CURSO DE TCP/IP: ICMP (PROTOCOLO DE MENSAJES DE CONTROL DE INTERNET).

- ICMP es, posiblemente, el protocola más utilizado por los hackers
- ICPM nos dará información MUY VALIOSA sobre el funcionamiento de las conexiones.
- Veremos que este protocolo nos permite detectar gusanos y ataques, tracear conexiones, destruir firewalls, degradar servicios, descubrir datos sobre nuestras "victimas"... ...

1. Introducción

Imagino que muchos de los lectores que han seguido fielmente el curso de TCP/IP desde sus comienzos estaríais esperando que este mes hablase del protocolo IP, ya que una vez explicados los protocolos de transporte clásicos (TCP y UDP) el orden lógico sería llegar a la capa inferior, o la capa de red.

Pero, para vuestra sorpresa, aún no vamos a llegar a esa parte tan esperada, pero también tan compleja; si no que vamos a seguir con el orden lógico, con un protocolo que, aunque no sea un protocolo de transporte, si que funciona por encima del protocolo IP.

La diferencia entre ICMP y otros protocolos que funcionen por encima de IP (como TCP, UDP, u otros que no hemos mencionado) es que ICMP se considera casi una parte de IP, y las especificaciones estándar de IP obligan a que cualquier sistema que implemente el protocolo IP tenga también soporte para ICMP.

¿Y por qué es tan importante ICMP para IP? Pues porque IP no es un protocolo 100% fiable (como nada en esta vida), e ICMP es precisamente el que se encarga de manejar los errores ocasionales que se puedan dar en IP.

Por tanto, ICMP es un protocolo de Control (**I**nternet **C**ontrol **M**essage **P**rotocol), que sirve para avisar de los errores en el procesamiento de los **datagramas**, es decir, de los paquetes IP.

Como ICMP funciona sobre IP, los paquetes ICMP serán siempre a su vez paquetes IP. Esto podría dar lugar a situaciones en las que se generasen bucles infinitos donde un paquete ICMP avisa de errores de otro paquete ICMP. Para evitar esto, hay una regla de oro, y es que, aunque los paquetes ICMP sirvan para arreglar errores de otros paquetes, jamás se enviarán paquetes ICMP para avisar de errores en otro paquete ICMP.

Aunque esto suene a trabalenguas, ya iremos viendo en detalle el funcionamiento del protocolo ICMP, y se aclararán muchas cosas.

Muchos de vosotros conoceréis ya algo sobre ICMP, pero quizá alguno se esté preguntando: "¿Pero realmente es tan importante este protocolo? ¡Si jamás he oído hablar de él! ¡Seguro que no se usa para nada!".

Pues por supuesto que se usa, y mucho, aunque es normal que no os suene, teniendo en cuenta que es un protocolo que no suele usar la gente en su casita, si no que es usado sobre todo por los routers.

Aún así, aunque los paquetes ICMP sean generados normalmente por un router, el receptor del paquete suele ser un usuario como tú, así que en realidad sí que estás usando el protocolo ICMP aunque no te des cuenta.

Por ejemplo, cuando intentas acceder a una IP que no existe, normalmente se te notificará por medio de un mensaje ICMP. Además, hay también una serie de aplicaciones, como Ping, o Traceroute, que utilizan ICMP, y en este caso sí que eres tú el que genera los paquetes.

Aparte de todo esto, para nosotros ICMP es especialmente interesante ya que, tal y como podéis leer en más de un documento de los que circulan en la red sobre ICMP, se dice de este protocolo que es mucho más útil para un hacker que para un usuario normal.

Para este artículo iré explicando uno a uno los diferentes mensajes ICMP, y contando en cada uno alguna curiosidad un poco más oscura para darle más gracia a la cosa (buscad los epígrafes en color rojo). Por último, como era de esperar, terminaré explicando cómo generar paquetes ICMP desde cero con las aplicaciones ya conocidas.

2. Directamente al grano

Igual que hacen en el RFC que explica el protocolo ICMP (**RFC 792** : ftp://ftp.rfc-editor.org/in-notes/rfc792.txt), no nos enrollaremos mucho con explicaciones previas, e iremos directamente al grano.

Como ya debes...

Como ya debes suponer, los RFCs son documentos escritos originalmente en inglés. Para quien no domine el idioma, contamos con la inestimable ayuda de la Web www.rfc-es.org

Aquí podrás encontrar la traducción al español del RFC 792 (Protocolo ICMP), concretamente en el enlace http://www.rfc-es.org/getfile.php?rfc=0792

No dudes en pasarte por www.rfc-es.org y, si te es posible, participar en su proyecto. Desde esta revista damos las gracias a quienes colaboran de forma totalmente desinteresada en este tipo de iniciativas que nos benefician a todos.

El protocolo ICMP consta simplemente de una serie de mensajes totalmente independientes que se generan para cada situación de error que se de en el procesamiento de un paquete IP, o bien por otras circunstancias especiales en las que un usuario desea conocer cierta información sobre una máquina.

Por eso, no hay que andar con complejas explicaciones sobre conexiones virtuales, ni nada de nada. Simplemente basta con ir explicando cada mensaje por separado.

2.1. Destino inalcanzable (Destination Unreachable)

Este es el primer mensaje ICMP, que sirve para avisar de un gran número de situaciones de error, aunque todas con una misma consecuencia: no se ha podido acceder al destino que se solicitaba.

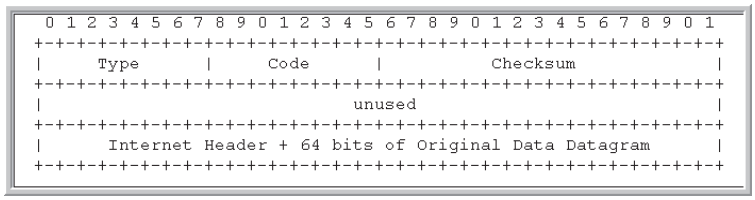
Cuando intentas acceder a una IP (una página Web, u otro servidor de cualquier tipo), puede haber muchos motivos por los que no puedas llegar a acceder y, siempre que estos motivos tengan que ver directamente con la parte de la que se encarga el protocolo de red (IP), cualquier dificultad que haya te será notificada mediante un mensaje ICMP de este tipo.

Por supuesto, puede haber otros motivos por los que no puedas acceder a los contenidos que buscas. Por ejemplo, al intentar acceder a una página Web, el Servidor Web podría responderte con un **error 404** (te recuerdo que expliqué todo esto en mi artículo sobre **HTTP** en la serie **RAW**, hace ya unos cuantos números...).

Esta situación de error no es responsabilidad de IP y, por tanto, tampoco de ICMP, por lo que este error no se notificaría mediante un mensaje ICMP, si no que sería responsabilidad del Servidor Web el generar una respuesta de error 404 de HTTP.

Un error que sí que sería responsabilidad de IP al intentar acceder a una página Web sería que la IP del Servidor Web que buscamos no existiese.

Hay muchos otros motivos por los que un destino puede ser inalcanzable, como veremos en detalle ahora mismo. De momento, vamos a ver el formato detallado del mensaje:



Campo: Type

El primer campo, Type, se encuentra en todos los mensajes ICMP, y especifica el tipo de mensaje. En el caso del mensaje **Destination Unreachable** el tipo de mensaje es **3**.

Campo: Code

El campo Code permite precisar el motivo por el que el destino es inalcanzable. Según el valor de este campo, podemos saber el motivo exacto, consultando la siguiente tabla:

Code	Descripción
0	Red inalcanzable.
1	Host inalcanzable.
2	Protocolo inalcanzable.
3	Puerto inalcanzable.
4	Paquete demasiado grande, y no puede ser fragmentado.
5	Error del router de origen.
6	Error desconocido en la red de destino.
7	Error desconocido en el host de destino.
8	Host de origen aislado (este mensaje está obsoleto).
9	Acceso no autorizado a la red de destino.
10	Acceso no autorizado al host de destino.
11	La red es inalcanzable para el tipo de servicio especificado.
12	El host es inalcanzable para el tipo de servicio especificado.
13	Comunicación no autorizada.
14	Violación de las reglas de precedencia de hosts.
15	Corte de precedencia.

Por supuesto, me imagino que no entendéis ni papa de los contenidos de esta tabla.

Explicar todos los códigos en detalle va a ser demasiado (y, realmente, poco interesante), así que explicaré sólo lo suficiente para que os hagáis una idea de lo más importante.

En primer lugar, no hay que ser muy observador para darse cuenta de que la mayoría de los errores están duplicados, refiriéndose siempre o bien a la **red,** o bien al **host**.

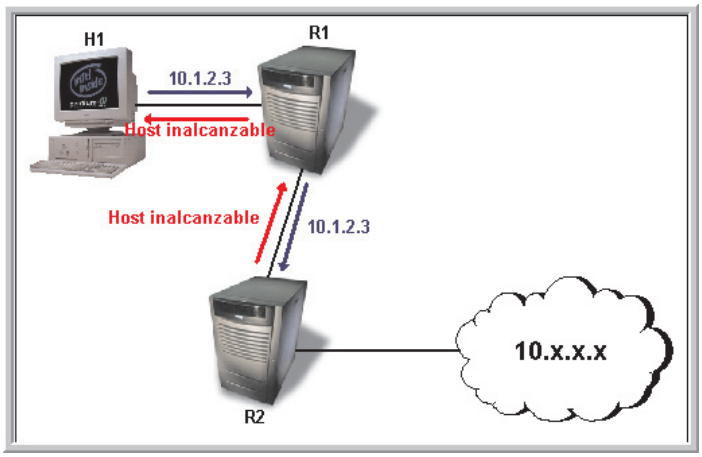
Imaginemos que estamos en una red local, cuyas direcciones IP abarcan desde la **192.168.1.1** hasta la **192.168.1.255**, pero en la cual sólo existen tres máquinas:
Máquina 1 ▸ *192.168.1.1*
Máquina 2 ▸ *192.168.1.5*
Y máquina 3 ▸ *192.168.1.100*

Si intentamos acceder a la máquina 192.168.1.200 recibiremos un mensaje de **Host inalcanzable** (***código 1***). La IP *192.168.1.200* forma parte de una **red** a la que tenemos acceso, pero en cambio ese **host** concreto no existe en la red.

