

Seguridad de la Información

Marzo 2025



Universidad de Deusto
Deustuko Unibertsitatea
University of Deusto

Deusto

Práctica 1

Escaneo de puertos

Miguel Acha Delicado

Índice

1	Introducción	1
1.1	¿Qué se pretende con esta práctica?	1
1.2	¿Qué es el escaneo de puertos?	1
2	Teoría	1
2.1	¿Cómo funciona la comunicación entre computadores?	1
2.2	Protocolos	3
2.2.1	TCP	4
2.2.2	UDP	5
2.3	¿Cómo se puede explotar esto?	6
2.3.1	Tipos de respuestas	6
3	Ejercicio	7
3.1	Formulación del ejercicio	7
3.2	Configuración del entorno	7
3.3	Resoluciones y sus consecuencias	9
3.3.1	Pasos previos	9
3.3.2	Resolución básica	11
3.3.3	Resolución silenciosa	13
3.3.4	Resolución zombie	17

1. Introducción

1.1. ¿Qué se pretende con esta práctica?

El objetivo de esta práctica es explicar superficialmente **cómo se comunican los computadores** a través de la red, **cuales son sus protocolos** y sus **funcionamientos**, y por último explicar **qué implicaciones conllevan** estas para la *seguridad cibernética* en el ámbito de la *enumeración o escaneo de puertos*.

1.2. ¿Qué es el escaneo de puertos?

El escaneo de puertos consiste en **analizar cómo se establecen las conexiones entre computadores** en una red para **identificar la configuración de los puertos** de uno o varios equipos, ya sea de forma local o remota. En otras palabras, se aprovechan los mecanismos de conexión para determinar qué puertos están abiertos, qué sistema operativo se utiliza y otros detalles relevantes. Esta información puede servir para **evaluar posibles vulnerabilidades** en la configuración del sistema o para **planificar un ataque**.

2. Teoría

2.1. ¿Cómo funciona la comunicación entre computadores?

Los computadores utilizan **protocolos de comunicación** para establecer *conexiones* o *simplemente mandar datos*. Los puntos de encuentro de estas conexiones se llaman *puertos*, por los que se envían o se reciben los datos. Las comunicaciones, en la mayoría de casos, utilizan un modelo moderno llamado **TCP/IP**, para gestionar las conexiones.

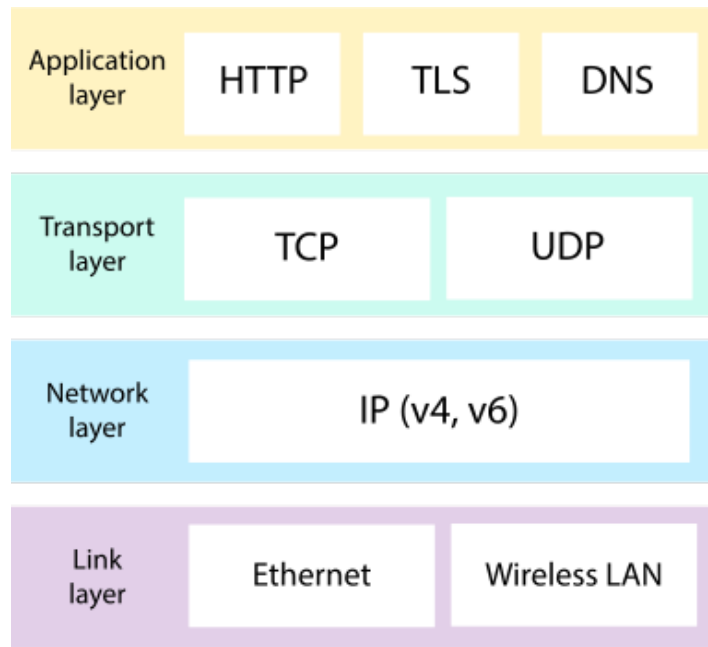


Figura 1: Diagrama de protocolos en las capas del modelo TCP/IP

Fuente: <https://cdn.kastatic.org/ka-perseus-images/6a0cd3a5b7e709c2f637c959ba98705ad21e4e3c.svg>

Este modelo está pensado para estandarizar y simplificar las conexiones entre dispositivos de diferentes fabricantes.

Los datos viajan por las diferentes **capas internas**, preparando su envío. Este envío se hace en **paquetes**, para después ser retransmitidos hacia donde se desee. Normalmente, los datos recorrerán varias de estas capas dependiendo de las necesidades de la conexión. Estos paquetes necesitan saber que destino tienen que alcanzar, por lo tanto, a la par que van avanzando las capas del modelo, se les asignan diferentes **direcciones** para que sepan que deben hacer.

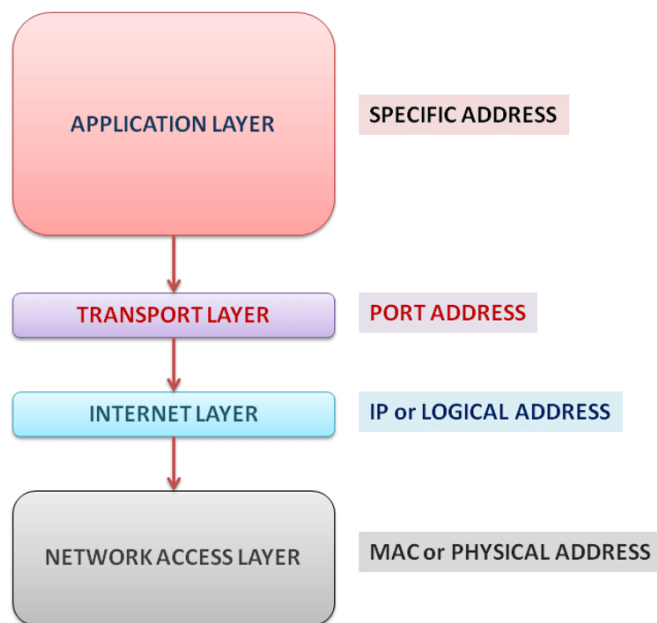


Figura 2: Diagrama de direccionamiento en las diferentes capas de modelo TCP/IP

Fuente: <https://www.sanfoundry.com/wp-content/uploads/2024/05/different-addressing-at-different-tcp-ip-model-layers.png>

Por norma, la **capa de aplicación** es la que se encarga de gestionar los datos que se envían y formatear los que se reciben. En otras palabras, dependiendo de como se representen al usuario estos datos tendrán diferente aspecto. La **capa de transporte** se encarga de empaquetar los datos que se envían según el tipo de conexión y de realizar los pasos previos necesarios en caso de que haya. La **capa de red** es donde se enrutan los paquetes de red. Por último, la de **enlace** las envía a través de diferentes medios, utilizando protocolos con o sin cable.

En esta práctica nos fijaremos en la **capa de transporte**, ya que por su naturaleza, es la que nos interesa hablar cuando nos referimos al comportamiento de los paquetes en la comunicación. Esto va a ser lo que nos permita determinar estados de los puertos, **el comportamiento**.

2.2. Protocolos

Para cumplir con los requisitos que demandan los diferentes tipos de conexiones, existen protocolos de conexión con diferentes **capacidades** y **atributos**, que a su vez, como hemos visto en la sección anterior, coexisten en las diferentes capas del modelo TCP/IP. Entre los más usados en la *capa de transporte* y los que se contemplarán en esta práctica están: *TCP* y *UDP*.

2.2.1. TCP

El protocolo TCP, **Transmission Control Protocol**, está principalmente diseñado para conexiones entre computadores, esta comienza mediante lo que se conoce como un *apretón de manos de tres direcciones*, este nombre viene del proceso por el cual para cerciorarse de que los paquetes han llegado existen códigos especiales llamados *Flags*, mediante los cuales un paquete puede tener un contexto sobre el estado de la conexión.

En esta práctica, no se va a dar una explicación extensa sobre todo el protocolo ni como funciona a su nivel más bajo, pero si se va a dar una breve explicación sobre los Flags, ya que es pertinente a la hora de visualizar la conexión a un alto nivel.

Un ejemplo es el *flag ACK*, proveniente de la palabra anglosajona *acknowledge*, que indica que el paquete anterior enviado ha llegado con éxito. Normalmente, los flags se combinan para proporcionar más contexto en la comunicación.

