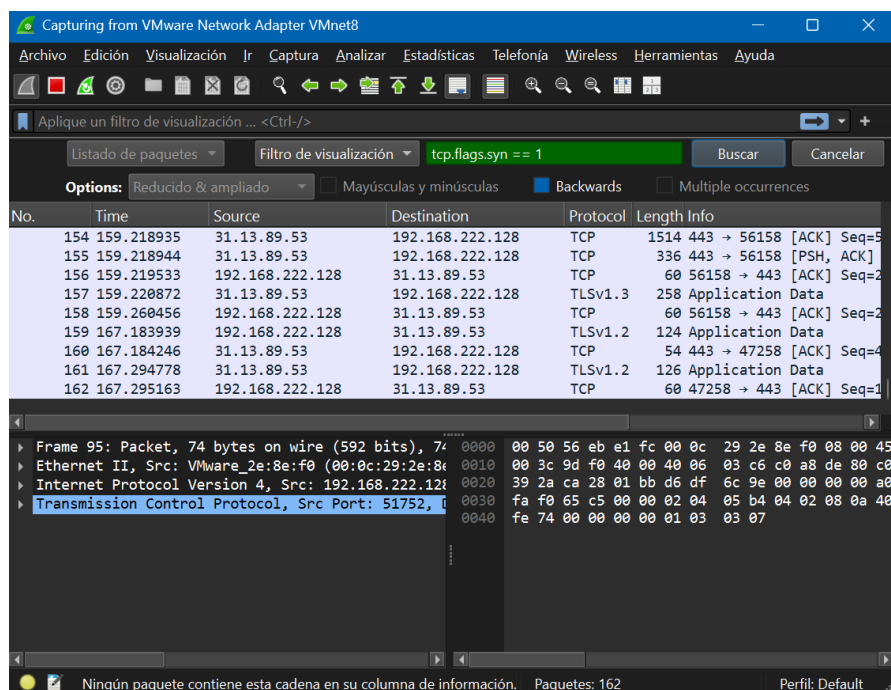
Filtrar tráfico hacia un host específico:

**ip.addr == 192.168.222.168**



Mostrar solo paquetes SYN (detección de escaneos):

