Máster Universitario en Ingeniería Informática

Administración de Sistemas y Seguridad

ATAQUES DOS Y DDOS

UNIVERSIDAD DE GRANADA

Juan Manuel Castillo Nievas
Carlos Morales Aguilera

Curso Académico 2020-2021

—

Granada, 9 de abril de 2021

Índice

1. Introducción

Los **ataques Dos y DDoS** son dos de los ataques que más amenazan a las empresas actuales. A pesar de ser muy parecidos, cuentan con algunas diferencias y algunas formas de atacar son más frecuentes en uno que en otro. Algunas encuestas de ciberseguridad aseguran que un ataque de este tipo puede llegar a costar entre 20.000-40.000 dólares a la hora [1].

1.1. Protocolos TCP y UDP

Es necesario hacer un breve repaso sobre el **modelo de interconexión de sistemas abiertos**, bien conocido como el **modelo OSI**, para poder entender mejor qué es lo que sucede en estos ataques.

El **modelo OSI** presenta las fases por las que pasan los paquetes de datos para viajar entre diferentes dispositivos sobre una red de comunicaciones. Este modelo se divide en 7 capas que pueden verse en la Figura 1.

#	Capa	Aplicación	Descripción
7	Aplicación	Datos	Procesamiento de red para la aplicación
6	Presentación	Datos	Representación de datos y cifrado
5	*Sesión*	*Datos*	*Comunicación entre hosts*
4	Transporte	Segmentos	Conexiones integrales y confiabilidad
3	Red	Paquetes	Determinación de la ruta y direccionamiento lógico
2	*Enlace de datos*	*Marcos*	*Direccionamiento físico*
1	*Físico*	*Bits*	*Medios, señal y transmisión binaria*

Imagen 1: Capas presentes en el modelo OSI [2]

Dentro de este modelo, los protocolos **TCP** y **UDP** están presentes en la **capa de transporte**.

1.1.1. TCP

El **Protocolo de Control de Transfrencia**, conocido como **TCP** (*Transfer Control Protocol*), es un protocolo **orientado a conexión** cuyo objetivo es **garantizar que los datos lleguen correctamente** a su destinatario, respetando siempre el orden en el que se envían. Que sea un protocolo orientado a conexión significa que ambos equipos deben aceptar la conexión previamente al intercambio de datos.

Para la aceptación de conexión de ambos equipos, en este protocolo se hace lo que se conoce como **3-Way Handshake**, es decir, un **establecimiento de la conexion** (ver Figura 2). Este primer paso asegura la comunicación entre los dos puntos. Esta negociación se divide en tres pasos [3]:

1. El cliente envía un paquete **SYN**. Es una forma de decirle al servidor que quiere establecer una conexión.
2. El servidor responde con un paquete **SYN/ACK**, creando en el paquete **SYN** un **bloque de control de la transmisión** (TCB). Esto le indica al cliente los parámetros que debe utilizar.
3. El cliente responde con un paquete **ACK**. En este momento se establece la conexión y ya puede darse el intercambio de datos.

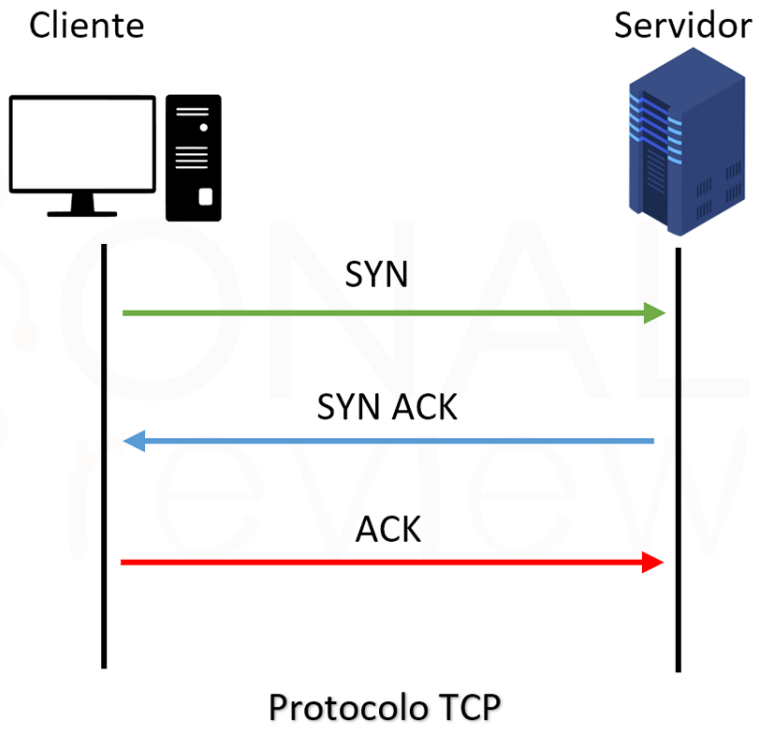

Imagen 2: *3-Way Handshake* en el protocolo TCP [4]

1.1.2. UDP

El **Protocolo de Datagramas de Usuario**, conocido como **UDP** (*User Datagram Protocol*), es un protocolo **no orientado a conexión**. Esto quiere decir que no se necesita hacer un establecimiento de conexión previo entre ambos equipos para el intercambio de datos (ver Figura 3). Durante el envío de datos no se realiza ningún tipo de seguimiento, y mucho menos se asegura que el orden en el que llega la información es correcto. La ventaja de este protocolo es que **los paquetes se envían de una forma mucho más rápida**, pero menos fiable.

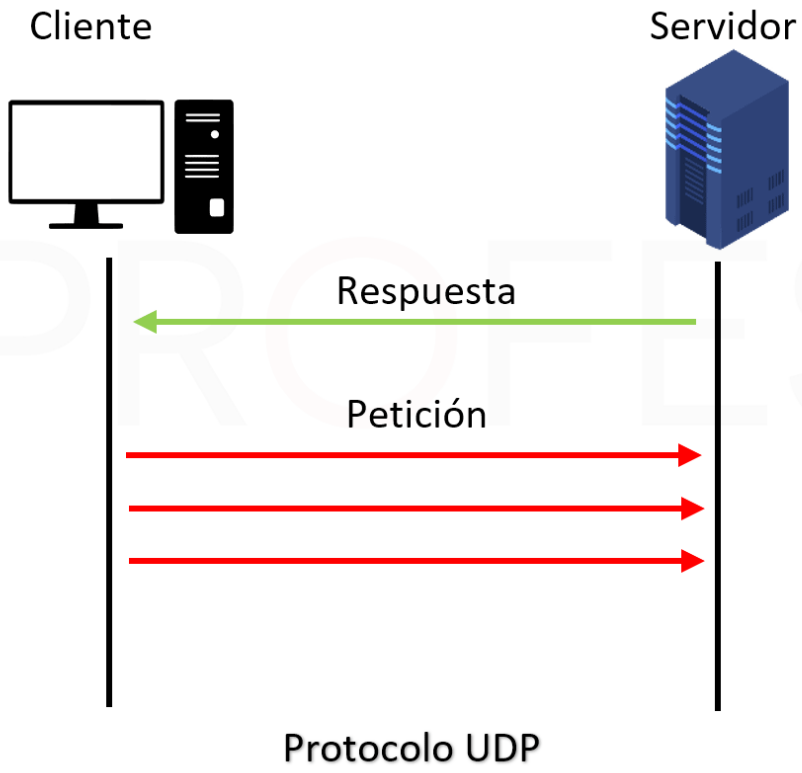

Imagen 3: Intercambio de datos en el protocolo UDP [4]

1.1.3. Diferencia entre TCP y UDP

Hay algunas diferencias entre estos dos protocolos, pero la principal es que TCP es **orientado a conexión**, es decir, necesita hacer un establecimiento de conexión para asegurar que los datos van a llegar correctamente y en el mismo orden en el que fueron enviados. Sin embargo, UDP es **no orientado a conexión**, es decir, no se asegura que los datos lleguen de forma correcta y/o en el mismo orden en el que fueron enviados, pero su transmisión es mucho más rápida. En la Figura 4 se puede ver una imagen que explica de manera muy clara esta diferencia.