En cambio, si intentamos acceder en el mismo escenario a la IP *10.12.200.1*, estaremos intentando acceder a una **red** que no es la nuestra, por lo que el router

no podrá darnos acceso, y tendrá que respondernos con un error de **Red inalcanzable** (***código 0***).

Con esta breve explicación de paso he explicado los dos primeros códigos. Sólo queda comentar que el **código 1** sólo lo enviará el último router dentro del camino que lleve desde nuestra máquina hasta el host de destino, ya que éste último router será el que sepa qué máquinas en concreto existen dentro de su red. En cambio, el **código 0** puede ser enviado por cualquiera de los routers que haya en el camino entre tu máquina y el host de destino, ya que en cualquier punto del camino se podría detectar que la red de destino es inalcanzable.



En la imagen la máquina H1 solicita acceder a la IP 10.1.2.3 al router R1. Éste pasa la bola a R2, el cual está conectado a la red 10.x.x.x, donde debería encontrarse la IP solicitada. Como R2 sabe que el host 10.1.2.3 no existe, responderá con un mensaje ICMP Host inalcanzable. El mensaje ICMP llegará a R1, que lo retransmitirá hasta H1.

Con respecto al siguiente código, Protocolo inalcanzable (*código 2*), en primer lugar hay que decir que no se genera en un router, si no en el propio host de destino. Este error se genera cuando se intenta acceder a un protocolo de transporte que no está implementado en el host de destino.

Por ejemplo, si enviamos un paquete UDP a una máquina que no tiene implementado UDP. Como la capa IP es la encargada de reconocer al protocolo de nivel superior, será responsabilidad suya, y por tanto de ICMP, el notificar de los errores de protocolos no implementados.

El mensaje de Puerto inalcanzable (*código 3*) se genera cuando se intenta acceder a un puerto en un protocolo de transporte (por ejemplo, TCP o UDP), y ese puerto no está accesible en el host de destino. En muchos casos, el propio protocolo de transporte se encargará de notificar este error (por ejemplo, en TCP mediante el flag **RST**), pero siempre que el protocolo de transporte no tenga implementado un mecanismo para notificar esta situación, será responsabilidad de ICMP hacerlo.

El mensaje de Paquete demasiado grande, y no puede ser fragmentado (*código 4*) tiene relación con un campo de la cabecera IP que veremos más adelante en el curso de TCP/IP. Este campo especifica si un paquete puede ser o no fragmentado en trozos más pequeños. En el caso de que esté especificado que el paquete no puede ser fragmentado, y el paquete sea demasiado grande como para ser enviado de una sola vez, el paquete no podrá ser transmitido.

De momento con esto creo que tenemos suficiente, que no quiero llenar el artículo entero sólo con el primero de los mensajes ICMP y, al fin y al cabo, ya he explicado los códigos más comunes.

Campo: Checksum

Pues aquí nos encontramos de nuevo con

otro checksum, como los que ya vimos en anteriores entregas cuando tratamos el UDP y TCP. En el artículo anterior detallé cómo generar un checksum para TCP, y también como realizar su comprobación para un paquete recibido.

En el caso de ICMP, los checksums (sumas de comprobación) se generan mediante la misma operación, pero en este caso no es necesario añadir una pseudocabecera, si no que directamente la operación se realiza únicamente con todos los bits que componen la cabecera ICMP.

Campo: Unused

Pues eso, ¿qué queréis que os cuente sobre un campo que no se usa? Simplemente lo rellenáis con ceros, y ya está.

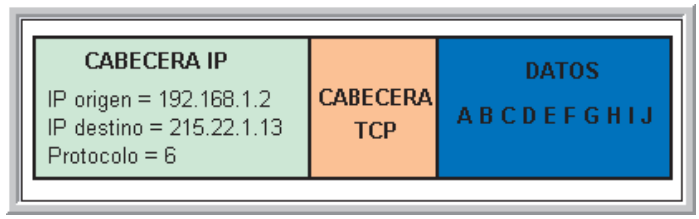
Campo: Internet Header + 64 bits of original data datagram

Aquí ya hay bastante más que decir, ya que además este campo es común a muchos mensajes ICMP. Este campo es especialmente importante, ya que es el que nos permite identificar de forma unívoca a qué datagrama (paquete IP) se refiere el mensaje ICMP.

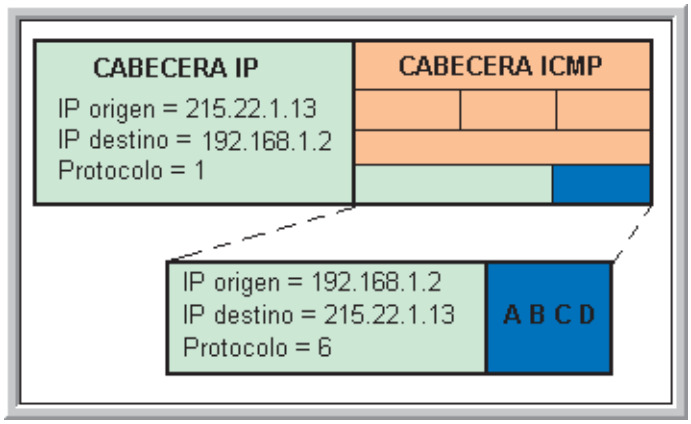
Como ya he dicho, un paquete ICMP puede ser una respuesta a un paquete IP que ha generado algún tipo de error. Si nos llegasen paquetes ICMP sin más no podríamos saber cuál de nuestros paquetes IP generó el error.

Por eso, muchos paquetes ICMP llevan incluidos en su propia cabecera la cabecera IP del paquete que generó el error. Además, para terminar de asegurarnos de que se trata de nuestro paquete, se incluyen también los 64 primeros bits del campo de **DATOS** del datagrama, por lo que el paquete puede quedar identificado sin lugar a dudas.

Por ejemplo, imaginemos que enviamos el siguiente paquete:



En el caso de que este paquete genere, por ejemplo, un error de Destino inalcanzable, éste será el paquete ICMP que nos llegará en respuesta:



Dentro de la cabecera ICMP podemos ver en verde la copia de la cabecera IP del paquete que generó el mensaje, así como los primeros datos del campo de **DATOS** (en azul). También es importante destacar que, como se ve en la imagen, el número de protocolo asignado a ICMP es el **1**.

Detectando gusanos mediante Destination Unreachable

Como os dije, comentaré algún detalle curioso sobre cada uno de los mensajes ICMP. Os voy a comentar una posible

señal de alarma en relación con los mensajes **Destination Unreachable.**

Si observamos los **logs** de nuestra conexión... ¿que qué es eso? Pues en un sistema que tenga implementada una buena política de seguridad es habitual tener un mecanismo de monitorización que guarde un registro (un log) de lo que ocurre con las conexiones de nuestra máquina.

En caso de que no tengáis nada de esto montado (sería motivo para varios artículos el explicar cómo hacerlo bien hecho), podéis hacer pruebas simplemente poniendo en marcha un **sniffer**, como los que he explicado ya en otros artículos (por ejemplo, **Iris** para Windows, o **Ethereal** o **Snort** para Linux).

Pues como decía, si observamos lo que ocurre en nuestra conexión con Internet, y detectamos que nos llegan una gran cantidad de mensajes **ICMP** de tipo **Destination Unreachable**, podría ser síntoma de que estamos infectados con algún **gusano** (Worm).

Un gusano es un tipo de virus informático cuya principal función consiste en reproducirse a toda costa al mayor número de víctimas posible. La gran mayoría de los virus que existen hoy día pueden ser calificados como gusanos, siendo buenos ejemplos los conocidos virus *Sasser*, *MyDoom*, etc.

Algunos de estos gusanos intentarán reproducirse por un sistema tan burdo como es el generar direcciones IP aleatorias, e intentar acceder a todas ellas para tratar de infectar una nueva máquina.

Esto es especialmente útil en gusanos como el *Sasser* (y, de hecho, el *Sasser* utiliza este sistema), que no necesitan ser ejecutados para infectar una máquina (por ejemplo mediante un correo electrónico), si no que basta con que den con una máquina vulnerable (es decir, que tenga cierta versión de Windows sin parchear).

Por supuesto, si el gusano genera direcciones IP que no existen, recibiremos un mensaje **Destination Unreachable** cada vez que el gusano intente infectar una de estas IPs. Por tanto, encontrar varios mensajes ICMP de este tipo cuyo origen desconocemos puede ser un síntoma de que tenemos un visitante no deseado en nuestro PC.

No sólo eso, si no que además analizando en detalle la cabecera ICMP podremos seguramente extraer información acerca del gusano, ya que estos mensajes ICMP, tal y como acabamos de ver, incluyen en su cabecera toda la cabecera IP del mensaje que generó el error, así como los primeros 64 bits de datos.