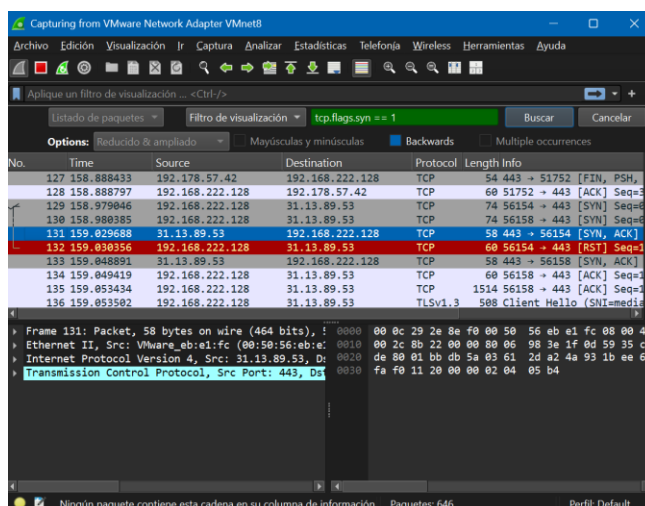
`tcp.flags.syn == 1`



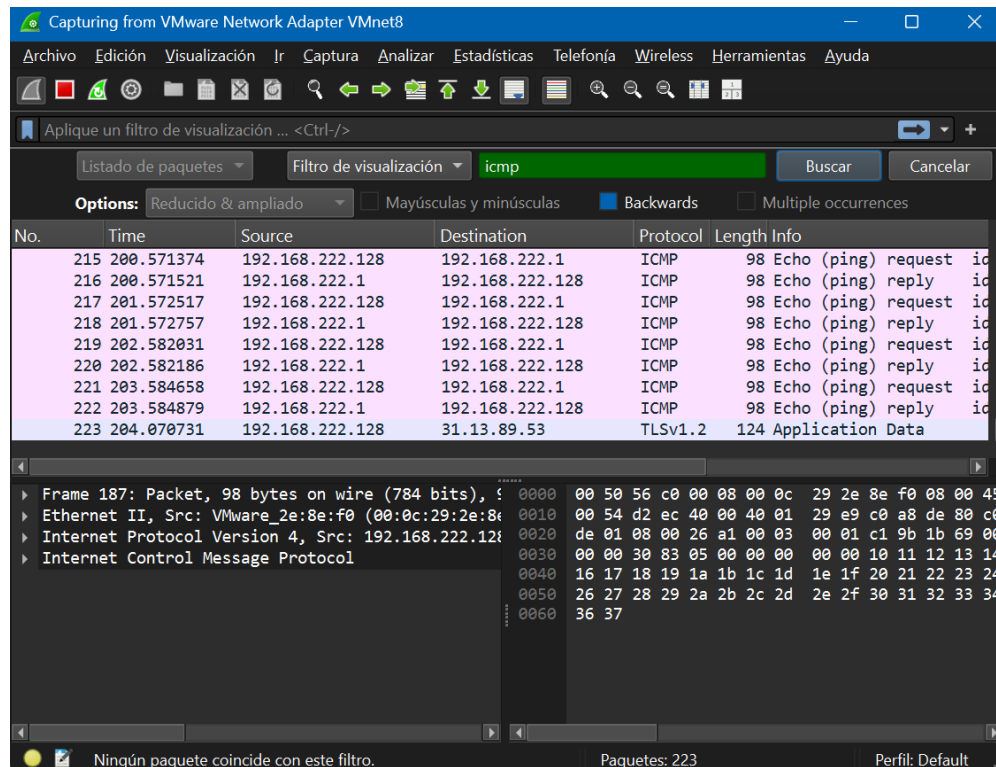
### 3.2.3 Detección de escaneos

Durante la ejecución de Nmap, Wireshark muestra:

- Paquetes SYN repetidos a diferentes puertos



- Solicitudes ICMP echo-request



Esto permite identificar en tiempo real un escaneo de red.

### 3.2.4 Monitoreo de tráfico anómalo

Ejemplos detectables:

