La vulnerabilidad del protocolo TCP sockstress

Por Gabriel.Verdejo.Alvarez@gmail.com - Febrero 2010

Este documento explica y analiza la nueva y "gravísima" [www1][www2][www3] vulnerabilidad del protocolo TCP/IP denominada ***sockstress*** que puede permitir el cese de servicio de cualquier dispositivo conectado a Internet. El escrito se encuentra dividido en dos partes claramente diferenciadas para que el lector pueda acceder rápidamente al contenido que sea de su interés:

- **Análisis técnico**: En la primera parte se realiza un repaso de los conceptos fundamentales de los protocolos TCP/IP así como de sus mecanismos de control. A continuación nos centramos en el análisis detallado del mecanismo que utiliza TCP para el establecimiento de una conexión ya que es a partir de este punto dónde reside la vulnerabilidad.

- **Impacto social**: En la segunda parte de este documento (página 16) se procederá a reconstruir cronológicamente la gestión de la vulnerabilidad *sockstress*. Se detallaran las distintas reacciones sociales que siguieron a su divulgación pública así como la visión de los protagonistas. En la parte final de este escrito (página 21) se analizará la gestión de la seguridad en Internet así como sus tendencias actuales.

El espíritu de este artículo está orientado hacia la divulgación pública de conocimiento por lo que los aspectos técnicos, si bien deberían ser rigurosos, son tratados de forma accesible para cualquier persona con un mínimo de interés. La extensa bibliografía aportada permite al lector, si así lo desea, profundizar hasta el nivel de detalle que desee.

"*Dime y lo olvido, enséñame y lo recuerdo, involúcrame y lo aprendo*" Benjamin Franklin

1. Análisis técnico de la vulnerabilidad sockstress

1.1 Introducción al protocolo IP

La transferencia de datos entre dos sistemas conectados a Internet se realiza de una forma conceptualmente sencilla. Troceamos los datos en bloques de un tamaño determinado, los encapsulamos uno a uno y los enviamos por la red.

El protocolo IP (*Internet Protocol*) [Stevens94][www4][www5][www6] es el encargado de realizar la transferencia de datos a través de Internet. Recoge los fragmentos de datos[1] que le proporciona el sistema y los envía a través de Internet con la esperanza de que alcancen su destino.

[1] A pesar de que estrictamente es cierto que son datos, estos no se corresponden *exactamente* con los que desea enviar el usuario.

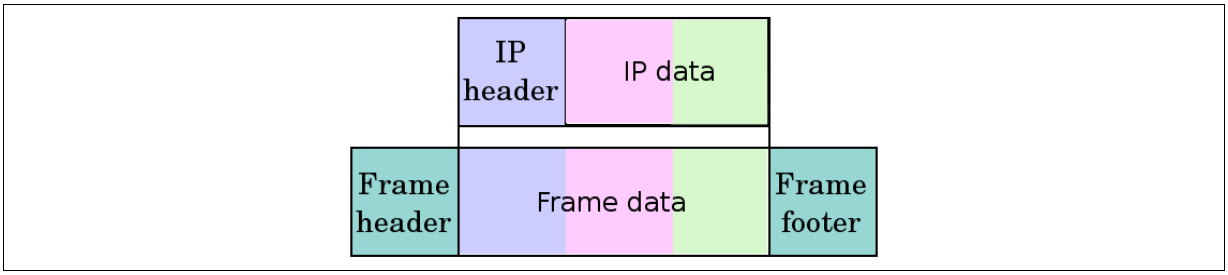


Fig. - 1: Encapsulación de protocolos en Internet [img1].

Para diferenciar los paquetes o datagramas IP que circulan por la red, el protocolo IP al igual que el resto de protocolos de Internet, añade una cabecera (*IP header*) con la información básica necesaria. Esta información hace referencia principalmente a las direcciones de origen y destino así como algunas opciones o *flags* para gobernar el control de la comunicación (ver figura 1).

De esta forma tenemos que el protocolo IP proporciona un servicio de transporte de datos a través de Internet sin conexión ni fiabilidad. No es fiable puesto que no tiene ningún mecanismo que asegure que los datagramas lleguen al destino (no se pide confirmación al destinatario) o verifica que estos se entreguen en orden. De igual forma este protocolo no utiliza ningún mecanismo de establecimiento de conexión para la comunicación con lo que los paquetes IP "fluyen" del origen al destino libremente.

En este punto, probablemente, el lector se pregunte por la utilidad de semejante sistema de "comunicación". La idea que justifica este modelo es el famoso axioma divide y vencerás. De esta forma en vez de crear un único protocolo de transporte complejo y sofisticado, la gente del DARPA [www7] decidió utilizar un sistema incremental dónde unos protocolos "sencillos" ofrecen servicio a otros de forma que cada nivel superior aumenta los servicios ofrecidos.

Modelo OSI		**Pila TCP/IP**	**Protocolos**
Capa de aplicación	(*application*)	Capa de aplicación	HTTP, SSH, IMAP…
Capa de presentación	(*presentation*)		
Capa de sesión	(*session*)		
Capa de transporte	(*transport*)	Capa de transporte	TCP, UDP…
Capa de red	(*network*)	Capa de internet	IP, ICMP…
Capa de enlace	(*link*)	Capa de red	Ethernet…
Capa física	(*physical*)		

Fig. - 2: Pila de protocolos TCP/IP.

Sin entrar en demasiado detalle, de momento, podemos observar que la comunicación en Internet se realiza mediante el uso de varios protocolos, cada uno de los cuales va añadiendo de forma incremental servicios a la capa inmediatamente superior (ver figura 2).

Esta organización, que es una simplificación del modelo OSI[2] [www10], se conoce como pila de protocolos TCP/IP (*TCP/IP stack*) y en los sistemas operativos es la parte encargada de la comunicación vía Internet.

Una vez definido el método de transporte básico para Internet concretaron a su vez una serie de protocolos auxiliares [www11][www12] para dotar de versatilidad a las comunicaciones por Internet. Esta enorme flexibilidad ha contribuido a su rápida expansión y al desarrollo de nuevos servicios. De todos los protocolos existentes, nuestro propósito es el análisis de la vulnerabilidad *sockstress*, nos centraremos exhaustivamente en el que gobierna las comunicaciones fiables y seguras, el protocolo TCP.