Este sistema no es definitivo, por supuesto, ya que no todos los gusanos utilizan este sistema para buscar víctimas, así que el hecho de no recibir **mensajes Destination Unreachable** no significa ni mucho menos que no podamos estar contaminados con algún otro tipo de gusano más "inteligente".👏

Podría comentar muchos otros aspectos oscuros de este tipo de mensajes ICMP, como los ataques **Smack**, **Bloop**, **WinNewK**, etc. Pero como el espacio es limitado, y además tengo que despertar un poco en vosotros la capacidad y el ánimo de investigar por vuestra cuenta, ahí os he dejado los nombres, para que vuestro amigo Google os dé más detalles. 👏

Path MTU Discovery (P-MTU-D)

Una aplicación importante de los mensajes **Destination Unreachable**, es el proceso de **Path MTU Discovery**.

Ya sabemos que entre dos máquinas que se comunican suele haber varias máquinas intermedias, que actúan como routers para encaminar los paquetes entre ambas máquinas.

El problema es que cada máquina de todas las involucradas en el proceso (incluyendo las máquinas de origen y destino) puede tener una configuración y unas características diferentes. Esto da lugar a que haya máquinas preparadas para manejar paquetes más o menos grandes.

El tamaño máximo de paquete que puede manejar una máquina se denomina **MTU** (**M**ax **T**ransmission **U**nit), y depende de la tecnología utilizada, tal y como veremos cuando hablemos sobre el nivel de enlace a lo largo del curso.

Por ejemplo, una tecnología Ethernet utiliza una MTU de 1500 Bytes, mientras que en X.25 se utiliza una MTU de sólo 576 Bytes. Pero bueno, ya explicaremos esto en otro número...

De momento, tenemos que tener la idea intuitiva de lo que es la **MTU** que, por cierto, está bastante relacionada con el **MSS**, es decir, el **tamaño máximo de segmento** que vimos en **TCP**. El tamaño máximo de segmento indica el tamaño máximo del campo de datos de un paquete TCP, el cual estará directamente relacionado con el tamaño de MTU de la máquina que envió el paquete TCP.

Si queremos que funcione correctamente una comunicación entre dos máquinas tendrá que haber un acuerdo sobre qué tamaño de MTU utilizar ya que, no sólo cada una de las dos máquinas tendrá una MTU diferente, si no que también podrían tener MTUs diferentes cada uno de los routers que se encuentran en el camino entre las dos máquinas.

Cuando un router se encuentra con un paquete cuyo tamaño es mayor que el de su MTU, tendrá que **fragmentarlo** en trozos más pequeños que sí que quepan en su MTU.

El problema con esto es que la fragmentación es muy poco deseable, por motivos que explicaré a lo largo del curso, por lo que conviene evitarla. Para poder evitar la fragmentación, una buena idea consiste en conseguir un acuerdo sobre el tamaño máximo de los paquetes entre todas las máquinas involucradas en la conexión.

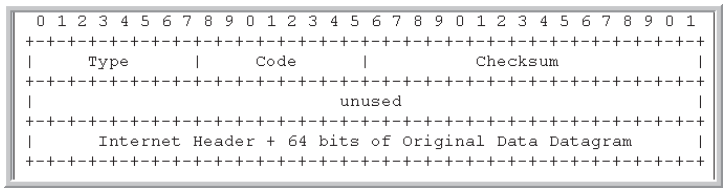
El mecanismo para conseguir este acuerdo se denomina **Path MTU Discovery**, y viene detallado en el **RFC 1191**.

A grandes rasgos, lo que se hace es enviar un paquete demasiado grande, pero forzando a que no sea fragmentado (mediante una opción que se puede especificar en la cabecera IP). Si alguna máquina del camino del paquete no puede manejar un paquete tan grande, al no poder tampoco fragmentarlo no le quedará más remedio que responder con un **ICMP Destination Unreachable Code 4**. A partir de ahí, se irá repitiendo el proceso con diferentes tamaños de paquete, hasta que se de con un tamaño que sí que sea aceptado por todas las máquinas.

Este tamaño máximo para todo el camino que se ha encontrado se denomina precisamente **Path MTU**.

2.2. Tiempo excedido (Time Exceeded)

Como podemos ver, la cabecera de este mensaje consta de los mismos campos que el mensaje anterior:



Para explicar la utilidad de este mensaje vamos a ver mejor los campos que lo forman.

Campo: Type

Para este tipo de mensajes, su valor ha de ser 11, en decimal, por supuesto.

Campo: Code

En este caso sólo existen dos posibles valores, ya que este mensaje puede ser enviado por dos motivos diferentes:
Code 0: Tiempo de vida superado.

Como veremos a lo largo del curso, en la **cabecera IP** existe un campo **TTL** (**T**ime **T**o **L**ive) que permite limitar el número de routers que atravesará un paquete para llegar a su destino.

Si no limitásemos este número de saltos, se podría dar el caso de que un paquete quedase circulando para siempre en la red en caso de que algún router mal configurado produjese un bucle cerrado en el que los paquetes van de un router a otro sin sentido, sin llegar jamás a su destino.

Cada paquete IP llevará, por tanto, un campo **TTL** con un valor numérico determinado, y este valor se irá decrementando en cada router por el que pase el paquete.

Así, si por ejemplo un paquete tiene un **TTL = 3**, el primer router que reciba el paquete modificará este valor haciendo **TTL=2**, y reenviará el paquete al próximo router del camino entre el origen y el destino.

En caso de que un paquete con **TTL=0** llegue a un router, significará que el paquete no ha podido llegar a su destino en un plazo limitado, por lo que el router tendrá que responder al transmisor del mensaje con un **ICMP** de tipo **Time Exceeded**, y **Code 0**.

Code 1: Tiempo de reensamblaje superado.

Otra situación en la que un paquete puede "caducar" tiene relación con la **fragmentación** que ya mencioné antes.

Como dije, un paquete demasiado grande puede ser dividido en fragmentos para facilitar su transmisión. Los fragmentos irán llegando al destino en cualquier orden, y éste se encargará de reensamblar el paquete. Por supuesto, hay un límite de tiempo por el que estará esperando el destino a que le lleguen todos los fragmentos, para evitar quedarse esperando indefinidamente en caso de que haya habido algún problema de transmisión.

Si un paquete que está en proceso de reensamblaje no se completa en un tiempo determinado, se enviará un mensaje **ICMP** de tipo **Time Exceeded**, y **Code 1**, indicando que no se ha podido completar el reensamblaje del paquete.

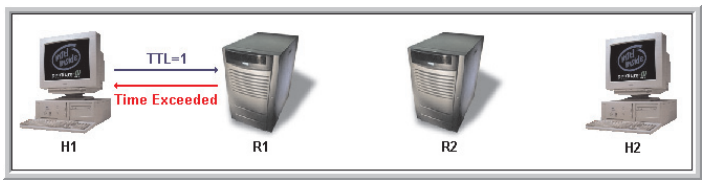
Trazado de rutas mediante Time Exceeded

Posiblemente habréis utilizado alguna vez la conocida herramienta **traceroute**.

Como sabréis, esta herramienta permite conocer todos los routers que existen en el camino entre un origen y un destino.

Esta herramienta funciona precisamente gracias a los mensajes **ICMP** de tipo **Time Exceeded**, concretamente a los de **Code 0**.

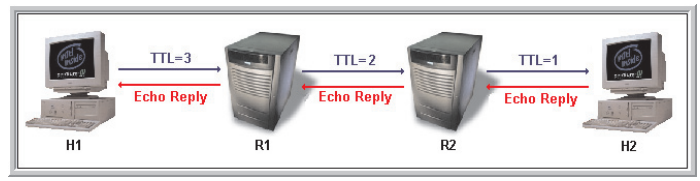
Lo que hace traceroute es empezar enviando un paquete cualquiera al destino que queremos trazar, pero utilizando en el paquete un **TTL=1**. En cuanto este paquete llegue al primer router que haya en el camino entre tu máquina y la máquina de destino, este router decrementará el **TTL**, por lo que lo dejará en **TTL=0**. Al no poder reenviarlo, ya que el tiempo de vida del paquete ha expirado, tendrá que enviar un **ICMP Time Exceeded Code 0** en respuesta.



Este paquete contendrá como **IP de origen** la del router que lo generó (**R1**), por lo que conocemos así ya la IP del primer router que nos hemos encontrado en el camino.

A continuación, traceroute enviará otro paquete similar, pero con un **TTL=2**. El primer router del camino, **R1** (cuya IP ya conocemos), decrementará el **TTL**, pero al ser ahora el **TTL=1**, podrá seguir reenviándolo, por lo que el paquete llegará al próximo router, **R2**. En cambio, en este segundo router ocurrirá lo mismo que antes, pues se encontrará con un paquete con **TTL=0** que no podrá reenviar, por lo que tendrá que responder con un **ICMP Time Exceeded Code 0**.