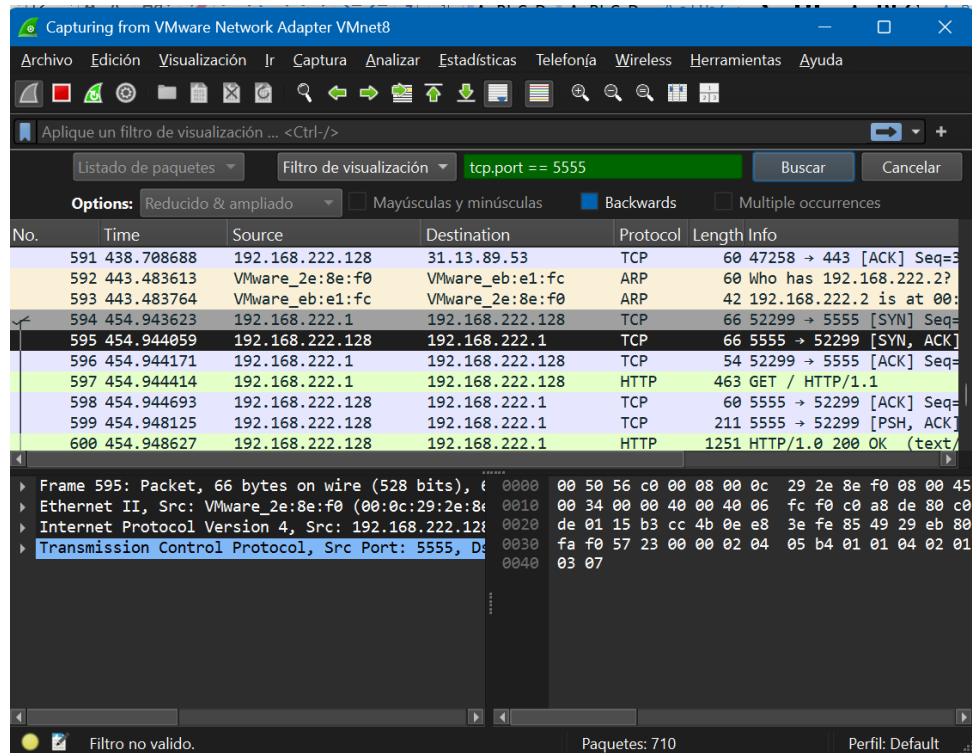
- Conexiones a puertos no usuales

Abrimos un servidor simple en Linux: `python3 -m http.server 5555`

Esto usará el puerto **5555**, que es inusual.

En el navegador de Windows abrimos: <http://192.168.222.128:5555>

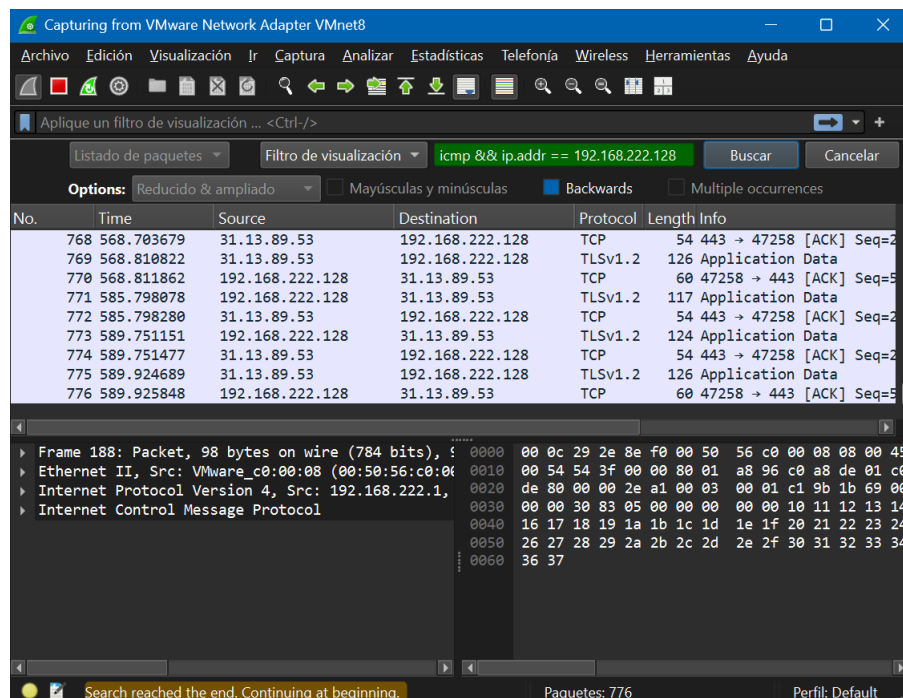
Y usamos el filtro `tcp.port == 5555`.



- Muchas peticiones seguidas al mismo host

En nuestra terminal linux pegamos el siguiente comando *for i in {1..20}; do ping -c 1 192.168.222.128; done*

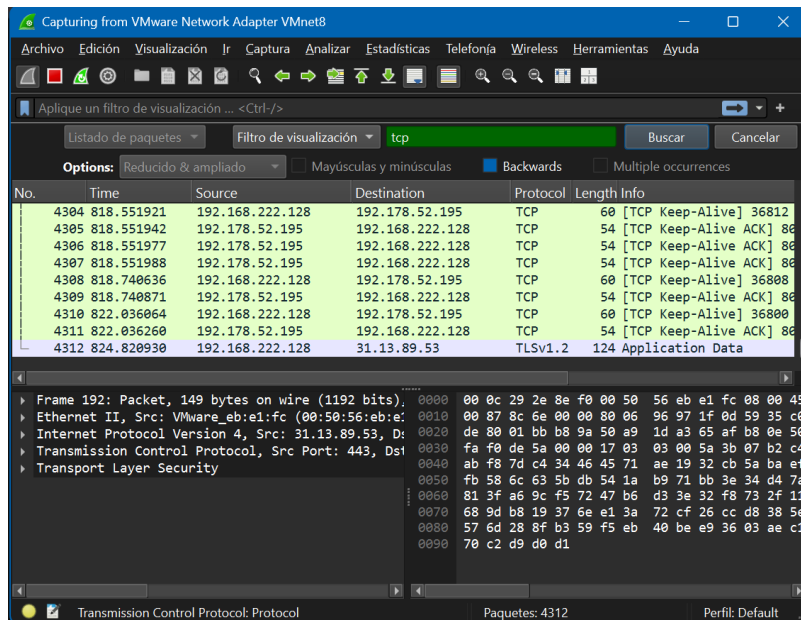
En Wireshark usamos el filtro: *icmp && ip.addr == 192.168.222.128*



