Tácticas de guerra en el IRC *v.4.1*

Por: ***RoMaN SoFt / LLFB*** **(20/09/99)**

##

0.- Contenido.

1.- *Introducción.*

Ultimamente la guerra en el IRC es, por desgracia, algo bastante común. Conviene, a lo menos, estar informado de las distintas técnicas que se suelen usar para "luchar", aunque sea simplemente para detectar que te están atacando, saber cómo lo están haciendo e intentar defendernos. No es mi intención profundizar demasiado en el tema aunque intentaré detallar algunos puntos que considere convenientes.

Todo lo que aquí se hable es extensible en general a cualquier red IRC. No obstante en algunos casos muy particulares me referiré a la red IRC hispana (*"*.irc-hispano.org"*). Ni que decir tiene que la información que se proporciona aquí es con fines educativos; en ningún caso debe ser usada salvo en circunstancias excepcionalmente justificadas. Un uso abusivo puede significar una *k/g-line* totalmente merecida. Yo no me hago responsable de un posible mal uso de esta info.

[Indice]

2.- *Principios.*

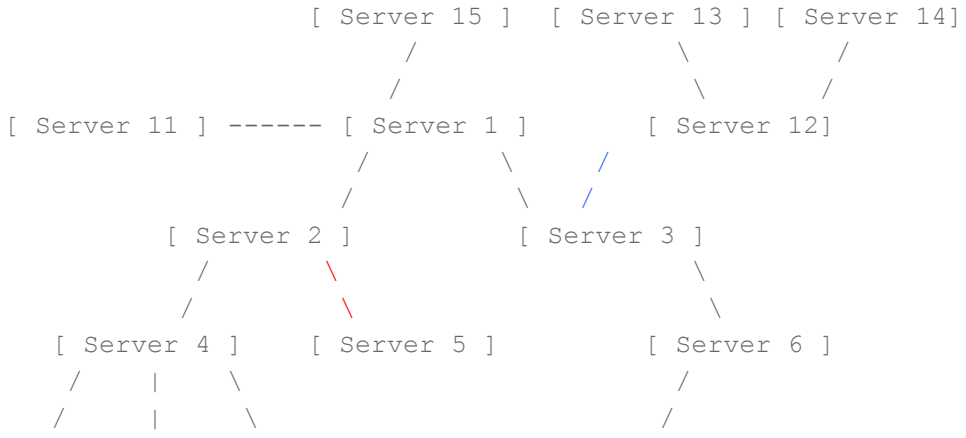
Muchas veces el objetivo de un ataque no es otra cosa que hacerse con un canal (*"tomarlo"*). Esto es relativamente fácil y hay diversas técnicas para ello. El objetivo es hacerse con el op en el canal. Los medios: tu inteligencia y astucia... ;-)

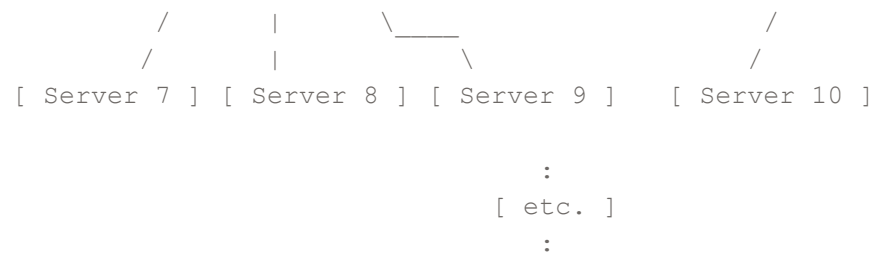
2.1.- Cómo funciona una red IRC.-

El *servidor* de IRC propiamente dicho no es más que un programa corriendo en background (un *daemon*) en una máquina determinada (en Unix correría el *"ircd"*). Los usuarios se conectan a dicha máquina y acceden al servidor en forma de *clientes*.

Una *red IRC* se compone de varios servidores corriendo en paralelo y enlazados entre ellos, de forma que se mantegan comunicados (puedan intercambiar mensajes entre ellos). Cuando un usuario se conecta a un servidor determinado, éste (el servidor) lo notifica a los demás servidores que forman parte de la red IRC. Igualmente, cualquier otra acción es notificada a todos los servidores, de forma que éstos actuan como una unidad. De esta forma el usuario se deja ver en todos los servidores aunque físicamente sólo esté conectado a uno. Esto permite tener muchos usuarios repartidos por diferentes servidores pero que virtualmente es como si estuvieran todos en uno sólo.

La *estructura* de la red IRC es en forma de árbol (es decir, no puede haber *bucles*, o "caminos cerrados": partiendo de un nodo no se llegue por ningún camino otra vez a dicho nodo) aunque un tanto especial: cada nodo se ve a sí mismo como el nodo raiz de la red y tiene un grafo en forma de árbol que le indica el camino a seguir para alcanzar cada uno de los restantes nodos. En la "literatura" esto se conoce como *"spanning tree"*, que podríamos traducir como *"árbol de expansión"*. Esto quiere decir que en un momento determinado un nodo cualquiera tendrá almacenada información para alcanzar cada uno de los otros nodos de forma unívoca (tiene un único camino posible hacia cada nodo). Esa información sería el árbol que está usando el nodo en cuestión. Pero además este árbol puede ser distinto para el mismo nodo en un instante diferente, es decir, puede cambiar (digamos que el nodo va reconfigurándose). Esto tiene la ventaja de que permite adaptarse a posibles variaciones (eventuales) de la topología de la red (así, si un nodo cae, los restantes nodos lo detectarán y se reconfigurarán de forma que los caminos que antes pasaban por dicho nodo dejen de hacerlo: se tomarían caminos alternativos con lo cual la red seguiría funcionando correctamente a pesar de la caida del nodo). Un ejemplo de topología de red podría ser:





El paso de un nodo a otro adyacente se conoce como *"hop"* (salto). Así para alcanzar el nodo 5 partiendo de 4 tendremos que dar 2 saltos (*hops*): uno de 4 a 2 y otro de 2 a 5.

Podemos visualizar el árbol que está usando el server al que estamos conectados usando el comando *"/links"*. Este sacará un listado por pantalla de los servidores alcanzables desde el nuestro, de forma jerarquizada, es decir, respetando la estructura del árbol. Normalmente se indica entre paréntesis al lado de cada servidor el número de *hops* que habría que dar para alcanzar cada uno de los nodos partiendo del nuestro.

Cuando se rompe uno de los eslabones (*links*) que unen 2 servidores el conjunto se divide en 2 subconjuntos, los cuales intentarán seguir funcionando normalmente aunque de forma aislada. Esto es, cada subconjunto permanece operativo y mantiene la comunicación entre los *servers* pertenecientes a dicho subconjunto. Pero por razones obvias los servidores de un subconjunto no ven a los del otro y viceversa. Esta situación se conoce como *net-split*.

En una sesión normal de IRC esto lo veríamos:

*[1:23] *** Case_Zer0 has quit IRC (fuego.irc-hispano.org io.irc-hispano.org)*

Esto indica que se han *spliteado* los dos servidores indicados entre paréntesis y que a consecuencia de ello el usuario Case_Zer0 [hi Case ;-)] ha salido "de nuestra red IRC" (lo que está ocurriendo es que se encuentra en el otro subconjunto de servidores: a todos los efectos es que como si se encontrase ya en otra red IRC).

Cuando el enlace caido se recupera (i.e. se reestablece la comunicación entre los servers spliteados) se habla de *net-merge*. Esto se vería en la sesión anterior como un *"join"* por parte del usuario que estaba en el servidor spliteado (tanto el *quit* como el *join* anteriores son mecanismos del propio IRC, es decir, el usuario anterior no dio ninguna orden de *quit* ni de *join*, es transparente a dicho usuario).

Hay programas que detectan y avisan cuando se produce algún *net-split* o *net-merge*: son los denominados *"link-lookers"*, y su utilidad es bastante obvia.

Por ejemplo, si el enlace dibujado en rojo (enlace server 2 <-> server 5) cayera, el servidor 5 estaría aislado de la red. Los usuarios de dicho servidor dejarían de ver a todos los demás pertenecientes a servidores distintos, y al contrario. Se dice que el servidor 5 está *spliteado*. Es fácil reconocer a un servidor en esta situación: si entras en una red a través de un determinado servidor y te encuentras a muy poca gente es muy normal que se deba a que está spliteado de la red.

Otra posibilidad es que el enlace azul (3 <-> 12) cayera. En este caso el servidor 12 se splitea de la red, pero también lo hacen los servidores 13 y 14 indirectamente, por conectarse a través del primero.

Para una información completa del funcionamiento y estructura de una red IRC, y del protocolo subyacente (*"Internet Relay Chat Protocol"*) os remito al *RFC1459*.

[Indice]

2.2.- Conseguir op sin que nadie te lo de.-

Cuando alguien se une a un canal donde no hay nadie (hace un */join #canal*) el servidor supone que se trata de un nuevo canal y le da op a dicho usuario. Se dice que ha creado un canal. Vamos a aprovechar esto para hacernos con el op en un canal ya existente. ¿Cómo? Fácil: solo hay que aprovechar un net-split. Los pasos serían los siguientes:

- Esperar un split (lo podemos detectar con un link-looker).
- Entrar (conectar) al servidor spliteado.
- /join #canal (donde *canal* es el canal donde queremos conseguir op).
- El server creará un nuevo canal y tendrás el op.
- Esperar a que se deshaga el split.

Si "hay suerte" (leer más abajo), al deshacerse el split conservaremos el op en los restantes servidores (el servidor spliteado se encarga de dar las órdenes correspondientes). Entonces se dice que hemos llevado a cabo un *"net-hack"*. Los usuarios presentes en el canal en el que hemos llevado a cabo la acción verán algo como:

*[1:41] *** irc.i3d.es sets mode: +o RoMaNSoFt*

(donde el servidor que nos da op es el que antes estaba spliteado).

Esto no siempre funcionará porque hay aspectos que todavía no he comentado. Paso a explicar el procedimiento y comentar algunos puntos negros. Supongo que habréis comprendido el procedimiento; es muy simple: aprovechar que el servidor spliteado no ve a los usuarios de otros servidores y por tanto al canal previamente creado. Esto **presupone** que no hay usuarios **del servidor spliteado** en el canal (en este caso no funcionaría) ya que al entrar nosotros por el server spliteado veríamos al canal como ya creado, con los usuarios de nuestro mismo servidor (a los otros los "esconde" el split) y por tanto el server no nos dará el op, como es habitual al entrar en cualquier canal ya existente.

También hay que tener en cuenta que actualmente todos los servidores tienen *protecciones anti-nethack*. En este caso, al deshacerse el split, los restantes servidores te quitarán el op a tí en vez de ser al contrario (imponer tu op en los restantes servers), protegiendo al canal **PERO** ésto lo harán únicamente en caso de que ya hubiera ops en el canal antes de tu intento de net-hack (aunque hay veces en que el server se equivoca y mantiene tu op, quitándoselo a los demás). Es decir, que el net-hack funcionará sólo para canales donde no haya op (*"opless channels"*). Por esta razón, si queremos el op, necesitaremos *tirar* previamente a los ops (echarlos del canal, de forma que pierdan su op) para luego llevar a cabo el net-hack. ¿Cómo tirarlos? De esto nos encargaremos más adelante, sigue leyendo };-)

[Indice]

2.3.- Otras formas de conseguir op.-

La otra alternativa para conseguir el op es que alguien te lo de ;-) . Puede ser un op del canal o un irc-op de la red, aunque para esto último tendrás que dar una justificación convincente (como por ejemplo que os acaban de tomar el canal, alguien os ha atacado, étc).

Para la primera alternativa entra en juego tu "don de la palabra": trata de hacerte amigo de algún op para que éste te lo pase. En ese momento ya estás capacitado para quitarle el op a todos los demás (*"mass-deop"*) y quedarte con el canal. Esto lo hacen automáticamente muchos scripts (*"automatic-deop"*): nada más darte el op el script automáticamente deopea a todos los ops (excepto a tí, claro).

 [Indice]

2.4.- Bot de servicio.-

Se trata de un "usuario" muy especial... un ro*bot* que se encarga entre otras cosas de proteger los canales. En la red hispana se llama *Scytale* (en CobraNet, por ejemplo, es *KingCobra*) y está dentro de muchos canales (registrados). Normalmente suele tener op, con lo cual el canal deja de ser opless y se evita el net-hack :-(Suele tener *ircop-status* [*channel-service status*] y además tiene otras funciones en las que no pienso entrar. Resumiendo: si hay bot, nuestro gozo a un pozo.

 [Indice]

3.- *Ataques.*

En esta sección entraremos en materia... }:-) Nuestro objetivo: *tirar* a alguien del server irc.

3.1.- Flood.-

Los servidores IRC tienen que controlar el tráfico de entrada (el que proviene del exterior) para evitar su congestión. Una de las formas de conseguirlo es no permitir que un cliente le mande más de una determinada cantidad de información en un pequeño intervalo de tiempo; o lo que es lo mismo: la velocidad con que un cliente puede enviar datos al servidor está limitada.

Cuando un cliente supera el límite preestablecido por el servidor, éste cierra la conexión con el cliente: lo echa del servidor porque no puede soportar tanto caudal de entrada. El servidor lo "explica" así:

*[1:59] *** _^TkLaS^_ has quit IRC (Excess Flood)*

Un *flood*, en general, no es otra cosa que mandar mucha información en poco tiempo a alguien para intentar que se sature. Se puede floodear una dirección IP, ..., o lo que ahora nos concierne: un servidor de IRC.

La manera de aprovechar el *flood* en nuestro favor consiste en mandar muchas peticiones de información a nuestra víctima, de forma que ésta, al contestar, supere el límite del servidor y éste lo eche. Por ejemplo, si le mandamos muchos */ctcp version*'s seguidos (requiriendo información sobre el programa cliente que está utilizando) la víctima floodeará al servidor cuando conteste porque mandará muchas veces (tantas como peticiones haya habido) el texto de respuesta al servidor (para que del servidor vaya al cliente que peticionó, i.e., al atacante).

En esto del flood juega un papel muy importante el número de peticiones que se reciben en un pequeño intervalo de tiempo. Cuantas más se reciban, más posibilidades hay de que el flood tenga éxito. Por ello no es ninguna tontería mandar peticiones desde varios puntos a la vez, y no desde uno sólo, es decir, varios usuarios (¡que podrían ser una misma persona!) de la red IRC manden

peticiones a la víctima todos a la vez en un determinado momento. Si los usuarios (nicks) corresponden a una misma persona (una misma dirección IP) se habla de *clones*. Por tanto, una posible forma de ataque sería crearnos muchos clones y peticionar a la vez desde todos ellos a la víctima.

Pero los servidores también suelen estar preparados para evitar muchos clones (cada clone ocupa, por decirlo de alguna manera, una "linea" de entrada al servidor, y esto consume recursos del mismo). Suele haber un máximo permitido (en el irc hispano es 2) denegándosele el acceso a la red a un tercer clone, o en caso de que éste lo consiguiese expulsándosele del servidor (*"matándolo"*) (el programa servidor revisa periódicamente las IP's conectadas y detecta cuando hay varios usuarios con una misma dirección IP):

*[1:32] *** _^Virus^_ has quit IRC (Killed (Clones!))*

Se puede cambiar el número máximo de clones admisibles desde una determinada dirección IP o dominio añadiendo una *I-Line* al servidor IRC (en caso de no existir *I-Line* para esa dirección IP en particular se usa el máximo genérico definido). Esto lo debe hacer algún administrador de la red IRC y es lo que habitualmente se usa para dar acceso a entidades con muchos ordenadores accediendo a Internet desde una misma IP (como es el caso de la mayoría de cyber-cafés).

¿Cómo provocar un flood con más de 2 clones entonces? La respuesta es simple: en principio no se puede. ¿Entonces? Pues la solución es que varias personas distintas se pongan de acuerdo para atacar a la vez a la víctima. Cada persona podría tener a su vez varios clones. Por ejemplo, si A (atacante) quiere atacar a V (víctima), A se pone de acuerdo con B y C (otras 2 personas atacantes). A su vez supongamos que cada atacante tiene 2 clones: i.1 e i.2 (donde i=A,B,C). Entonces tendremos 6 usuarios (conexiones IRC) distintos atacando a V, que serían A.1, A.2, B.1, B.2, C.1 y C.2. Pero hay un problema: ¿cómo sincronizarse para atacar? ¿Cómo "ponerse de acuerdo" para mandar las peticiones en un determinado momento? Para esto existe lo que se denomina *"floodnet"* que, como habrá adivinado nuestro ávido lector, es una "red" (asociación) de gente cuyo único objetivo es floodear a alguien. La ventaja que tiene es que la sincronización entre los distintos componentes de la *floodnet* es automática (lo hacen los scripts) lo cual resuelve el problema anterior. También existe lo que se denomina *"botnet"* y que es análogo a la floodnet pero usando *bots* (no confundir con los "de servicio"; estos últimos los ponen los servers de la red irc y no los usuarios) los cuales serán lanzados desde alguna *shell Unix* (intérprete de comandos en una máquina Unix). Los bots suelen estar prohibidos y cuando se detectan, a lo menos, son expulsados:

*[1:32] *** Viernes13 has quit IRC (Killed (You are not welcome to this network!))*

Hoy en día tanto los programas clientes de IRC como los scripts implementan *protecciones anti-flood* que dificultan enormemente el éxito de un ataque de tipo flood. Por ejemplo, cuando detectan varias peticiones seguidas mandan las respuestas espaciadas en el tiempo (con pausas) y no inmediatamente, con lo cual se evita el flood.