1.2 Introducción al protocolo ICMP

El protocolo ICMP (*Internet Control Message Protocol*) [www13] es un protocolo de la capa de Internet encargado de controlar el flujo de las comunicaciones IP y gestionar las anomalías o errores que se produzcan (ver figura 3).

Este mecanismo de comunicación es utilizado tanto por los equipos intermedios por dónde circula el tráfico como por los dos extremos de la comunicación. Los routers intermedios se comunican entre sí para regular los flujos de comunicación de datos entre ellos o para tareas de sincronización entre otras funciones. Los usuarios finales también utilizan este protocolo por ejemplo para el famoso "ping" que permite comprobar la conectividad con el otro extremo de la comunicación.

Como el resto de protocolos, ICMP también utiliza los servicios de transporte del protocolo IP manteniendo la estructura de encapsulamiento que hemos descrito anteriormente. A pesar de no tener un papel activo en la vulnerabilidad *sockstress* es importante conocer su existencia debido a que ya ha protagonizado varios ataques famosos en Internet como el ping de la muerte (*ping of death*) o la inundación de paquetes ICMP (*ICMP flood*) [www14][www15][www16].

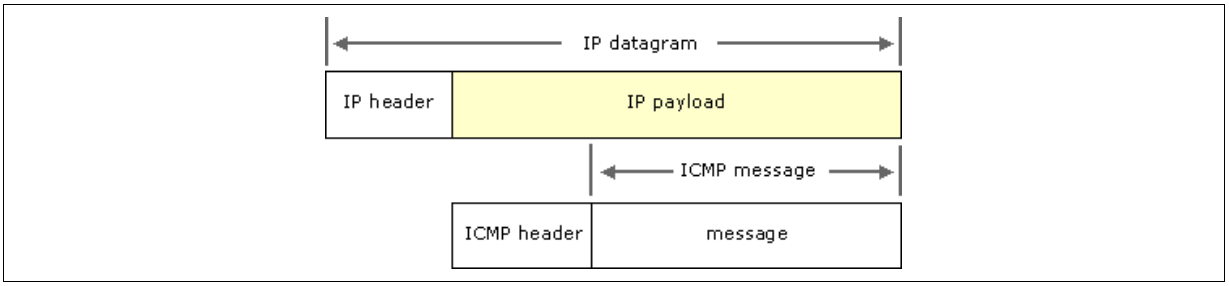


Fig. - 3: El protocolo ICMP [img2].

1.3 Introducción al protocolo UDP

El protocolo UDP (*User Datagram Protocol*) [www17] es un protocolo de comunicación simple, sin estados y orientado a datagrama. De esta forma cada envío de datos se corresponde con un datagrama que tiene un tamaño máximo[3] de 64Kbytes.

[2] Este modelo genérico establecido por la ITU establece el número de capas necesarias para una comunicación a través de redes de datos.
[3] Realmente serian 64Kbytes menos el espacio consumido por las cabeceras de los protocolos.

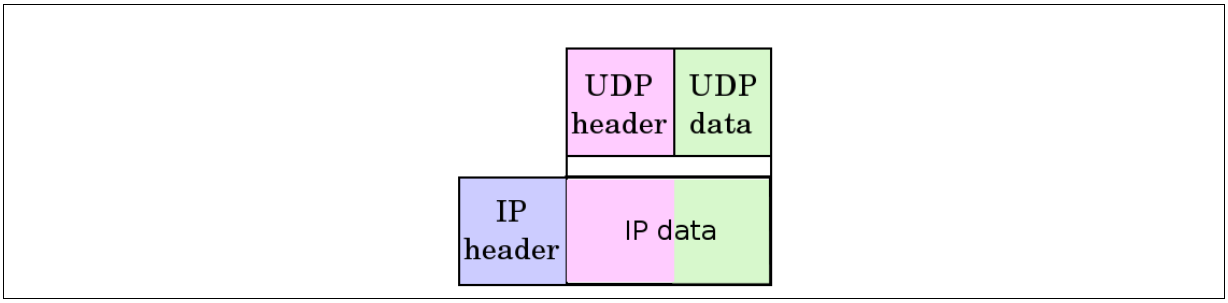


Fig. - 4: El protocolo UDP [img1].

Así mismo podemos observar como el protocolo UDP proporciona un acceso directo a los servicios ofrecidos por el protocolo IP. No realiza ninguna gestión sofisticada del flujo de control, lo que permite que si enviamos varios datagramas estos pueden llegar en orden distinto al que fueron enviados o incluso que algunos "se pierdan". Como es un protocolo fácil de utilizar y que no añade prácticamente *overhead* (sobrecoste de procesar el protocolo) al proceso de la comunicación (ver figura 4).

Aunque pueda parecer que en la práctica este protocolo carecería de sentido, esto no sólo no es así si no que es muy utilizado en aplicaciones de streaming de audio/video o telefonía IP. En estos servicios se busca una gran velocidad de interacción que UDP proporciona perfectamente y no suele ser un gran problema si algún datagrama es descartado por algún router intermedio (por ejemplo por colapso de la línea por exceso de tráfico) o algún paquete llega fuera de orden.

En el caso de desear un sistema fiable de transferencia, por ejemplo si transmitimos ficheros, no podemos asumir que se pierdan datos. Debemos implementar nosotros mismos un mecanismo de control mediante una aplicación propia que utilice UDP como medio de transporte de datos o directamente utilizar un protocolo fiable como TCP.

1.4 Introducción al Protocolo TCP

El protocolo TCP (*Transmission Control Protocol*) [www18] es un protocolo de comunicación orientado a conexión que proporciona un flujo fiable de bytes a través de Internet. Este protocolo se encarga de gestionar los mecanismos necesarios (establecimiento de conexión, transmisión y retransmisión de datos si fuera necesario, control de flujo y finalización de la conexión) para proporcionar un servicio transparente de transporte de datos a través de Internet utilizando el protocolo IP (ver figura 5).