Este proceso continuará, incrementando en uno cada vez el valor de **TTL**, hasta que finalmente sea el **host de destino** el que nos responda:

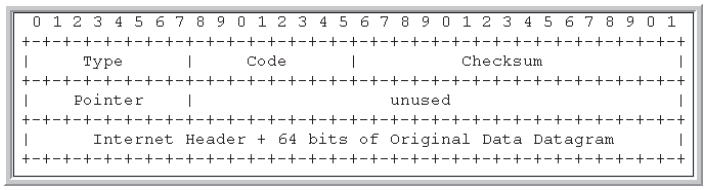


Como vemos en la imagen, el **host de destino** no nos responderá con un **Time Exceeded**, ya que el paquete sí que ha podido llegar hasta él antes de expirar. En lugar de eso, nos responderá con otro mensaje **ICMP**, de tipo **Echo Reply**, que veremos más adelante. Esto se debe a que los paquetes que va enviando **traceroute** son en realidad mensajes **ICMP** de tipo **Echo Request**. Como ya he dicho, más adelante comprenderemos de lo que estoy hablando.

2.3. Problema de parámetros (Parameter Problem).

Este mensaje se envía cuando algún parámetro de la cabecera IP tiene un valor incorrecto, siempre y cuando no sea un campo que tenga ya otro mensaje ICMP específico para notificar el error.

Este es el formato del mensaje:



Campo: Type

El valor en este caso es **12**.

Campo: Code

Aunque el RFC sólo especifica un posible valor para este campo, en realidad pueden ser implementados otros valores (como ya ha pasado con otros campos vistos anteriormente).

Estos son los posibles valores:

Code	Descripción
0	El campo **Pointer** especifica el error
1	Falta una opción requerida
2	Tamaño incorrecto de paquete o de cabecera

El valor más habitual es el **Code 0**.

En este caso, el campo **Pointer** indicará la **posición** de la cabecera IP en la que se encuentra el error. Conociendo la cabecera IP, podemos localizar el parámetro exacto cuyo valor es incorrecto.

El **Code 1** es usado cuando una trasmisión requiere opciones IP adicionales, como las usadas por el ejército de los EEUU al utilizar transmisiones seguras, en las que son requeridas ciertas opciones de seguridad en los datagramas.

Sobre el **Code 2** no hay mucho que decir, ya que se explica por sí sólo.

Campo: Pointer

Si el campo **Code** tiene valor **0,** este campo especifica la **posición en bytes** dentro de la **cabecera IP** donde se encuentra el parámetro erróneo.

Destruyendo firewalls

Como comentario sobre este tipo de mensaje ICMP comentaré un **exploit** para una aplicación en concreto, que siempre puede ser interesante.

La aplicación no es ni más ni menos que **Gauntlet Firewall**, un firewall de **Network Associates**, la compañía del famoso (y bastante triste) antivirus McAffee que, por cierto, hace poco apareció la noticia de que es posible que Microsoft compre también esta compañía. 🧐

El exploit que, por cierto, también afecta a otro producto de la misma compañía, **WebShield**, aprovecha una vulnerabilidad de estos programas que da lugar a una denegación de servicio (**DoS**) con sólo enviar un mensaje **ICMP Parameter Problem** malformado.

El exploit, y su explicación en castellano los tenéis en: http://www.hakim.ws/ezines/RazaMexicana/raza011/0x02.txt

2.4. Calmar al transmisor (Source Quench)

Este mensaje es útil para implementar un **control de flujo** rudimentario. Ya he explicado algo sobre el control de flujo a lo largo del curso.

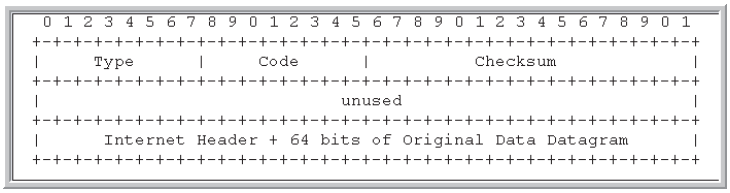
Como su nombre indica, este mensaje permite calmar al transmisor cuando está enviando demasiados paquetes, y estos no pueden ser procesados debido a la excesiva velocidad.

Cuando un paquete no pueda ser procesado adecuadamente debido a la

saturación del buffer de recepción, habrá que notificar esta situación mediante un **ICMP Source Quench**. Cuando el transmisor reciba uno o más **Source Quench**, debe bajar la tasa de transferencia para no seguir saturando al receptor.

Este mecanismo es más efectivo si el receptor empieza a enviar los **Source Quench** antes de que haya realmente un problema, es decir, cuando esté cerca del límite, pero sin haberlo superado aún.

El formato del mensaje nos es ya muy familiar:



Campo: Type

El valor es **4.**

Campo: Code

Aquí es siempre **0**.

Denegación de servicio y Source Quench

Algunos de los ataques que he comentado aprovechando el protocolo ICMP son ataques de tipo **Flood**, es decir, ataques **DoS** que funcionan mediante un bombardeo masivo de paquetes.

Es posible que un sistema que está siendo floodeado intente calmar al atacante enviándole mensajes **ICMP Source Quench**, pensando que el bombardeo no es malintencionado y, por tanto, avisando así de que no da a basto.

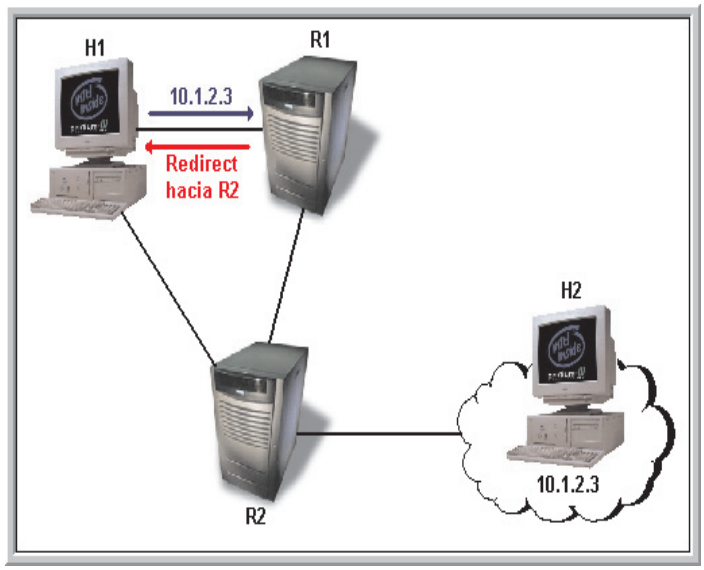
Si estamos realizando un ataque de **flood** y empezamos a recibir mensajes Source **Quench** de la víctima tenemos una buena prueba de que nuestro ataque está funcionando.

Por otra parte, el **Source Quench** mismo puede servir para realizar ataques **DoS**. Si enviamos mensajes **Source Quench** a la víctima le estaremos pidiendo que limite su ancho de banda porque supuestamente nos está saturando. Si conseguimos suplantar la personalidad de una máquina a la que esté conectada la víctima (mediante spoofing) podremos echar abajo esa conexión, limitando cada vez más el ancho de banda.

2.5. Redireccionar (Redirect)

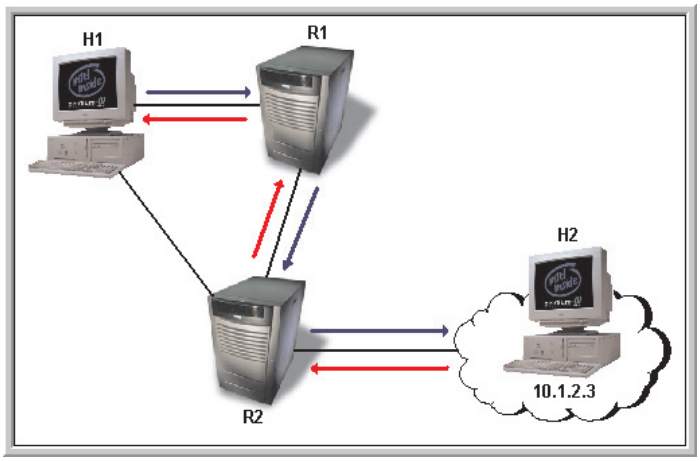
Este mensaje, bastante peligroso, se utiliza cuando un router **R** recibe un paquete que debe llevar hasta un destino, pero se da cuenta de que existe un camino más corto hasta ese destino que no pasa por **R**.

Veamos por ejemplo este caso:

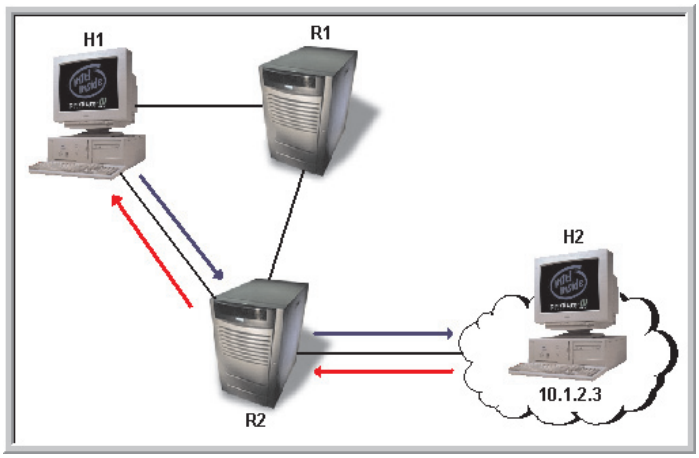


La máquina **H1** envía un datagrama al router **R1** para que lo transmita hasta la máquina **H2**. El router **R1** mira en su tabla de enrutamiento y descubre que para llegar a **H2** tiene que pasar primero por **R2**. Analizando las direcciones de **R2** y **H1** deduce que ambos pertenecen a la misma red, por lo que **H1** podría acceder directamente a **R2** sin necesidad de pasar por **R1**.