También hay técnicas de flood bastante optimizadas (por ejemplo, usando una *floodnet*) aunque en general un ataque flood no suele ser demasiado eficiente y es más costoso lograr su éxito que con algunas de las técnicas que se describen a continuación.

[Indice]

3.2.- Nick collide.-

Un *"nick collide"* ocurre cuando dos personas tienen un mismo *nick*. En principio esto no debería ser

posible (el servidor no deja usar un nick que ya está en uso) pero hay dos situaciones en las que podría darse el caso y que se describen en los dos puntos siguientes.

El resultado de un *nick collide* depende del servidor (*ircd*). En servidores antiguos (sin protección) el *collide* se resuelve matando a los dos usuarios con mismo nick (¡ambos!). En otros más inteligentes (con protección) el servidor guarda información acerca de los usuarios y puede saber que usuario tiene el nick con mayor antigüedad (i.e. quién se lo puso antes), matando únicamente al usuario con el nick más reciente (protegiendo al usuario más "veterano").

3.2.1.- Split server collide.-

Se basa en aprovechar un net-split:

- Esperar un split.
- Entrar (conectar) al servidor spliteado.
- Ponerse como nick el de la víctima.
- Esperar a que se deshaga el split.

Si todo va bien (el servidor no tiene protección), a la vuelta del split se detectará el collide y se matarán tanto al atacante como a la víctima. Lógicamente nuestro usuario atacante será un clone nuestro, con lo cual no pasa nada si es *killeado*.

3.2.2.- Lag collide.-

Consiste en aprovechar el lag de un servidor, o lo que es lo mismo, el retraso en recibir los mensajes de otros servidores. Esta técnica es más efectiva que la anterior, pues funciona en servidores con protección.

Los pasos serían los siguientes:

- Meter un clone en el servidor lageado.
- Esperar a que la víctima cambie de nick (esto lo detectamos desde **otro** servidor **no** lageado).
- Cambiar rápidamente el nick de nuestro clone y ponerle el que se acaba de poner la víctima (el nuevo).
- Esperar al lag. ;)

Lo que ocurre es que nuestra orden de cambiar el nick para nuestro clone llega antes al servidor (lageado) que la orden de cambio de nick de la víctima debido a que nuestra orden va directamente de nuestro cliente al servidor lageado mientras que la otra va a través de la red IRC (donde hemos supuesto que se introduce un lag notable). Esto hace que el servidor (lageado) tome a nuestro clone como "dueño" legítimo del nick y mande un kill al otro (la víctima). Esto ocurriría en caso de servidores protegidos; si es no protegido el resultado es que ambos mueren, resultado también aceptable, pues hemos acabado con nuestro objetivo. };-)


[Indice]

3.3.- Nuke.-

"Nuke" es la denominación genérica que se le suele dar a cualquier forma de ataque consistente en mandar paquetes arbitrarios a una determinada dirección IP (no es que sea una definición demasiado

ortodoxa pero bueno... :)). Realmente el término "nuke" siempre se ha referido al primero de los dos tipos que comentaremos, aunque aquí se ha preferido tomar una definición más amplia de dicha palabra.

3.3.1.- ICMP nuke.-

El más veterano de los nukes [:-)] usa un protocolo subyacente de IP, el *ICMP* (*"Internet Control Message Protocol"*: parte integral del protocolo de Internet [IP] que resuelve errores y controla los mensajes), para romper una conexión cliente-servidor de IRC (tirar a alguien del server). Para entender cómo funciona hay que hablar un poco de protocolos; es aburrido pero no hay más remedio...

Una conexión IRC (cliente-servidor, que es lo que nos interesa) utiliza el protocolo TCP (nivel 4 [transporte] en la torre OSI), el cual se apoya sobre IP (nivel 3 [red]). IP se encarga, entre otras cosas, de hacer el rutado de paquetes (*"datagramas IP"*), es decir, dado un destino ir enviando los paquetes por el camino apropiado hasta alcanzar el host destino. TCP no ve nada de esto, tan sólo el destino directamente (manda los *segmentos TCP* directamente al destino), porque IP lo oculta (hace que el rutado sea transparente a TCP). Lógicamente para que un protocolo de nivel superior funcione correctamente, también deberán hacerlo todos los que estén por debajo. En particular, para que nuestra conexión TCP (IRC) se mantenga "viva", IP debe funcionar perfectamente. Y aquí es donde interviene ICMP: se encarga de informar de posibles anomalías que se han producido en el nivel 3 (IP), como por ejemplo, *"host unreachable"*, que significaría que no se ha podido alcanzar el *host* (el paquete IP ha ido dando saltos [*"hops"*] de un nodo a otro, hacia el destino, y ha llegado un momento en el que un determinado nodo intermedio no sabía qué hacer con él o ha expirado el tiempo de vida de dicho paquete). En este caso, el paquete que informa del error (ICMP) lo envía el nodo intermedio que se ha dado cuenta del error hacia el "remitente" que lanzó el paquete original (que no se ha podido entregar a su destinatario). Los mensajes ICMP se situan dentro del campo de datos de un datagrama IP y se envían exactamente igual que si fueran datos IP (no son prioritarios). No es objetivo de este escrito tratar más a fondo este tema (para los interesados les aconsejo el libro *"Internetworking with TCP/IP, vol I"*, de *Douglas E. Comer*, disponible en castellano ya en su tercera edición).

Resumiendo: mediante ICMP informamos de que IP ha fallado, y por tanto, también los niveles superiores como TCP.

Comprendiendo lo anterior ya se puede intuir en qué consiste el *ICMP nuke*: mandar mensajes ICMP falseados, engañando al destino, haciéndole creer que el otro extremo ha detectado un error y por tanto, provocando un "cierre" de la comunicación. Vamos a explicar un poco mejor ésto.

En una conexión siempre tenemos dos extremos, lo que da dos posibilidades a la hora de engañar, según lo hagamos a uno u otro. En el caso de una conexión IRC, podemos llevar a cabo dos formas de ataque:

* ***Server nuking*** (nukear al server): los mensajes ICMP se mandan al servidor IRC, haciéndole creer que se ha producido un error al intentar comunicarse con el cliente. Como respuesta a este mensaje el server cierra la conexión que tenía con dicho cliente. El efecto producido es la "expulsión" del usuario por parte del servidor.

* ***Client nuking*** (nukear al cliente): esta vez se envían los ICMP's al cliente; éste cree que el servidor no está disponible y cierra la conexión (el cliente). El servidor no sabe nada en principio, pero detecta el cierre de conexión por parte del cliente, dando el correspondiente error y cerrando también la conexión por su parte.

En teoría las dos fomas de nuking son perfectamente válidas y eficientes, aunque hay que tener ciertas consideraciones en cuenta, como son:

- tanto servidor como cliente pueden tener protección anti-nuke y puede ser necesario atacar uno porque el otro esté protegido (ver más adelante).
- si atacas a un cliente, éste puede detectar quién le está atacando con un simple analizador de paquetes IP o *tracer*, y también podría responder con otro ataque de este o cualquier otro tipo (cuidado con quién te metes ;-)).
- si atacas al servidor, el cliente no tiene manera de saber quién le ha "atacado" porque los mensajes ICMP no le han llegado a él sino al servidor (ventaja); pero por otro lado, el servidor sí sabe quién ha hecho el ataque y puede resultar en una K/G-Line a dicho usuario por parte del servidor (el usuario podría ser baneado de toda la red de IRC).
- los inconvenientes de los dos puntos anteriores pueden ser solventados falseando la dirección origen de los mensajes ICMP que se envían. Esta técnica se conoce como *"spoofing"* (ver punto 4).

Hay diversos tipos de error ICMP que se pueden utilizar a la hora de hacer un nuke. En cuanto a la información práctica de cómo utilizar un *nuker* (programa "nukeador"), debemos tener en cuenta que además de suministrarle el tipo de error que se desea producir, juegan un papel muy importante los puertos, tanto origen como destino, que se elijan.

Una conexión IRC (TCP) queda definida unívocamente por los pares dirección IP origen-puerto origen y dirección IP destino-puerto destino. Estos datos son los que hay que suministrarles al programa nukeador. Puertos típicos del servidor de IRC (será el puerto destino en caso de server nuking o el fuente si se trata de un client nuking) son 6665-9, 4400-6, 7000 y 7777. En realidad cada servidor IRC tiene unos puertos oficialmente reconocidos (que son conocidos públicamente: los podemos leer en el *motd* [*"mensaje del dia"*] al entrar en el IRC) y otros que podríamos denominar como privados, y que se usan por ejemplo para las conexiones entre los distintos servidores que forman la red. Un usuario puede estar usando uno de estos puertos "fantasmas" (aunque el servidor también puede limitar el acceso a estos puertos) para esconderse de nukes, puesto que necesitamos conocer este dato para que el nuke sea efectivo.

También necesitamos conocer el puerto del cliente, aunque esto es más difícil porque varía mucho (no son fijos como en el caso anterior) dependiendo del sistema operativo que esté corriendo dicho cliente, los puertos que ya tuviera ocupados antes de establecer la conexión IRC, étc. Lo normal es hacer un barrido de estos puertos empezando por el 1024 (hay puertos que por convenio siempre se asignan a determinadas tareas y no se pueden usar arbitrariamente con lo cual no necesitamos barrerlos) y acabando en 4000, por ejemplo, aunque podría ser necesario aumentar este número.

Es también muy útil utilizar un *"port-scan"*: programa que va probando los distintos puertos de una dirección IP (destino) dada e informa de la respuesta recibida para cada uno de dichos puertos (así podemos saber, por ejemplo, qué puertos de un servidor están dedicados a aceptar conexiones IRC).

A continuación transcribo mensajes típicos de salida de nuestras potenciales víctimas en una sesión típica de IRC:

*[1:42] *** aRmiTaGe has quit IRC (Read error to aRmiTaGe[ig-183.arrakis.es]: Connection reset by peer)*
*[1:13] *** KoNtRoL has quit IRC (Read error to KoNtRoL[195.76.99.76]: EOF from client)*
*[3:17] *** BrOKeNn has quit IRC (Read error to BrOKeNn[194.224.57.171]: Protocol not available)*
*[5:25] *** Eli has quit IRC (Read error to Eli[info760.jet.es]: Network is unreachable)*
*[5:26] *** Eli has quit IRC (Read error to Eli[info760.jet.es]: Machine is not on the network)*
*[4:20] *** vp has quit IRC (Read error to vp[ia-176.arrakis.es]: Connection refused)*
*[2:41] *** Estrayk has quit IRC (Read error to Estrayk[ctv3011.ctv.es]: No route to host)*

La protección anti-nuke, a grosso modo, pasa por ignorar los mensajes ICMP que lleguen, aunque

ésto ya está limitando el propio funcionamiento del protocolo IP, en el sentido de que ICMP es parte integrante de IP y no se debería inhibir (¿qué ocurriría si llega un mensaje "de verdad" y es ignorado?). Se puede llevar a cabo más o menos "finamente": por ejemplo descartar solo los ICMP's de un tipo y no todos los posibles. Se podría lograr con un *firewall* (software o hardware encargado de filtrar los paquetes provinientes de la red en base a una reglas previamente definidas) convenientemente configurado.

Puedes echar un vistazo al listado fuente de un programa en C para mandar ICMP nuke's:

 puke.c

 [Indice]

3.3.2.- OOB nuke.-

También conocido como 'winnuke', ya que afecta sólo al sistema operativo Windows, en cualquiera de sus "sabores": 3.x, 95 y NT. Se basa en un *bug* que tiene este SO en la pila de protocolos por el cual el sistema se cuelga ("error de protección general...blah, blah...") cuando recibe un paquete con el flag OOB (*"Out of band"*) activado. El ataque es sencillo: mandar un paquete de este tipo a un puerto (normalmente el 139) de nuestrá víctima (ésta debe estar corriendo Windows, lo cual es muy normal hoy en día). Existen programas ya hechos a los cuales simplemente le das la dirección IP de la víctima y el programa lo hace todo.

La forma de protegerse es cerrar los puertos por los que nos puedan atacar (el 139 principalmente) o aplicar algún parche al SO para quitarnos el bug. Otra solución menos recomendable es la que llevan a cabo algunos ISPs (proveedores de Internet), y que consiste en filtrar todos los paquetes dirigidos al puerto 139 (inconveniente: nos están dejando inoperativo ese puerto). Hoy en día es muy popular este bug y normalmente está ampliamente parcheado (aunque siempre habrá algún que otro despistado que no lo tenga instalado }=)).

Como hemos dicho, este ataque no sólo consigue echar a la víctima del server sino que además le deja colgado el ordenador (tendrá que hacer un *reboot*), lo que lo hace especialmente peligroso. La víctima saldrá del IRC con un mensaje de tipo *ping-timeout* como:

*[19:56] *** Goku has quit IRC (Ping timeout for Goku[system.tech.arrakis.es])*

Aquí teneis una curiosa muestra en PERL de cómo implementar el ataque (autor: Randal Schwartz):

```
#!/usr/bin/perl
use IO::Socket;
IO::Socket::INET
      ->new(PeerAddr=>"bad.dude.com:139")
      ->send("bye", MSG_OOB);
```

Y por último, el programa definitivo en C, que tiene la ventaja de que *spoofea* la dirección origen:

 winbomber.c

* Puedes encontrar información oficial sobre el bug y links a los parches correspondientes (Win95 y NT) en:
http://premium.microsoft.com/support/kb/articles/q168/7/47.asp

[Indice]

3.4.- Ping.-

Algunos los llama también *"IP bombs"* (bombas IP). Un *ping* es un tipo de mensaje ICMP que se usa para ver si una máquina se encuentra operativa y accesible. El procedimiento es enviarle un ping a la máquina; ésta lo recibe y contesta. Al recibir la contestación ya sabemos que la máquina vive. Si no se recibe en un plazo dado se considera como no accesible (la máquina podría estar apagada, o todos los "caminos" en la red hacia ella cortados). Además podemos obtener más datos como el grado de saturación de una máquina o red (midiendo el tiempo de respuesta de la máquina, es decir, el tiempo transcurrido desde que una máquina origen envía el ping hasta que recibe la contestación de la otra).

La manera de usar esto de forma ofensiva consiste en mandar más pings a un usuario de los que su máquina pueda contestar, saturándole su conexión a Internet. Por tanto se debe de hacer desde un enlace más potente que el que pretendemos atacar.

Lo típico es que la víctima esté en su casa y tenga un modem. Por tanto, necesitamos una conexión a inet más rápida que eso. Lo normal es atacar desde una máquina ubicada ya en la red (conectada mediante *ATM*, *FDDI*, ...). Por ejemplo, puede valer la cuenta de la Universidad ;-). La forma de hacerlo sería abriéndonos un shell y tecleando:

$ ping -s 32000 <direcc. IP>

(la sintaxis puede variar ligeramente según sistema operativo).

Los paquetes que se envían al hacer ping son típicamente muy pequeños. Con el modificador *-s* estamos forzando un nuevo tamaño (32000 bytes es aceptable; también podeis probar con 64000).

Pensad: un modem de 28.8 tardará unos 18 segs. en recibir 64 Kbytes (sin considerar compresión), mientras que desde nuestra shell lo hemos mandado en ¡¡décimas de segundo!! Si consideramos además que el comando ping manda más de un paquete (los que queramos) ... ¡boom! Tendréis el modem de vuestra víctima trabajando a toda pastilla para nada y fastidiándole todo lo que esté haciendo. En particular, le estropearéis su conexión al IRC: en el mejor de los casos la víctima tendrá un *lag* horroroso y en el peor será expulsada del servidor por *"ping time-out"*.

Desgraciadamente la *solución* para evitar este ataque no suele ser fácil ya que no se trata de un bug que se pueda parchear sino de la propia mecánica de los protocolos TCP/IP. Aconsejo usar un *bouncer* localizado en una máquina potente capaz de absorber el ataque.