La gestión de la fiabilidad exige una serie de mecanismos "complejos" de control que introducen un sobrecoste en la comunicación. Este *overhead* implica que para cada comunicación TCP y paquete de datos enviado se debe guardar información sobre su estado, de forma que sea posible retransmitirlo si es necesario o descartarlo si llega duplicado o con errores. De esta forma tenemos que no únicamente se consumen recursos "extra" de CPU para procesar las comunicaciones, si no que también se consumen recursos de memoria del sistema. Esta necesidad de recursos es precisamente la característica que explotan los hackers para realizar ataques que en el mejor de los casos paraliza las comunicaciones del servidor y en el peor inutiliza todo el sistema.

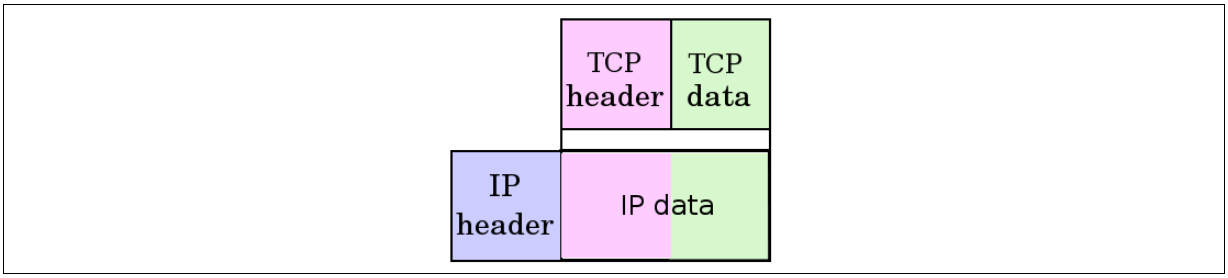


Fig. - 5: El protocolo TCP [img1].

De forma resumida podemos describir el proceso de una comunicación TCP como el compendio de las siguientes acciones:

1. **Establecimiento de conexión:** El emisor envía un paquete inicial señalando su voluntad de establecer una comunicación con el receptor. El receptor puede responder a la petición de forma negativa si no desea establecer ninguna comunicación, ya sea porque tiene todas las conexiones disponibles en uso o no tiene ningún servicio que ofrecer en este momento por ejemplo.

   En el caso habitual de respuesta afirmativa, tanto el emisor como el receptor inician un intercambio de mensajes en el que establecen los parámetros que gobernaran la comunicación. Esta negociación viene determinada por un protocolo de tres pasos (*three way handshake*) [Stevens94][www19][www20] que trataremos con detalle en el siguiente apartado, puesto que es tras este punto dónde se centra el ataque.

2. **Transmisión de datos**: Una vez determinados los parámetros de la comunicación, como por ejemplo la cantidad de datos que se pueden enviar cada vez, se inicia el intercambio de información. El emisor envía una parte de los datos y espera confirmación del receptor.

   Si el receptor recibe correctamente los datos envía una señal de conformidad (*ack o acknowledge*) al emisor especificando explícitamente la cantidad exacta de información que confirma. Si los datos recibidos total o parcialmente fuesen erróneos o presentasen cualquier anomalía, solicitaría la retransmisión de estos confirmando los que ha recibido correctamente.

   Además cuando el emisor o el receptor envían un datagrama por Internet también establecen un sistema de temporizador (*timer*) que en el caso de expirar (*timeout*) permite al emisor o receptor reenviar de forma automática los últimos datos.

3. **Finalización de la transmisión**: Una vez que el emisor ha enviado todos los datos necesarios debe indicar mediante una petición de desconexión su deseo de finalizar la comunicación existente. El receptor, en el caso habitual, responde afirmativamente confirmando la recepción de la solicitud de desconexión. El emisor confirma este mensaje y finaliza la conexión.

   También puede pasar que el receptor desee finalizar la conexión, haya terminado el emisor de enviar sus datos o no. En este caso el inicio de desconexión parte del receptor de los datos y el emisor únicamente debe confirmar la petición recibida.

Cabe destacar que gobernar todos los aspectos de una comunicación fiable, como hemos observado, es un proceso muy complejo debido a la gran cantidad de posibilidades que nos encontramos (ver figura 6).

Por ejemplo podemos encontrarnos en el caso de que al finalizar una conexión la petición de desconexión no llegue nunca a su destino (podría ser descartada por un router intermedio colapsado de trabajo) o que la confirmación final a esta petición sea descartada o transmitida erróneamente. Incluso puede pasar que el ordenador origen o destino se apagase antes de finalizar una comunicación. El mecanismo de los temporizadores permite liberar los recursos asociados a la conexión TCP para evitar el colapso de los sistemas.