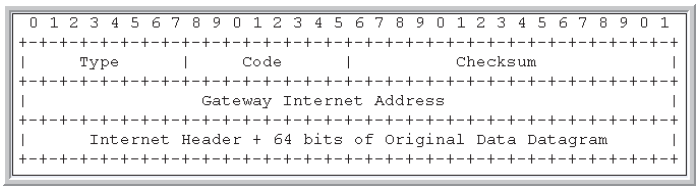
Si **no** notificase esta situación a **H1** toda la comunicación entre **H1** y **H2** seguiría el siguiente camino:



Mientras que si avisa de esta situación mediante un **ICMP Redirect**, podría ahorrarse un salto en el camino yendo de forma más directa:



Veamos el formato del mensaje:



Campo: Type

El valor es **5**.

Campo: Code

De nuevo, tenemos varios posibles valores para este campo:

Code	Descripción
0	Redireccionar todos los datagramas para una red
1	Redireccionar todos los datagramas para un host
2	Redireccionar todos los datagramas de cierto tipo de servicio para una red
3	Redireccionar todos los datagramas de cierto tipo de servicio para un host

El segundo caso, **Code 1**, es el explicado en el ejemplo anterior.

El **Code 0**, es lo mismo, pero en lugar de tratarse de un **host** de destino se trata de toda una **red**.

Para el **Code 2** y el **Code 3** hay que anticipar de nuevo algo sobre el protocolo IP, que aún no hemos visto.

Uno de los campos de la cabecera IP es el **TOS** (**T**ype **O**f **S**ervice), o **Tipo de Servicio**. Este campo permite definir diferentes servicios en función del ancho de banda, fiabilidad, e interactividad que requieren. Por ejemplo, un servicio de transferencia de archivos requiere mucho ancho de banda pero poca interactividad, mientras que un servicio de terminal remota (como un telnet) generalmente necesitará poco ancho de banda, pero mucha interactividad. Todo esto ya lo veremos en detalle cuando hablemos del protocolo IP.

En ciertos casos, un determinado router puede estar más dedicado a transmitir cierto tipo de servicio, por lo que se puede optimizar la comunicación utilizando los caminos adecuados para cada caso.

El **RFC 1349** da detalles sobre el campo TOS y su relación con ICMP, concretamente con los ICMPs **Redirect**, y **Destination Unreachable** (donde vimos que los **Code 11 y 12** también se refieren al tipo de servicio).

Campo: Gateway Internet Address

Para los que aún no lo sepáis, un gateway es lo que nosotros estamos llamando un router, por lo que este campo indica precisamente la dirección IP del router al que queremos redireccionar el tráfico.

Por ejemplo, en el caso citado anteriormente, el router **R1** enviaría un mensaje **ICMP Redirect** donde este campo contendría la dirección IP del router **R2**.

Super Source Quench, y otras historias

***Entre los mil usos ilegítimos que se pueden hacer de los mensajes* ICMP Redirect**, se encuentra el ataque conocido como **Super Source Quench**.

A pesar de que su nombre pueda confundirnos, este ataque no tiene nada que ver con el **ICMP Source Quench**, aunque sí que es cierto que, si bien un **Source Quench** puede ralentizar tu conexión, un **Super Source Quench** lo que hace es tirártela por completo.

El ataque consiste simplemente en enviar un mensaje ICMP **spoofeado** (para que aparentemente provenga de un router legítimo) de tipo **Redirect**, en el cual el campo **Gateway Internet Address** contenga la propia dirección IP de la víctima.

Si la víctima es vulnerable a este ataque, desde que reciba el paquete de **Super Source Quench**, redirigirá todo su tráfico hacia si mismo, dando lugar a un bucle infinito de tres pares de narices. 🙂

El Super Source Quench es, por supuesto, un ataque de tipo DoS, pero gracias a los paquetes **Redirect** de ICMP se pueden llevar a cabo todo tipo de ataques de lo más variopintos. Existen herramientas para automatizar este tipo de ataques, como **WinFreez**, que permiten realizar ataques DoS, ataques de suplantación *man in the middle*, etc, etc. En el momento que tienes el poder de redirigir el tráfico de una víctima hacia donde tú quieras los límites sólo los pone tu imaginación. 😉

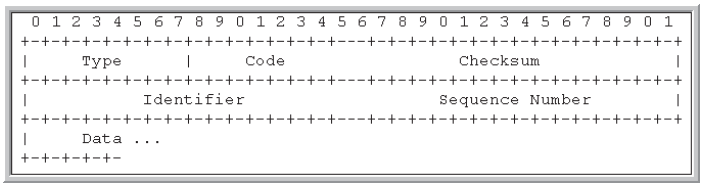
2.6. Petición de Eco (Echo Request)

Este es uno de los mensajes ICMP más interesantes, aunque en principio pueda parecer tan simple que carezca de interés.

Además de utilizarse en la herramienta **traceroute**, como ya vimos, es también la base del funcionamiento del famoso **PING**, ya que su función consiste simplemente en solicitar a otra máquina que nos devuelva lo que le decimos, como si de un eco se tratase.

Un mensaje **Echo Request** contendrá unos pocos datos de muestra (por ejemplo, se suele usar el abecedario: *abcdefghijklmnopqrstuvwxyz*), y el host que lo reciba debería responder mediante otro mensaje **ICMP** diferente, que es el

Echo Reply (lo veremos a continuación), que contendrá los mismos datos de muestra (el abecedario de nuevo, por ejemplo).



Como vemos, el formato del mensaje es más complejo que el de los anteriores tipos de ICMP.

Campo: Type

En este caso el valor es **8.**

Campo: Code

Siempre tomará valor **0.**

Campo: Identifier

Este campo simula en cierto modo el comportamiento de los puertos UDP, aunque de forma mucho más simple.

Simplemente es un número que sirve para identificar una "**sesión**" de mensajes de eco.

Se usa por ejemplo cuando hacemos un **ping** a una máquina, ya que siempre se hará más de un **Echo Request** a la misma máquina. Toda esta serie de **Echo Request** que componen un **ping** tendrán normalmente el mismo valor en el campo **Identifier**.

Realmente, el estándar no exige ninguna forma concreta de generar este campo, por lo que se deja un poco a las necesidades de cada aplicación.

Campo: Sequence Number

Este campo, combinado con el anterior, permite identificar unívocamente un mensaje concreto de **Echo Request**, para que el **Echo Reply** correspondiente pueda especificar a qué petición de eco en concreto está respondiendo.

El campo **Sequence Number** simula de manera muy rudimentaria **un número de secuencia** TCP, siendo por tanto un número que simplemente se va incrementando con cada **Echo Request** para un mismo **Identifier**.

De nuevo, el estándar nos deja libertad para que utilicemos este campo según las necesidades.

Campo: Data

Este campo contiene los datos de muestra que queremos que nos sean devueltos, como cuando gritamos *"Eco!"* y nos vuelve la respuesta *"Eco!..."*.

En lugar de *Eco* podríamos poner cualquier otra palabra, como el ya mencionado abecedario.

Así que ya sabéis, cuando vayáis por la montaña y encontréis a un tío que está berreando el abecedario "*abcdefghijklmnopqrstuvwxyz*", no debéis asustaros, ya que sólo será un friki con una indigestión de mis artículos que habrá terminado creyéndose que es un PC y está intentando hacer un ping

Sobre el mensaje **Echo Request** no voy a comentar nada de momento (en color rojo, me refiero , ya que comentaré luego juntos los mensajes **Echo Request** y **Echo Reply** después de hablar sobre éste último.

2.7. Respuesta de Eco (Echo Reply)

Este mensaje tiene exactamente el mismo formato, con los mismos campos.

Todos los campos de la cabecera deben llevar una copia exacta de los campos del mensaje **Echo Request** al que están respondiendo, excepto el campo **Type**, que debe ser **0,** que es el número asignado al tipo de mensaje **ICMP Echo Reply**.

La invasión de los pitufos

No se puede hablar de ICMP sin hablar de una de las técnicas más clásicas para explotar este protocolo con fines poco éticos, que es la conocida como "*el pitufo*". Jejeje, bueno, ésta sería la traducción al castellano. En realidad el término "técnico" es **smurf**.

La técnica de smurf consiste en un ataque **DoS** basado en los mensajes **Echo Request** y **Echo Reply**, y que se puede llevar a cabo sin necesidad de ninguna herramienta especial, ni siquiera un software para manipular **raw sockets** (como **Nemesis** o **Hping**, sobre los que hablaré al final del artículo), si no que basta con la herramienta **PING** que podemos encontrar en cualquier sistema operativo.

El ataque consiste exactamente en un **flood** mediante mensajes **Echo Reply** generados por máquinas que no son la del propio atacante, por lo que se hace bastante complicado para la víctima el encontrar al culpable.

Vamos a ver en qué consiste exactamente:

Todas las redes tienen definidas una dirección especial, que es la denominada dirección **broadcast**. Esta dirección IP especial está reservada para paquetes que estén dirigidos a **todas las máquinas de la red**, y no a una sola.

Como ya sabemos, todas las máquinas de una red están interconectadas, por lo que teóricamente todos los paquetes que circulan por la red llegan a todas las máquinas de la red. Cada máquina tendrá que ser capaz de discernir qué paquetes van dirigidos a ella y cuales no. Este es el motivo por el que un **sniffer** puede funcionar.

Lo que hace el sniffer es poner a la máquina en modo **promiscuo**, es decir, hacer que recoja todos los paquetes, aunque no sean dirigidos para esa máquina. Una máquina que no esté en modo promiscuo sólo debería recoger dos tipos de paquetes: los que van dirigidos a su dirección IP, y los que van dirigidos a la dirección **broadcast**.