Aquí teneis un par de programas que mandan los paquetes *spoofeados*:

pong.c

g00k.zip

[Indice]

3.4.1.- Ping of death.-

Este viejo ataque afecta a *kernels* muy antiguos (Linux 2.0.7 y 2.1.1, SunOS y HPUX antiguos, étc)

y consigue "congelarlos" en el acto o provocar un "reboot". Consiste en mandar a la víctima paquetes *ICMP* (de "*ping*", vulgarmente hablando) de un tamaño mayor al permitido por el kernel de la víctima. Lógicamente el ataque no se puede realizar desde una máquina de características similares a la víctima, por razones obvias. Sin embargo, sí se conseguía llevar a cabo desde una máquina Win95 ó WinNT, ya que éstas soportaban el tamaño de paquete requerido. El comando a usar desde estos SO's era:

> *ping -l 65510 host.victima*

Nótese que este comando se refiere exclusivamente a Windows antiguos. Así, por ejemplo, no funciona en NT 4.0 (da como error "valor incorrecto para la opción -l" ya que el máximo *ping* que puede lanzar es de longitud 65500), pero sí en versiones más antiguas y sin parchear. También se puede lanzar desde *Unix*'es no demasiado antiguos.

Para los *antiguos* salió una versión que emulaba el referido *ping*:

simping.c

Hoy en día este ataque está ampliamente superado y he decidido incluirlo en el documento por "razones históricas" :-). Aunque todavía es posible encontrar máquinas muy antiguas vulnerables a este ataque.

También es posible en la actualidad atacar algún Windows desde un Linux, p.ej.

[Indice]

3.5.- Ssping.-

Nos encontramos ante otro *bug* parecido al del *OOB*, que afecta a Win95 y NT (aunque no a todas las configuraciones), y cuya idea es que el "maravilloso" Windows "se lía" a la hora de reconstruir paquetes que le han llegado fragmentados y acaba con un cuelgue del ordenador. El ataque consiste en precisamente mandar esos paquetes fragmentados a la víctima.

Un *bug* de este tipo es viejo conocido de los sistemas *UNIX* (el ataque se conocía como *"ping of death"*); pero la novedad es que ahora lo sufren los *Windows*. Aunque son cosas técnicamente diferentes, la forma de proceder a la hora de atacar es análoga a *OOB*: solo hay que saber la dirección IP de la víctima y ¡boom!: le dejamos colgado el ordenador.

La *solución* pasa por parchear el S.O. En particular, este bug parece que no afecta al *Trumpet Winsock* así que si lo usais estareis protegidos.

A continuación teneis un *programa* que implementa el ataque con *spoofing* incluido:

jolt.c

[Indice]

3.6.- Teardrop.-

Este ataque nos recuerda al "*ssping*" ya que también está basado en un error al reensamblar paquetes fragmentados que le han llegado a la víctima. Esta vez la pila TCP/IP se lía al intentar ensamblar

paquetes fragmentados que se solapan entre sí (es decir, hay regiones de "datos" comunes en dos paquetes distintos) provocando el cuelgue del ordenador. A diferencia de "*ssping*" y "*OOB*" este bug afecta a multitud de S.O.'s, incluídos *Linux*, *Win95* y *NT*.

El programa que explota esta vulnerabilidad es el siguiente:

 teardrop.c

* La *solución* a este bug pasa por parchear o actualizar nuestro S.O, a saber:
- Linux: actualizar el *kernel* a 2.0.32 o superior.
- Windows 95 y NT: información y parches disponibles en:
http://premium.microsoft.com/support/kb/articles/q154/1/74.asp
ftp://ftp.microsoft.com/bussys/winnt/kb/Q154/1/74.TXT
http://premium.microsoft.com/support/kb/articles/q174/0/95.asp

Me parece que en Windows 95 basta con los siguientes parches (pero no os fieis demasiado, no los he probado):
http://support.microsoft.com/download/support/mslfiles/Vtcpupd.exe
http://support.microsoft.com/download/support/mslfiles/Vipup11.exe (para *Winsock 1.1*)
http://support.microsoft.com/download/support/mslfiles/Vipup20.exe (para *Winsock 2*)

(se recomienda hacer un backup de los ficheros *vip.386*, *vtcp.386* y *vnbt.386* antes de aplicar los parches, por lo que pudiera pasar).

 [Indice]

3.7.- Land.-

Consiste en mandar a un *host* y puerto especificado (víctima) un paquete *spoofeado* con el bit *syn* (petición de conexión) activo y con dirección y puerto fuente los mismos a los cuales va dirigido el paquete (i.e. dirección y puerto de la víctima). Resumiendo, hacemos como si la máquina víctima quisiera realizar una conexión a ella misma. Esto provoca el cuelgue de la máquina víctima en la mayoría de los casos o en otros que la carga de la CPU alcance valores insospechados.

Este bug afecta a una gran variedad de sistemas (*routers* incluídos). Una lista bastante completa (pero nunca al 100%) de sistemas afectados es la siguiente:

```
AIX 3                                  IS  vulnerable
AIX 3.2                                NOT vulnerable
AIX 4                                  NOT vulnerable
AIX 4.1                                NOT vulnerable
AIX 4.2.1                              NOT vulnerable
AmigaOS AmiTCP 4.0demo                 NOT vulnerable
AmigaOS AmiTCP 4.2 (Kickstart 3.0)     IS  vulnerable
AmigaOS Miami 2.0                      NOT vulnerable
AmigaOS Miami 2.1f                     NOT vulnerable
AmigaOS Miami 2.1p                     NOT vulnerable
AmigaOS Miami 2.92c                    NOT vulnerable
BeOS Preview Release 2 PowerMac        IS  vulnerable
BSDI 2.0                               IS  vulnerable
BSDI 2.1 (vanilla)                     IS  vulnerable
BSDI 2.1 (K210-021,K210-022,K210-024)  NOT vulnerable
```

BSDI 3.0	NOT vulnerable
DG/UX R4.12	NOT vulnerable
Digital UNIX 3.2c	NOT vulnerable
Digital UNIX 4.0	NOT vulnerable
Digital VMS ???	IS vulnerable
FreeBSD 2.1.6-RELEASE	NOT vulnerable
FreeBSD 2.2.2-RELEASE	NOT vulnerable
FreeBSD 2.2.5-RELEASE	IS vulnerable
FreeBSD 2.2.5-STABLE	IS vulnerable (fixed)
FreeBSD 3.0-CURRENT	IS vulnerable (fixed)
HP External JetDirect Print Servers	IS vulnerable
HP-UX 9.03	NOT vulnerable
HP-UX 10.01	NOT vulnerable
HP-UX 10.20	NOT vulnerable
IBM AS/400 OS7400 3.7	IS vulnerable (100% CPU)
IRIX 5.2	IS vulnerable
IRIX 5.3	IS vulnerable
IRIX 6.2	NOT vulnerable
IRIX 6.3	NOT vulnerable
IRIX 6.4	NOT vulnerable
Linux 1.2.13	NOT vulnerable
Linux 2.1.65	NOT vulnerable
Linux 2.0.30	NOT vulnerable
Linux 2.0.32	NOT vulnerable
MacOS MacTCP	IS vulnerable
MacOS OpenTransport 1.1.1	NOT vulnerable
MacOS 7.1p6	NOT vulnerable
MacOS 7.5.1	NOT vulnerable
MacOS 7.6.1 OpenTransport 1.1.2	IS vulnerable (not a compleate lockup)
MacOS 8.0	IS vulnerable (TCP/IP stack crashed)
MVS OS390 1.3	NOT vulnerable
NetApp NFS server 4.1d	IS vulnerable
NetApp NFS server 4.3	IS vulnerable
NetBSD 1.1	IS vulnerable
NetBSD 1.2	IS vulnerable
NetBSD 1.2a	IS vulnerable
NetBSD 1.2.1	IS vulnerable (fixed)
NetBSD 1.3_ALPHA	IS vulnerable (fixed)
NeXTSTEP 3.0	IS vulnerable
NeXTSTEp 3.1	IS vulnerable
Novell 4.11	IS vulnerable (100% CPU for 30 secs)
OpenBSD 2.1	(conflicting reports)
OpenBSD 2.2	NOT vulnerable
OpenVMS 7.1 with UCX 4.1-7	IS vulnerable
OS/2 3.0	NOT vulnerable
OS/2 4.0	NOT vulnerable
QNX 4.24	IS vulnerable
Rhapsody Developer Release	IS vulnerable
SCO OpenServer 5.0.2 SMP	IS vulnerable
SCO OpenServer 5.0.4	IS vulnerable (kills networking)
SCO Unixware 2.1.1	IS vulnerable
SCO Unixware 2.1.2	IS vulnerable
Salaris 2.4	NOT vulnerable
Solaris 2.5.1	NOT vulnerable
Solaris 2.5.2	NOT vulnerable

Solaris 2.6	NOT vulnerable
SunOS 4.1.3	IS vulnerable
SunOS 4.1.4	IS vulnerable
Ultrix ???	NOT vulnerable
Windows 95 (vanilla)	IS vulnerable
Windows 95 + Winsock 2 + VIPUPD.EXE	IS vulnerable
Windows NT (vanilla)	IS vulnerable
Windows NT + SP3	IS vulnerable
Windows NT + SP3 + simptcp-fix	IS vulnerable

Some misc stuff:

3Com Accessbuilder 600/700	NOT vulnerable
3Com LinkSwitch 1000	NOT vulnerable
3Com OfficeConnect 500	NOT vulnerable
3Com SuperStack II Switch 1000	IS vulnerable
Adtran TSU Rack	NOT vulnerable
Apple LaserWriter	IS vulnerable
Ascend 4000 5.0Ap20	NOT vulnerable
Ascend Pipeline 50 rev 5.0Ai16	NOT vulnerable
Ascend Pipeline 50 rev 5.0Ap13	NOT vulnerable
BayNetworks MARLIN 1000 OS (0).3.024(R)	NOT vulnerable
BinTec BIANCA/BRICK-XS 4.6.1 router	IS vulnerable
Cisco Classic IOS < 10.3, early 10.3, 11.0, 11.1, and 11.2	IS vulnerable
Cisco IOS/700	IS vulnerable
Cisco Catalyst	IS vulnerable
Digital VT1200	IS vulnerable
Farallon Netopia PN440	NOT vulnerable
HP Envizex Terminal	IS vulnerable
LaserJet Printer	NOT vulnerable
Livingston Office Router (ISDN)	IS vulnerable
Livingston PM ComOS 3.3.3	NOT vulnerable
Livingston PM ComOS 3.5b17 + 3.7.2	NOT vulnerable
Livingston PM ComOS 3.7L	NOT vulnerable
Livingston PM ComOS 3.7.2	NOT vulnerable
Livingston Enterprise PM 3.4 2L	NOT vulnerable
Livingston T1/E1 OR	IS vulnerable
Milkyway Blackhole Firewall 3.0 (SunOS)	IS vulnerable
Milkyway Blackhole Firewall 3.02(SunOS)	IS vulnerable
NCD X Terminals, NCDWare v3.1.0	IS vulnerable
NCD X Terminals, NCDWare v3.2.1	IS vulnerable
Netopia PN440 v2.0.1	IS vulnerable
Proteon GT60	NOT vulnerable
Proteon GT60Secure	NOT vulnerable
Proteon GT70	NOT vulnerable
Proteon GT70Secure	NOT vulnerable
Proteon GTAM	NOT vulnerable
Proteon GTX250	NOT vulnerable
Proteon RBX250	NOT vulnerable
Sonix Arpeggio	NOT vulnerable
Sonix Arpeggio +	NOT vulnerable
Sonix Arpeggio Lite	NOT vulnerable

Podeis comprobar la efectividad del ataque usando:

 land.c

* Información y parches para NT:
ftp://ftp.microsoft.com/bussys/winnt/winnt-public/fixes/usa/nt40/hotfixes-postSP3/land-fix/Q165005.txt

* Información y parches para Win95:
ftp://ftp.microsoft.com/bussys/winnt/winnt-public/fixes/usa/nt40/hotfixes-postSP3/land-fix/Q177539.TXT

 [Indice]

3.8.- Teardrop modificado.-

Nos encontramos ante una variante del temido *teardrop*. Esta vez lo que se hace es mandar paquetes *UDP* fragmentados a puertos aleatorios de la máquina víctima. La peculiaridad es que esta vulnerabilidad sigue existiendo en máquinas Windows ¡***ya*** parcheadas contra *teardrop*!

Aquí teneis el *source* para reproducir el ataque:

 newtear.c

* Como siempre, para evitar este ataque tenemos que aplicar los correspondientes parches:
ftp://ftp.microsoft.com/bussys/winnt/winnt-public/fixes/usa/NT40/hotfixes-postSP3/teardrop2-fix/

 [Indice]

3.9.- Smurf.-

Consiste en mandar paquetes *ICMP* de *"echo request"* (como los de un *ping*) con una falsa dirección de origen (la IP de la víctima) a una dirección de difusión (*"broadcast"*). De esta forma alcanzarán muchas máquinas, cada una de las cuales enviará sus paquetes de respuesta a la dirección origen de la petición (cuyos datos estaban falseados y apuntaban a la víctima). El resultado es que la máquina víctima se ve inundada de paquetes IP, resultando en una saturación de su enlace (algo así como si le hubieran hecho un *ping* a lo grande).

Desgraciadamente la víctima no puede hacer nada para evitarlo. La solución está en manos de los administradores de red, los cuales deben configurar adecuadamente sus *routers* para filtrar los paquetes ICMP de petición indeseados (*broadcast*) o bien configurar sus máquinas para que no respondan a dichos paquetes. Es decir, que lo que se parchea son las máquinas/redes que puedan actuar de intermediarias (inocentes) en el ataque y no la máquina víctima.

De igual forma también se podría evitar el ataque si el *router*/*firewall* de salida del atacante (p.ej. el *ISP* al que pertenece) estuviera convenientemente configurado para evitar *spoofing*. Esto se haría filtrando todos los paquetes de salida que tuvieran una dirección *source* (origen) que no perteneciera a la red interna (desde la cual salimos).

He aquí el programa que implementa el ataque:

 smurf.c

Para más información sobre "*smurf*" os remito a:
http://www.quadrunner.com/~c-huegen/smurf.txt

También podeis echar un vistazo a listas de redes mal configuradas y que pueden "colaborar" con nosotros para lanzar el ataque. Son lo que podemos llamar "*smurf amplifiers*". Se hacen públicas para poner en evidencia a los administradores de dichas redes y que éstos no tengan más remedio que parchearlas. Toma nota:
http://netscan.org
http://www.powertech.no/smurf/ ("Smurf Amplifier Registry")

 [Indice]

3.10.- Bonk (/Boink).-

La mecánica de este ataque es parecida a la del *Teardrop*. De hecho, se puede hablar de *Teardrop a la inversa*. Esta vez lo que se fuerza es a que el offset del fragmento sea mayor que la longitud de la cabecera. El intento de reensamblaje por parte de la víctima conlleva el cuelgue del ordenador. Afecta a Win95 (y quizás NT) incluso una vez parcheado contra *Teardrop2*.

A continuación teneis el programa original que da nombre al ataque. Este ataca al puerto 55 de la víctima. Existe una variante, llamada "*boink*", que barre un rango de puertos, pero el fundamento es el mismo y no creo que merezca la pena plasmar aquí el fuente.

 bonk.c

 [Indice]

3.11.- Syndrop.-

Tal y como reza el comentario del siguiente source este bug es una mezcla de otros dos: el basado en secuencias con el flag SYN y el Teardrop. Es decir, es una variante más del *Teardrop*.

Si quereis podeis echarle un vistazo al *source*:

 syndrop.c

 [Indice]

3.12.- Overdrop.-

Afecta al kernel 2.0.33 de Linux y es más que nada molesto.

Consiste en mandar paquetes incorrectos de forma que el kernel víctima produzca los correspondientes mensajes de error, que normalmente son logeados en fichero. Se consigue pues

llenar el disco duro o la consola de la víctima con mensajes de este tipo.

Aquí teneis el *fuente* para llevar a cabo el ataque:

 overdrop.c

 [Indice]

3.13.- Nestea (/Nestea2).-

Se trata de una nueva variante del *Teardrop*.

Ofrezco la versión 2 de *Nestea* que no es otra cosa que la primera pero con un poco más de estética (colores y demás):

 nestea2.c

 [Indice]

3.14.- Trojans.-

Un "*troyano*" (o "*trojan*", en inglés) es un programa que "cuelan" en tu ordenador y que tiene otras funciones (normalmente perniciosas) en vez de o además de las funciones normales (para lo que se supone que sirve el programa). En este sentido, es análogo a un virus, solo que no se extiende por sí sólo (normalmente): hay que introducirlo "manualmente" a la víctima y forzarla o convencerla para que lo ejecute.

En el caso del IRC lo que se hace usualmente es mandárselo a la víctima vía DCC. Cuando ésta ejecuta el programa se ve infectada o le ocurre algo pernicioso }:-).