Estos son los flags y sus significados:

Flag	Descripción
CWR	Indica que la ventana de congestión ha sido reducida. Se usa en mecanismos de control de congestión.
ECE	Tiene doble función dependiendo del estado de SYN: - Si SYN = 1, indica que el host es compatible con ECN. - Si SYN = 0, indica congestión en la red (ECN=11 en la cabecera IP).
URG	Señala que el campo <i>Urgent Pointer</i> es significativo.
ACK	Indica que el campo <i>Acknowledgment</i> es válido. Todos los paquetes después del SYN inicial deben llevarlo activado.
PSH	Solicita que los datos almacenados en buffer sean enviados inmediatamente a la aplicación de destino.
RST	Reinicia la conexión. Se usa para abortar conexiones activas.
SYN	Usado para sincronizar números de secuencia al inicio de una conexión. Solo el primer paquete de cada lado debe llevarlo activado.
FIN	Indica que el remitente ha terminado de enviar datos y desea cerrar la conexión.

Cuadro 1: Flags en TCP y su función

El proceso inicial asegura que los dos computadores están **listos para empezar la conexión**. Después de este primer encuentro se empieza el **envío de datos**, en donde cada paquete de datos que ha sido enviado por el emisor necesita una confirmación por parte del receptor. Y por último, para finalizar la conexión se envía una **solicitud de finalización de conexión**, la cual necesita dos respuestas consecutivas por parte del receptor de esta solicitud, y una última respuesta por la parte del emisor para cerciorar que ha recibido esta última respuesta.

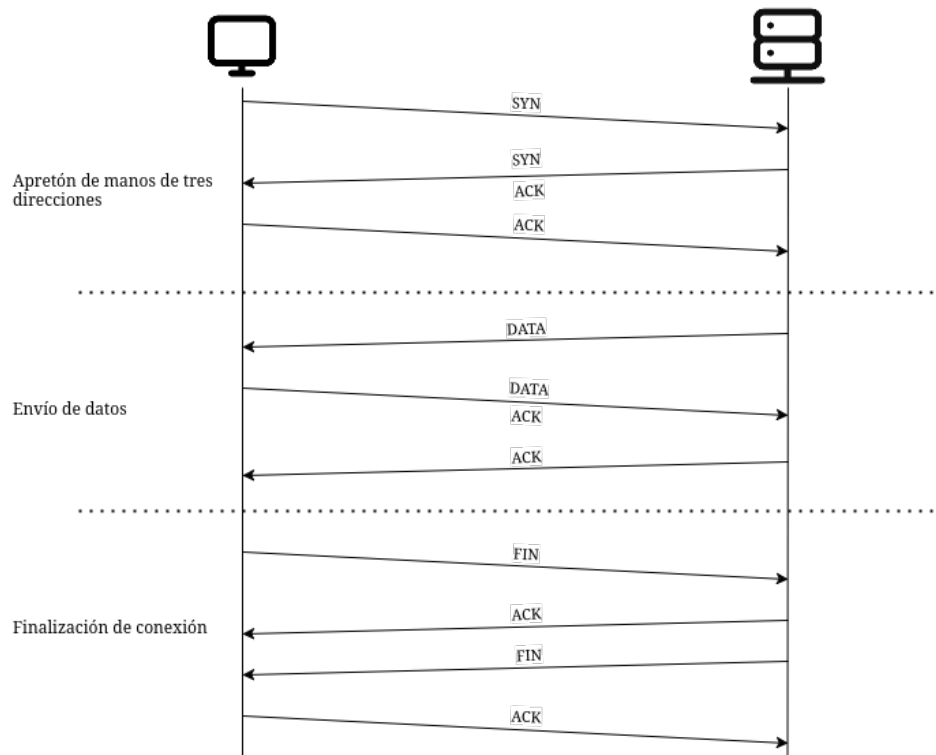


Figura 3: Conexión entre computadores utilizando el protocolo de transporte TCP

2.2.2. UDP

El protocolo UDP, **User Datagram Protocol**, está diseñado para comunicaciones rápidas y eficientes entre computadores sin necesidad de establecer una conexión previa. A diferencia de TCP, UDP no garantiza la entrega de los paquetes, ni verifica si estos han llegado correctamente a su destino.

Como se ha dicho anteriormente, en esta práctica no se aborda todos los detalles del protocolo, solo las claves para comprender su papel en la comunicación entre dispositivos.

Un concepto fundamental en UDP es que los paquetes, llamados **datagramas**, se envían sin establecer una conexión previa con el receptor. Esto significa que no hay necesidad de un *apretón de manos*, ni confirmaciones de recepción, lo que reduce la latencia pero también introduce el riesgo de pérdida de datos.

En UDP, cada datagrama es enviado de forma **independiente sin esperar respuesta** del receptor. Si bien esto puede generar pérdida de paquetes, muchas aplicaciones que usan UDP implementan sus propios mecanismos de control de errores y recuperación de datos. Al no requerir confirmaciones ni establecer conexiones, UDP permite una **transmisión de datos más rápida y eficiente**.

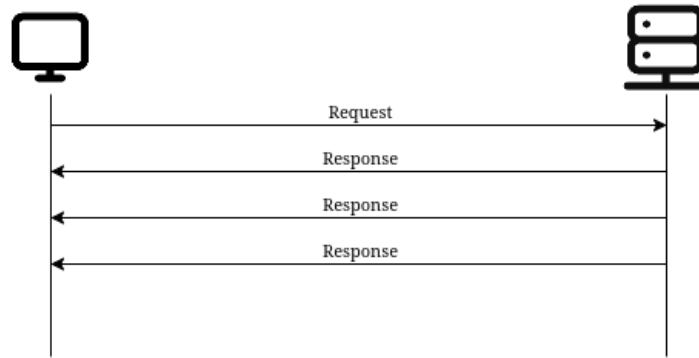


Figura 4: Comunicación entre computadores utilizando el protocolo de transporte UDP

2.3. ¿Cómo se puede explotar esto?

Para el **escaneo de puertos** toda información que pueda devolver el servidor cuando mandamos una solicitud, independientemente de que protocolo usemos, es útil para poder **determinar** o **inducir** estados tanto de la *máquina* como de los *servicios y disponibilidad en un puerto*. En el caso del protocolo TCP, hay varias formas de enviar este primer contacto, por ejemplo con paquetes usando flags como CONNECT, SYN, ACK, etc. Y dependiendo de si la respuesta es positiva, se determina. Con UDP, con enviar una solicitud, y si se recibe o no una respuesta, se supone que este servicio está disponible o no.

Qué tipo de servicio es el que reside en el puerto, se suele saber analizando el *Banner* de respuesta a la solicitud. Aunque hay técnicas más avanzadas, e incluso se podría intuir cual es teniendo en cuenta que hay números de puertos usuales para cada servicio.

2.3.1. Tipos de respuestas

Cuando estás intentando saber si un puerto está ofreciendo servicios o no, hay principalmente **tres tipos de respuestas**: *positiva*, *negativa* y *filtrada*.

Una respuesta positiva, es en el caso en el que el servidor nos responde con un estado que nos lleva a determinar que ese servicio en el puerto está activo. En el caso negativo, el servidor nos informará de lo contrario, el puerto no está activo. En el tercer caso, no nos responderá con nada, por lo tanto, está saneado para este tipo de ataques. Los administradores se han tomado las molestias de dificultarnos nuestro escaneo. Esto no siempre supone un punto y final a esta tarea, ya que la **seguridad por obscuridad** no funciona.

3. Ejercicio

3.1. Formulación del ejercicio

Haz un **escaneo de los puertos** con **Nmap** a los puertos número 80 y 81 a una máquina víctima, del tipo **TCP SYN Scan**, revisando los paquetes en **Wireshark**. Explica que está pasando, e intenta mejorar este escaneo para que sea lo más '*silencioso*' posible.