- Reenvíos sospechosos

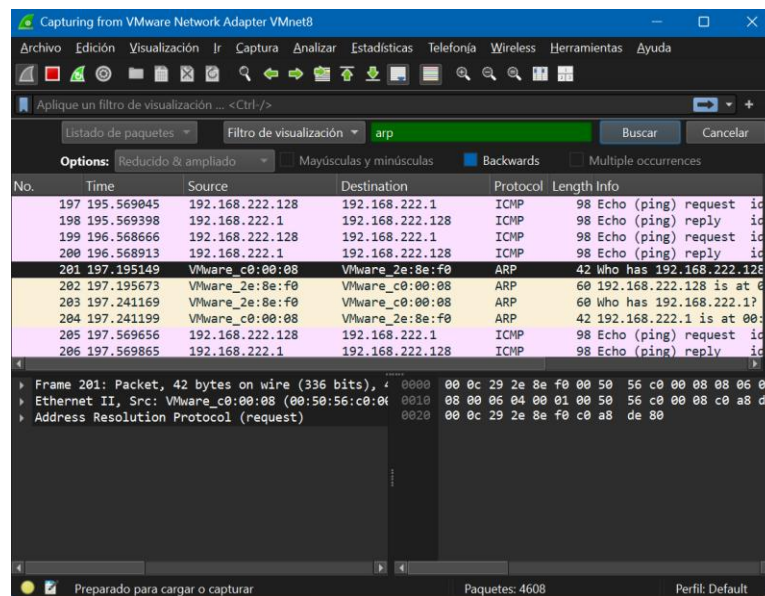
Se abre una página cualquiera desde Linux, algo como: <http://example.com>

Y en Wireshark usamos el filtro **tcp**



- Paquetes modificados o extraños

El paquete más simple de capturar es **arp**, así que podemos usarlo como palabra clave en el filtro.



Wireshark se convierte así en una herramienta defensiva.

## 4. Conclusiones

Al finalizar la práctica se obtuvo lo siguiente:

1. El sandbox creado con Windows 11 y VMware permitió realizar actividades de escaneo y monitoreo sin riesgos.
2. Nmap demostró ser una herramienta poderosa para auditorías, permitiendo identificar hosts, puertos abiertos, servicios y posibles vulnerabilidades.
3. Los distintos tipos de escaneo ofrecen usos tanto ofensivos (auditorías) como defensivos (detección de intrusiones).
4. Wireshark permitió visualizar tráfico en tiempo real y detectar patrones característicos de un escaneo.
5. La combinación de Nmap (ataque controlado) y Wireshark (defensa/monitoreo) ayuda a comprender cómo se realizan auditorías de seguridad.
6. Se comprobó cómo un administrador puede implementar medidas para detectar ataques, así como técnicas para realizar análisis preventivos.

## Referencias

Kretzschmar, F. (2020). Nmap 7: Network scanning. Red Publishing.

Orebaugh, A., Ramirez, G., & Beale, J. (2006). Wireshark & Ethereal network protocol analyzer toolkit. Syngress Publishing.

Lyon, G. F. (2018). Nmap Network Scanning: The official Nmap project guide. Insecure.Org.

Wireshark Foundation. (2024). Wireshark User Guide. <https://www.wireshark.org/docs/>

Nmap Project. (2024). Nmap Reference Guide. <https://nmap.org/book/man.html>