Para que se entienda el concepto: imaginad que yo creo un fichero llamado "*sorpresa.bat*" con lo siguiente:

```
@echo off
del c:\autoexec.bat
del c:\config.sys
echo Sorpresa...
```

Ahora se lo envío a alguien y éste lo ejecuta. Entonces se le borrarían los ficheros de arranque };-)

Esto que se ve tan simple se puede implementar de forma mucho más elaborada creando un *.exe* de forma que en principio no podemos saber lo que hace el programa internamente hasta que lo ejecutamos.

Solución: como normal general no acepteis ficheros ejecutables de extraños, o por lo menos, no los ejecuteis. Si os van a pasar algún programa extraño es aconsejable que os pasen el *source* para que se le pueda echar un vistazo al programa, asegurarnos de que no hace "cosas raras" y entonces compilarlo nosotros mismos. Esto último es lo más seguro.

 [Indice]

3.14.1.- Mirc.ini.-

Se trata del fichero de configuracion principal del cliente mIRC. Es muy potente porque permite *scripting*, o sea, crearte tus propios comandos a medida y automatizar tareas.

Algunos graciosos se dedican a pasar por DCC una versión "modificada" (troyana) que circula por ahí y que entre otras cosas añade ciertos comandos para que remotamente un intruso pueda hacerte cosas en tu ordenador. Sólo funcionará si se consigue que el archivo troyano reemplace al archivo *mirc.ini* original, lo cual tampoco es fácil ya que tendríamos que tener configurado nuestro mIRC para que la zona de download apunte al directorio "raiz" de mIRC (de forma que el fichero recibido por DCC vaya al directorio donde se encuentran los ejecutables y ficheros de configuración de mIRC) y además aceptar el DCC (o tener activada la opción de "*auto-get*", o sea, de aceptar automáticamente los DCC's, cosa que no aconsejo en absoluto).

Un ejemplo de script troyano puede ser éste:

 script.ini

Solución: no uses el "*auto-get*" ni aceptes ficheros de extraños o que no sepas bien lo que hacen.

 [Indice]

3.14.2.- Dmsetup.exe.-

Este troyano se copia a sí mismo en diversos lugares del disco duro, crea su propio *mirc.ini* / script asociado, y se añade a un fichero de sistema, para que no pueda ser eliminado fácilmente. El script lo único que hace es mandar el fichero troyano ("*dmsetup.exe*") a todo el que entra en nuestro canal, aparte de provocarnos un */quit* del servidor IRC al escuchar cierta palabra clave.

Los pasos que lleva a cabo el troyano dependen de la unidad en que tengamos instalado el mIRC:

A) Unidad C:

(es decir, si tienes c:\mirc\)

La primera vez que "dmsetup.exe" se ejecuta:

1. Se copia a sí mismo en:
   c:\
   c:\mirc
   c:\windows
   c:\program files

2. Crea c:\configg.sys.
3. Añade una linea con "dmsetup" en el fichero "autoexec.bat"
4. Copia "mirc.ini" a "backup0412.ini"
5. Crea "mirc.ini" y "mircrem.ini"
6. Da un error de tipo 0 (para que pienses que el download está roto).

Las siguientes veces que se ejecuta "dmsetup.exe" hace los mismos pasos, excepto 2 y 3, debido a la existencia del fichero "configg.sys".

+ Para eliminar el troyano de un sistema infectado:

1. Hacer "unload" del fichero "mircrem.ini".
2. Abrir "c:\autoexec.bat" con el bloc de notas, y quitar la linea "dmsetup". Grabar y salir.
3. Borrar:
    c:\dmsetup.exe
    c:\configg.sys
    c:\mirc\dmsetup.exe
    c:\mirc\mircrem.ini
    c:\mirc\backup0412.ini
    c:\windows\dmsetup.exe
    c:\progra~1\dmsetup.exe

B) Otra unidad que no es C:

El troyano sigue los pasos 1 a 3 de arriba, y después devuelve un error de tipo 1. Aunque al no existir c:\mirc, el paso 1 varía un poco: el troyano se copia precisamente a esta localización (cambia de nombre). Como no modifica el fichero "mirc.ini", no sabremos en principio que estamos infectados (hasta que cambiemos nuestro mIRC de unidad).

- - -

Todo esto ha sido el funcionamiento de la ***versión original*** de este troyano. Sin embargo, se han sucedido nuevas versiones del mismo, cuyas actuaciones y modos de proceder varían. A continuación, veremos los síntomas que padecerá un usuario infectado con las diferentes versiones, para que podamos identificarlas.

* **Original**:

- distribuye "dmsetup.exe", de unos 57,556 bytes.
- algunas veces resulta en un *username* de una sola letra, como "s".
- crea "dmsetup.exe" en varios sitios, como c:\, c:\mirc, c:\windows, c:\program files.
- crea el fichero "c:\configg.sys".
- añade una linea de referencia a "dmsetup" al fichero "autoexec.bat".
- si tu mIRC está en C:, crea c:\mirc\mircrem.ini y c:\mirc\backup0412.ini .

* **2ª variante**:

- el nombre del fichero que se distribuye varía, siendo de unos 81,084 bytes.
- la barra de título de mIRC dice "your mirc is buggy".
- crea "filename.ini".
- crea la linea "filename -inauto" en el "autoexec.bat".
- crea c:\ni.cfg, bakupwrks.ini.
- crea dos carpetas en el directorio de download del mIRC: "_dm2yif" y "suck_it".

* **4ª variante**:

- el nombre del fichero que se distribuye varía, siendo de unos 81,084 bytes (llamémosle "loquesea.exe").
- la barra de título de mIRC dice "your mirc is buggy".
- crea "loquesea.ini", c:\mirc\loquesea.ini, c:\ni.cfg, c:\mirc\bakupwrks.ini y c:\mirc\mircrem.ini (nota: "loquesea" puede ser cualquier cadena de caracteres).
- crea la linea "loquesea -inauto" en c:\autoexec.bat.
- crea dos carpetas en el directorio de download del mIRC. El nombre de las carpetas varía, pero si intentas borrarlas no te dejará y te dirá que la carpeta está vacía.

- si el usuario intenta entrar en un canal de ayuda, automáticamente saldrá del canal.
- "loquesea.exe" será instalado en: C:\, c:\mirc, C:\windows, c:\windows\system, C:\games, C:\doom, c:\doom2, c:\quake, c:\quake2, y c:\picsd. Si estos directorios no existen, en su lugar se creará un fichero con ese nombre (quake, doom, doom2, étc).

*** 5ª variante**:

- muy parecida a la 4ª variante explicada más arriba, excepto que puede ser distribuída como "loquesea.jpg.com", que desde Win95/98 se ve sólo como "loquesea.jpg". De esta forma, el usuario cree que se trata de un dibujo.
- distribuye "loquesea.jpg.com", de unos 80 Kbytes.
- crea una copia de sí mismo llamada C:\Windoom.exe y C:\Windows\Freeporn.exe.
- crea un fichero llamado "taged.lmr", que contiene una sóla línea: "Ni!" (mirar la linea n41 de pusysex.ini).
- copia dos lineas al final de "c:\autoexec.bat".
- crea los siguiente ficheros en el directorio del mirc.ini: remote.ini, var.ini y Buttfuck.ini.
- si el usuario intenta entrar en un canal de ayuda (excepto #mirc), éste será expulsado del servidor (/quit).
- crea sobre unos 200 directorios en C:\.
- crea aproximadamente 20,000 directorios en C:\program files (C:\progra~1), dependiendo del espacio libre que tengas en el disco duro.
- crea un nuevo mirc.ini en C:\mirc y hace backup del viejo como bakupwrks.ini.


[Indice]

3.14.3.- Back Orifice.-

También conocida como "*BO*", por sus iniciales, es una herramienta escrita por el grupo "*Cult of the Dead Cow*" que permite la administración remota de una máquina con Windows 95 / 98. Respetando el tradicional esquema cliente / servidor, el programa servidor es instalado en la máquina a administrar y el cliente desde otra máquina puede enviar comandos al servidor de forma que se obtiene un control total sobre la primera máquina: podemos ejecutar comandos, listar ficheros, arrancar servicios de forma transparente, compartir directorios, subir y bajar ficheros, manipular el registro, matar o listar procesos, étc.

La página oficial del *BO* es:
http://www.cultdeadcow.com

El problema viene cuando un atacante se las ingenia para que una posible víctima instale el software servidor en su máquina, sin darse cuenta, al más puro estilo troyano. Esto es posible si por ejemplo el atacante le envía a la víctima el programa '*boserver.exe*" renombrado a digamos "*antinuke.exe*", y ésta, inocentemente, lo ejecuta creyendo que se trata del último y fabuloso programa anti-nuke ;-). A partir de ese momento, el ordenador de la víctima se encuentra "infectado" con lo que técnicamente se denomina un "*back-door*" ("*puerta trasera*"), y el atacante o cualquiera que tenga el cliente de *BO* (y que sepa o adivine ciertos datos acerca de la configuración del servidor) podrá acceder al ordenador de la víctima. Se trata de un grave problema de seguridad.

Cuando ejecutamos el programa servidor, se instala el *back-door*, siguiendo los siguientes pasos:
- se copia el programa servidor en el directorio de sistema de Windows (normalmente "*c:\windows\system*") con el nombre especificado por el usuario (/atacante).
- se crea una entrada de registro en "*HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices*" apuntando al nombre de fichero del servidor referido anteriormente y con una descripción de "*(Default)*" o bien la que haya especificado el usuario (/atacante).

- el servidor comenzará a escuchar el puerto *31337 UDP* o el especificado por el usuario (/atacante), atendiendo a las peticiones del cliente (/atacante). El protocolo usado utiliza una débil encriptación, fácil de romper.

* *Cómo detectarlo*: accedemos a la entrada del registro arriba nombrada, usando '*regedit.exe*', y buscamos un servicio que no hayamos instalado nosotros. Si el fichero al que apunta es de una longitud aproximada de 124,928 bytes, entonces es posible que se trate de *BO*.

Otra forma es usar el comando "*netstat*":

```
C:\WINDOWS>netstat -an | find "UDP"
  UDP    0.0.0.0:31337          *:*
```

* *Cómo eliminarlo*: simplemente tenemos que borrar tanto el programa servidor del directorio de sistema de Windows como la entrada del registro que apunta hacia él (anteriormente mencionada). Teneis instrucciones aquí:
http://www.nwi.net/~pchelp/bo/bo.html

* También existen diversos programas que detectan y/o eliminan el *BO*, aunque hay que tener cuidado con ellos porque muchos no son más que nuevos troyanos (ej.: "*BOsniffer.zip*" o "**.exe*"). La manera más fácil y cómoda de detectar / eliminar este *back-door* es usando un *anti-virus* actualizado, como puede ser el "*AntiViral Toolkit Pro*" (*AVP*):
http://www.avp.ru/ (página *oficial*)
http://www.ontinet.com/avp/avp.htm(versión en *castellano*)

* *Cómo determinar la configuración de un BO instalado*: ver el fichero del servidor usando un editor de texto (como el "*Bloc de notas*"). Si la última linea del fichero es *'8 8$8 (8,8084888<8@8D8H8L8P8T8X8\8'8d8h8l8'* entonces está usando la configuración por defecto. En caso contrario, la configuración estará en las últimas lineas del fichero, en este orden:

<nombre del fichero>
<descripción del servicio>
<número de puerto>
<password>
<información sobre plug-in adicional>

* *Plug-in*'s: se trata de código que añade funcionalidad extra al servidor. Se conocen como "*BUTTplugs*". Así por ejemplo existe un *plug-in* que hace que la víctima conecte secretamente a un servidor y canal (ej.: *#bo_owned* en *EFNET*) de IRC dado, anunciándose ante los demás usuarios del canal como infectados. De esta forma, el atacante puede conectar a dicho servidor y canal de IRC (que se supone previamente conocido), y encontrarse las *IP*'s de numerosas víctimas.

Algunos *plug-in*'s conocidos:

- *Speakeasy*: es el comentado del IRC.
- *Silk Rope*: enlaza el *BO* a casi cualquier programa existente.
- *Saran Wrap*: esconde el *BO* dentro de un programa instalador estándar "*InstallShield*".
- *Butt Trumpet*: manda al atacante un email con la dirección *IP* de la máquina víctima, una vez que *BO* se ha instalado.

Aquí podeis encontrar los últimos *plug-in*'s:
http://www.cultdeadcow.com/tools/bo_plugins.html

* Últimamente se han descubierto evidencias que nos inclina a pensar que también el programa cliente del *BO* puede ser un *back-door* en favor de los creadores del programa. Si monitorizamos el tráfico de red mientras corremos este programa (con un "*netstat*", sin ir más lejos) se puede observar que se abre una conexión hacia el puerto *80* (*httpd*) del site de los creadores del programa, lo que en principio parece no tener demasiado sentido en lo que respecta a las funciones del cliente.


[Indice]

3.14.4.- NetBus.-

Se trata de una herramienta similar al *BO*, en ciertos aspectos incluso más avanzada, ya que aparte de soportar todas las funciones del *BO*, presenta funcionalidades extra como pueden ser abrir/cerrar la unidad de CDROM, mandar cuadros de diálogo interactivos para poder hacer *chat* con el sistema comprometido o escuchar el micrófono de dicho sistema. Funciona en Windows 95 / 98 / NT.

Esta es la página oficial del programa:
http://members.spree.com/SIP/NetBus/

NetBus escucha los puertos *12345* y *12346* de *TCP*. No usa encriptación y los comandos tienen el siguiente formato:
<Comando>;<Arg1>;<Arg2>;...;<Argn>

Por defecto, el servidor NetBus se llama "*Patch.exe*". Es posible ponerle *password*, el cual se guardaría sin cifrar en la entrada de registro "*HKEY_CURRENT_USER\Patch\Settings\ServerPwd*". El cliente se autentificaría mandando una cadena con el formato "*Password;0;mi_password*", aunque si en vez de un *0* ponemos un *1*, conseguiremos autentificarnos con cualquier *password*. Esto es un *back-door* que contiene el propio programa.

* *Cómo detectarlo*:

```
C:\WINDOWS>netstat -an | find "12345"
  TCP    0.0.0.0:12345          *:*
```

* *Cómo eliminarlo*: hay diferentes formas para hacerlo, dependiendo de la versión de que se trate.

Versión 1.5x:
http://members.spree.com/NetBus/remove_1.html

Versión 1.6:
http://members.spree.com/NetBus/remove_2.html

Para esta última versión, también es posible eliminarlo remotamente haciendo *telnet* al puerto 12345 y mandando lo siguiente (sin las comillas):
"Password;1;"<Enter>"RemoveServer;1"<Enter>

Esto último funciona incluso existiendo *password*, aunque no se borra el fichero *.exe*, lo que deberá hacerse manualmente.

* Al igual que ocurría con el *BO*, lo más cómodo para *detectar / eliminar* este *back-door* es usar un *anti-virus* actualizado, como puede ser el "*AntiViral Toolkit Pro*" (*AVP*).


[Indice]

3.15.- Hanson.

Afecta al cliente mIRC 5.3x con el puerto 113 (ident) abierto. Se consigue matar al programa cliente o ralentizar mucho el sistema (Windows 95/NT). La idea es mandar al puerto especificado cadenas más largas de las que admite el buffer interno de mIRC provocando un *"crash"* del programa.

 hanson.c

 [Indice]

3.16.- Modem DoS.-

Las técnicas que a continuación paso a explicar atacan directamente al modem o al puerto al cual está conectado el mismo y consiguen romper la conexión con nuestro *ISP*.

3.16.1.- /Msg ComX.-

Funciona con algunos clientes de IRC (preferentemente en versiones antiguas). Consiste en que cualquier mensaje (via */msg*) que se envíe a un usuario cuyo nick coincida con un nombre lógico de puerto (*com1*, *com2*, ..., *lpt1*, ..., *aux*, ...) es directamente enviado al puerto en cuestión y no al usuario de IRC. Estamos, pues, mandando datos arbitrarios a un puerto, lo cual tendrá resultados impredecibles y conducirá a error. Si el puerto usado es aquel al cual está conectado el modem (usualmente *com2*), estropearemos la conexión que se estuviera llevando a cabo en ese momento. Queda claro que la víctima de este bug es quien realiza el */msg*; por tanto, nuestra intención como atacante será engañar a la víctima para que mande ese */msg*.

* *Ejemplo*:

Atacante:
/nick com1
/msg victima Oye, ¿estás ahí?

Víctima:
/msg com1 Sí, dime.

(ahora si todo va bien se le habrá colgado el ratón, que es el dispositivo que suele estar en *com1*).

Atacante:
/nick com2
/msg victima ¿Me respondes o no?

Victima:
/msg com2 ¡Ya lo he hecho antes!

(ahora se habrá quedado sin conexión, puesto que *com2* suele ser normalmente el modem).