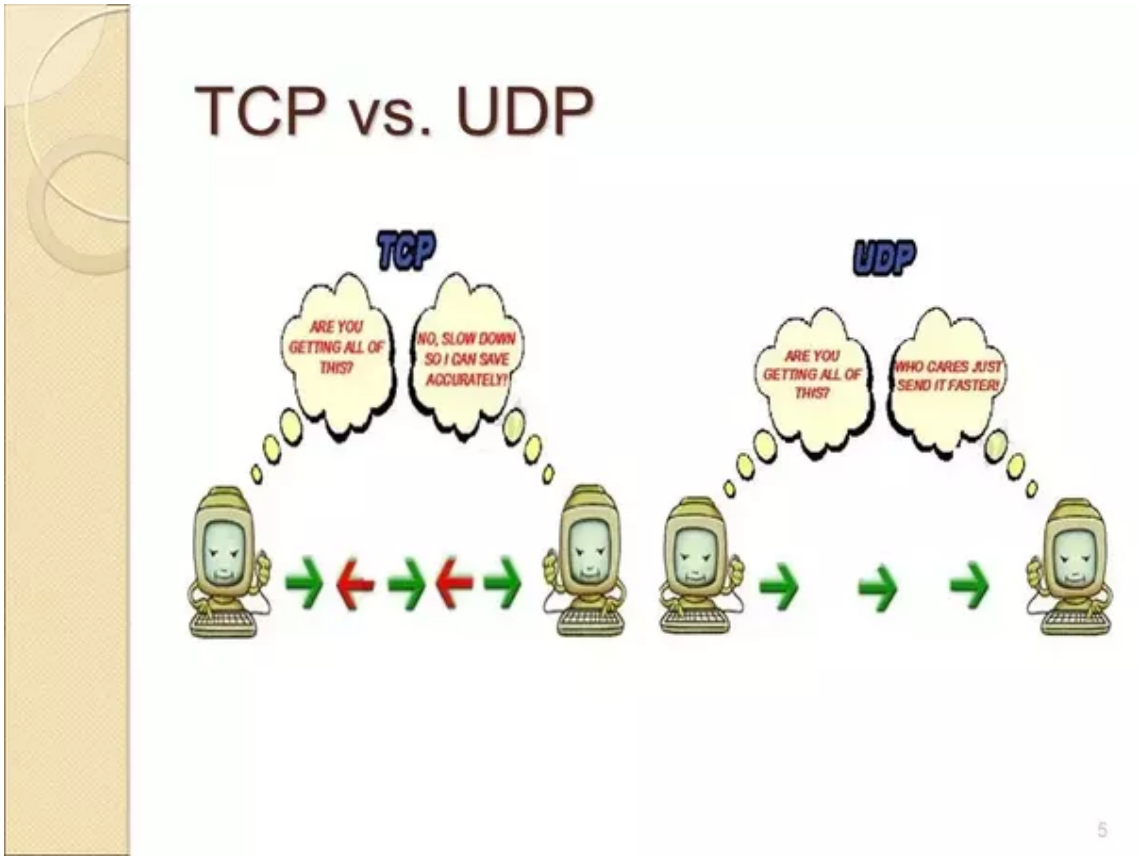

Imagen 4: TCP vs. UDP [4]

2. Ataque DoS

Para entender qué es un ataque **DDoS**, primero debemos empezar por el ataque más simple, **DoS**. Un ataque **DoS**, en inglés *Denial-of-service*, es un ataque de denegación de servicio donde se usa un ordenador para inundar un servidor con paquetes TCP y UDP [1].

Durante este tipo de ataque se envía una cantidad enorme de paquetes a lo largo de la red, de manera que el servidor entra en un estado de **sobrecarga** hasta llegar a un punto en el que el servidor queda **fuera de servicio** para otros dispositivos y usuarios de la red. Su objetivo principal es apagar los servidores y redes de forma que ningún usuario pueda acceder a ellos.

2.1. Ataques más frecuentes

A lo largo de esta Sección se van a mostrar algunos de los ataques más frecuentes DoS [1].

2.1.1. Buffer Overflow

El **desbordamiento de búfer** es el ataque DoS más común. El atacante sobrecarga la red de forma que queda inaccesible.

Los *buffers* son regiones de memoria que almacenan temporalmente datos mientras se envían de una localización a otra. El desbordamiento ocurre cuando el **volumen de los datos supera la capacidad del *buffer***. El resultado es que el programa **escribe en localizaciones adyacentes a la memoria**.

Por ejemplo, cuando se hace log-in en alguna aplicación, los campos de usuario y contraseña pueden estar diseñados de manera que esperan entradas de 8 bytes. Si un usuario hace log-in e introduce 10 bytes (ver Figura 5), la aplicación escribe en localizaciones adyacentes a la memoria [5].

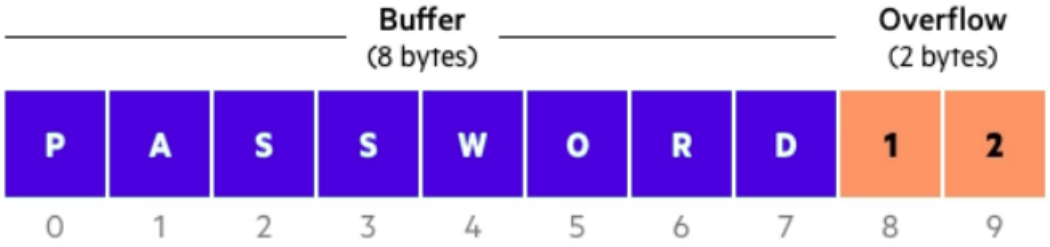

Imagen 5: Ejemplo de debordamiento de búfer [5]

De esta manera, los atacantes se aprovechan del desbordamiento de búfer para sobreescribir la memoria de una aplicación. Esto cambia la ejecución del programa, provocando

respuestas que puedan dañar o exponer información privada. Si los atacantes conocen la estructura de la memoria del programa, pueden reemplazar código ejecutable del programa con su propio código.

2.1.2. Ping de la muerte

Este ataque forma parte de los **ataques más antiguos de Internet** y, aunque producía la caída inmediata de los sistemas más vulnerables, desde el año 1998 ha dejado de tener efecto en la mayoría de dispositivos.

En este ataque lo que se hace es **mandar un paquete de datos malicioso** que cuando el destinatario lo abre provoca la destrucción inmediata de este. Este paquete se envía a través del comando **ping**, que es usado para comprobar si un dispositivo está al alcance de la red.

Para realizar el ping de la muerte es necesario utilizar el **protocolo de mensajes de control de Internet** (ICMP) [6]. El atacante crea un paquete ICMP que supera el máximo tamaño autorizado, que es **65.536 bytes**. Este tamaño es mil veces mayor al límite por paquete establecido en el **protocolo IP**, lo que provoca la caída del sistema y, por lo tanto, la **degenación de servicio**. En la Figura 6 se puede ver un esquema del funcionamiento de este ataque

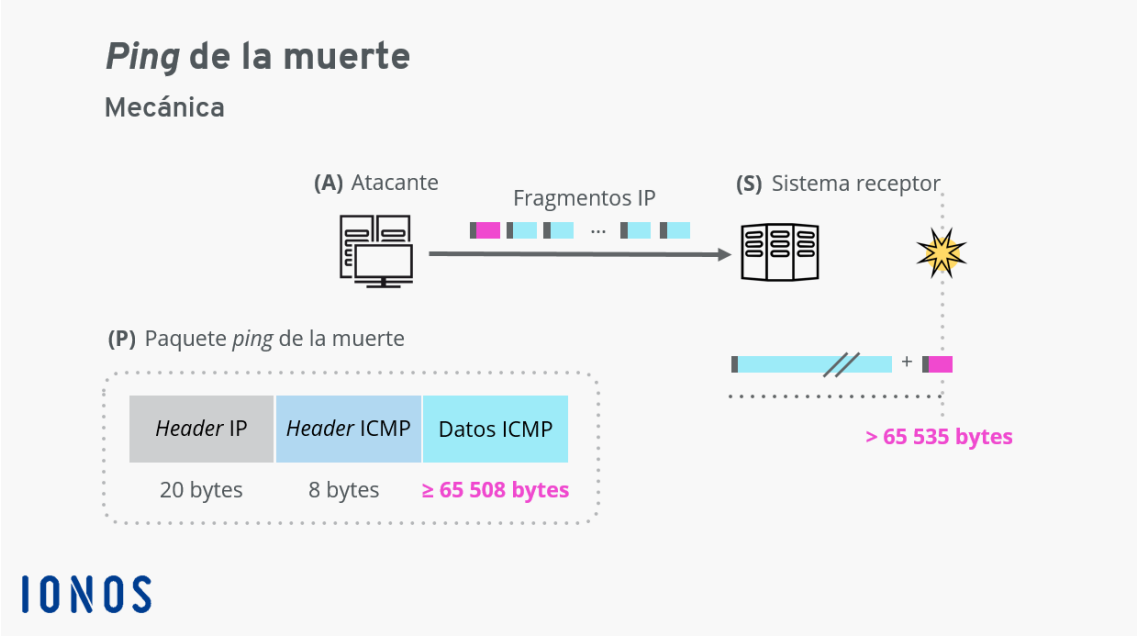

Imagen 6: Esquema del funcionamiento del ping de la muerte [6]

Este ataque se puede realizar mediante una línea de comando muy simple, puesto que lo único que se debe hacer es cargar un paquete con un tamaño de al menos **65.535 bytes**. En sistemas operativos Linux/UNIX/macOS, el ping de la muerte se puede provocar mediante el comando [6]:

```
ping <dirección ip> -s 65500 -t 1 -n 1
```

Debido a que los sistemas operativos ya pueden garantizar que no se exceda el tamaño máximo y que los paquetes maliciosos ya se pueden filtrar al pasar por la red, este tipo de ataque ya no supone ningún problema para casi todos los sistemas modernos.