3.2. Configuración del entorno

Para simular el escenario para el desarrollo de este ejercicio, utilizaremos un programa muy conocido para *virtualizar* máquinas físicas con diferentes sistemas operativos: **Virtualbox**. Aunque esta es solo una sugerencia, hay muchos programas similares. En este caso, solo tendré que virtualizar la máquina víctima, en este caso, una máquina Windows Server 2000. Ya que en mi máquina principal, desde donde estoy escribiendo esto, tengo instalado ya **Nmap**. Si esto no es así, y se prefiere no instalar nada en la máquina principal, se podría virtualizar otra para este propósito.

Para la instalación de la máquina víctima, extrapolable para instalar cualquier máquina, necesitaremos iniciar Virtualbox y conseguir un archivo ISO o uno VBOX, con los cuales podremos construirla, con un archivo VBOX, o crearla, con un archivo ISO, siguiendo los pasos recomendados por el desarrollador del programa (https://www.virtualbox.org/manual/topics/Introduction.html#add_vm).

Una vez instalada la máquina víctima, y si se requiere la máquina atacante, la iniciamos (<https://www.virtualbox.org/manual/topics/Introduction.html#intro-running>). La máquina víctima tiene que estar configurada para que algún puerto esté abierto, en este caso utilizaremos los puertos 80 y 81, con servicios activos para estos. Y en la máquina atacante debe de estar instalado Nmap y Wireshark.

```
local ~
>> nmap --version
Nmap version 7.95 ( https://nmap.org )
Platform: x86_64-pc-linux-gnu
Compiled with: liblua-5.4.6 openssl-3.4.1 libssh2-1.11.0 libz-1.3.1 libpcap-1.10.5 nmap-libdnet-1.12 ipv6
Compiled without:
Available nsock engines: epoll poll select

local ~
>> wireshark --version
Wireshark 4.4.5.

Copyright 1998-2025 Gerald Combs <gerald@wireshark.org> and contributors.
Licensed under the terms of the GNU General Public License (version 2 or later).
This is free software; see the file named COPYING in the distribution. There is
NO WARRANTY; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Compiled (64-bit) using GCC 14.2.1 20250207, with GLib 2.82.5, with Qt 6.8.2,
with libpcap, with POSIX capabilities (Linux), with libnl 3, with zlib 1.3.1,
with zlib-ng 2.2.4, with PCRE2, with Lua 5.3.6, with GnuTLS 3.8.9 and PKCS #11
support, with Gcrypt 1.11.0, with Kerberos (MIT), with MaxMind, with nghttp2
1.64.0, with nghttp3 1.8.0, with brotli, with LZ4, with Zstandard, with Snappy,
with libxml2 2.13.6, without libsmi, with Minizip 1.3.1, with QtMultimedia, with
QtDBus, without automatic updates, with binary plugins.

Running on Linux 6.13.6-arch1-1, with 12th Gen Intel(R) Core(TM) i5-1235U (with
SSE4.2), with 15663 MB of physical memory, with GLib 2.84.0, with Qt 6.8.2, with
libpcap 1.10.5 (with TPACKET_V3), with zlib 1.3.1, with PCRE2 10.45 2025-02-05,
with c-ares 1.34.4, with GnuTLS 3.8.9, with Gcrypt 1.11.0, with nghttp2 1.65.0,
with nghttp3 1.8.0, with brotli 1.1.0, with LZ4 1.10.0, with Zstandard 1.5.7,
with LC_TYPE=en_US.UTF-8, binary plugins supported.
```

Figura 5: Comprobación de que los programas están instalados en la máquina atacante

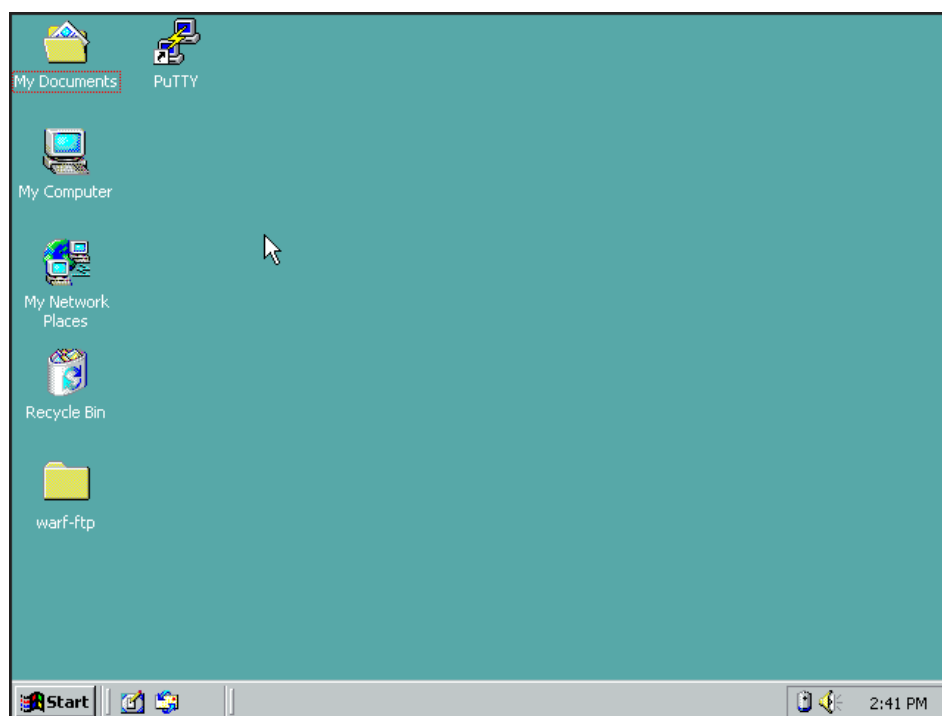


Figura 6: Comprobación de que la máquina víctima está corriendo sin problema

Ya está todo listo para empezar a resolver el problema.

3.3. Resoluciones y sus consecuencias

3.3.1. Pasos previos

Antes de nada, debemos saber la dirección IP local de la máquina víctima si queremos hacer un escaneo dirigido a esta, si no, no sabríamos donde mandar estos paquetes. Para esto, como tenemos el control de la máquina víctima podemos comprobar esta con un simple comando para comprobar que direcciones IP están asignadas a cada interfaz física o virtual en esta. En Windows tenemos que abrir la terminal de comandos de windows, el **CMD**, y escribir el comando 'ipconfig'.

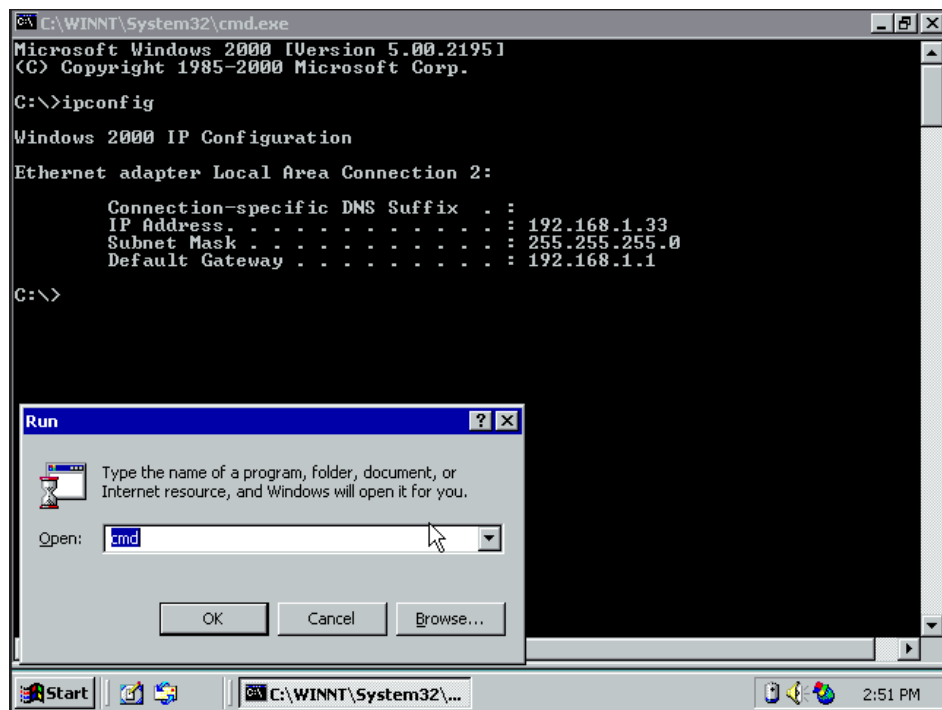


Figura 7: Comprobación de la dirección IP de la víctima con un comando en dicha máquina