* La mejor forma de *evitar* este ataque es tener un cliente mínimamente actualizado. Otra posible manera es ignorar los mensajes del atacante, para no caer en tentación :-). Por ejemplo, poniendo en */ignore* a *com1!*@*, com2!*@**, étc.

[Indice]

3.16.2.- +++ATH0.-

Este es un viejo ataque pero que sorprendentemente sigue siendo perfectamente válido para colgar un modem (y por tanto, la conexión al *ISP*) en muchas ocasiones. Es bastante poderoso, así que por favor, no abuses de él.

Para comprender su funcionamiento hay que explicar antes algo sobre modems. Vamos a ello.

La mayoría de modems usan un juego de comandos conocido como "*Hayes*". A través de estos comandos se puede enviar cualquier orden interna al modem o incluso configurarlo. Un comando-ejemplo muy conocido es "ATDTxxxxxx", que le indica al modem que marque por tonos el número xxxxxx.

El comando que nos será de utilidad ahora será "ATH0". La respuesta del modem a este comando es que rompe la conexión actual, o sea, "cuelga el teléfono". Si logramos que la víctima mande ese comando a su propio modem, éste desconectará. Esa es la idea.

Para que el modem entienda que los datos que se le está mandando son comandos, éste debe estar en lo que se conoce como "*modo de comandos*". Sin embargo, nosotros vamos a intentar mandar el comando estando el modem ya conectado al *ISP* y funcionando, por lo que éste no está en principio aceptando comandos sino "datos". Tenemos que, de alguna forma, forzar a que el modem entre en el *modo de comandos*. Para ésto, basta con lanzar una secuencia de "*códigos de escape*". Esta secuencia o evento fuerza a que se conmute al *modo de comandos*, de forma que los datos que vengan a continuación el modem los entiende como comandos. El *código de escape* está definido en el registro *S2* del modem y por defecto, es el 43 ("+"), de forma que la secuencia para entrar en *modo de comandos* es "+++". Detrás iría el comando a lanzar, que según lo visto, va a ser un "*ATH0*". Así y con todo, la cadena de caracteres completa que la víctima debe enviar a su modem queda como "+++*ATH0*", que es el nombre que usualmente se da al ataque.

Visto ésto, queda por saber cómo podemos aprovecharnos de esta característica. ¿Cómo podemos lograr que la víctima mande esa cadena a su propio modem?

Basándonos en que el protocolo *PPP*, que es el normalmente usado para conectar por modem a nuestro *ISP*, por defecto no comprime los paquetes *IP* que viajan sobre él, si la víctima manda un paquete *IP* de datos hacia el exterior que contenga la cadena "+++*ATH0*", lo que ocurrirá es que dicho paquete se encapsulará en una o varias tramas *PPP*, y dichas tramas se enviarán hacia el modem, para que éste las transmita por la linea telefónica, pasando la cadena "+++*ATH0*" de forma transparente a través del modem. Entonces el modem interpretará el "+++" como la secuencia de escape y ejecutará el comando "*ATH0*", que viene a continuación, que obliga al modem a desconectar.

Desde el punto de vista del atacante, para forzar a que la víctima mande la cadena "+++*ATH0*", hay muchas maneras posibles. Veamos unos ejemplos.

* *Ejemplo 1*:

```
[maquina@atacante]$ telnet 192.168.1.1 21
Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is '^]'.
220 foo FTP server (Version wu-2.4.2-academ[BETA-15](1) Fri Dec 12
```

```
20:41:
USER +++ATH0
^]
telnet> close
Connection closed.
[maquina@atacante]$ telnet 192.168.1.1 21
Trying 192.168.1.1...
telnet: Unable to connect to remote host: Network is unreachable
[maquina@atacante]$
```

La explicación es simple. Hemos hecho un *ftp* via *telnet*. Cuando nos ha preguntado el usuario, le hemos dicho que era "*+++ATH0*". Acto seguido el servidor ftp, o sea, la víctima, debió responder con algo como "*331 Password required for +++ATH0.*". Vemos que la víctima ha mandado sin querer la temida cadena para que el modem cuelgue ;-).

* *Ejemplo 2*:

Conectamos al puerto 25 ("*sendmail*") de la víctima y le mandamos:

```
HELO blah.com
VRFY +++ATH0
```

La víctima mandaría la maléfica cadena en una respuesta como:

```
550 +++ATH0... User unknown
```

* *Ejemplo 3*:

Desde mIRC:

```
//raw NOTICE NickVictima : $+ $chr(1) $+ PING +++ATH0 $+ $chr(1)
```

* *Ejemplo 4*:

```
[maquina@atacante]# ping -p 2b2b2b415448300d -c 5 xxx.xxx.xxx.xxx
PATTERN: 0x2b2b2b415448300d
PING xxx.xxx.xxx.xxx (xxx.xxx.xxx.xxx): 56 data bytes

--- xxx.xxx.xxx.xxx ping statistics ---
5 packets transmitted, 0 packets received, 100% packet loss
[maquina@atacante]#
```

La opción "*-p*" permite especificar un "*pattern*" que servirá para rellenar el paquete de ping ("*ICMP ECHO_REQUEST*"). La víctima responderá con un "*ICMP ECHO_REPLY*", conteniendo el mismo *pattern*. El *pattern* está codificado en hexadecimal y se traduce como "*+++ATH0<CR>*". No todos los programas de *ping* soportan esta opción.

* *Ejemplo 5*:

El mismo método anterior implementado en C:

 gin.c

- - - - - -

* *Nota*: en todos los casos, el atacante también envía la cadena "*+++ATH0*", por lo cual si éste es vulnerable, su modem caerá y el ataque se habrá vuelto contra el propio atacante.

* *Modems vulnerables*: no todos los modems son vulnerables a este ataque. Los modems "buenos", o sea, que soportan "Hayes" estrictamente (la patente) no son vulnerables. Esto es porque la norma especifica que después de recibir la secuencia de escape y antes de ningún comando, debe pasar un intervalo de tiempo "vacío" de 1 seg. Lógicamente, en nuestro ataque va todo (secuencia + comando) seguido, sin pausa, por lo que el ataque no sería efectivo. Un ejemplo de modem *no* vulnerable es el *US Robotics*.

Sin embargo, hay otros muchos modems que no siguen la patente; en particular, parece que todos aquellos que contienen el chipset *Rockwell*, que son bastante frecuentes hoy en día. Ejemplo: *Diamond SupraExpress 56k*.

* En todo caso, una *forma de protegerse* es deshabilitar la entrada manual al *modo de comandos*. Esto se hace cambiando el código de escape (que por defecto es un 43 ["+"]), y poniendo un valor de *255*. La mayoría de los modems reconocen un valor por encima de 127 como deshabilitador manual del *modo de comandos*, pero otros necesitan un valor superior. Para estar más seguros, usamos 255. El comando completo sería:

```
ATS2=255
```

y se podría introducir en la cadena de inicialización del modem (configurable normalmente en *propiedades* del modem o en la utilidad de *dial-up*), de forma que se "ejecute" siempre antes de marcar hacia nuestro *ISP*.

* También es posible parchear remotamente un *host* dado usando la misma técnica del "*ping -p*" vista para atacar:

```
ping -c 1 -p 2b2b2b415453323d32353526574f310d host
```

(el *pattern* usado equivale a: "*+++ATS2=255&WO1*").

De forma análoga, podemos mandar otros comandos remotos, provocando, por ejemplo, que el modem de la víctima marque a donde nosotros queramos, étc. ¡Muy peligroso!


[Indice]

3.17.- Fragmento de longitud 0 - Linux DoS.-

Este bug afecta al kernel de *Linux*, desde la versión *2.1.89* a la *2.2.3*. El ataque consiste en mandar a la víctima una serie de *fragmentos IP* de forma que el primero de ellos tenga *longitud nula*. Esto provoca que una cuenta de referencia en la información de rutado cacheada para el originador de ese paquete sea incrementada una vez de más. Esto hace imposible liberar la entrada de destino y borrarla del caché. Si llenamos las *4096* entradas de destino del caché, la máquina víctima no podrá reservar nuevas entradas y *la comunicación IP habrá quedado deshabilitada*.

Para que el ataque sea fructífero, es necesario dejar el programa funcionando durante unos minutos. Además habrá que ajustar convenientemente el retraso de forma que los paquetes no sean desechados.

Como no, aquí tenéis a vuestra disposición el *exploit*:

 sesquipedalian.c

 [Indice]

3.18.- Paquetes ICMP aleatorios - Linux DoS.-

Parece ser que el kernel *2.2.x* de *Linux* (probado en *2.2.7* y *2.2.9*) se bloquea ("*kernel panic*") cuando se le manda un gran número de paquetes *ICMP* específicos. No está claro el origen de este bug, pero se cree que aparece al combinar longitudes de cabecera mutiladas, y a la aleatoriedad de los paquetes *ICMP* que se mandan (*tipo*/*subtipo* y *dirección origen* aleatorios).

Puedes compilar el siguiente *exploit* con: g++ -- g++ random_icmp.c -o random_icmp.

 random_icmp.c

 [Indice]

3.19.- ICMP/Redirect-host DoS.-

Los paquetes *ICMP* de tipo *"Redirect Host"* son usados en el procedimiento de rutado (encaminamiento). Su misión consiste en informar al nodo al que van dirigidos de que debe usar otro camino (ruta). El nodo (un *host*, un *router*, étc) entonces modificará su tabla de rutas de acuerdo con la información recibida, de forma que pueda usar la ruta recién descubierta. Se consigue corregir rutas erróneas u obtener una ruta más óptima.

Para entenderlo mejor, supongamos dos nodos: H y R. Nuestro host va a ser H mientras que R va a ser un router perteneciente a nuestra LAN. Supongamos también que H tiene que enviar datos a un nodo cualquiera X. Entonces H construiría los correspondientes datagramas IP, y miraría en su tabla de rutas para ver hacia qué router debe enviar los paquetes que tengan como destino X. Asumamos que dicho router resulta ser R, por lo que H decide enviar esos datagramas (que tenían destino X) hacia R. Este es el procedimiento habitual de *encaminamiento* o *rutado*. Ahora supongamos que ha caido algún nodo intermedio en el camino de H a X, de forma que R recibe la información de que esa ruta no es válida y por el medio que sea (no voy a entrar aquí) sabe que existe una ruta alternativa, a través de R2 (supongo que R2 pertenece a la misma LAN que R y H, que es distinta a la de X). Entonces R necesita decirle a H que *"a partir de ahora mande sus paquetes (con destino final X) al router R2 en vez de a R"*. Esto lo hace enviando el mensaje *"Redirect Host"*, mediante ICMP. H leerá este paquete y comprenderá que debe alterar su tabla de rutas para que use R2 en lugar de R.

Una vez visto el procedimiento usual de rutado (o al menos la parte que nos interesa), veamos en qué consiste el ataque. Este afecta a máquinas con **Win9x / NT(sp4)**. El programa que lo implementa se dedica a mandar al host víctima una lluvia de paquetes de redirección, modificando la dirección origen (spoofeando) para que parezca que los mensajes son auténticos y vienen del router adyacente. Según lo explicado, nuestra víctima tendrá que procesar dichos mensajes y alterar su tabla de rutas una vez por cada mensaje. Como se mandan muchísimos, el resultado es que se sobrecarga al S.O., consiguiendo ralentizar extremadamente a nuestra víctima y con suerte, bloqueándole la máquina.

Debido a la naturaleza de este ataque, sólo funcionará si la víctima está dentro de nuestra LAN, lo que restringe en gran medida el campo de acción de este ataque.

Aquí teneis tanto la versión original del *exploit* como una modificada para que funcione desde *Solaris*:

winfreez.c

winfreez.c (*Solaris* port)

[Indice]

3.20.- Paquetes fragmentados IGMP en Windows.-

Microsoft no deja de sorprendernos. Esta vez la pila TCP/IP de la mayoría de sus productos **Windows** (incluidos **95**, **98**, **98se** y **2000**) se ve afectada por un bug que provoca desde la pérdida de red (es decir, que la red deja de funcionar) hasta un *reboot*, pasando por una *pantalla azul*. Se trata de una seria vulnerabilidad ya que la gran mayoría de usuarios que se conectan al IRC lo hacen desde alguno de estos sistemas Windows.

El bug se debe a un incorrecto trato por parte de Windows a los *paquetes fragmentados de tipo IGMP*.

No todos los ordenadores Windows son vulnerables, no se exactamente a qué es debido esto. Ha habido reportes que incluyen de todo, incluso que puede afectar a NT4. No está del todo claro. Se habla de un 85% de probabilidad de éxito de este ataque contra un Windows. Además, puede ser necesario lanzar el ataque repetidas veces o durante un cierto periodo de tiempo.

Desde que se hizo público el ataque, se han sucedido distintos *exploits* que lo implementan. A continuación tenéis algunos de ellos:

kod.c (Exploit original del *advisory*)

kox.c (Permite *spoofing*)

pimp.c

flushot.c

Existen *versiones para Windows* (al menos yo he visto y probado una) del *exploit original*, pero debido a la peligrosidad de un arma así corriendo de mano en mano (de HD en HD), he decidido no pasarlo a nadie bajo ningún concepto, y mucho menos colgarlo de la web. Cualquier petición será rotundamente rechazada, así que por favor, ni os molestéis en intentarlo.

* Una *solución* posible al problema es filtrar los paquetes *IGMP* con la ayuda de algún *firewall*. La alternativa (y recomendable) es aplicar el *parche* de Microsoft correspondiente. Para más información os remito a:

http://www.microsoft.com/security/bulletins/MS99-034faq.asp

Desde allí podréis saltar al boletín correspondiente (*MS99-034*), donde se anuncia la disponibilidad y localización del parche.

[Indice]

3.21.- Paquetes fragmentados ICMP-type 13 en Windows.-

El bug es similar al de paquetes fragmentados IGMP, aunque esta vez se debe a un incorrecto trato por parte de Windows a los *paquetes fragmentados de tipo ICMP-type 13 (timestamp request)*.

La prueba:

 moyari.c (Exploit original *Unix*)
 moyari_patch (Parche para *Linux*)

* Una *solución* posible al problema es filtrar los paquetes *ICMP-type 13* con la ayuda de algún *firewall*. La alternativa (y recomendable, cuando esté disponible) es aplicar el *parche* de Microsoft correspondiente (lo encontraréis en su web), para erradicar completamente el problema.

 [Indice]

4.- *Spoofing.*

Esta técnica no es un ataque en sí pero permite mejorar y perfeccionar cualquier ataque (de los anteriores, por ejemplo). También puede ser la base de algún ataque, como ocurre con el *IP Spoofing* que los hackers suelen emplear (me desviaría del tema de este escrito si siguiese escribiendo...).

Se trata de *"spoofear"* (=falsear) la dirección IP de origen de los paquetes que se mandan a la víctima, de forma que ésta crea que el origen de dichos paquetes es otro (el que nosotros le indiquemos). De esta forma protegemos nuestro anonimato, y en general podemos llevar a cabo cualquier acción que se nos pueda ocurrir y que se derive de una falsa dirección *source* (origen). Por ejemplo, podemos nukear a alguien, con la dirección fuente de otro, haciendo creer a la víctima (si ésta tiene un analizador de paquetes o algo parecido) que es el otro el que le está atacando }:-).

El *spoofing* es (o podría ser) una funcionalidad más del programa de ataque (del nuker, del ping, ...). Como consiste en manejar IP desde un muy bajo nivel en muchos casos se requieren privilegios especiales. Por ejemplo, en el caso de máquinas Unix, se necesita abrir *"raw sockets"* y ésto requiere de privilegios de superusuario (*root*).

Para los interesados en el tema os remito a las siguientes direcciones:

* IP-spoofing Demystified Trust-Relationship Exploitation by daemon9 / route / infinity
http://www.fc.net/phrack/files/p48/p48-14.html

* IPSPOOF.C (Spoofing source)
http://www.ilf.net/Toast/files/unix/ipspoof.c

* This site contains both spoofing code, an analysis of that spoofing code and a good description of how to implement such attacks:
http://main.succeed.net/~coder/spoofit/spoofit.html

 [Indice]

5.- *Técnicas avanzadas.*

Aquí me propongo a describir con la mayor simplicidad posible algunas técnicas más o menos avanzadas (algunas son realmente difíciles de implementar y suelen ser usadas por *hackers*) para que por lo menos sepais de su existencia.

5.1.- Bouncing.-

Consiste en usar una máquina (*host*) intermedia para mantener una comunicación TCP "virtual" entre dos máquinas (situadas una a cada lado). El esquema sería algo así como:

```
Host A   <--->   Bouncer  <---> Host B
          (1)              (2)
```

Todos los datos que A tiene para B son enviados al *bouncer*. Este se encarga de reenviarlos a su destino (B). De igual forma ocurre en el otro sentido. El *bouncer* actúa de elemento redireccionador en ambos sentidos.