2.1.3. SYN flood

Al igual que el ping de la muerte que se ha visto en la anterior Sección, la **inundación SYN** es un ataque al protocolo. Este ataque lo que hace es **manipular el *3-Way Handshake* del protocolo TCP**. En vez de establecer una conexión entre el cliente y servidor, lo que se hace es crear múltiples conexiones semiabiertas en el servidor, lo que provoca que los recursos del servidor dejen de estar disponibles para un uso real. [3].

El mecanismo del ataque de inundación SYN es el siguiente [3] (ver Figura 7):

1. El atacante envía un paquete **SYN** con una **dirección IP falsificada**.
2. El servidor crea un **TCB** en la **cola SYN**, ocupando almacenamiento en el servidor.
3. El servidor envía un paquete **SYN/ACK** a la dirección IP falsificada.
4. Como la dirección IP está falsificada, el cliente (el atacante) no recibe ningún paquete **ACK** que confirme la conexión y el servidor envía más paquetes **SYN/ACK**, manteniendo de esta forma la conexión semiabierta.
5. Como el servidor se queda a la espera de una respuesta, el atacante sigue enviando paquetes **SYN** que quedan registrados en la **cola SYN**.
6. Cuando ya no queda espacio en la **cola SYN**, el servidor rechaza todos los paquetes **SYN** entrantes provocando la inaccesibilidad desde cualquier conexión real.

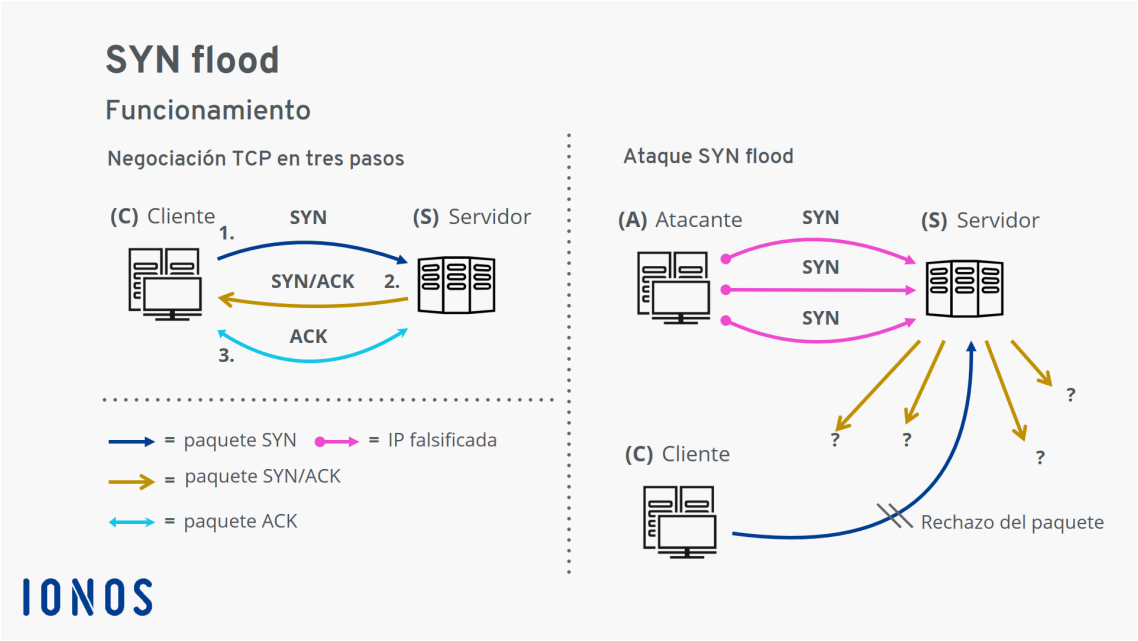

Imagen 7: Esquema del *3-Way Handshake* y la inundación SYN [3]

Para realizar un ataque de este tipo se debe usar un software especial como la herramienta **hping**, que permite simular diferentes ataques a la red. Un código aproximado para simular una inundación SYN es el siguiente [3]:

```
hping -syn -flood -rand-source -p <puerto><dirección IP>
```

Las opciones del comando son las siguientes:

- `-syn`: indica que se debe utilizar el protocolo TCP y enviar paquetes SYN.
- `-flood`: para enviar los paquetes lo antes posible.
- `-rand-source`: se introduce una dirección IP aleatoria y falsa.

Estos ataques se conocen desde el año 1994 y, aunque hay medidas defensivas muy eficaces, es complicado distinguir los paquetes SYN maliciosos de los legítimos.

2.1.4. Teardrop

Se conoce como un **ataque de goteo**, y durante este ataque se envían **paquetes fragmentados** al servidor. La red intenta recomponer los fragmentos para volver a montar el paquete pero en muchos casos este proceso agota al sistema y termina quedando fuera de servicio.

El motivo por el que la reconstrucción de los fragmentos termina colapsando el sistema es porque se diseñan de forma que **confundan al sistema** y no puedan reconstruirlos de forma correcta [7] (ver Figura 8).

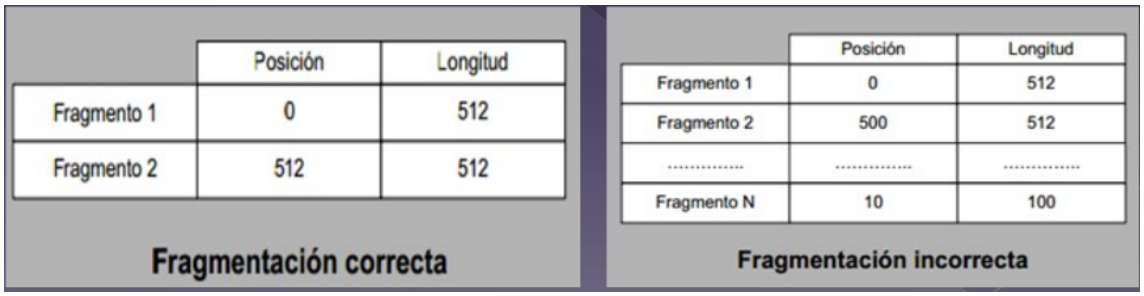

Imagen 8: Fragmentación en el ataque *teardrop* [7]

3. Ataque DDoS

Realmente un ataque **DDoS** es un tipo de ataque **DoS**. Recibe el nombre de **denegación de servicio distribuido**. Es uno de los ataques **DoS** más comunes y, durante su ataque, muchos sistemas apuntan a un sólo servidor con datos maliciosos.

Los ataques **DDoS** son mucho más complicados de parar debido a la cantidad de ordenadores esclavos (bots) que se usan para atacar remotamente a las víctimas. Todos estos bots forman una red de dispositivos conectados y se conoce como lo que son las **botnets**.

Una **botnet** es un grupo de ordenadores controlados por un hacker [8]. Esto implica que el daño que hace el hacker se multiplica, y además el origen del ataque no es el propio ordenador del atacante, haciendo casi imposible rastrearlo.

Las botnets se pueden usar casi para cualquier ataque, pero sí que es cierto que su mayor uso se da en los ataques **DDoS**, además de los **fraudes publicitarios en las páginas webs** y el **correo spam**. La cantidad de ordenadores que forman una botnet van desde los **100.000 hasta los 5.000.000**.

3.1. Ataques más frecuentes

Los ataques **DDoS** también se pueden hacer de la misma forma que los ataques **DoS** vistos en la anterior Sección, pero los más comunes son los que se presentan a lo largo de esta Sección [1].

3.1.1. UDP floods

Tal y como su nombre indica, este ataque se produce por una **inundación de UDP**. Su objetivo principal es saturar el sistema de destino con una entrada masiva de datos [9].

En la Figura 9 se puede ver un esquema de cómo funciona la inundación UDP. Su proceso es el siguiente [9]:

1. El atacante envía paquetes **UDP** con una **dirección IP falsificada** a puertos alea-

torios.