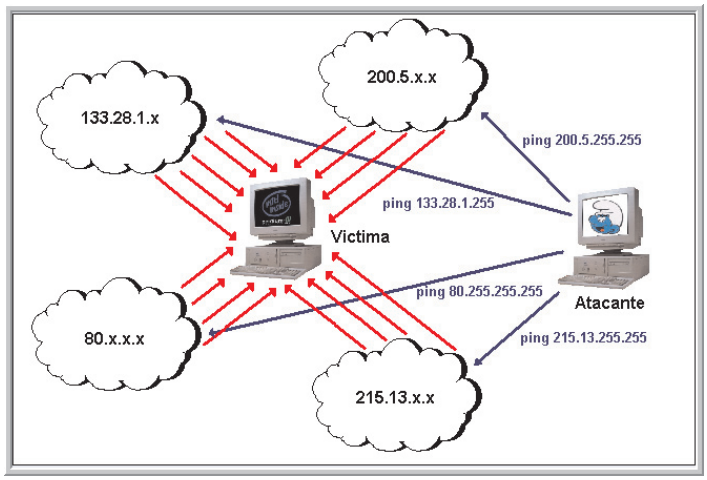
Por tanto, ¿qué pasaría si enviásemos un mensaje **Echo Request** a la dirección **broadcast** de una red? Pues teóricamente todas las máquinas de la red tendrían que recibir ese mensaje y, por tanto, enviar un **Echo Reply** en respuesta. Imaginemos que estamos en una red con mil máquinas. Con sólo enviar nosotros un mensaje, habremos recibido mil mensajes en respuesta...

¿Qué pasaría entonces si enviásemos ese **Echo Request** con la **IP de origen spoofeada** para aparentar que quien manda el mensaje no somos nosotros, si no la máquina de nuestra víctima?

Pues, teóricamente, la víctima recibiría 1000 mensajes **Echo Reply** que no tiene ni idea de dónde narices han salido. Sabrá que las respuestas llegan de las máquinas de la red que hemos utilizado, pero no podrá saber quién dio la "orden" a esa red de que le mandase los mensajes.

Si lanzamos varios **Echo Request** a varias redes diferentes, podremos conseguir que llegue un auténtico bombardeo de

mensajes **Echo Reply** a la víctima, con lo que podríamos llegar a saturar su conexión.



Como vemos en la imagen, lo único que tiene que hacer el atacante es lanzar unos cuantos **pings**, uno por cada red que quiera utilizar como **amplificadora** del ataque, utilizando en cada uno la dirección de **broadcast**, que es la que vemos en la imagen.

Una dirección broadcast normalmente se obtiene poniendo un **255** en aquellos bytes de la IP que puedan variar según el tipo de red, es decir, poniendo un **255** donde en la **máscara de red** hay un **0.**

Por ejemplo, para una red *192.168.1.1*, con máscara de red *255.255.255.0*, la dirección **broadcast** será **192.168.1.255**.

En general, para una red de **clase A** habrá que poner un **255** en las **3 últimas cifras** de la IP, en una red de **clase B** un **255** en las **2 últimas cifras** de la IP, y en una red de **clase C** un **255** sólo en la **última cifra** de la IP. ¿Qué no sabéis de qué hablo? Pues esperad al próximo artículo del curso, que tratará sobre el protocolo **IP**.

Para evitar convertir nuestra red en una amplificadora de ataques smurf, tenemos que configurarla para que las máquinas no respondan a mensajes ICMP dirigidos a la dirección broadcast.

El ping de la muerte

Que nadie se eche las manos a la cabeza diciendo: "¡Pero si el ping de la muerte está más pasado de moda que las patillas de Curro Jiménez!".

Es cierto que el ping de la muerte es un ataque que difícilmente puede funcionar hoy día, pero en su momento fue una auténtica **"bomba"** así que merece al menos una mención en este artículo en el que no sólo se trata la actualidad, si no también la historia de ICMP.

Digo lo de "bomba" porque es importante diferenciar entre un **DoS** de tipo **flood**, y un **DoS** de tipo **nuke**.

Un **flood** ya hemos visto lo que es: tirar abajo un sistema a base de saturarlo con miles de paquetes. En cambio, un **nuke** es un ataque que también puede tirar abajo un sistema, pero en este caso con sólo uno o unos pocos paquetes.

Vimos un ejemplo de **nuke** al hablar del **ICMP Parameter Problem**, y un ejemplo de **flood** por ejemplo en la técnica de **smurf** mencionada anteriormente.

El ping de la muerte (**ping of death**), y sus variantes (como el **jolt**, o el **IceNewk**) es un ataque DoS de tipo **nuke** que explota una limitación de los sistemas operativos, ya que estos suelen tener limitado a **65536** el tamaño máximo de paquete que pueden manejar.

No confundamos este tamaño con la **MTU**, ni con el **MSS**, ya que en este caso se refiere al tamaño que puede manejar el

propio sistema operativo, y no el tamaño de los paquetes que se pueden transmitir a través de una conexión.

Como la MTU siempre será menor de 65536 la forma en que se conseguía enviar el ping de la muerte era utilizando la **fragmentación** de la que ya hemos hablado. El ping de la muerte consistía simplemente en un mensaje **Echo Request** de un tamaño mayor que el que podía manejar el sistema operativo receptor, que se enviaba fragmentado para que pudiese circular hasta su destino. Una vez en el destino, el sistema receptor intentaba **reensamblar** el paquete uniendo los fragmentos y, al encontrarse con que el paquete era más grande de lo que podía manejar, el sistema directamente cascaba.

Este problema fue "descubierto" en 1996, y en 1997 la mayoría de los sistemas operativos ya disponían de parches para arreglar este problema, pero.... ¿quién sabe cuántos sistemas pueden quedar ahí fuera sin parchear?...

Túneles para pasar información a través de firewalls

Un sistema que es utilizado por programas maliciosos, como troyanos, demonios de redes DDoS, o cualquier otra aplicación que necesite pasar información a través de un firewall, es el crear un túnel mediante mensajes ICMP.

¿Qué os ha sonado muy raro eso de los demonios de **redes DDoS**? Jeje, he conseguido despertar vuestra curiosidad una vez más soltando por ahí un término raro como quien no quiere la cosa.

Pues si queréis saciar vuestra curiosidad podéis leer este documento en castellano: http://www.fi.upm.es/~flimon/ddos.pdf Es un documento muy interesante y fácil de leer, en el que encontraréis también una descripción detallada del ataque smurf que mencioné anteriormente.

Antes de que me vaya más del tema, estaba diciendo que se puede aprovechar un mensaje ICMP para pasar información sin que un firewall se entere.

Si tenemos, por ejemplo, un **troyano** en nuestro sistema, pero tenemos un **firewall**, lo que no podrá hacer el troyano es abrir un puerto **TCP** en escucha para que su controlador se conecte remotamente a nuestra máquina para hacernos todo tipo de judiadas.

Como los firewalls son cada día más comunes, los troyanos tienen que buscar otros sistemas más ingeniosos para poder comunicarse con su controlador. Uno de estos sistemas consiste precisamente en aprovechar los mensajes **Echo Request** y **Echo Reply**.

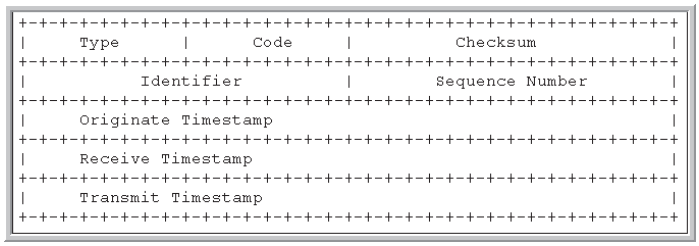
Como hemos visto, ambos mensajes contienen un campo **DATA** en el que se puede meter cualquier cosa... pues simplemente intercambiando mensajes de **Echo** que contengan todos los datos que queramos transmitir entre el troyano y su controlador, estos podrán realizar una comunicación bastante similar a la que podrían llevar a cabo a través de un puerto UDP.

Otra historia es si el firewall filtra también los mensajes ICMP, pero en la mayoría de los sistemas se deja abierto el **Echo Request** hacia el exterior, y el **Echo Reply** desde el exterior, por lo que el troyano podría utilizar Echo Request para enviar sus datos, y el controlador del troyano podría utilizar **Echo Reply** para lanzar sus órdenes.

2.8. Petición de Sello de Tiempo y Respuesta de Sello de Tiempo (Timestamp Request y Timestamp Reply)

Estos mensajes, al igual que los anteriores, también son una pareja.

El **Timestamp** sirve para medir los tiempos de respuesta en una comunicación entre dos máquinas. El mensaje **Timestamp Request** envía un dato con el instante en que el mensaje fue enviado, y el mensaje de respuesta **Timestamp Reply** contendrá otros datos informando sobre el tiempo que tardó el paquete en ser procesado, tal y como veremos en breve.



Campo: Type

El tipo para el mensaje **Timestamp Request** es **13**, y para el mensaje **Timestamp Reply** es **14.**

Campo: Code

Es siempre **0** en ambos mensajes.