La máquina intermedia corre un programa especial (*daemon*) llamado *"bouncer"*. A menudo nos referiremos también con el mismo término a la máquina intermedia (i.e. la que corre el soft especial), de la misma forma que se ha asumido al dibujar el esquema. Dicho programa usa dos puertos TCP previamente definidos, uno por cada conexión (A<->Bouncer, Bouncer<->B), y hace de pasarela entre ellos (tal y como se ha explicado). Por tanto, mantiene por separado dos conexiones TCP diferentes pero las "enlaza" de forma que en su conjunto parezca una única conexión TCP (de ahí lo de "virtual") entre A y B.

La ventaja que ésto conlleva es que ocultamos nuestra identidad a los ojos de B, ya que éste solo ve la comunicación que mantiene con el host intermedio (2) (y no tiene por qué saber que se trata de un host haciendo de *bouncer*).

A esta técnica se le puede dar diferentes usos, todos ellos basándonos en la principal ventaja descrita en el párrafo anterior. En lo que al IRC concierne es ampliamente usada para saltar bans o K/G-Lines. Lo vemos con un ejemplo. Supongamos que nosotros somos el host A y que no podemos entrar a ningún servidor (B) de una red IRC por estar baneados permanentemente. ¿Qué hacer? Pues hacemos uso de una *shell* en otro *host* (que va a ser el intermedio) y colocamos allí un *bouncer*. Al programa, que se quedará corriendo en segundo plano (*background*), le tenemos que dar dos datos: el puerto que estará permanentemente escuchando (por donde recibirá nuestra conexión) y el host y puerto destino (a donde intentará conectarse) correspondientes a B. Ahora, cuando queramos conectarnos al IRC lo hacemos al primero de los puertos predefinidos (correspondiente al host que alberga el bouncer). Automáticamente el bouncer redireccionará nuestra petición hacia B (el servidor IRC) y estaremos dentro (¡saltándonos el ban o lo que sea!) ya que hemos llegado a través de un host que no tenía ningún *ban*. A todos los efectos el servidor IRC cree que nuestro host es el bouncer. Así, al hacernos cualquier otro usuario un /whois ¡verá el *hostname* del bouncer! (ésto es otra ventaja ya que no nos pueden mandar ataques como OOB's o Sspings, por no saber nuestra verdadera dirección IP). Es una buena forma de garantizar nuestro anonimato =).

He aquí un sencillo ejemplo de *IRC bouncer*:

 ts2.c

Hay que tener en cuenta que un *bouncer* es algo fácil de "robar":
1.- Exploramos la red IRC en busca de algún usuario que esté usando *bouncer*. Para ello basta con hacer */whois* a distintos usuarios y fijarnos en el *hostname* que nos sale. Podemos asumir que

presumiblemente se trata de un *bouncer* si es una IP fija (alguna máquina de una universidad, por ejemplo) y al hacer */finger @host* no vemos ningún usuario conectado. Esto último no siempre se puede ver ya que muchos hosts deshabilitan este servicio por el potencial problema de seguridad que puede acarrear.
2.- Hacemos un *PortScan* a dicho *host* (se van mandando sucesivas peticiones de conexión TCP a los distintos puertos de la víctima, recorriendo el rango de puertos deseado, y a la vez escuchando las respuestas que nos llegan; sólo contestarán los puertos "activos" así que es una manera de ver los puertos abiertos que tiene una máquina).
3.- Normalmente un *bouncer* se coloca en los puertos no privilegiados (es decir, a partir del 1024), típicamente en el rango 1024 a 10000, por decir algo, aunque en realidad en principio no hay ninguna restricción en cuanto al número de puerto usado. Así que podemos seleccionar a ojo unos cuantos puertos que son candidatos posibles a bouncer :).
4.- Intentamos conectarnos a dichos puertos con nuestro cliente IRC habitual y vemos si alguno/s lo hace/n.
5.- Si conseguimos entrar al IRC por alguno de esos puertos ya tenemos bouncer =).

Para evitar abusos como el anterior se suele usar algún tipo de autentificación en el *bouncer*, como podría ser poner *password* de entrada. Esta característica y alguna que otra más también interesante la soporta este otro programa de *bouncer*:

bnc.zip

Otro ejemplo de bouncer con password:

vhost.c

[Indice]

5.2.- Wingate.-

Wingate es un programa comúnmente usado para dar conectividad a Internet a varios ordenadores usando un único modem. (supongo que tendrá más funciones pero tampoco interesa para nuestros propósitos). Es un programa bastante peligroso por los problemas de seguridad que implica, al menos en versiones no muy modernas.

Uno de los *bugs* que tiene es que permite acceso vía puerto 23 (telnet) sin pedir password y desde ahí se puede saltar a cualquier otra máquina. Es decir, podemos usar *Wingate* para hacer *bouncing* sin necesidad de preocuparnos de buscar o hackear una shell de por ahí =). ¿Cómo hacerlo?

A continuación teneis un ejemplillo práctico, que funciona desde cualquier cliente IRC que soporte la introducción de comandos *"raw"* (nos permita mandar comandos "directos" al *server*). Esto se hace en mIRC con el mandato *"/quote"* o desde BitchX con *"/raw"*. Las siguientes lineas de código funcionan perfectamente desde mIRC (y son fácilmente portables a otros clientes sin más que cambiar los mandatos "*/quote*" por otros similares, como "*/raw*"):

1.- **/server <ip_wingate> 23**

[Conectamos nuestro cliente IRC al *wingate*. El cliente iniciará el handshake automáticamente, es decir, mandará los comandos *NICK* y *USER* basados en los datos de configuración de dicho cliente IRC. Estos comandos no son "entendidos" por *wingate* -ya que éste espera un *hostname* y no ningún comando- y generan los correspondientes errores. Es interesante ver los comandos que nuestro cliente manda ;-)]

2.- **/quote irc.servidor.org 6667**

[*Wingate* espera un *hostname* a donde saltar, así que se lo mandamos*. Veremos el mensaje de bienvenida del servidor de IRC. Dicho servidor IRC se queda a la espera de "autentificación".]

3.- **/quote NICK RoMaNSoFt**

[Comenzamos la identificación. Le mandamos el *nickname*. En este punto el servidor IRC responde con "PING PONG" ;-)]

4.- **/quote USER RoMaNSoFt RoMaNSoFt host RoMaNSoFt :IRT rlz**

[Terminamos la autentificación mandando el *username* y campo de info sobre el usuario ("*realname*"). Todo lo demás (*host*, *server*), aunque hay que mandarlo, no aporta info relevante al servidor, ya que esta info se usa en conexiones server-server. Un server no debe tragarse lo que el cliente le diga acerca de su *hostname*: sería un spoofing muy fácil, ¿no? :-)

A partir de aquí ya estaríamos dentro del IRC. Por tanto, la cosa ha sido tan fácil como encontrar algún host que corra *Wingate* (que los hay).

También hay otro bug que te permite el acceso al HD del host corriendo *Wingate* aunque no será materia de este artículo al no tener nada que ver con el IRC. Busca en páginas de *hack* :-P.

Solución: aparte de usar siempre la última versión de *Wingate*, aconsejo mejor usar otras alternativas más eficientes, menos peligrosas y más baratas como puede ser un *Linux* corriendo *IP-masquerade*.


[Indice]

5.3.- DNS Inject.-

Con esta técnica conseguimos *spoofear* nuestro *hostname*, es decir, que un servidor nos asocie el *nombre de host* que queramos (y no el que tiene realmente, que es único) (en realidad, puede haber más de uno, pero sólo uno es el real, lo que se conoce como "*canonical name*" mientras que los demás son simples "*aliases*").

La explicación detallada es compleja y excede los objetivos de este documento. A título informativo paso a esbozar o perfilar lo que sería el *Inject*.

Cada servidor (ya sea de IRC, ftp, étc) tiene asociado uno o más *"servidores de nombres"* (*NS*, del inglés: *Name Server*) que resuelven todas las peticiones de conexión que le llegan: para cada IP que peticiona conexión buscan y almacenan el *hostname* correspondiente para su posterior uso. El proceso sería el siguiente:

- un usuario de la máquina A intenta conectarse a un puerto de la máquina B (el servidor IRC, ftp, ...).
- B recibe la dirección IP desde la que peticionamos (la de A) y se la entrega a C (otro host: el NS) para que la resuelva.
- C resuelve y devuelve su respuesta a B.
- B se "cree" la respuesta y la cachea en memoria. Ahora tiene un par IP-*hostname*. A partir de este momento nosotros estamos identificados en el servidor con el *hostname* dado anteriormente por el NS.

Como se puede intuir el truco está en forzar al NS a que de una respuesta falsa (siempre en favor nuestro :-)). Digamos que modificamos los datos del NS para que responda a nuestra IP con el *hostname* que nosotros queremos y no con el que debería hacerlo realmente. Se dice que hemos

"inyectado" (*injectado*, en inglés :-P) el NS.

El cómo hacer esto último es lo complicado y sólo voy a decir que el truco se basa en la *"resolución recursiva"* de los NS (cuando un NS no sabe la respuesta a la petición que le ha llegado lanza una nueva petición a otro NS, que posiblemente sí sepa la respuesta, y así sucesivamente --> de ahí la "recursividad") y en tener finalmente un NS trucado que de respuestas falsas y que nosotros reprogramamos a nuestro gusto.

La principal ventaja es una vez más el anonimato (aparte de poder ir vacilando por el IRC con una identidad del tipo *root@nasa.gov* :-O).

[Indice]

5.4.- Recursos compartidos.-

El protocolo *Netbios* permite compartir recursos (impresoras, discos duros, étc) de un ordenador con otros conectados al primero a través de la red. De esta forma, podemos acceder remotamente a los recursos de una máquina dada: leer ficheros, modificarlos, escribirlos, enviar algún documento a una impresora remota... El abanico de posibilidades es amplio.

Los problemas vienen cuando este servicio está disponible "públicamente" sin contraseña de acceso: nuestro disco duro estará "abierto al público" }:-). Cualquier hacker podrá leer ficheros nuestros, borrarlos, introducir troyanos... en fin, nada bueno.

Un gran fallo de *Windows 95* (corregido en versiones posteriores, creo que a partir del OSR2) es que los recursos no tienen password por defecto con lo cual si activamos *Netbios*, ya sea para utilizarlo o por descuido, y se nos olvida proteger los recursos con password tendremos un gran agujero de seguridad en nuestra máquina. Esto no ocurre así en *Windows 98* ni *Windows NT*.

A pesar de que este problema es ampliamente conocido todavía se ve por el IRC a mucha gente con su disco duro compartido (¡pobres... y ellos sin saberlo!).

Los pasos para ver si una máquina (IP) dada comparte recursos y para utilizarlos son, ***desde Windows NT***, los siguientes:

1.- Asegurarnos de que tenemos el soporte *Netbios* instalado.
2.- Desde el shell (*command.com*, *ndos*, *4dos*, étc) hacer:

c:\> nbtstat -A 195.69.69.69

Notas:

- la IP dada se supone que es la de la víctima
- es importante poner -A y no -a
- no vale un *hostname*, sólo dirección IP.

3.- Para que tengamos posibilidades deberá salir algo como lo siguiente:

```
      NetBIOS Remote Machine Name Table
   Nombre            Tipo        Estado
---------------------------------------------
SKORPIO       <03>  UNIQUE      Registrado
```

ADMINISTRADOR <03> UNIQUE   Registrado
SKORPIO   <00> UNIQUE   Registrado
INSULTOZ   <00> GROUP   Registrado
SKORPIO   <20> UNIQUE   Registrado
Dirección MAC = 00-00-00-00-00-00

Debe haber al menos una entrada de tipo ***<20>*** para que tengamos algo que hacer ;-).

4.- Menú Inicio -> Ejecutar. Introducimos en el cuadro de diálogo:
\\195.69.69.69

5.- Ahora puede que nos aparezca otro cuadro de diálogo solicitándonos contraseña (en cuyo caso en principio no tenemos nada que hacer... aunque podríamos intentar passwords comunes como el nombre o nick de la persona, étc) o bien directamente se abrirá una nueva ventana donde se muestran los recursos compartidos. Pinchando sobre el que queramos tendremos acceso a él.

La *solución* a este grave problema de seguridad es tan simple como proteger los recursos con contraseña o simplemente no activar el servicio *Netbios* si no se va a necesitar. También podemos bloquear los puertos que ofrecen este servicio (en especial el 139) mediante *firewall*.

[Indice]

6.- *Defensa.*

En esta sección me gustaría dar unas pinceladas generales acerca de cómo defendernos de ataques e intrusiones. Las formas de defensa ante determinados ataques en particular no se tratan aquí pues se suponen cubiertas en los distintos apartados correspondientes a los ataques.

6.1.- Firewall.-

Un "*firewall*" o "*muro de fuego*" es un programa (software o hardware) que filtra los paquetes que entran o salen a través de los distintos interfaces de red seleccionados, en base a unas reglas ("*rules*") previamente definidas.

Se tienen 2 tipos de reglas:
- de *entrada*: afectan a los paquetes entrantes, es decir, a los que provienen del exterior.
- de *salida*: afectan a los paquetes salientes, es decir, que van hacia el exterior.

Otra clasificación es:
- *"accept"*: si el paquete cumple los requisitos dados por la regla éste se dejará pasar.
- *"deny" o "ignore"*: si el paquete cumple los requisitos dados por la regla éste no se dejará pasar (se ignorará).
- *"reject"*: si el paquete cumple los requisitos dados por la regla éste no se dejará pasar y además se notificará al remitente del paquete el correspondiente mensaje de error.

Entre otras cosas, lo que caracteriza a un *firewall* es la flexibilidad a la hora de decidir si un paquete debe atravesar o no el *firewall*, o lo que es lo mismo, la capacidad de definir y tratar diferentes reglas para reflejar diferentes situaciones.

Las reglas se pueden basar en cualquiera de los campos que pueda tener un paquete: tipo de paquete, longitud, protocolo, dirección origen, dirección destino, étc. También es posible combinar varios de

ellos. Por ejemplo, una regla podría ser: *"no dejar pasar paquetes IP entrantes provinientes de la subred IP 195.69.69.* y que vayan destinados al puerto 23 TCP"*. En este caso tendríamos una regla de entrada que bloquea el servicio *telnet* (habitual del puerto 23).

Además de las funciones de bloqueo, un *firewall* suele tener funciones añadidas como el *logeo* de los distintos eventos que ocurran (por ej. "se ha tirado [*drop*] el paquete proviniente de tal dirección en base a la regla *x*").

Centrándonos en el tema que nos concierne -el IRC- podemos usarlo por ejemplo para filtrar *client icmp-nukes*. La idea es añadir una regla de entrada diciendo algo así como *"ignorar todos los paquetes ICMP de tipo unprotocol"* (en este caso filtraríamos el nuke normal y no otros tipos).

¿Qué *firewall* es el mejor para nuestros propósitos? Deberíamos buscar algo a la vez flexible y simple (características que suelen ir enfrentadas; habrá que llegar a un compromiso entre las dos), que cumpla con nuestros requisitos de la forma más eficiente, funcione con nuestro modem o tarjeta de red y que sea lo más barato posible. La penúltima característica es muy importante ya que muchos firewalls no tienen soporte para modem y sólo funcionan con tarjeta de red, la cual no es habitual que se tenga si únicamente tenemos un ordenador conectado a Internet por *PPP* mediante algún *ISP*.

Hay diversas ofertas y no voy a tratarlas todas. Tan sólo aconsejaré unos cuantas y daré las razones que me han llevado a ello.

En **Linux** se puede usar el paquete "*ipfwadm*", que viene por defecto en la mayoría de distribuciones. Es un poco tedioso a la hora de configurar pero es totalmente gratuito.

Para **Windows 95** la mayoría de usuarios optan por el "*ConSeal PC fw*", que es fácil de usar y que aunque es comercial se puede descargar de la web en versión demo y aplicarle alguno de los cracks disponibles en Astalavista, para hacerlo totalmente operativo de forma gratuita.

En **Windows NT** he tenido más problemas para buscar *firewall*, aunque al final... :-). El problema es que no se encuentran por ningún lado serials/cracks para los firewalls existentes o en caso de haberlos no soporta el uso de modem. Por nombrar algunos de los mejores y que creo que soportan modem tenemos:

- "Check Point FireWall-1"
- "TIS Gaunlet"
- "NetGuard Guardian"
- "*ConSeal PC fw*"

[Indice]

7.- *Resources.*

Este apartado está dedicado, brevemente, a los distintos programas disponibles y útiles a nuestra causa :-).

Por ahora no voy a subir ningún programa a este servidor ya que puedes encontrar ***TODO* tipo de programas de IRC-War** (*icmp-nukers*, *bouncers*, *flooders*, *link-lookers*, *port-listeners*, *port-bombers*, *port-scanners*, étc) en:
http://www.warforge.com/html/windows/dos/index.shtml.