2. El servidor de destino repite el siguiente proceso para cada paquete entrante:

   a) Comprueba si el puerto especificado está ocupado.

   b) Envía un paquete **ICMP** *"Destination Unreachable"* a la dirección IP falsificada, que suele llegar a un tercero.

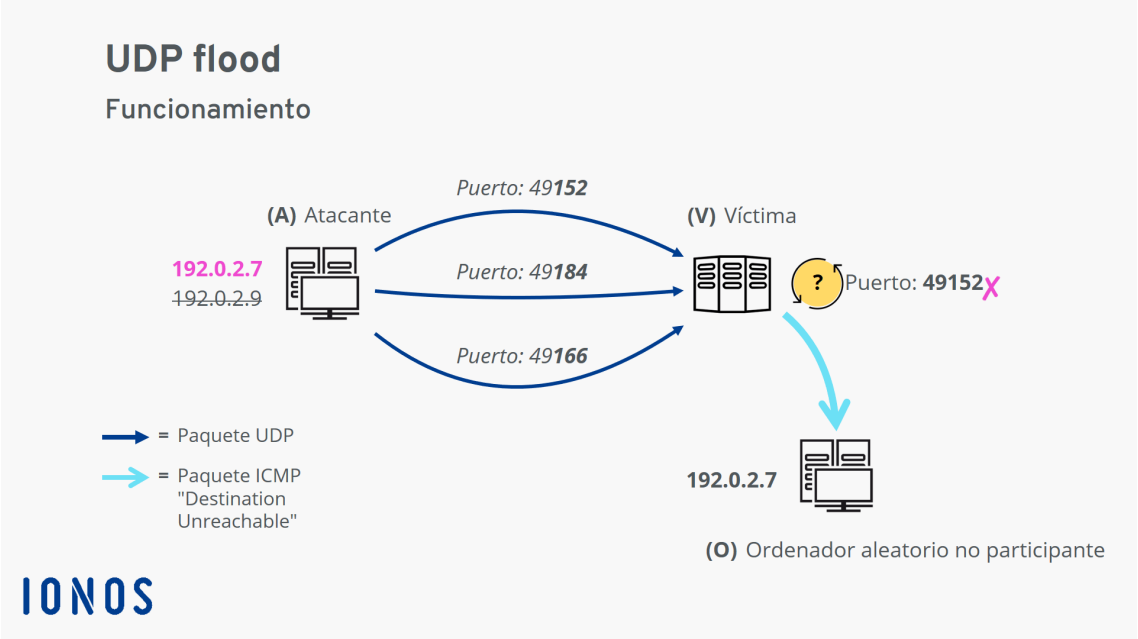

Imagen 9: Esquema de la inundación UDP [9]

Una posible solución para mitigar estos ataques es la **distribución del tráfico de red a una enorme cantidad de centros de datos** en todo el mundo, recurriendo para ello a **servicios en la nube**. Esto permite tener más ancho de banda, amortiguando mejor la gran cantidad de paquetes entrantes.

3.1.2. Slowloris

El principal objetivo de este ataque es sobrecargar un servidor manteniendo **abiertas** muchas **conexiones HTTP simultáneas** [10]. En la Figura 10 se puede ver de manera gráfica el proceso del ataque.

Slowloris ataca en la **capa de aplicación** del modelo OSI. Lo que hace es abrir una conexión a un sitio web e intenta mantenerla activa el máximo tiempo posible. De esta manera intenta tirar un servidor **sin usar mucho ancho de banda**.

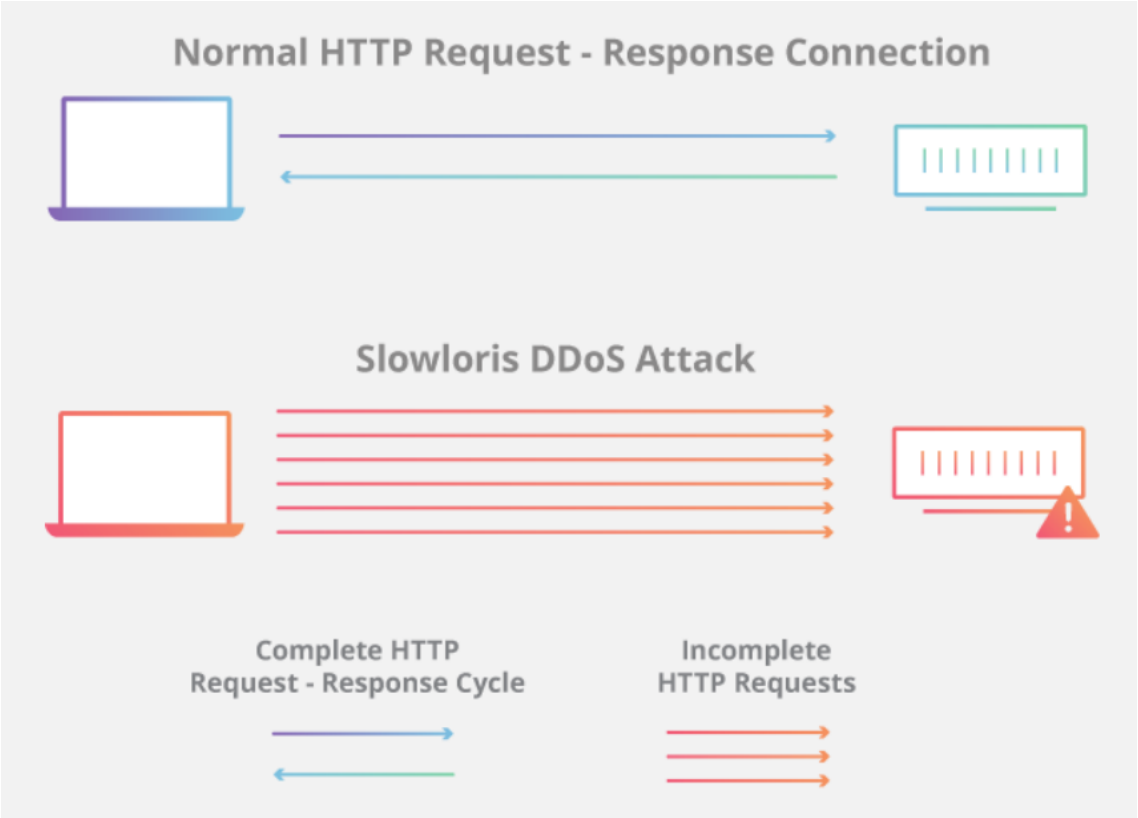

Imagen 10: Ataque DDoS Slowloris [10]

Este ataque se divide en cuatro etapas [10]:

1. El atacante abre **múltiples conexiones** del servidor web enviando muchas **peticiones parciales HTTP**.
2. El servidor abre **una hebra para cada petición** intentando cerrar cada una cuando la conexión finaliza. Si una conexión toma demasiado tiempo, el servidor dará *timeout* liberando la hebra correspondiente.
3. El atacante envía periódicamente las **cabeceras de las peticiones** parciales para **mantener la solicitud viva** y así prevenir el *timeout*.
4. El servidor nunca libera estas peticiones y por lo tanto llegará un momento en el que no podrá atender una petición de alguien real.

Para los servidores más vulnerables, existen tres formas de mitigar estos ataques [10]:

1. **Incrementar la disponibilidad del servidor**: se incrementa el número de clientes que el servidor puede atender al mismo tiempo.
2. **Limitar el tráfico de las peticiones entrantes**: hay diversas técnicas como limitar el máximo número de conexiones que una dirección IP puede hacer, limitar

el máximo tiempo que un cliente puede estar conectado al servidor y restringir la velocidad de transferencia.

3. **Protección basada en la nube**: se usa un servicio que funciona como un proxy invertido, protegiendo el servidor original.

3.1.3. HTTP Flood

Este ataque se encarga de **sobrecargar al servidor mediante peticiones HTTP**, saturando a este y haciendo que el servidor no pueda responder al tráfico normal de usuarios reales (ver Figura 11). Al igual que **Slowloris**, este ataque opera en la **capa de aplicación** del modelo OSI.