O también podríamos descubrir esta dirección IP haciendo un **escaneo de la red** con la propia herramienta **Nmap**. Esta es una de las tantas posibilidades con esta herramienta, y es bastante efectiva. El comando que se va a utilizar no se va a explicar, es otro tema diferente.

```
local A ~ /nmap_tests
)) sudo nmap -O 192.168.1.1/24 -oN nmap_01
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-16 15:12 CET
Nmap scan report for 192.168.1.1
Host is up (0.0007s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
443/tcp    open  https
MAC Address: 1C:80:44:4D:76:ED (Askey Computer)
Aggressive OS guesses: Linux 2.6.32 - 3.13 (98%), Linux 2.6.32 - 3.10 (94%), OpenWrt 22.03 (Linux 5.10) (93%), Linux 2.6.39 (93%), Infomir MAG-250 set-top box (92%), OpenWrt 19.07 (Linux 4.14) (92%), AVM FRITZ!WLAN Repeater 450E (FritzOS 6.51) (92%), Hitron CVE-30360 router (92%), DD-WRT v24 or v30 (Linux 3.10) (92%), Linux 3.2 - 3.8 (92%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

Nmap scan report for 192.168.1.33
Host is up (0.00072s latency).
Not shown: 986 closed tcp ports (reset)
PORT      STATE SERVICE
7/tcp     open  echo
9/tcp     open  discard
13/tcp    open  daytime
17/tcp    open  qotd
19/tcp    open  chargen
21/tcp    open  ftp
25/tcp    open  smtp
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
1026/tcp  open  LSA-or-nterm
MAC Address: 08:00:27:F6:F7:D1 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows 2000|XP
OS CPE: cpe:/o:microsoft:windows_2000::- cpe:/o:microsoft:windows_2000::sp1 cpe:/o:microsoft:windows_2000::sp2 cpe:/o:microsoft:windows_2000::sp3 cpe:/o:microsoft:windows_2000::sp4 cpe:/o:microsoft:windows_xp::- cpe:/o:microsoft:windows_xp::sp1
OS details: Microsoft Windows 2000 SP0 - SP4 or Windows XP SP0 - SP1
Network Distance: 1 hop

Nmap scan report for 192.168.1.103
```

Figura 8: Comando para escanear la red y ver la estimación de que sistema operativo usa cada una de las máquinas

Tenemos un resultado para nuestra máquina Windows Server 2000:

```
Nmap scan report for 192.168.1.33
Host is up (0.00072s latency).
Not shown: 986 closed tcp ports (reset)
PORT      STATE SERVICE
7/tcp     open  echo
9/tcp     open  discard
13/tcp    open  daytime
17/tcp    open  qotd
19/tcp    open  chargen
21/tcp    open  ftp
25/tcp    open  smtp
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
1026/tcp  open  LSA-or-nterm
MAC Address: 08:00:27:F6:F7:D1 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows 2000|XP
OS CPE: cpe:/o:microsoft:windows_2000::- cpe:/o:microsoft:windows_2000::sp1 cpe:/o:microsoft:windows_2000::sp2 cpe:/o:microsoft:windows_2000::sp3 cpe:/o:microsoft:windows_2000::sp4 cpe:/o:microsoft:windows_xp::- cpe:/o:microsoft:windows_xp::sp1
OS details: Microsoft Windows 2000 SP0 - SP4 or Windows XP SP0 - SP1
Network Distance: 1 hop
```

Figura 9: Máquina que se asimila a nuestra máquina víctima

Esto lo que va a hacer es un escaneo de puertos de todas formas, y nos da los 'resultados' que estamos esperando con la práctica directamente. Esto no es un problema, ya que se podría modificar el comando para hacerlo con las técnicas que veremos a continuación, solo que para todas las máquinas activas en el rango de la red especificado. No pasa nada, solo es otra opción posible para localizar la máquina víctima.

3.3.2. Resolución básica

Ahora que sabemos la dirección IP de la máquina víctima, podemos hacer un escaneo básico de estos puertos como pide el enunciado. También veremos sus consecuencias.

Para construir el comando utilizaremos varios argumentos del programa Nmap para lograrlo, estos son los que se utilizarán:

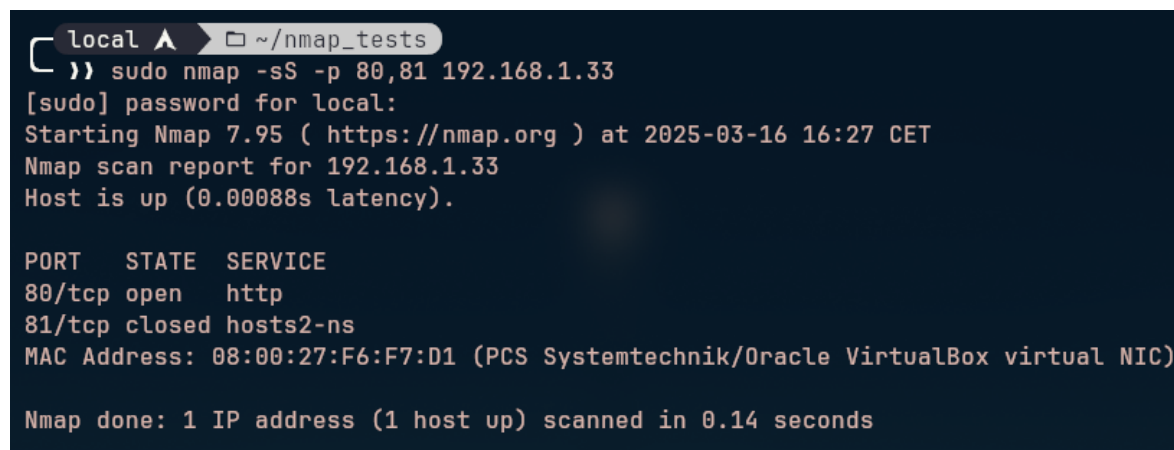
Comando	Descripción
-p	Permite especificar que puertos escanear.
-sS	Si el protocolo utilizado es TCP, utiliza paquetes SYN y no realiza la conexión completamente.

Cuadro 2: Comandos a utilizar durante el ataque

Por lo que quedaría este comando:

```
#!/bin/bash
nmap -sS -p <Puertos> <IP_Victima>
```

Este es el resultado de este escaneo:



```
local ^ ~ /nmap_tests
)) sudo nmap -sS -p 80,81 192.168.1.33
[sudo] password for local:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-16 16:27 CET
Nmap scan report for 192.168.1.33
Host is up (0.00088s latency).

PORT      STATE SERVICE
80/tcp    open  http
81/tcp    closed hosts2-ns
MAC Address: 08:00:27:F6:F7:D1 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.14 seconds
```

Figura 10: Nmap - Escaneo de puertos básico

ip.dst==192.168.1.33 or ip.src==192.168.1.33					
No.	Time	Source	Destination	Protocol	Length Info
19	4.100683564	192.168.1.172	192.168.1.33	TCP	58 50839 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
20	4.100701957	192.168.1.172	192.168.1.33	TCP	58 50839 → 81 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
21	4.101488348	192.168.1.33	192.168.1.172	TCP	60 80 → 50839 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
22	4.101541808	192.168.1.172	192.168.1.33	TCP	54 50839 → 80 [RST] Seq=1 Win=0 Len=0
23	4.101566317	192.168.1.33	192.168.1.172	TCP	60 81 → 50839 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Figura 11: Nmap - Paquetes de escaneo de puertos básico

Vamos a ver que está pasando en este escaneo:

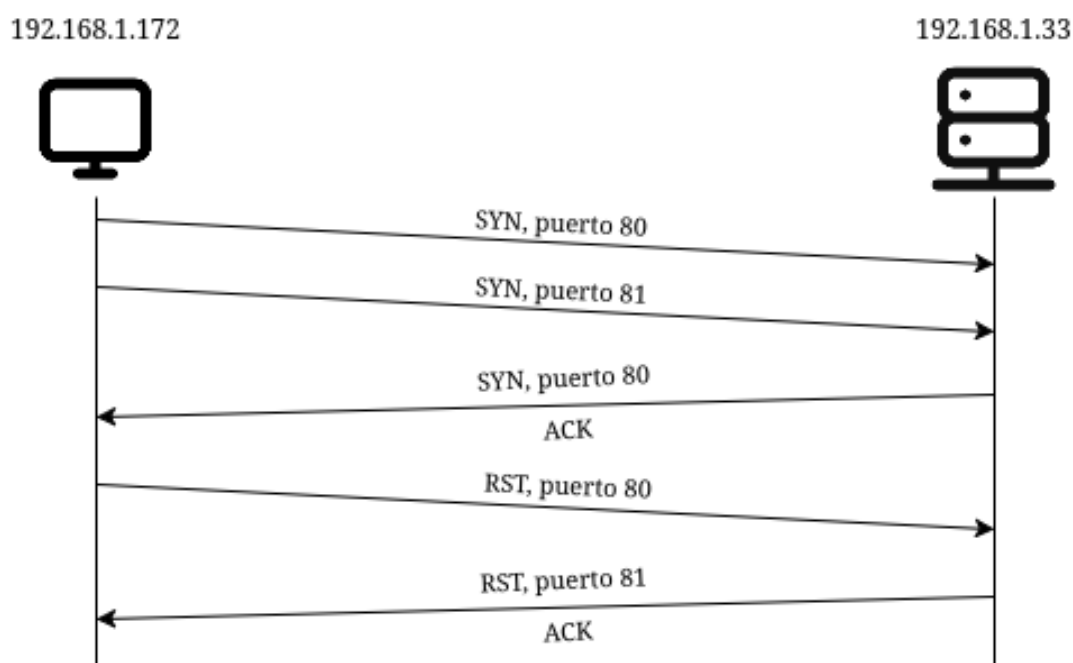


Figura 12: Intercambio de paquetes escaneo puertos básico

Primero, como hemos puesto el escaneo en modo TCP SYN, se le mandan los paquetes pertinentes a la víctima. Esta para el puerto 80 que está abierto nos devuelve un SYN con ACK, es decir, que está lista para la sincronización de la conexión y le ha llegado nuestro anterior paquete, a lo que nosotros le respondemos con un RST, es decir, no queremos seguir con la conexión, ya tenemos lo que queríamos sabiendo que está abierto el puerto. Al siguiente paquete que nos mandan le configuran un RST y un ACK, lo que quiere decir que no pueden continuar con la conexión y les ha llegado bien nuestro paquete.

La conclusión es que sabemos que el puerto 80 está abierto y el 81 no. Nmap también

asume que el puerto 80 tiene servicios HTTP por el número de puerto que contiene, pero con el argumento '-sV' se conectaría al puerto y lo intentaría descubrir por el *Header* o *Banner* de las respuestas, esto también se puede hacer con el script 'http-headers'.

3.3.3. Resolución silenciosa

El escaneo que estamos haciendo ya es bastante silencioso por utilizar la técnica TCP SYN, ya que no crea la conexión completa y por lo tanto puede que los sistemas no lo apunten en sus historiales. Pero puede ser mejor, vamos a usar técnicas más sigilosas junto a este. Estas serán las elegidas:

Comando	Descripción
-p	Permite especificar que puertos escanear.
-sS	Si el protocolo utilizado es TCP, determina utilizar paquetes tipo SYN.
-D	Envía paquetes desde direcciones IP falsas junto a las reales para confundir a la víctima.
-f	Fragmenta los paquetes, envía más paquetes más pequeños para intentar evadir técnicas de detección.
-T1	Usa el modo más lento para reducir la posibilidad de detección.
-Pn	Evita hacer PING al principio de la conexión.
-n	Evita hacer DNS lookup innecesarios.

Cuadro 3: Comandos a utilizar durante el ataque

Así se va a ver el comando final:

```
#!/bin/bash
nmap -sS -T1 -Pn -n -f -D RND:3 -p <Puertos> <IP_Victima>
```

En este caso el parámetro '-D' tiene puestas tres direcciones IP falsas aleatorias, 'RND:3'. Y aquí el resultado de este escaneo:


```
local A ~ /nmap_tests
)) sudo nmap -sS -T1 -Pn -n -f -D RND:3 -p 80,81 192.168.1.33
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-16 17:36 CET
Nmap scan report for 192.168.1.33
Host is up (0.00045s latency).

PORT      STATE SERVICE
80/tcp    open  http
81/tcp    closed hosts2-ns
MAC Address: 08:00:27:F6:F7:D1 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 45.12 seconds
```

Figura 13: Nmap - Escaneo de puertos silencioso

ip.dst==192.168.1.33 or ip.src==192.168.1.33						
No.	Time	Source	Destination	Protocol	Length	Info
77	19.494971673	192.168.1.1	192.168.1.33	ICMP	98	Echo (ping) request id=0x1d2f, seq=0/0, ttl=64 (reply in 78)
78	19.495792124	192.168.1.33	192.168.1.1	ICMP	98	Echo (ping) reply id=0x1d2f, seq=0/0, ttl=128 (request in 77)
116	35.763590157	192.168.1.172	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=0, ID=35f1) [Reassembled in #118]
117	35.763628670	192.168.1.172	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=8, ID=35f1) [Reassembled in #118]
118	35.763634920	192.168.1.172	192.168.1.33	TCP	42	38803 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
119	35.763642199	162.16.224.6	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=0, ID=35f1) [Reassembled in #121]
120	35.763647866	162.16.224.6	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=8, ID=35f1) [Reassembled in #121]
121	35.763653168	162.16.224.6	192.168.1.33	TCP	42	38803 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
122	35.763658971	187.69.179.223	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=0, ID=35f1) [Reassembled in #124]
123	35.763664239	187.69.179.223	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=8, ID=35f1) [Reassembled in #124]
124	35.763670872	187.69.179.223	192.168.1.33	TCP	42	38803 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
125	35.763677490	50.191.218.174	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=0, ID=35f1) [Reassembled in #127]
126	35.763686258	50.191.218.174	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=8, ID=35f1) [Reassembled in #127]
127	35.763692088	50.191.218.174	192.168.1.33	TCP	42	38803 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
128	35.764663807	192.168.1.33	192.168.1.172	TCP	60	80 → 38803 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
129	35.764714498	192.168.1.172	192.168.1.33	TCP	54	38803 → 80 [RST] Seq=1 Win=0 Len=0
130	35.764743082	192.168.1.33	162.16.224.6	TCP	60	80 → 38803 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
131	35.764780861	192.168.1.33	187.69.179.223	TCP	60	80 → 38803 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
132	35.764793609	192.168.1.33	50.191.218.174	TCP	60	80 → 38803 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
133	39.013202003	192.168.1.33	162.16.224.6	TCP	60	[TCP Retransmission] 80 → 38803 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
134	39.013283284	192.168.1.33	50.191.218.174	TCP	60	[TCP Retransmission] 80 → 38803 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
135	39.013296052	192.168.1.33	187.69.179.223	TCP	60	[TCP Retransmission] 80 → 38803 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
143	45.576147772	192.168.1.33	162.16.224.6	TCP	60	[TCP Retransmission] 80 → 38803 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
144	45.576220579	192.168.1.33	50.191.218.174	TCP	60	[TCP Retransmission] 80 → 38803 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
145	45.576243197	192.168.1.33	187.69.179.223	TCP	60	[TCP Retransmission] 80 → 38803 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
153	49.972594542	192.168.1.1	192.168.1.33	ICMP	98	Echo (ping) request id=0x1d90, seq=0/0, ttl=64 (reply in 155)
155	49.973001048	192.168.1.33	192.168.1.1	ICMP	98	Echo (ping) reply id=0x1d90, seq=0/0, ttl=128 (request in 153)
156	50.764588326	192.168.1.172	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=0, ID=7ecf) [Reassembled in #158]
157	50.764624714	192.168.1.172	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=8, ID=7ecf) [Reassembled in #158]
158	50.764631660	192.168.1.172	192.168.1.33	TCP	42	38803 → 81 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
159	50.764639115	162.16.224.6	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=0, ID=7ecf) [Reassembled in #161]
160	50.764645636	162.16.224.6	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=8, ID=7ecf) [Reassembled in #161]
161	50.764650852	162.16.224.6	192.168.1.33	TCP	42	38803 → 81 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
162	50.764656932	187.69.179.223	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=0, ID=7ecf) [Reassembled in #164]
163	50.764662271	187.69.179.223	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=8, ID=7ecf) [Reassembled in #164]
164	50.764667319	187.69.179.223	192.168.1.33	TCP	42	38803 → 81 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
165	50.764673405	50.191.218.174	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=0, ID=7ecf) [Reassembled in #167]
166	50.764678725	50.191.218.174	192.168.1.33	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=8, ID=7ecf) [Reassembled in #167]
167	50.764684924	50.191.218.174	192.168.1.33	TCP	42	38803 → 81 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
168	50.765393309	192.168.1.33	192.168.1.172	TCP	60	81 → 38803 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
169	50.765472216	192.168.1.33	162.16.224.6	TCP	60	81 → 38803 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
170	50.765513752	192.168.1.33	187.69.179.223	TCP	60	81 → 38803 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
171	50.765526416	192.168.1.33	50.191.218.174	TCP	60	81 → 38803 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
181	53.888733607	192.168.1.33	224.0.0.9	IGMPv2	60	Membership Report group 224.0.0.9
182	53.888766711	192.168.1.33	224.0.0.9	IGMPv2	60	Membership Report group 224.0.0.9