Podréis encontrar muchísimos *scripts* para *mIRC* y una sección interesante sobre el *bot Eggdrop* en:

http://www.xcalibre.com.

Los últimos *exploits* y programas que explotan los bugs más recientes aparecen en:
http://www.rootshell.com.
http://www.insecure.org.

El mejor buscador de *cracks* y otros *h/p/c/v resources*:
http://astalavista.box.sk.

Como he visto que hay interés por el tema del "*DNS-Inject*" y no estoy por la labor de ampliar dicho apartado (perdonad la flojera: queda prometido para futuras revisiones del doc., *sources* incluidos), os remito a un excelente artículo sobre el tema:
http://hispahack.ccc.de/mi004.htm.

Una herramienta comercial llamada "*Aggressor*", que permite **spoofear** y lanzar multitud de ataques (*land*, *jolt*, étc) **desde Windows** la podeis ver en:
http://www.aggressor.net.

Más información técnica sobre IRC (ataques, defensas y mucho más) -todo en español- y con multitud de links en:
http://www.argo.es/~jcea/irc.

Todo sobre la *red Hispana* de IRC (servidores, info, étc) en:
http://www.irc-hispano.org.

Por último me gustaría recordar la "Home Page" de este documento, donde encontrareis siempre la versión más reciente del mismo (es la que debeis usar en vuestros enlaces). Esta es la ***página oficial de "Tácticas de guerra en el IRC"***:
http://www.rs-labs.com/papers/tacticas/

[Indice]

8.- *Feedback (FAQ).*

En este apartado incluiré las preguntas más frecuentes que me habeis formulado via e-mail algunos de vosotros, para que no se vuelvan a repetir. No pienso contestar mails de forma particular así que no espereis respuesta sino a la próxima versión del documento. Ni qué decir tiene que antes de preguntar leas esta sección; lo mismo alguien tuvo tu duda antes que tú ;-).

1.- ¿Qué script de guerra me aconsejas?

R.- Un script es simplemente algo para simplificar/automatizar tareas. Ningún script es realmente imprescindible tanto para atacar como para defenderse, aunque como digo, en algunos casos sí que puede facilitar la tarea. En este sentido no deberíamos preocuparnos tanto por el script que usamos, sino más bien de tener nuestro sistema correctamente parcheado y protegido y de estar al dia en programas/utils de guerra. Hoy en día (casi) todos los scripts cumplen perfectamente en cuanto a defensa se refiere. Para el ataque un script es tanto mejor cuanto más utilidades (=programas) traiga. Por si os sirve de algo yo uso el *7th Sphere 2.666*, ya bastante anticuado, pero que me sigue valiendo casi como el primer día.

2.- ¿Cómo puedo compilar los programas que se listan en el artículo en mi Windows?

R.- Simple: NO puedes. Estos programas están escritos para entornos UNIX, así que solo podrás compilarlos y correrlos en sistemas operativos de esa familia (como por ejemplo, Linux). No obstante, en algunos casos hay versiones disponibles para otros sistemas operativos. Por ejemplo, cuando el ataque *OOB* se puso de moda surgieron diversas utilidades que lo implementaban desde Win95.

3.- ¿Y dónde puedo encontrar dichas utilidades?

R.- En la sección de "*resources*" hay una URL que es muy buena y donde podrás encontrar de todo. Otra opción es utilizar algún buscador y localizar alguna página de IRC-War. Los programas típicos se encuentran en la mayoría de estas páginas, por muy malas que sean. También puedes buscar webs de H/P/C/V; normalmente todo el material *underground* se encuentra focalizado en diversos puntos de la Red centrados ampliamente en esa temática. Quiero decir que donde hay cosas de Hacking suele haber también cosas de IRC-War, Phreaking, Virus, Cracking... en fin, todos esos temas que tanto nos atraen suelen estas relacionados :-).

4.- ¿Cómo detectar a alguien que me haga un nuke?

R.- Todos los paquetes que llegan a tu ordenador por el interface de red pueden ser *logeados* o "visualizados" con la herramienta correspondiente. Sin embargo, hay que tener en cuenta que estos paquetes pueden venir falseados (*spoofeados*) con lo cual la información que obtengamos no siempre es verídica. De hecho, esto es lo que ocurre con la mayoría de los últimos ataques, en los cuales siempre se usa *spoof*.

No obstante, un *icmp-nuke* normal no suele venir *spoofeado* y se puede detectar fácilmente (obviamente me estoy refiriendo a un *client nuke*; si el paquete no te llegara a tí sino al servidor tú no te enterarías de nada). Cualquier programa analizador de paquetes puede valernos. Valgan como ejemplo:
- las propias funciones de logeo que implementa "de casa" el dialer y pila TCP/IP "*Trumpet Winsock*" (para los que conecten de esta forma, sin usar el dialup de Windows).
- idem para cualquier programa *firewall*, como el "*Conseal*".
- "*tcpdump*" o "*sniff-it*" para UNIX.
- "*ICMP Watch*" para Windows.

5.- ¿Puedo evitar los (client) icmp-nukes?

R.- Sí, solo tienes que filtrar los mensajes *ICMP* que te envíen. Esto lo puedes hacer con cualquier software *firewall*. En Win95 es bastante usado el *Conseal*.

6.- En la red xxxxxx hay un tío que siempre me está molestando y no deja de atacarme. ¿Puedes ayudarme?

R.- Si he escrito este documento es para ayudar a la gente y sobre todo, para que lo lea. :-P Así que usa la información que aquí se encuentra e intenta sacarle provecho por tí mismo en vez de pedir ayuda a terceros. No tengo tiempo para centrarme en casos particulares, a lo mejor ni siquiera conozco la red xxxxxx. Si tienes alguna duda contacta conmigo y si lo considero conveniente aparecerá solucionada en esta misma sección en la siguiente revisión del documento.

7.- ¿Cómo puedo garantizar mi anonimato en IRC? Actualmente he recibido e-mails con amenazas e insultos de gente de los canales que frecuento, ¿cómo puedo evitarlo?

R.- Una de las ventajas del IRC es precisamente que guarda el anonimato. Todos los datos que el servidor IRC tiene y puede ofrecer sobre un usuario dado son los que previamente uno ha

introducido en su cliente IRC, a excepción de la dirección IP desde donde nos conectemos. Por tanto, podemos introducir datos falsos: nombre falso, dirección de correo falsa, étc. Lo único que en principio no se puede esconder es tu dirección IP, que la toma el servidor en el momento de hacer tu conexión TCP al servidor de IRC. Sin embargo, hasta esta última se puede esconder usando algunas de las técnicas descritas en este documento, como el *bouncing* o el *DNS Inject*, lo que nos haría completamente anónimos.

8.- El otro día entré en un canal y me dijeron que tenía el puerto 139 abierto... que podían resetearme... ¿Cómo lo puedo cerrar?

R.- Puedes tener el puerto 139 abierto sin que "te puedan resetear": es cuestión de tener el parche contra *OOB* instalado en tu sistema. De hecho, es un parche obligado. Además no solo es vulnerable el puerto 139. No obstante, si lo quieres cerrar, solo tienes que desinstalar el soporte *Netbios* (en la configuración de red). Sólo necesitas tener soporte TCP/IP para conectarte a Internet; todo lo demás sobra. Cuantos más servicios ofrezca una máquina más insegura y propensa a ser hackeada será. Hay que procurar abrir únicamente los servicios que realmente nos hagan falta.

9.- A un amigo le entraron en su Windows y le empezaron a borrar carpetas. ¿Cómo lo hicieron?

R.- Tendría archivos o unidades compartidas sin password. Windows 95, por defecto, no pone password a los recursos compartidos, lo que es un grave problema de seguridad. Si usas recursos compartidos, ¡no olvides poner un password! Y si no los usas pues ¡no los tengas activados! Para esto último basta con desactivar *Netbios* (una razón más para tener cerrado el puerto 139). De lo contrario, cualquiera podrá entrar en tu HD: leer lo que quiera, borrar ficheros, modificarlos... ¡Muy peligroso!

10.- ¿Se puede mandar ataques como land o jolt, o spoofear desde Windows?

R.- La pila TCP/IP que trae Windows tiene ciertas limitaciones que impiden, por ejemplo, el spoof. Sin embargo, hay un programa en fase de pruebas que puede spoofear y mandar muchos de estos ataques desde Windows. Lo hace implementando su propia pila TCP/IP y no usando las funciones de Winsock normales. Por ahora solo he visto una versión beta que únicamente funciona en Win95. Existe también una versión comercial o al menos estaba en desarrollo. El programa se llama "*Aggressor*" y teneis su *URL* en la sección de *Resources*.

11.- Me gustaría saber más sobre los K/G-Lines: en qué consisten, lo que tienes que hacer para que no te hagan uno...

R.- Simplemente es como un *ban* pero aplicado a hosts o dominios enteros. Es decir, si tu dirección IP o el dominio al que pertenece tienen alguna de estas *lines* no podrás conectarte a la red IRC, no te dejará. La diferencia entre un *K-Line* y un *G-Line* es que ésta última es global, o sea, que se aplica a todos los servidores de la red IRC y por tanto no habrá forma, en principio, de entrar a esa red IRC. Un *K-Line* se aplica a un servidor en particular de la red; es decir, que se nos deniega el acceso a la red IRC a través de ese servidor, lo que no implica que no se pueda entrar por cualquier otro. También es posible que una misma *K-Line* esté activa en distintos servidores a la vez (o incluso en todos, lo cual sería equivalente a una *G-Line*).

Estas *lines* se suelen poner como respuesta a abusos por parte de uno mismo o de algún otro usuario perteneciente a tu dominio y la duración de las mismas depende de la política que sigan los administradores de la red de IRC. Para evitarlo, lo único que hay que hacer es portarse bien o al menos ser suficiente bueno y que no te pillen mientras haces cosas malas };-).

12.- ¿Qué es una I-Line?

R.- Es una entrada en el fichero de configuración del *ircd* que permite variar el número de clones admisibles desde una determinada dirección IP o dominio. La añade un administrador del servidor IRC para por ejemplo permitir la entrada al IRC desde cybercafés (donde normalmente todos los ordenadores se ven como una misma dirección IP) o máquinas multiusuario (universidades, étc).

13.- ¿Qué es un script Perl?

R- *Perl* es un lenguaje de programación interpretado. Un *script* (a secas) es algo que automatiza tareas, una especie de programa (así tenemos scripts de IRC, de shell [Unix], étc). Un *script Perl* es un script de Unix que en vez de usar una shell para su ejecución usa el propio programa de Perl. En definitiva, se trata de un programa escrito para Perl.

14.- Do you have an English translation of this page?

A.- Not at this moment, I hope it will be available soon though. I'm looking for somebody to translate this doc since I'm very busy for doing it by myself.

15.- Tengo información sobre ataques / bugs / fixes que no se listan en este documento. ¿Puedo ayudar?

R.- Por supuesto. Este documento jamás estará 100% completo porque ni soy perfecto ni tengo todo el tiempo del mundo. Pero intentaré actualizarlo con la mayor frecuencia posible. Toda la ayuda y *support* que pueda recibir de parte vuestra serán bien acogidos. Puedes enviar tus comentarios a roman@deathsdoor.com.

16.- Me gustaría saber más acerca del "DNS inject". ¿Alguna idea?

R.- Sí, claro. Mira la *URL* dada en la sección "*Resources*" para una completa información acerca de su funcionamiento y posible puesta en marcha (desde un punto de vista práctico). De todas formas, creo oportuno decir que es una técnica bastante costosa de implementar y bastante desagradecida, ya que la mayoría de servidores de nombres (*DNS*) están ya parcheados (en particular, los relacionados con redes importantes de IRC).

17.- Acabo de instalar Windows 98. ¿También necesita parches, winsocks, étc? De momento le he renombrado el "vnbt.386" a bak, porque he visto que ese archivo sigue estando en W98.

R.- Mal hecho. Veamos. Un *parche* (*"patch"*, en inglés) -como regla general- tiene un uso muy específico: va destinado a un *S.O.* o programa en particular, a una versión concreta del mismo y normalmente *fixea* (arregla) problemas también específicos. Así pues no puedes aplicar un parche sacado para Win95 a un Win98. Los resultados serían impredecibles; es más, casi con toda seguridad estropearías o empeorarías tu *S.O.* o al menos la parte afectada por el parche.

Un *parche* suele ser una especie de "chapuza" rápida para solventar algún problema encontrado. Cuando se descubre un *bug*, los autores del programa o *S.O.* son informados del mismo, estudian el problema y las formas de solucionarlo. Unas veces el *bug* es algo trivial y se arregla sin problemas: se saca un *parche* que será más o menos definitivo. Otras veces, cuando el *bug* es más "raro" o es menos común, la solución que sale en forma de *parche* no es algo definitivo sino temporal. Tras un periodo de pruebas ("*testing*") es fácil que saquen el mismo parche pero más actualizado, que viene a ser como una versión mejorada del parche original, al cual sustituye. También es posible que el nuevo parche se haya de aplicar sobre el parche anterior. Es importante leerse las "instrucciones" que

acompañan a cada parche.

Cuando un parche está lo suficientemente probado y se le da el "visto bueno" pasa a formar parte del *S.O.* o programa, es decir, ya las nuevas versiones del mismo estarán *fixeadas* del correspondiente *bug* y el sistema habrá dejado de ser vulnerable al problema en cuestión (o al menos debería).

Según lo anterior, *Win98* incorpora todos los parches importantes de *Win95*, claro está, anteriores a la fecha de lanzamiento de *Win98*. Si se descubre un nuevo *bug* que afecte a ambos *S.O.*'s, Microsoft debería de sacar dos parches: uno para cada *S.O.*, y no serían intercambiables entre sí.

Lo mismo ocurre con la pila *TCP/IP* (*Winsock*): *Win98* trae una versión más actualizada que *Win95*, como es lógico, por lo cual no vamos a cambiarla por una más antigua, ¿no? ;-)

Lo que todos estáis deseando escuchar: *Win98* *no* es vulnerable a ataques tan molestos como *ssping*, *land*, *teardrop*, étc. Desde el punto de vista de la seguridad es mucho mejor que *Win95*; quizás de las pocas cosas buenas que tiene. :-) Pero eso no quita que puedan salir nuevos *bugs*, así que habrá que estar informado para estar siempre al día con los últimos parches.

18.- ¿Cómo se configura el "Conseal PC Firewall"? ¿Podrías pasarme tu fichero de reglas?

R.- Para saber configurar un *firewall* hay que tener unos conocimientos mínimos sobre redes. Empieza por ahí. Mucha gente se instala el *Conseal* solo para evitar *client-nukes*, sin saber ni lo que es un *firewall* ni cómo funciona. Poco recomendable.

Mi fichero de reglas ("*rules*") es eso, mío. No lo paso porque no considero que esté optimizado y que deba ser ejemplo a seguir. No quiero confundir a nadie con mis posibles errores. Por dar algunas pistas, yo lo que hice fue partir del fichero de reglas ("*default.fwr*"), que ya incluía algunas protecciones como la de los nukes (al menos, a partir de la versión 1.2, que es la que yo ví). A partir de ahí ya le he ido añadiendo nuevas reglas o modificando otras. En particular, yo he metido las siguientes reglas para el tema de los nukes:
- denegar todos los *icmp*'s entrantes (cualquier tipo y de cualquier *host*) como norma general (ésta es la que venía ya por defecto, y es la regla más segura).
- permitir todos los tipos de *icmp*'s entrantes *exclusivamente* a los servers de más uso, como por ejemplo el de correo, news, étc. Si deniegas estos *icmp*'s las operaciones como mandar o leer correo, news, étc. pueden verse afectadas por la regla anterior y no funcionar correctamente. Además te ahorras unos cuantos *warnings* y *beep*'s debidos a los abundantes *icmp*'s generados por estos *hosts* "de uso frecuente".
- permitir todos los *icmp*'s salientes (cualquier tipo y hacia cualquier *host*). Esta regla , entre otras cosas, te permite mandar *icmp-nukes* (las reglas originales no lo permitían).

19.- ¿Por qué no compila bien el source del OOB Nuke ("WinBomber 0.9") en algunos Linux'es?

R.- Este fuente es bastante viejo y está escrito para *Libc5*. Compila perfectamente en un Red Hat 4.2, por ejemplo. Sin embargo, los Linux de hoy en día ya no usan esta librería e incluso han cambiado *includes* y demás. La línea que dará error en un Linux moderno es:

```
#include <netinet/protocols.h>
```

ya que ahora no existe tal fichero. Pero esta linea ***no* es necesaria** y la podemos omitir (porque

esta info se encuentra en otro archivo). Una vez borrada la linea, el *source* compilará sin problemas con el comando:

```
$ gcc -D_BSD_SOURCE -o winbomber winbomber.c
```

Nótese que el modificador "*-D_BSD_SOURCE*" es ***necesario*** para compilar todos los programas que contienen código para manejar *raw sockets* (necesario para el *spoof*) como el que viene en *Winbomber.c*.