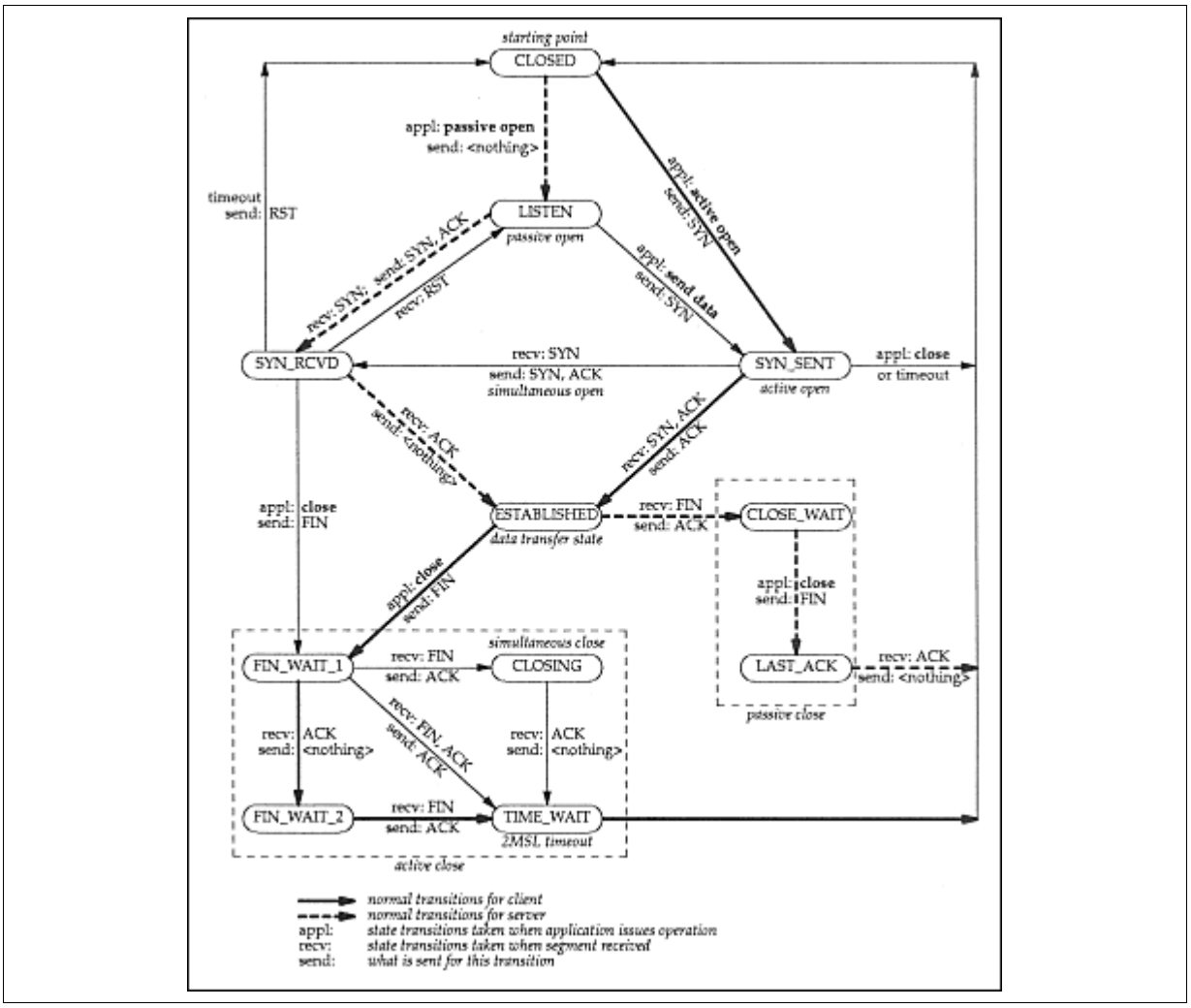


Fig. - 6: Diagrama de estados del protocolo TCP [img3].

Una vez presentados los aspectos fundamentales de las comunicaciones por Internet y presentada la problemática de definir un protocolo fiable y seguro, pasaremos a centrarnos en los aspectos técnicos del ataque *sockstress*. En la segunda parte de este documento se analizaran en profundidad los aspectos sociales derivados de esta vulnerabilidad.

Como ya hemos avanzado este ataque se basa en el consumo de los recursos de red, de forma que el sistema atacado quede en el mejor de los casos sin conectividad a Internet y en el peor se colapse totalmente.

1.4.1 Establecimiento de conexión: Three-way-handshake

El protocolo de los tres pasos [www18][www21][www22][www23] es el mecanismo utilizado en el TCP para el establecimiento de una conexión. Este paso, a diferencia de lo que sucede con UDP, es indispensable para el intercambio de datos y siempre viene iniciado de forma activa. De esta forma es obligatorio que el receptor de la petición de conexión tenga un servicio escuchando ya que de lo contrario el paquete queda descartado por el receptor automáticamente.

Obviamente tanto el emisor como el receptor pueden tener varias comunicaciones simultáneas activas, ya sea entre ellos o con otros servidores. Para diferenciar entre las distintas comunicaciones que puede utilizar a la vez un mismo sistema se asigna un número de puerto (*port*) que va del 1 al 65535 (16 bits). Cabe señalar que los puertos inferiores a 1024 se consideran reservados [www28] o del sistema y están asociados a diferentes servicios de Internet (http en el 80, https en el 443…).

La pareja formada por la dirección IP que identifica únicamente a un sistema[4] y el puerto asociado a la comunicación se denomina *socket*. Por consiguiente tenemos que las comunicaciones entre sistemas conectados a internet se producen entre el *socket* del emisor y el *socket* del receptor.

En el primer paso del establecimiento de una conexión TCP, el cliente (A) selecciona un número aleatorio (j) de secuencia denominado ISS (*Initial Send Sequence*) y envía una solicitud de conexión al servidor (B). Para indicar que esta petición es el primer paso de la conexión se activa la señal o *flag* de sincronización SYN (ver figura 7).

A continuación el servidor (B), asumimos que tiene un servicio escuchando en el socket TCP correspondiente, elige otro número de secuencia aleatorio (t) y responde enviando su número de secuencia y la confirmación (*ACK*) del número de secuencia anterior (j+1). Para indicar que estos datos pertenecen a una solicitud de conexión se activa también el *flag* SYN. Recordemos que el mecanismo de confirmación utilizado en TCP es incremental.

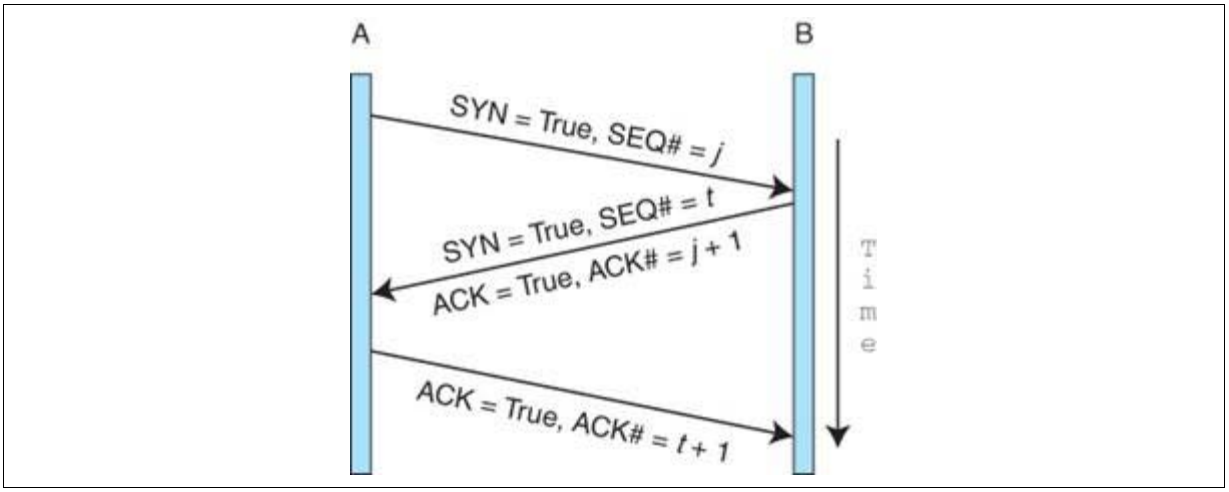


Fig. - 7: Protocolo de tres pasos (*3-way handshake*) [img4].