Figura 14: Nmap - Paquetes de escaneo de puertos silencioso

Como se puede ver el escaneo es 31 segundos más lento por la carga extra y por hacerlo con la opción 'T1' puesta, pero con todas estas técnicas extra podemos ser aun más indetectables. Vamos a ir paso a paso para ver que está ocurriendo en esta comunicación, solo vamos a ver la del puerto 80, ya que son iguales en cuanto a estructura:

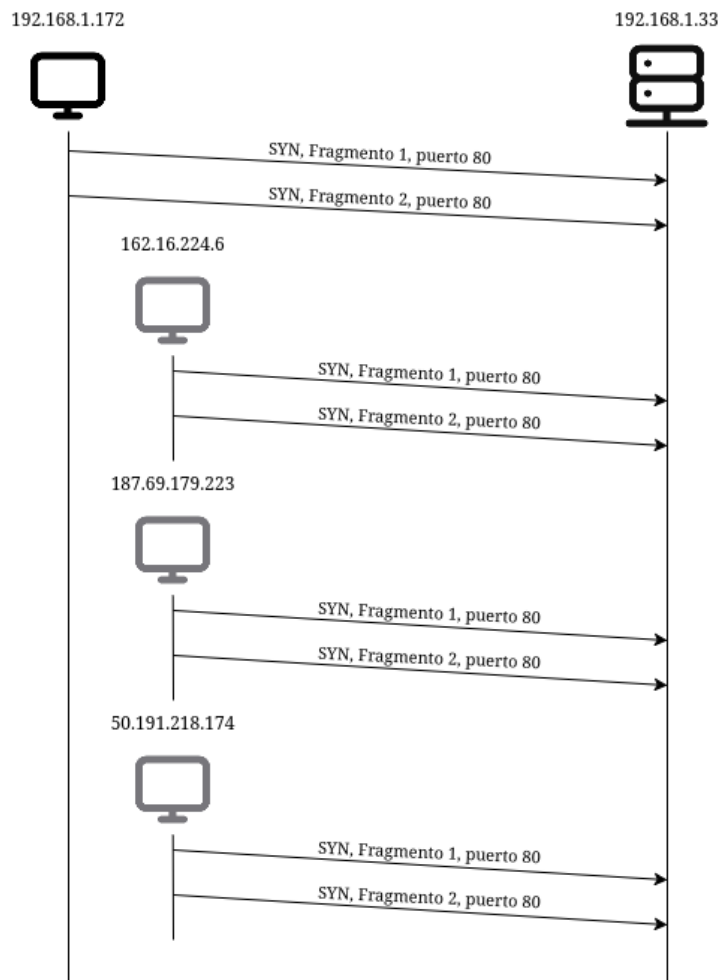


Figura 15: Intercambio de paquetes escaneo puertos silencioso 1

Primero nuestra máquina manda un paquete SYN fragmentado en dos, lo que continúa es el envío del mismo paquete desde tres dispositivos con direcciones IP inventadas por Nmap, también fragmentadas.

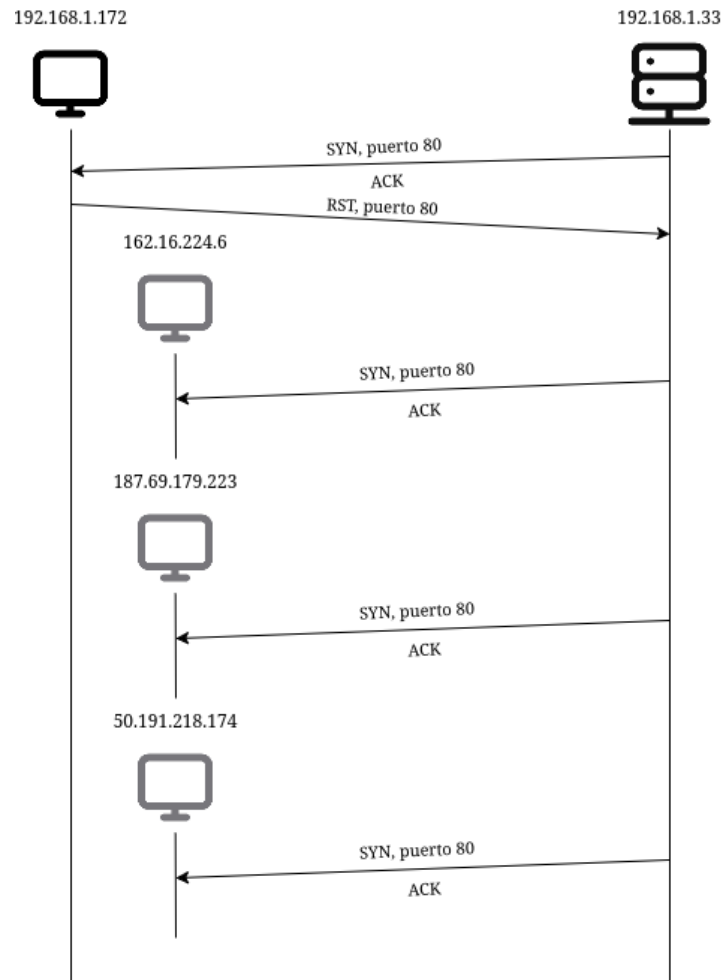


Figura 16: Intercambio de paquetes escaneo puertos silencioso 2

La respuesta del servidor es la natural a nuestra máquina, a la cual respondemos con un RST para terminar la conexión y que no nos registren. Y a las demás con lo mismo, pero estas, al ser inventadas no responden, o mejor dicho, no hacemos que respondan.