20.- ¿Cómo puedo saber la dirección IP y el Hostname de un usuario cuyo nick conozco?

R.- Para conseguir respuesta a preguntas tan triviales como ésta, lo mejor es que busqueis por la Red alguna guía o FAQ sobre mIRC. Sin que sirva de precedente, voy a contestar, dado el gran volumen de gente que me ha formulado la misma pregunta.

* Hostname: basta con hacer "***/whois nick***". Ejemplo: /whois RoMaNSoFt

```
RoMaNSoFt is roman@bart.us.es * {##-InSuLToZ ReSe@rCh / TiGeR Te@M-##}
RoMaNSoFt using irc.llfb.org Servidor de pruebas...
RoMaNSoFt has been idle 10 seconds, signed on Thu May 13 16:11:42
End of /WHOIS list.
```

* Dirección IP: "***/dns nick***" (desde mIRC). Ejemplo: /dns RoMaNSoFt

```
*** Looking up bart.us.es
-
*** Resolved bart.us.es to 193.147.160.132
```

Por razones en las que no entraré, hay veces en las que al hacer */whois* se nos muestra directamente la dirección IP en lugar del *hostname*. Esto ocurre cuando el servidor IRC no ha podido determinar el *hostname*, a partir de la IP (cuando un servidor IRC recibe una conexión exterior lo único que ve, en principio, es una dirección IP), o dicho con otras palabras: "no ha podido resolver la IP" (normalmente será debido a que el dominio del cliente no dispone de "*resolución inversa*").

21.- No puedo obtener la IP con ninguno de los métodos anteriores. Me aparece algo del estilo "200.45.34.xxx". ¿Qué está ocurriendo?

R.- Hay servidores de IRC que protegen la dirección IP de sus usuarios, de forma que la muestran parcialmente escondida. Para verla completamente se requiere de privilegios de operador. Tenéis un ejemplo en *irc.ciudad.com.ar 6667*. Para averiguar la IP en este caso (sin ser operador) puedes intentar provocar que la víctima te haga *dcc* (te mande algún fichero, por ejemplo), con lo cual tendrás su IP en los paquetes que te envíe.

22.- No me he enterado bien de "cómo obtener op sin que nadie te lo de". ¿Puedes explicarlo mejor? ¿Qué es un link-looker? ¿Dónde conseguir uno?

R.- Es increible la de gente que me ha formulado esta misma pregunta. Me parece innecesaria, ya que todo esto viene explicado en el doc. Más bien, se debe a gente que no se ha molestado en leer tranquilamente y asimilar el documento (o al menos la parte donde describe ciertos fundamentos

necesarios para la comprensión del mismo) y ha saltado directamente a esta sección, sin tener los conocimientos previos requeridos. También hay gente que ni siquiera ha leido la FAQ y pregunta cosas que ¡vienen en ella! Señores: ¡la FAQ está para algo! No esperéis encima respuesta a vuestros mails en este caso ni os enfadéis porque no conteste.

No obstante, he revisado la sección, a ver si es que estaba mal explicada. No lo veo así, pero de todas formas voy a intentar explicarlo con otras palabras para ver si despejo alguna incógnita. El texto original de la sección he decidido dejarlo íntegro en su sitio y aquí (y sólo aquí) añadiré los nuevos comentarios al tema. Vamos a ello.

Lo primero, decir que esta técnica está obsoleta, y no creo que a nadie le funcione hoy en día. Así que todos aquellos que hubieran visto el cielo abierto, que se vayan bajando de la parra. La técnica se basa en el hecho de que dos servidores spliteados (separados temporalmente, se ha roto el enlace que los une) son -a nuestros efectos- dos servidores aislados e independientes. ¿Qué quiere decir ésto? Pues que uno puede entrar tanto en uno como en otro servidor y crear canales, cambiarse de nick, étc, y ninguno de estos cambios se verán reflejados en el otro servidor. Según esto, podrá haber dos usuarios con el mismo nick (uno en cada servidor), dos canales con el mismo nombre (uno en cada servidor) y distintas personas en él, étc.

La clave es: ¿qué ocurre si los dos servidores (que estaban separados) se "juntan" de nuevo (es decir, se reestablece el enlace caído)? Ahora los dos servidores forman un todo (pertenecen a una misma red de IRC, han dejado de estar aislados) y no pueden existir dos usuarios distintos con el mismo nick, ni haber canales duplicados (esto es de cajón). ¿Cómo se resuelve ésto? En el caso de usuarios duplicados, la red de IRC (y en particular, uno de los servidores de dicha red) expulsa a uno de los usuarios cuyo nick está duplicado. Esto sería un "*nick collide*" y está explicado en la sección correspondiente.

Para el caso de canales duplicados, la red IRC asume que se trata del mismo (y único) canal, y en consecuencia, los usuarios de ambos canales duplicados se suman a un mismo canal que mantendrá el mismo nombre. Ahora bien, cada uno de los canales duplicados, antes de juntarse, disponía de sus propios operadores. ¿Quién mantiene el op ahora? La respuesta resumida es que el servidor intentará ser lo suficientemente inteligente como para discernir entre los ops "legítimos" y los "no legítimos", y conservar el status de operador sólo en el primero de los casos.

¿Qué se entiende por op "legítimo"? Dos servidores pertenecientes a una misma red de IRC no pueden tener un canal duplicado (éste es un caso excepcional y no es más que una patología debida al propio funcionamiento interno de la red). Si este caso se dá, es porque antes de producirse el *split* o separación de los dos servidores, ya existía un canal con ese nombre, y es este canal el que se subdivide en dos canales ahora, al producirse el *split*. Los operadores "legítimos" son aquellos que tenían el op antes de producirse el split. Si un usuario consigue su op durante el split, se considera como "no legítimo", porque esta asignación de privilegios no se ha visto en toda la red, sino únicamente en el servidor spliteado, que es donde se ha logrado el op, y podría haberse obtenido de forma ilegítima, "sin el consentimiento de toda la red". ¿Se entiende ahora?

Pues bien, imaginemos que la red no es muy lista que digamos y no distingue bien entre estos ops legítimos e ilegítimos. En este caso, al deshacerse el split (al juntarse los dos servidores separados) los ops ilegítimos se mantendrán en el canal resultante y pasarán a ser legítimos. En este caso, está claro que si conseguimos ser op ilegítimo, podremos elevar nuestros privilegios a op legítimo. ¿Cómo se hace ésto? ¿Cómo me hago op ilegítimo sin la ayuda de otro op que me lo dé? Fácil, el único requisito es que en alguno de los servidores separados no exista el canal en cuestión (esto ocurrirá si al producirse el *split*, ninguno de los usuarios del canal habían entrado a la red por dicho servidor). En este caso, sólo tenemos que crear el canal nosotros (en dicho servidor spliteado) y al ser nosotros los creadores del canal, tendremos nuestro op correspondiente (seremos el único usuario del canal, en ese servidor). Ya hemos conseguido ser op de forma ilegítima, y con suerte, al

deshacerse el split, lo mantendremos como legítimo.

Espero que ahora haya quedado suficientemente claro; me he esmerado todo lo que he podido.

El *link-looker* no es más que una herramienta que nos avisa cuando se produce algún *split*, y nos lista los servidores spliteados. Esta información la podemos aplicar directamente a la técnica vista, para intentar conseguir op sin que nadie te lo dé, intentando crear el canal en el servidor spliteado, tal y como se ha explicado más arriba.

La herramienta anterior suelen incorporarla la mayoría de los scripts, aunque de todas formas puedes encontrar diversos programas que la implementan, sin ir más lejos, en la página de WarForge, cuyo link aparece desde hace tiempo en la sección de "*Resources*" de este documento (y que -déjame adivinar- no has debido de mirar).

Tengan en cuenta mis queridos lectores que no pienso volver a explicar cosas que -considere- estén bien explicadas en alguna parte del documento. Ya que yo me he molestado en escribirlo *íntegro*, tómese la molestia de leerlo *íntegro* (que seguro que es mucho menor que la primera) o al menos no proteste si no entiende algo, por haberse saltado alguna sección que debería haber leído. Se puede decir más alto, pero no más claro.

23.- Tengo intención de instalarme Win NT-4 Workstation. Como quiera que me gusta chatear por IRC, la pregunta es: ¿necesita tantos parches como W95 para estar mínimamente protegido? Si así fuera, ¿donde puedo conseguir información y/o bajarme dichos parches?

R.- En general, NT es mucho más robusto y estable que W95, sobre todo en el tema de redes y conectividad a las mismas. Sin embargo, ambos están hechos "de la misma madera", o sea, provienen de Microsoft, y comparten similitudes de diseño. Dicho esto, no es de extrañar que un fallo en la pila TCP/IP de W95 afecte también a NT, y viceversa. Y de hecho, así ocurre normalmente. La similitud no siempre es perfecta. Como ejemplo, valga el ataque de fragmentación IGMP: Microsoft ha admitido que afecta tanto a los W9x como a NT; sin embargo, en NT es mucho más difícil de explotar y los programas más extendidos, que funcionan contra W9x, no lo hacen en NT.

Respecto a los parches, NT usa los llamados *"Service Packs" (SP)*, que no son otra cosa que un conjunto de parches, todos ellos agrupados en un mismo fichero. Periódicamente, Microsoft saca un SP nuevo. Los SP's son acumulativos, es decir, no hace falta tener instalado SP1 para poder instalar el SP2 o el SP3. El último SP contendrá siempre **todos** los parches aparecidos hasta la fecha. Entre el *release* de cada SP, Microsoft va sacando los llamados *"hot-fixes"*, que son parches sueltos para dar solución inmediata a problemas recientemente descubiertos, y que se aplican sobre un determinado SP. El siguiente SP que salga contendrá ya estos hotfixes.

Un gran inconveniente, si pretendes usar NT, es que los parches para la versión española tardan mucho en salir: una vez descubierto un bug, y publicado el parche para la versión USA, el correspondiente parche español puede tardar ¡meses! (periodo durante el cual continúas siendo vulnerable al bug descubierto).

24.- Necesito montar un servidor IRC para hacer mis propias pruebas y no se por donde empezar. ¿Alguna idea?.

R.- Si buscas lo fácil, o sea, todo en Windows, simplemente vete a Tucows

(http://www.tucows.com), y busca algún programa servidor de irc como el "Conference Room". Es facilísimo de configurar. Lo dejas corriendo en uno de los windows, en el puerto especificado y listo: ya puedes conectar desde cualquier otro ordenador que tenga conectividad ip con el mismo, sin más que ejecutar mIRC y conectar con el servidor y puerto especificado (aconsejable 6667).

Si lo tienes que hacer en Unix, buscate el código (source) de algún ircd, p.ej., en la web de alguna de las redes más famosas de IRC (http://www.undernet.org, de memoria). No es difícil pero tendrás más tarea de configuración que con la opción anterior (Conf. Room en Win).


[Indice]

9.- *Agradecimientos.*

Quisiera dar las gracias a *jcea@argo.es* por echarle un vistazo a la primera versión de este texto y por sus comentarios desinteresados sobre él. Así mismo, a todos los que me habeis apoyado con vuestros comentarios u opiniones acerca de este escrito. ¡No dejéis de seguir haciéndolo!

También me gustaría saludar desde aquí a todos mis amigos y conocidos del IRC, en especial a los canales *#insultos*, *#hack*, *#hackers, #hack_novatos, #amiga y #llfb* de la red *irc-hispano*.


[Indice]

10.- *Modificaciones.*

* **Cambios** en este documento en la versión 1.0:

- Añadida esta sección. :-)
- Mejor explicado lo del *"spanning tree"* (estructura de una red IRC).
- Añadido el concepto de *"hop"*.
- Añadidas un par de *URL's* en la sección de "*resources*".
- Añadida sección "*técnicas avanzadas*" incluyendo *"bouncing"* y *"DNS Inject"*.
- Añadidos diversos *fuentes* en C: *ICMP nuke*, *OOB*, *ssping*, *bouncer*, ...
- Añadidas direcciones donde encontrar numerosos *parches*.
- Añadidas *URL's* de referencia en la sección de "*spoof*".
- Añadidos nuevos ataques: *teardrop*, *land*, *teardrop modificado* y *smurf*.

* **Cambios** en este documento en la versión 2.0:

- Variación de las condiciones de distribución de este documento.
- Sustituido código fuente de los diferentes programas por enlaces a los mismos (se han puesto en ficheros independientes).
- Añadidos los ataques: *bonk*, *syndrop*, *overdrop, nestea y hanson*.
- Añadido cómo "robar" un bouncer así como nuevo código de bouncer con soporte de password.
- Añadida sección "*feedback (FAQ)*", con las respuestas a las preguntas más frecuentes formuladas al autor acerca de este documento.
- Añadida sección "*defensa*", con algunos consejos generales para defendernos de distintos ataques.
- Añadido apartado "*Wingate*".
- Añadido concepto de "*i-line*" en la sección introductoria.
- Añadida sección "*trojans*".

- Añadido apartado "*recursos compartidos*".
- Revisadas las distintas secciones y ampliadas algunas con más info o algún *link*.

* **Cambios** en este documento en la versión 3.0:

- Añadido apartado "*Ping of death*".
- Añadida sección "*Modem DoS*", con los apartados "*/Msg ComX*" y "*+++ATH0*".
- Añadido source "*script.ini*".
- Añadido apartado "*Back Orifice*" en la sección de troyanos.
- Añadido apartado "*NetBus*" en la sección de troyanos.
- Añadido *RFC1459*.
- Añadidos pequeños cambios en la "*FAQ*" y en "*Resources*".

* **Cambios** en este documento en la versión 3.1:

- *Minor fixes*.
- Añadidos enlaces a los distintos apartados del texto, desde el índice.
- Añadidos dos sites con listas de redes "*amplificadoras de smurf*".
- Actualizada la *FAQ*.

* **Cambios** en este documento en la versión 3.2:

- Añadido ejemplo práctico de *bouncing*, en la sección de *Wingate* (5.2).
- Reformada completamente la sección 3.14.2 sobre el troyano *"dmsetup.exe"*.
- Actualizada la *FAQ*.

* **Cambios** en este documento en la versión 4.0:

- Añadidos nuevos ataques:
  + "*Fragmento de longitud 0 - Linux DoS*".
  + "*Paquetes ICMP aleatorios - Linux DoS*".
  + "*ICMP/Redirect-host DoS*".
  + "*Paquetes fragmentados IGMP en Windows*".
- Actualizado el enlace a *WarForge*.
- Añadidos enlaces al índice desde los distintos apartados del texto, para que sea más fácil moverse por él.
- Actualizada la *FAQ*, incluyendo una nueva explicación del tema de "obtener op sin que nadie te lo dé".

* **Cambios** en este documento en la versión 4.1:

- Añadidos nuevos ataques:
  + "*Paquetes fragmentados ICMP-type 13 en Windows*".
- Añadido link hacia el *parche* de Microsoft para *Kod* (*paquetes IGMP fragmentados*).
- Añadido un nuevo fuente: *gin.c* (sección *3.16.2*: *+++ATH0*).
- Borrado el crack para Conseal 1.0, para evitar posibles problemas legales (en su lugar he puesto la dirección de Astalavista, para que busquéis el crack vosotros mismos).
- Actualizada la *FAQ*.

 [Indice]

11.- *Notas finales.*

Espero que este artículo os haya servido al menos para comprender un poco más el funcionamiento del IRC.

También me gustaría recordar que el IRC no es un campo de batalla, y que no se debe atacar a la gente así porque así, salvo "en defensa propia" y pocos casos más justificados. O:-) En cualquier caso un ataque **nunca** está justificado desde el punto de vista de los *ircops*, así que cuidado con las *K/G-lines* ;-)

Se agradecería toda clase de comentarios y sugerencias para mejorar el documento: *bug reports*, asuntos que creáis no están bien explicados, temas que faltan y se deberían incluir, ..., y en definitiva todo lo que os gustaría que se incluyese en el documento. También se aceptan *O-lines* ;-).

Se rechazará estrictamente todo aquello que salga de la temática así como dudas personales. Si lo creo conveniente y las dudas son generalizadas las intentaré aclarar en posteriores revisiones de este documento (ver sección "*Feedback*") pero (y repito) no a nadie en particular. Teneis que comprender que no tengo todo el tiempo del mundo.

Podeis contactar con el autor de este escrito por *email*: roman@rs-labs.com. O localizarme en el IRC hispano bajo el nick de *RoMaNSoFt*. Podeis consultar la web de la red Hispana de IRC para averiguar servidores por donde entrar (ver sección "*Resources*").

c'U l8er!!!!!

[Indice]

##########