[4] Esta simplificación es cierta de forma general, aunque hay casos como el uso de *NAT* [www24][www25] y *Proxies*[www26][www27] que requerirían matizar esta afirmación. Dejamos a criterio del lector profundizar en este aspecto puesto que sale del ámbito de este trabajo.

Finalmente el cliente (A) envía la confirmación (*ACK*) del número de secuencia recibido del servidor en el paso anterior (t+1) y finaliza el proceso de establecimiento de conexión TCP.

Cabe destacar la importancia de la aleatoriedad de los números de secuencia, ya que son los que permiten realizar el control del flujo de la comunicación. El número de secuencia es un número de 32 bits, lo que nos permite 4.294.967.296 posibilidades. Como el número de secuencia se utiliza en el establecimiento de conexión y para mantener el flujo de control (cada vez que se envía/recibe un byte se incrementa en uno) es importante asegurarse que la probabilidad de se escojan dos números iguales sea pequeña. Por otro lado, para evitar el problema de la recepción de datos duplicados o secuencias fuera de orden, el número ha de ser suficientemente grande como para no permitir colisiones cuando la secuencia cicle.

La elección de números "aleatorios" para las secuencias de conexión ha sido explotado con éxito en el pasado para permitir el robo de sesiones TCP [www29]. No trataremos esta cuestión con más detalle por alejarse del propósito de este documento. Sin embargo sí citaremos para los lectores que deseen profundizar más en el tema el ejemplo más famoso de este tipo de ataques, el protagonizado por el conocido hacker Kevin Mitnik [www30][www31].

1.4.2 Syncookies

Una vez descrito con detalle el proceso de establecimiento de una conexión TCP, explicaremos un procedimiento ya adoptado por prácticamente todas las implementaciones del protocolo TCP para evitar el consumo de recursos. Como hemos observado en el punto anterior tanto el emisor como el receptor han de guardar una información para mantener el estado de la conexión TCP que consume recursos. Esto es especialmente crítico en los servidores puesto que pueden recibir miles de peticiones de conexión simultáneamente.

El mecanismo denominado *syncookies* [www32][www33][www34] permite retrasar la reserva de recursos de memoria hasta la conclusión total de la conexión. De esta forma evitamos tanto los ataques consistentes en iniciar miles de conexiones y no finalizarlas, como las peticiones fallidas o anómalas recibidas por el servidor. La idea consiste en que el servidor, en vez de seleccionar un número de secuencia "aleatorio" para su respuesta a la petición inicial del cliente, conteste con un número de secuencia compuesto por cierta información escogida cuidadosamente. De esta forma al recibir el último paso de la conexión el servidor puede recuperar todos los datos del protocolo de conexión a partir de este número de secuencia especialmente construido. Finalmente se procede a la reserva de los recursos de esta conexión legítima.

El uso de *syncookies* se realiza únicamente por parte del servidor por lo que es totalmente transparente para el cliente que no percibe ningún cambio en la petición de conexión TCP. El protocolo de los tres pasos con *syncookies* queda de la siguiente forma:

1. El primer paso del establecimiento de la conexión TCP por parte del cliente es idéntico. Éste selecciona un número de secuencia aleatorio y lo envía al servidor (ver figura 7):

   SYN = True, SEQ# = j

2. En el segundo paso, el servidor recibe la petición de conexión y calcula un número de secuencia de la siguiente forma:

   t = (Valor resultante de una función que mire la fecha y hora del sistema) módulo 32

   Generalmente a esta función se le denomina *timestamp* [www35][www36] y es la encargada de generar un número natural a partir del día, mes, año, hora, minuto, segundo actual.

   Se realiza la operación matemática de módulo 32 (2^5) para obtener un número de 5 bits. No discutiremos las matemáticas [www37] asociadas a estas operaciones por estar fuera del propósito de este documento.

   m = Valor del sistema asignado al MSS (*Maximum Segment Size*) codificado en 3 bits.

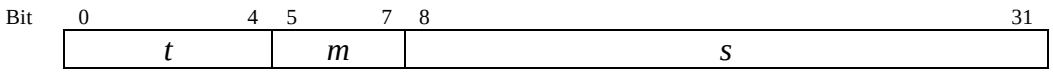
   El MSS [www38] es el valor que establece en la comunicación TCP el tamaño máximo de datos que se aceptará en un único datagrama. Este valor puede ir variando a lo largo de la comunicación.

   Es decir, si yo quiero transmitir 20 bytes y el MSS del receptor es de 5 bytes, puedo enviarle por ejemplo 4 datagramas de 5 bytes o 20 datagramas de 1 byte. Pero no puedo enviarle en ningún momento un datagrama con más de 5 bytes.

   Como este valor es un número de 3 bits (2^3 = 8 posibilidades) y los valores del MSS pueden ser muchos, el sistema que usa *syncookies* suele codificar unos valores por defecto. Por ejemplo:

   0 = MSS de 100, 2 = MSS de 500, 3 = MSS de 1000…… 7= MSS de 1500

   s = Valor resultante de una función criptográfica que incluye los *sockets* del cliente y del servidor, así como el valor de *t*. Esta función crea un número de 24 bits.



Número de secuencia de 32 bits generado mediante el uso de *Syncookies*.

   A continuación realiza el envío de la confirmación de la petición de conexión así como el número de secuencia generado por el sistema de *syncookies*.

   SYN = True, SEQ# = *Syncookie* , ACK = True, ACK# = j+1

3. Finalmente en el tercer paso el cliente envía la confirmación del número de secuencia +1 recibida desde el servidor. Es en este momento cuando el servidor realiza la función inversa recuperando los valores codificados en el paso 2. Se procede entonces a la reserva de los recursos de memoria necesarios y queda establecida la conexión.