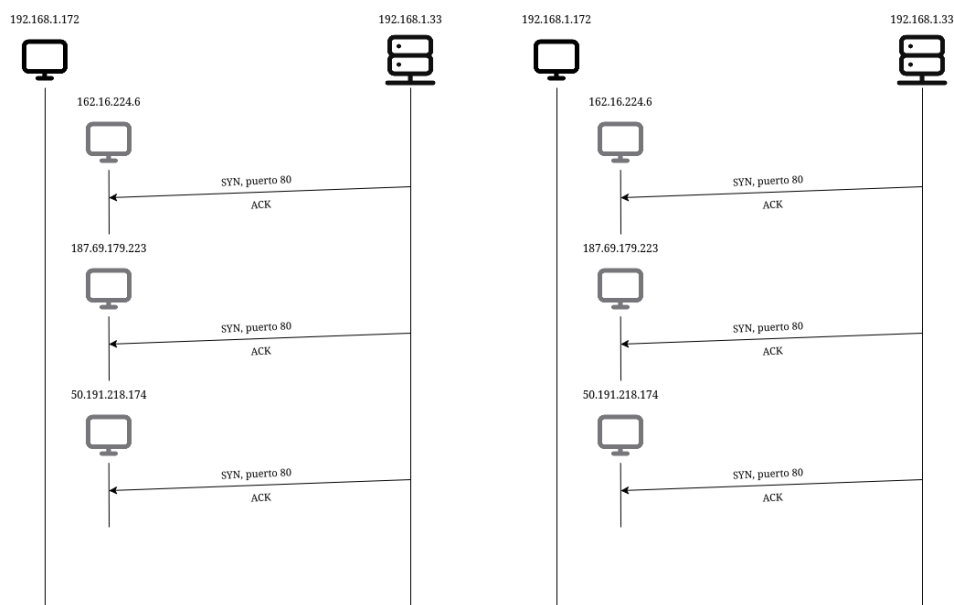


Figura 17: Intercambio de paquetes escaneo puertos silencioso 3

Como las máquinas inventadas no responden, la víctima va a intentar reenviar el paquete dos veces más a ver si con suerte si pueden recibirlo, teniendo en cuenta que el rezonamiento que tiene la víctima es de que el paquete se ha perdido antes de llegar al destinatario. Y al tratarse de una conexión TCP, los va a intentar reenviar.

3.3.4. Resolución zombie

Para la última resolución utilizaremos un tipo de escaneo llamado zombie. Este se aprovecha de una vulnerabilidad en algunos dispositivos 'inactivos' por la cual se incrementa el IP ID, un campo del header de los paquetes, de forma predecible e incremental. Este campo es un conteo del número de transacción de paquete

Por lo tanto, si copiamos la IP del servidor con esta vulnerabilidad haciendo primero una petición para saber su IP ID, y le mandamos a la víctima un paquete con la IP del servidor zombie, la víctima creerá que el servidor zombie le ha hecho una petición y le mandará el resultado. Esto hace que el IP ID aumente, aumentará más o menos dependiendo de la respuesta. Entonces podemos volver a hacer una petición al servidor zombie, y ver en cuanto ha aumentado este número, por lo tanto, sabremos si el puerto de la víctima está abierto o cerrado.

Hay que tener en cuenta que aquí estamos haciendo un escaneo con una máquina externa, que no es nuestra ¡**Necesitamos permiso del dueño!** Por lo que se va a utilizar un equipo propio.

Creamos una segunda máquina Windows Server 2000, la iniciamos y registramos cual es su dirección IP igual que anteriormente.

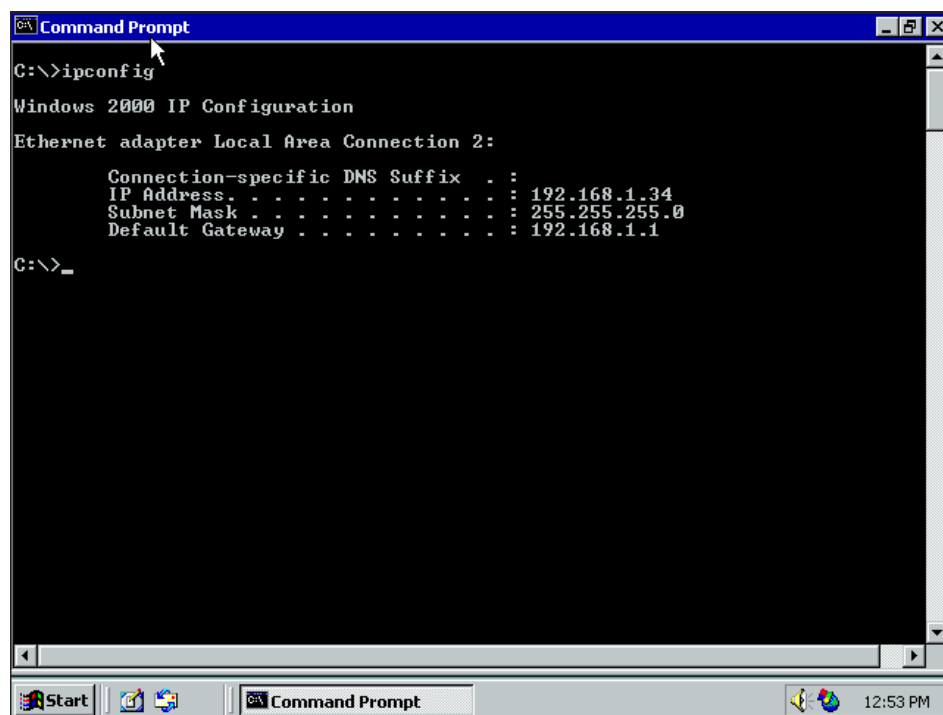


Figura 18: Comprobación de la dirección IP del zombie con un comando en dicha máquina

Así quedaría el comando para este tipo de escaneo, teniendo en cuenta que utilizaremos varios de los comandos anteriores para ser más silencioso:

Comando	Descripción
-p	Permite especificar que puertos escanear.
-Pn	Evita hacer PING al principio de la conexión.
-n	Evita hacer DNS lookup innecesarios.
-sI	Activa el modo zombie, requiere la IP del zombie.

Cuadro 4: Comandos a utilizar durante el ataque

```
#!/bin/bash
nmap -Pn -n -p 80,81 -sI 192.168.1.34 192.168.1.35
```

Nota: La dirección de la víctima ha cambiado por que he tenido que reiniciar la máquina, no cambia nada más.

Ahora vamos a ver los resultados del escaneo:

```
local A ~ /nmap_tests
)) sudo nmap -Pn -n -p 80,81 -sI 192.168.1.34 192.168.1.35
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-17 13:01 CET
Idle scan using zombie 192.168.1.34 (192.168.1.34:80); Class: Incremental
Nmap scan report for 192.168.1.35
Host is up (0.0069s latency).

PORT      STATE      SERVICE
80/tcp    open       http
81/tcp    closed|filtered hosts2-ns
MAC Address: 08:00:27:F6:F7:D1 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.95 seconds
```

Figura 19: Nmap - Escaneo de puertos zombie

ip.addr==192.168.1.34 or ip.addr==192.168.1.35						
No.	Time	Source	Destination	Protocol	Length	Info
20	1.740884326	192.168.1.172	192.168.1.34	TCP	58	52104 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
21	1.741629474	192.168.1.34	192.168.1.172	TCP	60	80 → 52104 [RST] Seq=1 Win=0 Len=0
22	1.771937694	192.168.1.172	192.168.1.34	TCP	58	52105 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
23	1.772538770	192.168.1.34	192.168.1.172	TCP	60	80 → 52105 [RST] Seq=1 Win=0 Len=0
24	1.802902109	192.168.1.172	192.168.1.34	TCP	58	52106 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
25	1.803510650	192.168.1.34	192.168.1.172	TCP	60	80 → 52106 [RST] Seq=1 Win=0 Len=0
26	1.833744168	192.168.1.172	192.168.1.34	TCP	58	52107 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
27	1.834361146	192.168.1.34	192.168.1.172	TCP	60	80 → 52107 [RST] Seq=1 Win=0 Len=0
28	1.864584140	192.168.1.172	192.168.1.34	TCP	58	52108 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
29	1.865056936	192.168.1.34	192.168.1.172	TCP	60	80 → 52108 [RST] Seq=1 Win=0 Len=0
30	1.895262385	192.168.1.172	192.168.1.34	TCP	58	52109 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
31	1.895738098	192.168.1.34	192.168.1.172	TCP	60	80 → 52109 [RST] Seq=1 Win=0 Len=0
32	1.895864654	192.168.1.35	192.168.1.34	TCP	58	52103 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
33	1.946136189	192.168.1.35	192.168.1.34	TCP	58	[TCP Port numbers reused] 52103 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
34	1.996255915	192.168.1.35	192.168.1.34	TCP	58	[TCP Port numbers reused] 52103 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
35	2.046560195	192.168.1.35	192.168.1.34	TCP	58	[TCP Port numbers reused] 52103 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
36	2.346861381	192.168.1.172	192.168.1.34	TCP	58	52355 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
37	2.347565887	192.168.1.34	192.168.1.172	TCP	60	80 → 52355 [RST] Seq=1 Win=0 Len=0
38	2.347807133	192.168.1.34	192.168.1.35	TCP	58	80 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
39	2.398067621	192.168.1.172	192.168.1.34	TCP	58	52189 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
40	2.398677173	192.168.1.34	192.168.1.172	TCP	60	80 → 52189 [RST] Seq=1 Win=0 Len=0
41	2.398895727	192.168.1.34	192.168.1.35	TCP	58	80 → 81 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
42	2.449118867	192.168.1.172	192.168.1.34	TCP	58	52223 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
43	2.449640897	192.168.1.34	192.168.1.172	TCP	60	80 → 52223 [RST] Seq=1 Win=0 Len=0
44	2.474216300	192.168.1.172	192.168.1.34	TCP	58	52139 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
45	2.474866306	192.168.1.34	192.168.1.172	TCP	60	80 → 52139 [RST] Seq=1 Win=0 Len=0
46	2.475109229	192.168.1.34	192.168.1.35	TCP	58	[TCP Retransmission] 80 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
47	2.525289200	192.168.1.172	192.168.1.34	TCP	58	52166 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
48	2.525878363	192.168.1.34	192.168.1.172	TCP	60	80 → 52166 [RST] Seq=1 Win=0 Len=0
49	2.526010470	192.168.1.34	192.168.1.35	TCP	58	[TCP Retransmission] 80 → 81 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
50	2.576197765	192.168.1.172	192.168.1.34	TCP	58	52257 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
51	2.576892455	192.168.1.34	192.168.1.172	TCP	60	80 → 52257 [RST] Seq=1 Win=0 Len=0
52	2.605239378	192.168.1.172	192.168.1.34	TCP	58	52239 → 80 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0 MSS=1460
53	2.605849902	192.168.1.34	192.168.1.172	TCP	60	80 → 52239 [RST] Seq=1 Win=0 Len=0