Campos: Identifier y Sequence Number

Tienen el mismo significado que en el caso de **Echo Request** y **Echo Reply.**

Campo: Originate Timestamp

Este campo es generado en el mensaje **Timestamp Request**. Especifica el momento exacto en el que el emisor del **Timestamp Request** envió el mensaje. Este tiempo se mide en milisegundos transcurridos desde la medianoche

Campo: Receive Timestamp

Este campo se generará en el mensaje **Timestamp Reply**. Especifica el momento exacto en el que el receptor del **Timestamp Request** (que, por supuesto, será el emisor del **Timestamp Reply**) recibió el mensaje **Timestamp Request.**

Campo: Transmit Timestamp

Este campo se genera también en el mensaje **Timestamp Reply**. Especifica el momento exacto en el que el emisor del **Timestamp Reply** envío el mensaje. Comparando este campo con el anterior podemos hacernos una idea del tiempo que se ha tardado en procesar los mensajes.

Explotando sistemas de seguridad basados en el tiempo

Muchos sistemas de seguridad dependen de la generación de números pseudoaleatorios.

Un ordenador es una máquina totalmente determinista, por lo que es imposible conseguir que genere un número totalmente aleatorio, es decir, sin estar basado en ninguna regla matemática que lo genere.

Lo que se suele hacer para simular esta aleatoriedad es aprovechar un parámetro que está en permanente cambio: el tiempo.

Los números pseudoaleatorios que genera un ordenador suelen estar basados en una fórmula matemática que incluye como

una de sus variables el tiempo exacto en el que se está generando el número (en milisegundos, o cualquier otra medida). Esto permite que dos números generados consecutivamente con la misma fórmula den resultados diferentes.

Por tanto, el permitir que cualquier máquina nos pregunte la hora exacta (con precisión de milisegundos) que tenemos en nuestra máquina, es facilitarle en gran medida la explotación de cualquier sistema que maneje números aleatorios. Recordemos por ejemplo lo que conté en el artículo sobre TCP acerca de los números de secuencia, y de las graves consecuencias que tendría que un atacante conociese los números pseudoaleatorios que hemos utilizado para generar los números de secuencia en nuestras conexiones TCP.

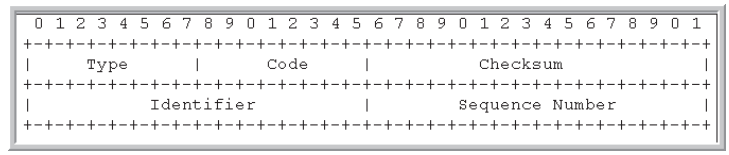
También conté algo sobre la adivinación de números pseudoaleatorios en el artículo sobre DNS de la serie RAW, así que a los aficionados a las matemáticas les recomiendo que le echen un vistazo.

2.9. Petición de Información y Respuesta de Información (Information Request, e Information Reply)

Esta nueva parejita de mensajes está en realidad obsoleta, ya que su funcionalidad ha sido sustituida y mejorada por otros sistemas, como **BOOT** o **DHCP**.

Al igual que en estos protocolos, el objetivo de estos mensajes es conseguir que una máquina entre en una red de forma automatizada, es decir, sin conocer previamente la configuración de la red. Este sistema era útil en estaciones de trabajo sin disco que arrancaban directamente en red, y nada más arrancar necesitaban conocer la red en la que entraban.

El mecanismo consistía en enviar un datagrama que tuviese ceros en las direcciones IP de origen y de destino (**Information Request**). Este paquete, al ser recibido por la máquina que se encargase del asunto dentro de la red, sería respondido con otro mensaje que sí que tuviese direcciones IP de origen y de destino (**Information Reply**). La dirección IP de destino del **Information Reply** sería la IP asignada a la máquina que la solicitó.



Ya conocemos el significado de todos los campos, así que sólo hay que decir que el campo **Type** para **Information Request** es **15**, y para **Information Reply** es **16**. El campo **Code** para ambos será **0.**

Buscando víctimas potenciales

Uno de los usos más clásicos de ICMP para fines oscuros es el de la detección de máquinas en una red, para apuntar víctimas potenciales.

No he hablado de ello hasta ahora porque el sistema más obvio y más común es utilizar simplemente un **escaneo de pings** que vaya recorriendo todas las posibles IPs para ver cuáles de ellas responden al **ping**. Cada IP que responda es una máquina que está funcionando en la red y, por tanto, una víctima potencial.

Como este sistema es bien conocido por cualquier administrador que vigile mínimamente la seguridad, es fácil

encontrarse con sistemas que tengan cerrados los mensajes de **Echo**, por lo que no responderían al **Echo Request**, aunque la máquina estuviese vivita y coleando. Un escaneo de pings nos diría que esas máquinas no existen, ya que no han respondido a nuestra llamada.

Existen herramientas que aprovechan otros tipos de mensajes ICMP menos comunes para hacer exactamente lo mismo, con la ventaja de que, al ser un mensaje poco conocido y, aparentemente inocente, es más probable que el administrador no los haya cerrado en el firewall. Un ejemplo de estas herramientas es **ICMPEnum**, que utiliza precisamente no sólo Echo Request para hacer los escaneos, si no **también Timestamp Request**, e **Information Request**.

Como ya hemos visto que los dos primeros tienen otros problemas potenciales de seguridad, quizá podamos tener suerte y encontrarnos con que el administrador ha considerado totalmente inofensivo el **Information Request**, y haya dejado aquí la puerta abierta que buscábamos.

¿Cuál es la conclusión que podemos sacar de este "comentario rojo" y de todos los demás? Pues, abreviando, que si queremos estar seguros lo mejor es cerrar en nuestro firewall CASI TODOS los mensajes ICMP. Como mucho, podemos dejar entrar los mensajes de respuesta (como **Echo Reply**), para poder hacer **ping** y **traceroute** nosotros desde nuestra máquina, pero pocos motivos hay para dejar abiertos sus contrarios, los mensajes que, en lugar de respondernos, nos preguntan a nosotros desde fuera.

Yo personalmente tengo abiertos sólo los mensajes **Echo Reply** (para poder hacer ping y traceroute), **Time Exceeded** (para poder hacer traceroute), y **Destination Unreachable** (para poder hacer P-MTU-D, entre otras cosas).

Si queréis asegurar de verdad vuestro sistema o vuestra red os aconsejo que investiguéis bien el tema y lleguéis a establecer una decisión sobre vuestra política de seguridad con ICMP.

2.10. Otros mensajes ICMP.

Aunque el **RFC 792** ni siquiera los menciona, existen otros tipos de mensaje ICMP definidos. Al no formar parte del estándar definido en el RFC, sólo nombraré alguno de ellos por si os interesa buscar información sobre alguno en concreto.

Personalmente, yo no me he encontrado nunca con casi ninguno de estos mensajes ICMP, así que no sé hasta qué punto será útil conocerlos.

Type	Nombre
6	Alternate Host Address
9	Router Advertisement
10	Router Solicitation
17	Address Mask Request
18	Address Mask Reply
30	Traceroute
31	Datagram Conversion Error
32	Mobile Host Redirect
33	IPv6 Where Are You
34	IPv6 Here I Am
35	Mobile Registration Request
36	Mobile Registration Reply
37	Domain Name Request
38	Domain Name Reply
39	SKIP Algorithm Discovery Protocol
40	Photuris, Security Failure

Como de costumbre, la lista completa de números asignados a ICMP es mantenida por el **IANA** (**I**nternet **A**ssigned **N**umbers **A**uthority), y la podéis consultar en:

http://www.iana.org/assignments/icmp-parameters

Buscando huellas

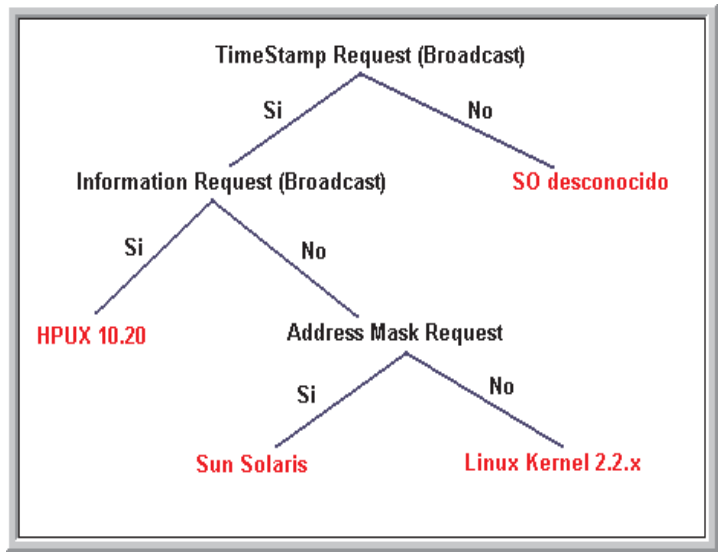
He dejado para el final esta técnica que es en realidad una de las que dan más utilidad a ICMP a la hora de encontrar información sobre un sistema para un posible ataque. Lo he dejado para el final porque es una técnica que está relacionada no con uno sólo, si no con muchos tipos de mensajes ICMP.

La técnica conocida como **OS Fingerprinting** consiste en analizar las reacciones de un sistema ante distintos tipos de mensajes ICMP y, conociendo cual es la reacción de cada sistema operativo conocido, poder realizar así una comparación que nos lleve a deducir qué sistema operativo está corriendo en la máquina que estamos analizando.

Hay una panda de chiflados (con cariño) que se dedican a hacer tablas que reflejan el comportamiento de cada sistema ante cada tipo de mensaje, aunque no llegan a estar tan chiflados como para tratar de usar es información a palo seco... lo que hacen es introducir toda esta información en una aplicación que automatiza la tarea de comparar las reacciones del sistema con la información conocida sobre cada sistema operativo.