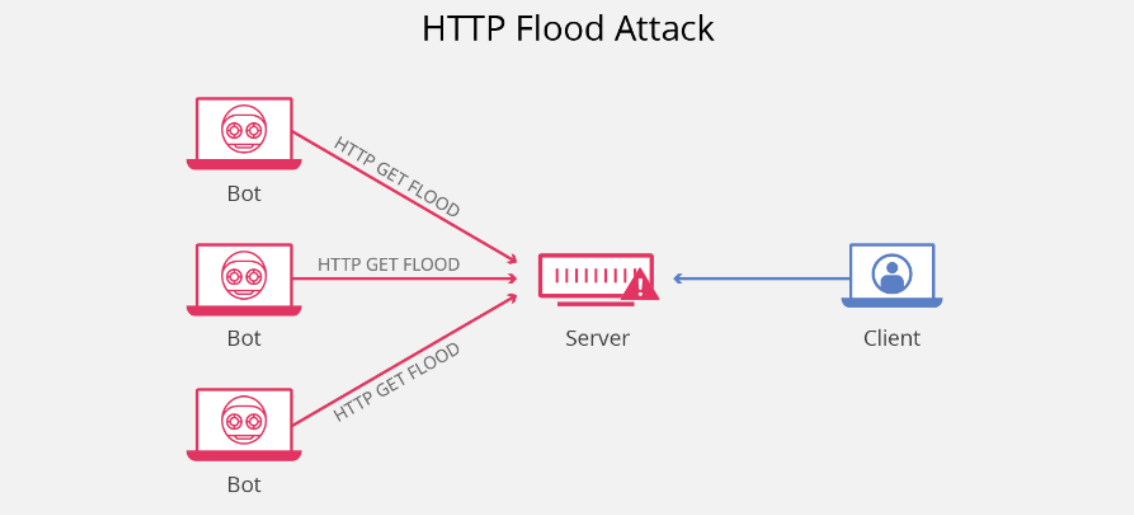

Imagen 11: Ataque DDoS HTTP Flood [11]

Hay dos tipos de inundaciones HTTP [11]:

1. **Ataque HTTP GET**: en este ataque, muchos ordenadores o dispositivos se coordinan para enviar peticiones de imágenes, archivos u otros recursos del servidor, provocando la denegación de servicio.

2. **Ataque HTTP POST**: las peticiones POST ocurren cuando el usuario rellena un formulario en un sitio web. Normalmente el servidor envía los datos de la petición a la base de datos. Este proceso de resolver la petición POST es mucho más intenso que el hecho de enviar la petición en sí por parte del usuario. El atacante envía muchas peticiones POST saturando el servidor y provocando la denegación de servicio.

Este tipo de ataques es difícil de mitigar. Una posible solución es implementar los famosos *captcha* para determinar si un usuario es o no es un bot.

3.1.4. Ataques de día cero

Estos ataques se aprovechan de **vulnerabilidades de seguridad que aún no han sido parcheadas**. Su nombre se debe a que las empresas u organizaciones disponen de **0 días** desde que se descubre la amenaza para parchear dichas vulnerabilidades [12]. Estos tipos de ataque son especialmente **devastadores** ya que la víctima **no tiene ninguna forma de prevenirlos**.

Cuando un atacante identifica una vulnerabilidad que aún no ha sido descubierta, escribe código que tiene como objetivo dicha vulnerabilidad y lo **empaqueta en un *malware*** que cuando es ejecutado daña al sistema.

Hay varias maneras de hacer que un atacante pueda distribuir este código. La táctica más común es la de utilizar *phising* mediante correos que contienen **archivos adjuntos o enlaces** que contienen el ataque.

Aunque actualmente no hay ninguna forma que prevenga al 100 % los ataques de día cero, existen dos tecnologías muy importantes que ayudan a prevenir la explotación de vulnerabilidades [12]:

- **Aislamiento del navegador**: el objetivo de esto es hacer que el código malicioso no se ejecute en el dispositivo del usuario. Se puede hacer de tres formas:
  1. **Aislamiento remoto del navegador**: las páginas webs se cargan y su código se ejecuta en un servidor en la nube, lejos de los dispositivos de los usuarios finales.
  2. **Aislamiento del navegador en local**: funciona de manera muy similar al anterior, pero esta vez ejecutando el código de la página web en un servidor interno.
  3. **Aislamiento del navegador del lado del cliente**: se utiliza la técnica de seguridad informática **sandboxing**, que ejecuta aplicaciones en un espacio virtual limitado, asegurando que el código se ejecuta de forma independiente al dispositivo.
- **Firewall**: se utiliza un *firewall* para bloquear el tráfico que sea sospechoso de ser un atacante de la vulnerabilidad.

4. Diferencia entre DoS y DDoS

La principal diferencia entre **DoS** y **DDoS** es que este último utiliza múltiples conexiones mientras que en el primero se hace uso de una sola conexión (ver Figura 12). Los ataques **DDoS** son más difíciles de detectar porque no se puede saber el origen del ataque.

Hay que remarcar que la forma de ejecutarse también es diferente. En los ataques **DDoS** se hace uso de las **botnets**, mientras que en los ataques **DoS** se lanzan desde un script o alguna herramienta como **Low Orbit Ion Cannon** [1].

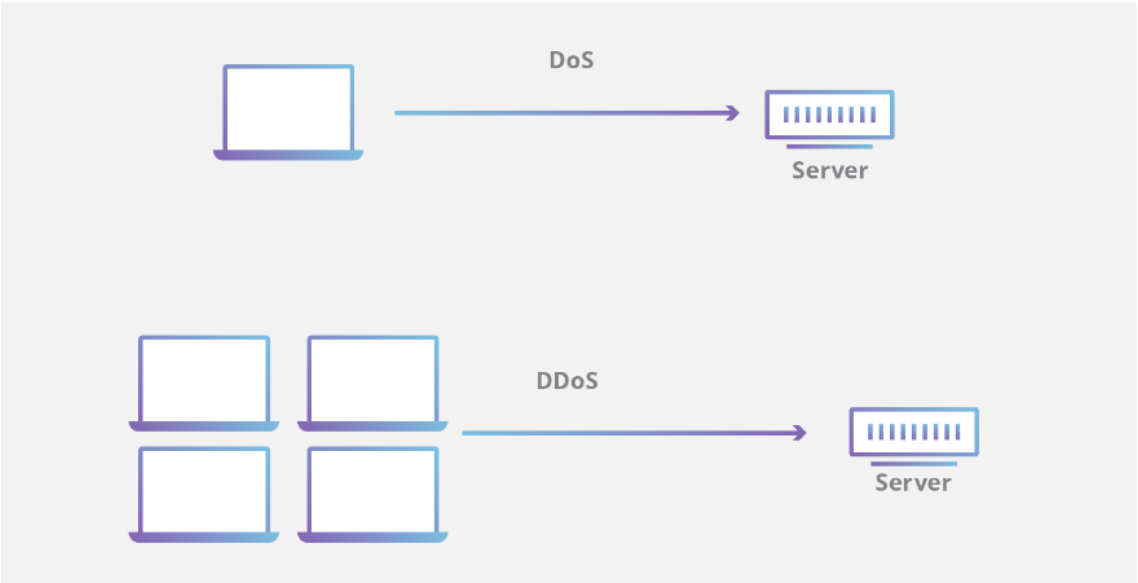

Imagen 12: **DoS vs. DDoS** [13]

5. Casos reales ataques DoS

5.1. Casos reales Buffer Overflow

5.1.1. Caso de SQL Slammer

Uno de los principales casos conocidos de ataques de tipo **Buffer Overflow** se dió a comienzos del año 2003, cuando nace un gusano informático, el cual enviaba un paquete de aproximadamente 380 bytes al puerto **1434 UDP**, y provocando que los sistemas vulnerables empezaran a enviar el mismo paquete provocando dicho ataque de denegación de servicio. Este ataque se produjo en productos de **Microsoft SQL Server** donde no se encontraba instalado el **Service Pack 3**.

Este ataque llegó a registrar casos en un total de 170 países, haciendo que países como Corea del Sur sufriera un apagón de Internet y que en Estados Unidos cerca de 13.000 cajeros automáticos se vieran afectados, denegando su servicio. El origen de dicho ataque fue reportado en países como México, China, Vietnam o Ucrania.

Las máquinas domésticas fueron afectadas en menor medida, salvo que tuvieran instalado **Microsoft SQL Server Data Engine**, ya que al ser un código tan pequeño no tenía capacidad para copiarse en discos duros, por lo que permanecía en memoria y podía ser eliminado fácilmente. Curiosamente existía cerca de 6 meses antes un parche de seguridad que protegía de dicha vulnerabilidad pero no fue implementada [14].

5.1.2. Caso del Gusano Morris

Considerado por muchos como el primer **Malware** de la historia de Internet (por aquel entonces **ARPANET**), creado en 1988 y que supuso una infección sobre aproximadamente el 10 % de todos los equipos conectados a la red, protagonizando la primera condena de la historia por **violación de la Ley de Fraude y Abuso Informático**.

El bug originalmente se trataba de una prueba para medir el tamaño de Internet mediante la infección de sistemas **UNIX**, pero por un **error de programación** fue liberado y difundido por toda ARPANET mediante conexiones **TPC** y **SMTP** y se basaba en la explotación de vulnerabilidades, implicando la ralentización de equipos, caída de sistemas, robos de credenciales, etc.