Figura 20: Nmap - Paquetes de escaneo de puertos zombie

Vamos a analizar uno a uno los paquetes, a ver que sucede en este escaneo:

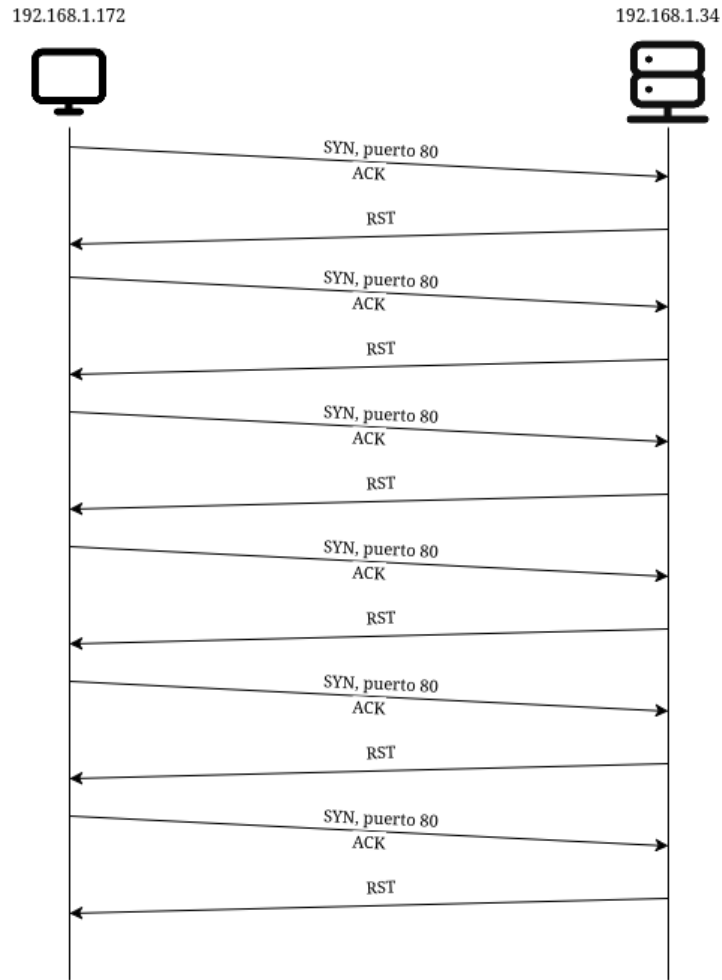


Figura 21: Intercambio de paquetes escaneo puertos zombie 1

Primero que todo, tenemos que comprobar que la máquina zombie cumple con los requisitos suficientes para poder efectuar este ataque. Esto lo hace primero mandando a esta máquina paquetes TCP SYN ACK, lo cual es inválido, con lo que el zombie le responde con RST, exponiendo así su contador IP ID. Con esto, si todo es correcto, podemos comprobar si la máquina nos sirve. En este caso si ha cumplido con los requisitos.

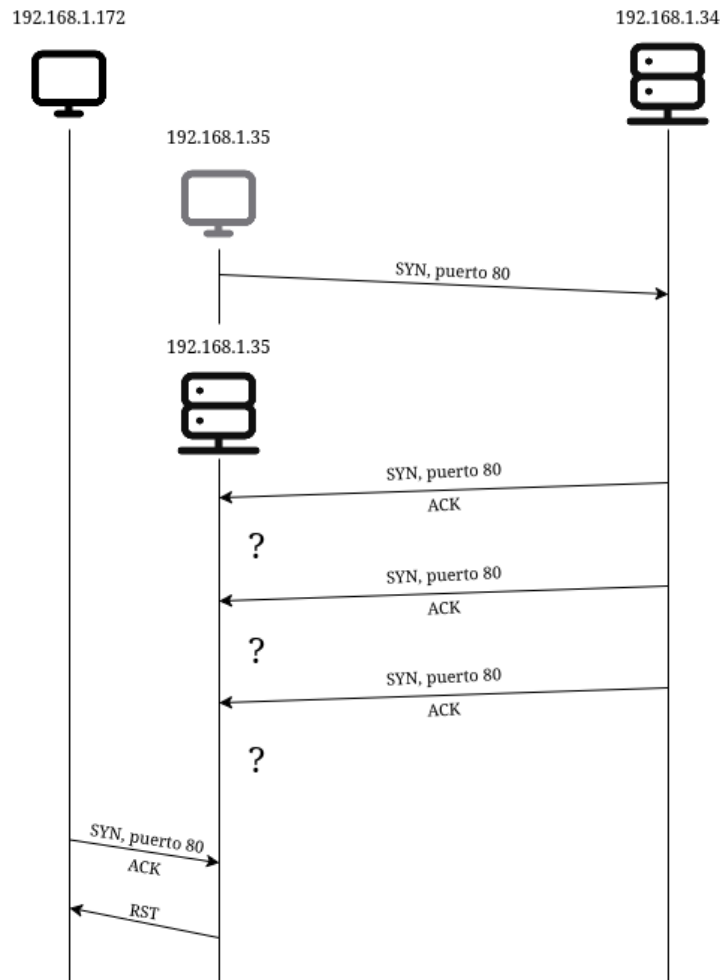


Figura 22: Intercambio de paquetes escaneo puertos zombie 2

Ahora viene lo interesante, ya sabemos el IP ID del zombie, y también la dirección IP, entonces, nos hacemos pasar por el zombie y le mandamos un paquete TCP SYN a la víctima. Esta le responderá con el estado de ese puerto a la máquina zombie real. El zombie no sabrá que hacer con esta respuesta, ya que desde su perspectiva no ha hecho ninguna petición, por lo tanto la ignorará. Al no recibir un paquete de respuesta la víctima reenviará estos paquetes, ya que recordemos que estamos utilizando el protocolo TCP. Después de esto, nosotros contactaremos de nuevo con el zombie real, veremos como ha cambiado el IP ID. Dependiendo de como haya cambiado este sabremos el estado aproximado del puerto que hemos escaneado ¡Jaque mate!

Esto se repite con el puerto 81 igual, con diferente resultado. Y un apunte más, no se sabe si un puerto es filtrado o está cerrado con seguridad, así que Nmap te lo pone en duda.