La más conocida de esas aplicaciones es **NMap**, de la que ya hemos hablado bastante en la revista. Nmap contiene gran cantidad de información sobre las reacciones de cada SO ante determinados paquetes, y utiliza esta información para hacer un **OS Fingerprinting**, es decir, para deducir cuál es el sistema operativo de la máquina que estamos analizando.

Para que os hagáis una idea del tipo de información de la que estoy hablando, podemos mostrar por ejemplo este árbol:



Aquí vemos que ante un mensaje **ICMP TimeStamp Request** a la dirección **broadcast** de una red, podemos obtener información precisa sobre el sistema operativo, siempre que la máquina responda con un **TimeStamp Reply**.

En caso de que no responda, se considerará (de momento) un **SO desconocido**, y habrá que hacer otras pruebas similares.

Si ha respondido, tenemos 3 posibles SOs: **HPUX 10.20**, **Sun Solaris**, o **Linux Kernel 2.2.x**. En ese caso, habrá que continuar con la prueba, haciendo ahora un **Information Request** a la dirección broadcast. Si responde, ya sabemos que se trata de un **HPUX 10.20**. Si no responde, habrá que continuar probando, esta vez con un **Address Mask Request** (uno de los ICMPs que no hemos visto). Si responde, se trata de un **Sun Solaris**, y si no, de un **Linux Kernel 2.2.x.**

Esta información también se puede mostrar en tablas más complejas, como ésta, que muestra el valor del campo TTL que usa cada sistema operativo al enviar mensajes ICMP de Echo:

Sistema Operativo	TTL en Echo Request	TTL en Echo Reply
Microsoft Windows 2000	128	128
Microsoft Windows 95	32	32
Otros Windows	32	128
Linux Kernel 2.0.x	64	64
Linux Kernel 2.2.x y 2.4.x	64	255
Otros tipo *NIX	255	255

Como ya he dicho, esta información no nos hace falta conocerla, ya que viene automatizada en herramientas como **Nmap**, o **Xprobe**, pero si queréis información detallada y tenéis los c*j*n*s... como los de un toro, podéis leeros la biblia del ICMP, que la tenéis en http://www.sys-security.com/archive/papers/ICMP_Scanning_v3.0.pdf .

3. RAW Sockets ICMP

Para terminar, volvemos con nuestros amigos Nemesis y Hping para jugar esta vez con el protocolo ICMP.

Existen herramientas dedicadas para explotar el protocolo ICMP, como algunas de las ya mencionadas, o la herramienta **SING** (**S**end **I**CMP **N**asty **G**arbage), pero por no romper la tradición vamos a seguir en nuestra línea, utilizando las herramientas ya conocidas.

3.1. Nemesis

Empezamos con Nemesis, cuyas instrucciones tenemos en el archivo **nemesis-icmp.txt** de la carpeta en la que instalamos la aplicación.

Resumo aquí las opciones más básicas:

-i : permite especificar el valor del campo **Type.**
-c : permite especificar el valor del campo **Code.**
-e : permite especificar el valor del campo **Identifier.**
-P : permite especificar un archivo con los **datos,** por ejemplo para un **Echo Request.**
-G : permite especificar el campo **Gateway Address** para los mensajes **Redirect.**
-qE : inyecta un paquete de **Echo.**
-qU : inyecta un paquete **Destination Unreachable.**
-qX : inyecta un paquete **Time Exceeded**.
-qR : inyecta un paquete **Redirect.**
-qT : inyecta un paquete **Timestamp.**

Si, por ejemplo, queremos lanzar un ataque **Super Source Quench** a la IP *192.168.2.2* para cortar su conexión **TCP** con la IP *215.22.69.22*, podríamos hacer:

```
nemesis icmp –i 5 –c 1 –G 127.0.0.1 –v –b 192.168.2.2 –B 215.22.69.22 –D 192.168.2.2 –S 215.22.69.22 –p 6
```

El parámetro **–i 5** especifica que se trata de un mensaje de tipo **Redirect**, que sería como utilizar la opción **–qR**. El parámetro **–c 1** especifica un **Code 1**, es decir, redireccionar los datagramas para el **host** de destino.

El parámetro **–G 127.0.0.1** es la base del ataque **Super Source Quench**, ya que indicamos aquí a la víctima que redirija sus paquetes a la dirección de **loopback**, **127.0.0.1**, es decir, a sí mismo.

El parámetro **–v** es el clásico **verbose** que nos muestra información de lo que estamos haciendo.

El parámetro **–b** no lo he explicado, y forma parte del campo **Internet Header + 64 bits of Original Data Datagram**. Es concretamente la **IP de origen** del

datagrama que generó el supuesto error que dio lugar a que se generase el mensaje **Redirect**.

El parámetro **–B** forma parte también de ese campo, y contiene la dirección **IP de destino** del datagrama original que supuestamente originó el **Redirect.**

Los parámetros **–D** y **–S** ya los conocemos de haber usado otras veces **Nemesis**, y son la **IP de destino y de origen** respectivamente para este mismo paquete.

El parámetro **–p 6** forma parte también del campo **Internet Header + 64 bits of Original Data Datagram**, ya que es el **número de protocolo** de transporte que había especificado en el datagrama original, el cual, al tratarse de **TCP**, será **6**.

Por supuesto, os muestro este ejemplo sólo para mostrar el funcionamiento de Nemesis ICMP, y no garantizo que vaya a funcionar. 🙄

3.2. Hping2

Vamos a ver algunas de las opciones que nos da hping2 para ICMP.
En primer lugar, veamos las **opciones generales**:

--verbose : el clásico modo **verbose** para ver información en pantalla.

--count : permite especificar el **número de paquetes** que queremos enviar.

--traceroute : este es un modo especial que permite hacer un **traceroute.**

Con respecto a la parte **IP** que, por supuesto, también es necesaria para generar paquetes ICMP, tenemos:

--spoof :permite especificar cualquier **IP de origen** (IP spoofing).

--rand-source : permite hacer un IP spoofing con direcciones **IP aleatorias**.

--rand-dest : permite utilizar direcciones **IP de destino aleatorias**. Algo parecido a lo que expliqué que hacían algunos gusanos, aunque en realidad lo normal es que un gusano utilice algún tipo de heurística para generar las IPs de una forma más inteligente que una aleatoriedad pura y dura.

--dont-frag : obliga a que el paquete **no** sea **fragmentado**. Esta opción puede ser útil para realizar un **P-MTU-D.**

--tos : permite definir el **tipo de servicio** que, como hemos visto, está relacionado con algunos mensajes ICMP.

--ttl : permite marcar un **tiempo de vida** para el paquete IP. Como ya sabemos, esto nos permite, entre otras cosas, hacer un **traceroute.**

--tr-keep-ttl : esta opción, combinada con la anterior, permite investigar un router en concreto del camino. Esta opción **fija el valor de ttl**, por lo que podríamos enviar varios paquetes, por ejemplo, al cuarto router del camino haciendo: **--ttl 4 --tr-keep-ttl.**

Por último, algunas de las opciones específicas para **ICMP.** Por supuesto, tenéis todo mucho mejor explicado en **man hping2** :

--icmptype : permite especificar un **Type** en la cabecera ICMP.

--icmpcode : permite especificar un **Code** en la cabecera ICMP.

--icmpipproto : podemos especificar también los valores del campo **Internet**

Header + 64 bits of Original Data Datagram. Por ejemplo, este parámetro es para especificar el **número de protocolo** del datagrama que originó el mensaje ICMP.

--icmpcksum : nos permite generar un **checksum** inválido. Por defecto, si no ponemos esta opción, hping2 generará automáticamente el checksum correcto.

--file : permite especificar un fichero para el campo de **DATOS**.

Vamos a ver un ejemplo sencillísimo, para realizar un ataque **smurf** a la IP *217.138.2.2*, utilizando como amplificadora la red *209.5.x.x* :

```
hping2 209.5.255.255 --icmp --verbose --spoof 217.138.2.2
```

¡Así de simple! Por cierto, que tenéis disponibles listas de redes que han sido probadas y se sabe que pueden funcionar como **amplificadores smurf** (es decir, que responden a paquetes **Echo Request** a la dirección **broadcast**). Podéis encontrar un registro de estas redes por ejemplo en: http://www.powertech.no/smurf/ .

Aunque pueda parecer que este tipo de listas son mantenidas por lamers que dedican su tiempo a ir destruyendo máquinas ajenas, en realidad su cometido suele ser justo el contrario. Se publican estas listas no para ayudar a los atacantes, si no para que la gente que quiera configurar su firewall pueda filtrar directamente todas las redes que sean susceptibles de convertirse en fuentes de un ataque smurf.

Pero claro, en realidad al final el resultado suele ser el contrario del esperado. Está claro que no todo el mundo consulta estas páginas para protegerse, y en cambio basta con que una sola persona consulte esta lista con fines perversos para que pueda atacar a cualquier otra persona que no haya tenido en cuenta la lista. Así que, una vez más, no se sabe si es peor el remedio o la enfermedad.

Espero que este artículo os haya resultado interesante, y que por lo menos haya despertado en vosotros la curiosidad por muchísimas cosas que he esbozado para que vosotros ampliéis información de lo que más os haya llamado la atención.

Por este artículo han circulado muchos nombres que pueden ser fácilmente consultados en Google, así que os animo a que lo hagáis.

Autor: PyC (LCo)