Por otro lado también el malware se protegía a sí mismo para evitar su intercepción, cifrando datos de la memoria entre otros métodos, llegando a provocar la organización de grupos de trabajos en las Universidades de **Berkley** y el **MIT**. Por suerte, fue finalmente solucionado en los dos días siguientes [15].

5.2. Casos reales SYN flood

5.2.1. Caso de Panix

Uno de los primeros ataques documentados públicamente se lanzó en septiembre de 1996 contra **Panix**, el proveedor de servicios más antiguo de Internet en Nueva York. El ataque se dirigió a diferentes equipos de la red del proveedor, como los servidores de correos, noticias, nombres, etc. Esto supuso entre otras cosas la caída de las máquinas de login de usuarios.

Este ataque, conocido como el famoso del tipo **SYN Flood** agotó las conexiones de red disponibles y evitaba que usuarios se conectaran a los servidores de Panix, requiriendo cerca de 36 horas de trabajo frenético por diferentes especialistas de diferentes lugares del mundo para recuperar el **control de los dominios** de Panix.

Este ataque fue realizado mediante una dirección IP falsa que saturó los servidores de la empresa con paquetes de tipo SYN falsos, obligando a dejar de procesar solicitudes reales y la posterior caída del sistema con todos las pérdidas que esto supuso [16].

5.3. Casos reales Teardrop

5.3.1. Caso de Windows Vista

Más que un ataque como tal, se trata de un caso donde se reportó una de las vulnerabilidades más conocidas sobre ataques de tipo **Teardrop**. Se trataba de un código que explotaba un fallo sobre el reinicio remoto, haciendo uso de la implementación del protocolo **SMB2**, el cual expuso a los usuarios de **Windows 7** y **Windows Vista** a ataques de este tipo.

Esta vulnerabilidad permitía a un atacante bloquear de forma remota cualquier máquina con estas versiones con **SMB** habilitado, sin requerir siquiera una acción por parte del usuario. Esto se debía a la exposición del driver vulnerable, por lo que versiones previas como **XP** que no lo mostraban accesible, no eran vulnerables a ataques de este tipo [17].

Por curiosidad, el error que se mostraba era el siguiente:

SRV2.SYS fails to handle malformed SMB headers for the NEGOTIATE PROTOCOL REQUEST functionality. The NEGOTIATE PROTOCOL REQUEST is the first SMB query a client send to a SMB server, and it's used to identify the SMB dialect that will be used for further communication.

6. Casos famosos de ataques DDoS

6.1. Google

Uno de los casos más sonados en la actualidad, fue para sorpresa de muchos un caso bastante desconocido en su día, el cual es el caso de Google en 2016, el cual recibió un

ataque de tipo DDoS, concretamente de **UDP Flood** según reportaban el 16 de Octubre en un artículo en su propio foro. Este ataque fue detectado por el **Grupo de Análisis de Amenazas de Google (TAG)**, el cual afirma que el ataque fue realizado patrocinado por el estado mediante una amenaza.

El ataque fue detectado con origen en China, dentro de cuatro diferentes proveedores de servicios de Internet chinos, y que llegó a alcanzar la cifra de **2.54Tbps**, siendo este el pico de un ataque con una duración de aproximadamente 6 meses con el objetivo de tumbar la infraestructura de servidores de Google, según reportan algunos ingenieros de la propia empresa.

El atacante utilizó diferentes redes para falsificar **167 Mpps (millones de paquetes por segundo)** a **180.000 servidores CLDAP, DNS, SMTP** que se encontraban expuestos y que como consecuencia enviarían grandes conjuntos de respuestas.

Finalmente el equipo de Google TAG quiso reportar un ataque tan grande en la sombra con el objetivo de crear conciencia de una tendencia cada vez mayor de grupos de piratas informáticos estatales que utilizan este tipo de ataques para interrumpir objetivos, y advierte que este tipo de ataques tenderá a intensificarse en el futuro con el aumento del ancho de banda de Internet [18].

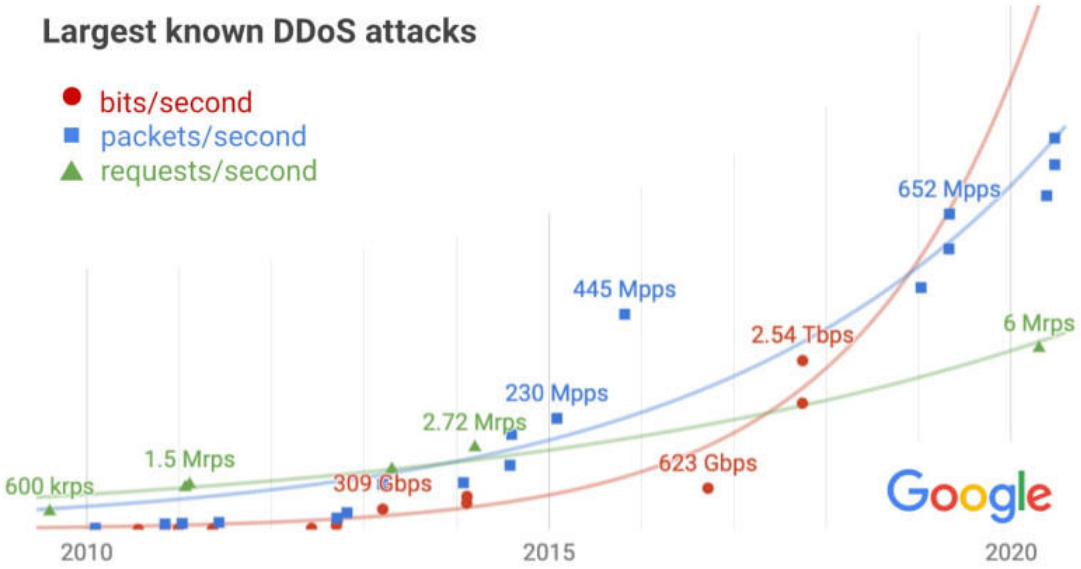

Imagen 13: Tendencia sobre ataques **DDoS** según **Google**.

6.2. AWS

En 2020 se alcanzó uno de los casos más sonados y recientes en lo que a ataques DDoS se refiere, registrado por el gigante **Amazon Web Services (AWS)**. Este ataque llegó a alcanzar los **2.3Tbps** el primer trimestre de 2020, superando el ataque récord previo por casi un **70** %.

La propia compañía llego a revelar que los atacantes trataron de tirar los servidores utilizando el protocolo **CLDAP (Connection-less Lightweight Directory Access**

Protocol), llegando a suponer una gran amenaza para el equipo de **Amazon AWS Shield**, durante el periodo de tres días.

Según la información revelada por la compañía, los atacantes trataron de tumbar los servidores utilizando el protocolo CLDAP (Connection-less Lightweight Directory Access Protocol), causando una gran amenaza para el personal de Amazon AWS Shield, durante tres días, a mediados de febrero de este año.

Este ataque llegó a suponer un récord y se hizo un gran eco, ya que no sólo habían sido capaces de mitigar dicho ataque, sino por el peligro que suponía el tamaño de dicho tipo de ataques, e incluso a día de hoy se desconoce cual era el cliente objetivo real de dicho ataque [19].

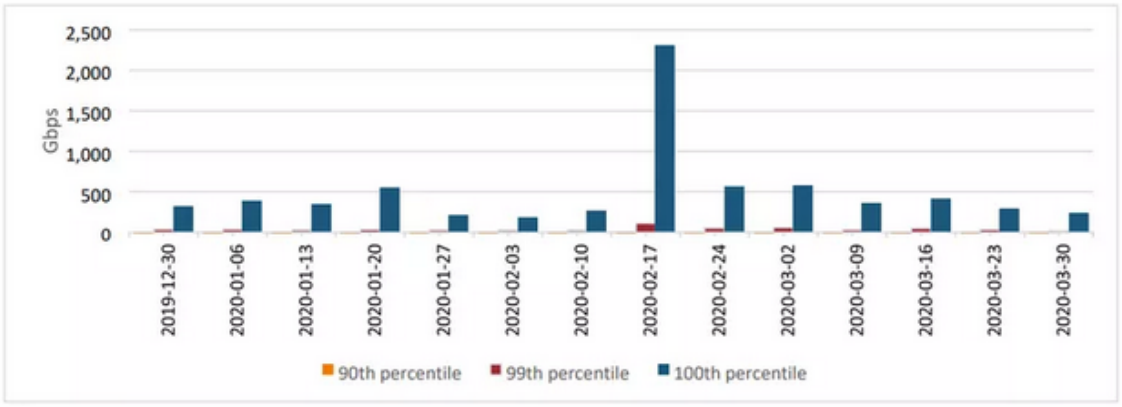

Imagen 14: Estadísticas del ataque a **AWS** en **Gbps**.