   ACK = True, ACK# = *Syncookie+1*

Como hemos podido comprobar, el sistema de *syncookies* permite de una forma fácil y eficiente eliminar los problemas asociados al establecimiento de conexión. La aplicación de este sistema por parte de los servidores, sin embargo, presenta algunos inconvenientes que si bien no son graves sí es importante tener en cuenta.

El número de valores que se pueden negociar para el MSS queda reducido a 8 posibilidades debido a que únicamente se han destinado 3 bits para su codificación. Aunque esto puede afectar en algunos casos al rendimiento, su incidencia real es nula ya que podemos definir unos valores precalculados que proporcionen buenos resultados.

Las opciones extra o especiales del protocolo TCP no pueden utilizarse. Debido a que *syncookies* no guarda el estado parcial de las solicitudes de conexión, estas se pierden en el proceso de los tres pasos. En casos muy concretos estas opciones pueden ser necesarias, pero su impacto al igual que en el caso anterior en prácticamente nulo para casi todas las conexiones de Internet.

Cabe destacar que en la mayoría de sistemas operativos actuales el soporte *syncookies* no se aplica por defecto. Cuando el servidor recibe un número alto de peticiones (depende de cada sistema) este soporte se activa generalmente de forma automática. Al bajar el número de conexiones simultáneas que el servidor mantiene activas, el sistema recupera la gestión "tradicional" de establecimiento de conexiones TCP.

Para finalizar este punto destacaremos la existencia de unas técnicas defensivas destinadas a retrasar la reserva de recursos por parte de los servidores denominadas *Tarpit* [www39][www40]. Estas prácticas, que se pueden aplicar tanto a los protocolos de red como a los servicios de Internet, permiten minimizar el impacto de los ataques de denegación de servicio [www41][www42][www43] destinados a consumir los recursos de los servidores.

1.5 La vulnerabilidad *Sockstress*

Las claves para entender este ataque al protocolo TCP se describen en los puntos anteriores debido a que el objetivo del *sockstress* es consumir los recursos del servidor atacado. Sin embargo, en vez de realizar un enfoque de ataque "clásico" dónde el servidor atacado agota la memoria asociada para gestión de conexiones de red, el atacante intenta consumir todos los temporizadores (*timers)* del sistema.

Ya hemos introducido anteriormente en que momento y por qué el protocolo TCP utiliza este mecanismo. Sin embargo es importante destacar que dentro de un sistema operativo, concretamente en el *kernel* [Tanembaum87][Tanembaum03][www44][www45] [www46], los *timers* son utilizados para múltiples funcionalidades críticas.

Sincronizaciones de partes vitales del sistema (el planificador de procesos, gestión de E/S…) utilizan este mecanismo. Dependiendo de cómo se haya diseñado y programado el *kernel* del sistema, este ataque fulmina todo el sistema operativo y por tanto todos los servicios que proporciona. Debido a que esta parte excede el ámbito del documento nos centraremos únicamente en el ataque al protocolo TCP, queda a criterio del lector profundizar en la bibliografía proporcionada.

Otra característica fundamental del ataque *sockstress* es que se realiza posteriormente a la conexión. Generalmente los ataques al protocolo TCP se centran en intentar falsear o envenenar el protocolo de los tres pasos, ya sea solicitando miles de conexiones que no se finalizan o realizar violaciones del protocolo. De esta forma los mecanismos protectores como *Syncookies* no sirven para evitar este tipo de ataques.

Si tenemos presente el ciclo de vida de una comunicación TCP (ver figura 8) podemos ver claramente diferenciadas las fases que la gobiernan: Establecimiento de conexión o *3 way handshake*, intercambio de datos y finalización de la conexión. Es al iniciar el intercambio de datos dónde se consumen los recursos del servidor, por lo que el atacante necesita establecer varias conexiones completas.

En este punto cabe remarcar, ya que *sockstress* requiere que el atacante mantenga establecidas conexiones activas con el receptor, en que punto se reservan los recursos por parte del emisor y el receptor de la comunicación TCP. El emisor/cliente/atacante reserva los recursos en el primer paso, ya que como inicia él la conexión necesita mantener el estado de su solicitud. El receptor/servidor/atacado reserva los recursos generalmente en el segundo paso y en el caso de utilizar *Syncookies* estos se reservan en el tercero.