6.3. Mirai botnet

Antes de hablar de los ataques cometidos, es necesario definir que es el **Mirai botnet**, el cual es un malware que se aprovecha de centenares de miles de dispositivos inteligentes o de **IoT** conectados. Instal el malware, se hace con el control y básicamente recluta un ejército mundial obteniendo acceso a dispositivos con contraseñas predeterminadas. Finalmente se lanza un ataque de DDoS a un objetivo común.

El primer ataque conocido se fecha el 20 de Septiembre de 2016, en el conocido blog del experto en ciberseguridad **Brian Krebs**, el cual ha sido atacado en numerosísimas ocasiones, pero esta vez fue distinta ya que se llegó a alcanzar una carga de **620 Gbps**, algo inalcanzable hasta ese preciso momento utilizando el malware de Mirai.

El siguiente ataque utilizando el Mirai botnet fua hacia uno de los principales servicios de hosting OVH, sobre un único cliente y llevado a cabo por aproximadamente unos **145.000 bots**, generando un tráfico de **1Tbps**, durante casi 7 días.

Por último cabe destacar el caso más conocido de Mirai botnet, poco más tarde en ese mismo año, un usuario divulgó el código del malware y se realizaron numerosas copias con diferentes mutaciones del algoritmo. El 21 de Octubre se produjo el mayor ataque conocido por este malware, a **Dyn**, una empresa proveedora de **DNS**, llegando a alcanzar un tráfico de **1.5Tbps** como pico, dejando páginas y servicios como GitHub, HBO, Twitter, Reddit,

PayPal, Netflix y Airbnb inaccessibles. Se llegó a detectar un total de **10 millones de IPs** asociadas al Mirai botnet [19].

6.4. GitHub

En el año 2018 recibió un ataque del tipo DDoS, llegando a alcanzar un total de **1.35Tbps** provocando durante unos minutos el colapso total del sistema. Pese a encontrarse normalmente preparados para este tipo de ataques, como ellos mismos afirmaban, era una situación imprevisible y a la que pese a su respuesta, fue inevitable dicha catástrofe ya que era impredecible un ataque a tan gran escala. Para ello se utilizó una amplificación de vector que utilizaba **Memcached** sobre **UDP**.

Fue uno de los casos más sonados no solo por la escala del ataque sino también porque este consistía en la explotación de un comando estándar basado en la memoria caché, el sistema de caché de la base de datos y su relación con las redes y webs. La técnica es más conocida como **Memcached DDoS attack technique** resultando realmente efectiva.

Según reportes propios de GitHub, se detectarón sobre miles de **sistemas autónomos (ASNs)** diferentes sobre miles de **endpoints únicos**. En su propias declaraciones además afirmaban investigar en este y otros incidentes con el fin de mejorar su infraestructura y seguridad de cara a posibles futuros ataques [20].

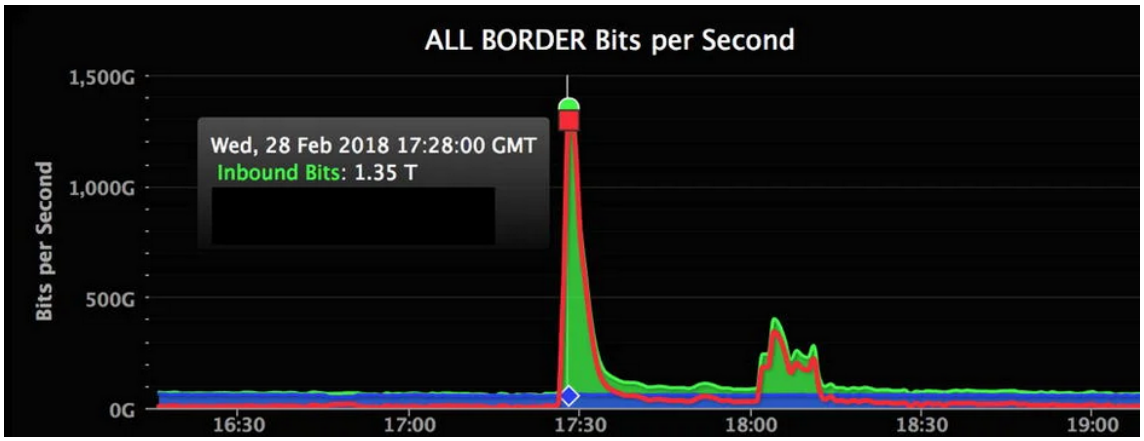

Imagen 15: Estadísticas del ataque a **GitHub**.

6.5. SpamHaus

En 2013, el servicio antispam **SpamHaus** se vió afectado por uno de los ataques DDoS más grandes vistos hasta la época. Este ataque llegó incluso a producir cierta lentitud en Internet, y alcanzó hasta un pico de unos **300 Gbps** de tráfico, siendo esto aproximadamente tres veces más grande que los previos ataques realizados hasta ese momento.

En su momento, Spamhaus no respondió de inmediato a una solicitud de comentarios por los diferentes medios, y llegó a requerir ayuda de empresas como **CloudFlare** para poder afrontar dicho ataque. Según las noticias posteriores, los ataques comenzaron después de que SpamHaus agregara al proveedor de hosting **Cyberbunker** a su lista negra

global.

Ya sea como queja, o ataque premeditado, la realidad es que Cyberbunker era una empresa que operaba desde un búnker de la OTAN abandonado y era conocida en su momento por albergar sitios web emisores de SPAM, y podía llegar a alojar páginas casi todo tipo de webs de dudosa moralidad.

Este ataque muestra claramente los peligros que corre una empresa de cara a interactuar negativamente con otras empresas y como con los suficientes recursos se puede llegar a producir un ataque de una gran escala, lo suficiente como para tirar todo un servicio con su página web y servicio de mensajería [21]-

6.6. Otros casos relevantes

Aunque existen una gran cantidad de casos importantes, a modo de resumen se incluyen a continuación varios de los ataques más importantes que se han dado recientemente:

- El ataque DDoS a seis bancos estadounidenses en 2012 [22].
- Code Red, el gusano que provocó un DDoS a La Casa Blanca en 2001 [23].
- El ataque DDoS a CloudFlare en 2014 [19].
- Ataque DDoS que tumbó Yahoo!, Amazon, eBay y muchos otros portales por Michael Calce (MafiaBoy) [24].

7. Simulación de ataques DDoS

Para la realización de ambas demostraciones, se ha optado por tomar una página web llamada `Firefoxbug.net`, la cual ha sido diseñada específicamente para realizar pruebas de ataques DDoS (ver Figura 16)

Hoy en día la mayoría de sitios web se encuentran protegidos ante ataques sencillos como los que hemos llevado a cabo, por lo tanto una página web con poca protección y expuesta para realizar pruebas es el entorno ideal para comprobar el funcionamiento de ataques **DDoS** simples.

7.1. HTTP Flood

Para la realización de un ataque **HTTP Flood** se ha utilizado un script en **Python**, el cual se ha adaptado a **Python3**, para poder realizar las pruebas. Dicho script se ha obtenido de un repositorio público de GitHub del usuario `Firefoxbug` [25].

Este script realiza un ataque parametrizable del tipo inundación de HTTP, consistente en enviar paquetes de forma constante durante un tiempo determinado provocando la saturación del sitio web al no poder responder a todos esos paquetes al mismo tiempo.

Uno de las principales características es la utilización de diferentes agentes de forma aleatoria para los ataques, pretendiendo camuflar la identidad del atacante, como se haría en un caso real. Además mantienen activa la conexión durante el tiempo del ataque.

Imagen 16: `Firefoxbug.net`

Como se puede observar, al realizar el ataque desde una terminal utilizando diferentes hebras, y durante un tiempo de 30 segundos, es suficiente para poder saturar la página web, la cual se queda bloqueada al no poder atender todas las peticiones. Una vez finalizado el ataque hay que esperar para que sea capaz de procesar todas las peticiones atrasadas y finalmente atender nuestra petición real:

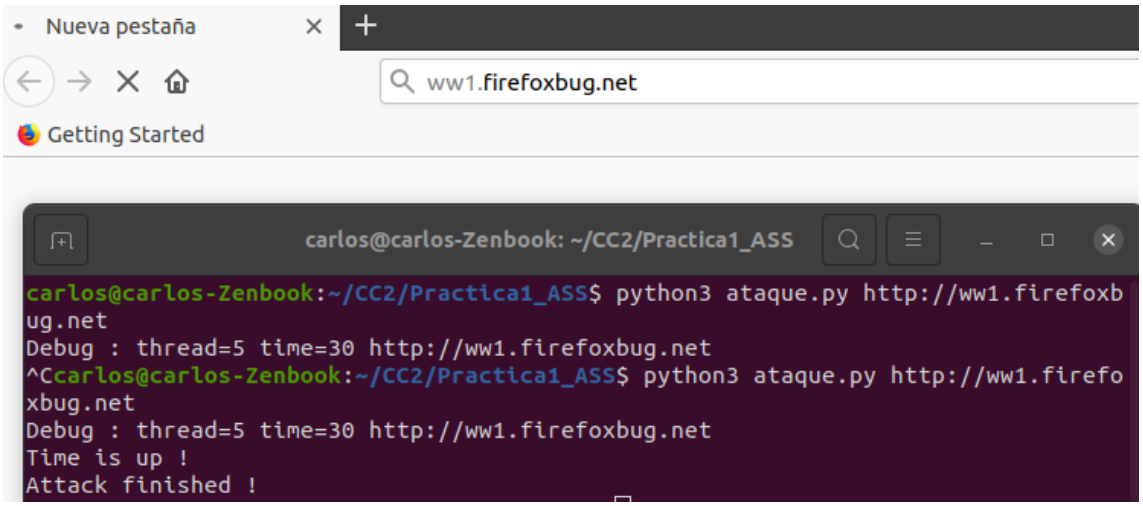

Imagen 17: Ataque **HTTP Flood**.

7.2. Slowloris

Para la realización de un ataque **Slowloris** se ha utilizado un programa en **Python3**, para poder realizar las pruebas. Dicho programa se ha obtenido de un repositorio público de GitHub del usuario `gkbrk` [26].

Tal y como se define en el propio repositorio, Slowloris es básicamente un ataque de denegación de servicio HTTP que afecta a los servidores con subprocesos. Funciona así:

1. Comenzamos a realizar muchas solicitudes HTTP.
2. Enviamos encabezados periódicamente (cada 15 segundos) para mantener abiertas las conexiones.
3. Nunca cerramos la conexión a menos que el servidor lo haga. Si el servidor cierra una conexión, creamos una nueva y seguimos haciendo lo mismo.

Esto agota el grupo de subprocesos de los servidores y el servidor no puede responder a otras personas. Nuevamente volvemos al ejemplo anterior y observamos que la página está disponible y funciona correctamente (ver Figura 16).

Al realizar el ataque nuevamente desde una terminal, se ve que se crean una serie de sockets que mantienen las **conexiones HTTP** vivas con el sitio web. Continuamente se encuentra enviando cabeceras para mantener estas conexiones y mantener el servidor saturado con comunicaciones sin ningún objetivo más que el de mantenerlo ocupado. Al igual que con el ataque anterior, el servidor es incapaz de responder a más peticiones y deja de cargar cuando el usuario solicita ver la página:

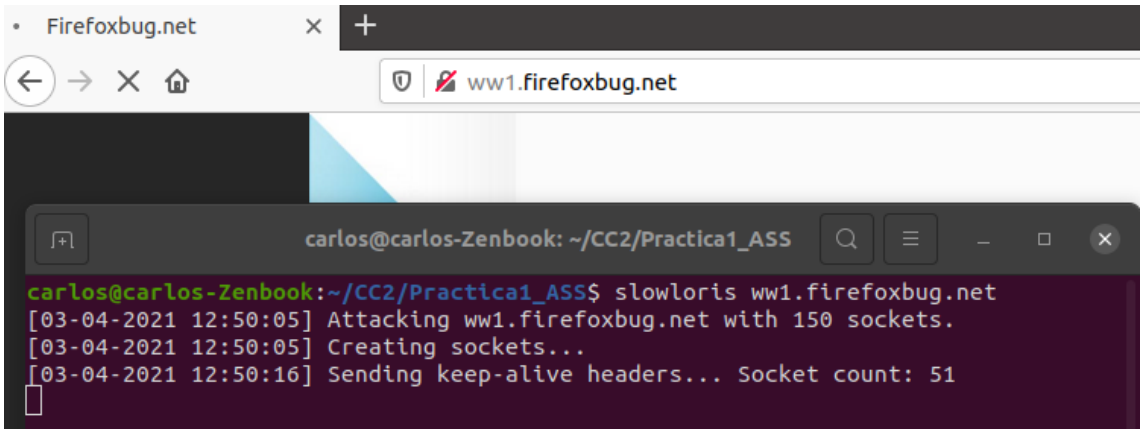

Imagen 18: Ataque **Slowloris**.

8. Bibliografía

[1] Tim Keary. Dos vs ddos attacks: The differences and how to prevent them. https://www.comparitech.com/net-admin/dos-vs-ddos-attacks-differences-prevention/, 2020.

[2] Amazon Web Services. ¿qué es un ataque ddos? https://aws.amazon.com/es/shield/ddos-attack-protection/.

[3] IONOS. Syn flood: variantes y medidas defensivas. https://www.ionos.es/digitalguide/servidores/seguridad/syn-flood/, 2020.

[4] José Antonio Castillo. Protocolo tcp/ip - qué es y cómo funciona. https://www.profesionalreview.com/2020/03/21/protocolo-tcp-ip/, 2020.

[5] Imperva. Buffer overflow attack. https://www.imperva.com/learn/application-security/buffer-overflow/.

[6] IONOS. El ping de la muerte: uno de los primeros ataques de red. https://www.ionos.es/digitalguide/servidores/seguridad/ping-de-la-muerte/, 2020.

[7] Lucio Tappi Diego Cejas, Iñaki Perea. Seguridad en redes - ataque y defensa. https://slideplayer.es/slide/10974297/, 2016.

[8] Stephen Cooper. What is a botnet and how to avoid being part of one. https://www.comparitech.com/blog/information-security/what-is-a-botnet/, 2018.

[9] IONOS. Udp flood. https://www.ionos.es/digitalguide/servidores/seguridad/udp-flood/, 2020.

[10] Cloudflare. Slowloris ddos attack. https://www.cloudflare.com/es-es/learning/ddos/ddos-attack-tools/slowloris/.

[11] Cloudflare. Http flood attack. https://www.cloudflare.com/es-es/learning/ddos/http-flood-ddos-attack/.

[12] Cloudflare. What is a zero-day exploit? https://www.cloudflare.com/es-es/learning/security/threats/zero-day-exploit/.

[13] Cloudflare. What is a denial-of-service (dos) attack? https://www.cloudflare.com/es-la/learning/ddos/glossary/denial-of-service/.

[14] Fran Ramírez. Slammer, el virus que colapsó internet en 2003. https://derechodelared.com/slammer-el-virus-que-colapso-internet-en-2003/.

[15] GenBeta. El "gusano morrisçumple 27 años; así fue el primer malware de la historia. https://www.genbeta.com/a-fondo/el-gusano-morris-cumple-27-anos-asi-fue-el-primer-malware-de-la-historia.

[16] GlobalDots. Panix - syn flood and domain hijacking (1996 and 2005). https://www.globaldots.com/blog/15-most-dangerous-ddos-attacks-that-ever-happened.

[17] Zdnet. Windows 7, vista exposed to 'teardrop attack'. https://www.zdnet.com/article/windows-7-vista-exposed-to-teardrop-attack/.

[18] Zdnet. Google says it mitigated a 2.54 tbps ddos attack in 2017, largest known to date. https://www.zdnet.com/article/google-says-it-mitigated-a-2-54-tbps-ddos-attack-in-2017-largest-known-to-date/.

[19] Zdnet. Aws said it mitigated a 2.3 tbps ddos attack, the largest ever. https://www.zdnet.com/article/aws-said-it-mitigated-a-2-3-tbps-ddos-attack-the-largest-ever/.

[20] Wired. Github survived the biggest ddos attack ever recorded. https://www.wired.com/story/github-ddos-memcached/.

[21] ComputerWorld. Update: Spamhaus hit by biggest-ever ddos attacks. https://www.computerworld.com/article/2495967/update--spamhaus-hit-by-biggest-ever-ddos-attacks.html.

[22] CIO. Ddos attacks against us banks peaked at 60 gbps. https://www.cio.com/article/2389721/ddos-attacks-against-us-banks-peaked-at-60-gbps.html.

[23] ComputerHoy. Code red, el gusano que provocó un ddos a la casa blanca (2001). https://computerhoy.com/listas/tecnologia/mayores-ataques-ddos-historia-internet-555187#cuatro.

[24] Wikipedia. Michael calce (mafiaboy). https://en.wikipedia.org/wiki/Michael_Calce.

[25] Wang (Firefoxbug). Http flood. *github.com*, 2013.

[26] Gokberk Yaltirakli. Slowloris. *github.com*, 2015.