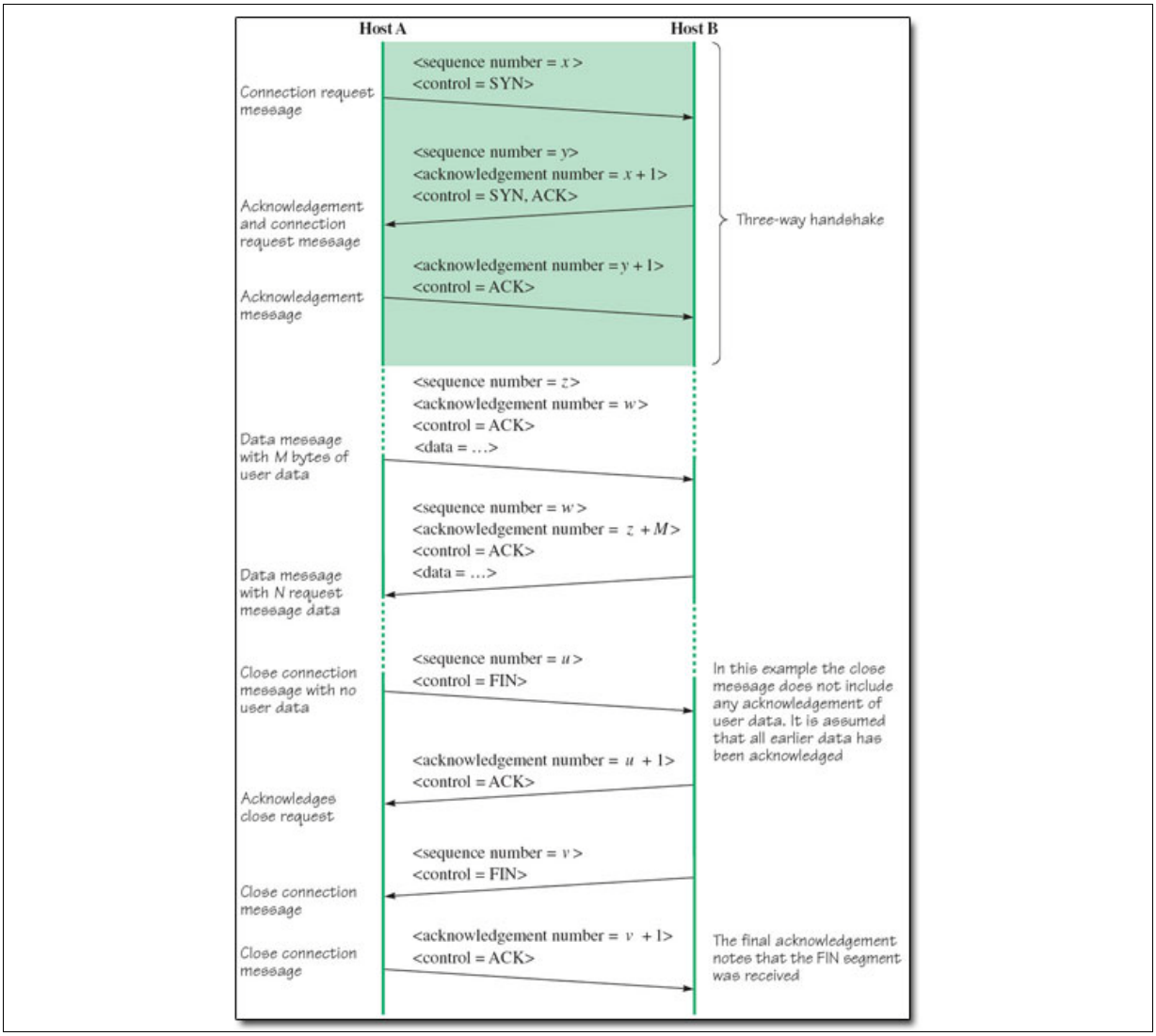


Fig. - 8: Ciclo de vida "típico" de una conexión TCP [img5].

El ataque *sockstress* [www2] fue descubierto inicialmente por Jack C. Louis [www47] en el año 2005 pero no se difundió oficialmente hasta el 20 de Agosto de 2008 junto a Robert E. Lee [www48]. Posteriormente, el 2 de Octubre del mismo año, se publicó como aviso oficial en el CERT de Finlandia [www52][www53]. Una vez presentado en sociedad y tras una demostración práctica, realizaron una famosísima entrevista de radio[5] el día 30 de Septiembre de 2008 [www50][www51] en la que se describía con cierto detalle este ataque captando toda la atención de la comunidad de expertos en seguridad.

Retomando el análisis inicial cabe destacar que es necesario que el atacante deba mantener establecidas varias conexiones que manipulará para consumir los temporizadores del servidor atacado. Por tanto el atacante realiza las conexiones desde su propia IP y no puede falsear la dirección de origen de forma que existe la paradoja de que ¡también consume sus propios *timers*!.

Para evitar el colapso de los recursos propios y cambiar el escenario dónde el atacante únicamente podría colapsar sistemas con menos recursos de temporizadores disponibles que su propio sistema, Louis decidió utilizar el mecanismo de *Syncookies* en el cliente [www77].

De esta forma el atacante no necesita, a diferencia del esquema tradicional de conexión TCP de la figura 8, reservar recursos cada vez que realice conexiones. La idea es que el atacante crea un número de secuencia o *cookie* especial de forma que cuando reciba la contestación del servidor pueda verificar que es correcta –y no está siendo atacado o contraatacado…- y pueda obtener los datos necesarios para mantener la comunicación.

Llegados a este punto es conveniente recapitular para establecer el marco concreto del ataque. En primer lugar seleccionamos un *socket* (dirección IP + puerto de servicio) TCP de un servidor en Internet al que deseamos atacar, por ejemplo el servicio web. También tenemos por otro lado a un atacante que desea establecer varias conexiones TCP para perpetrar un ataque *sockstress* pero sin utilizar los recursos de su ordenador para evitar realizarse un auto-ataque.

La solución pasa porque el atacante utilice un mecanismo denominado *raw-sockets* [Stevens94][www67][www68] que proporcionan de una u otra forma prácticamente todos los sistemas operativos. Esta modalidad de acceso a la red permite a las aplicaciones del usuario crear artesanalmente sus datagramas IP y enviarlos directamente sin utilizar las facilidades TCP o UDP que proporciona el sistema operativo. Consecuentemente el atacante genera una *cookie* para reconocer las respuestas "legítimas", rellena todos los campos de la cabecera TCP y envía directamente el paquete por Internet sin consumir recursos propios.

Si tenemos en cuenta que la comunicación IP se establece entre los *sockets* del emisor y del receptor podemos intuir rápidamente que este sistema presenta un problema. Como el atacante no utiliza una conexión TCP "normal" su sistema no tiene abierto un socket por el cual recibir las respuestas. Por tanto cuando llegue el paquete de respuesta quedará descartado por el sistema operativo. La solución consiste en que el atacante tenga acceso a todo el tráfico que recibe su ordenador utilizando, por ejemplo, alguna regla en el sistema de filtrado de paquetes para que los paquetes no solicitados no sean descartados. Otra opción complementaria es el uso del "*modo promiscuo*" [www69][www70] en las tarjetas de red que permite tener acceso a todo el tráfico recibido, sea o no para nuestro ordenador.

[5] El inicio de la entrevista se realiza en holandés pero a partir de los 5'10'' el idioma es inglés.

Podemos encontrar en Internet diferentes herramientas que nos permiten investigar y simular la vulnerabilidad *sockstress*. El lector que desee profundizar en la práctica puede consultar ejemplos de programación de *raw sockets* en C [www72][www73], programas de inspección de paquetes de red [www74][www75] y herramientas de escaneo de puertos [www71][www76].

Con el uso de las técnicas anteriormente descritas se puede realizar el ataque sin peligro de colapsar el propio sistema y prácticamente sin coste alguno. Según la entrevista radiofónica que concedieron en el 2008 en pocos segundos y ¡con menos de 40 paquetes por segundo pudieron colapsar cualquier sistema que probaron! (Windows, Linux, MacOs. *BSD,…). Teniendo en cuenta que un datagrama tiene un máximo de 64k y un mínimo de 64 bytes aproximadamente por las cabeceras, podemos calcular que un atacante necesita entre 2.5Kbytes y 20Mbytes por segundo, velocidades que actualmente ya se superan en muchos hogares.

Datagrama de 64Kbytes * 40 paquetes/s = 2560Kbytes = 2.5Mbytes/s = 20 Mbits
Datagrama de 64bytes * 40 paquetes/s = 2560 bytes = 2.5Kbytes/s = 20 Kbits

La idea fundamental detrás del ataque *sockstress* consiste en bloquear para cada conexión existente parte de los recursos de los temporizadores. Como este mecanismo principalmente se utiliza cuando la conexión experimenta problemas, el objetivo es simular una comunicación dónde se pierden gran cantidad de paquetes, obligando a sobrecargar el mecanismo de retransmisión y control del servidor.

A pesar de que desde el punto de vista del protocolo IP se pueden enviar datagramas de hasta 64Kbytes, estos no suelen utilizarse a menudo ya que generalmente las redes intermedias tienen tamaños de paquetes inferiores. Esto obliga a que un datagrama sea fragmentado en trozos más pequeños que deben ser reensamblados en su destino final. Por ejemplo en el caso de las redes Ethernet [www54][www55] el tamaño máximo de datos que puede transmitir son 1500bytes, incluyendo todas las cabeceras.

Cada uno de estos fragmentos viaja independientemente con lo que es un punto crítico ya que si es descartado o se entrega con problemas, invalida todo el datagrama al no poder construirse. Mecanismos como el *MSS* del TCP permiten ajustar los valores de la transmisión para optimizar el intercambio de datos. Como es de esperar la fragmentación de datagramas IP es muy golosa y ha sido utilizada para distintos ataques de denegación de servicio. Para profundizar en este tema se puede consultar la bibliografía adjunta [www56][www57] [www58].

En el caso que la comunicación TCP entre el emisor y receptor presente problemas en cualquiera de los puntos intermedios del camino, supongamos un escenario dónde hay mucho tráfico de la red, el mecanismo de los temporizadores se encarga de solicitar los datos "perdidos" mientras mantiene la conexión abierta, y por tanto consumiendo recursos.

Un ejemplo legítimo de sobrecarga de red es cuando se producen noticias importantes que llegan incluso a colapsar los servicios con peticiones de usuarios reales. Por ejemplo las noticias del 11-S o cuando Microsoft publica un nuevo *ServicePack*. El tráfico de la red se incrementa notablemente debido a los millones de peticiones de usuarios.

Aprovechando esta funcionalidad definida en el protocolo TCP el atacante modifica los parámetros de la comunicación y mantiene la conexión "ocupada" simulando pérdida de paquetes en la red. A su vez juega con los valores de la ventana de transmisión fijando valores que impiden una comunicación fluida.

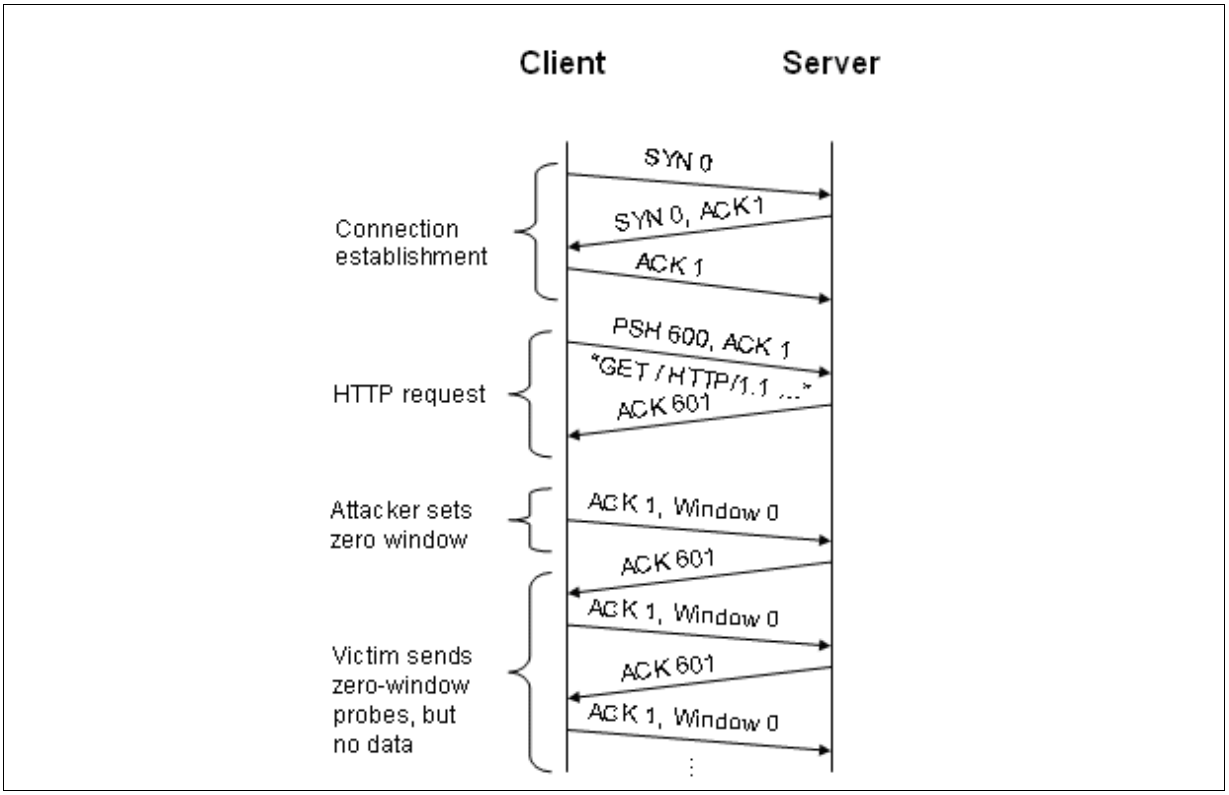


Fig. - 9: Ejemplo de escenario de ataque *sockstress* [img6].

Según comentaron en la entrevista radiofónica existen cinco escenarios de *sockstress* distintos que obtenemos jugando con los parámetros dentro del mismo tipo de ataque. Un ejemplo de estos ataques (ver figura 9) se produce estableciendo una ventana de intercambio con valor 0, lo que impide el intercambio real de datos.

Otro punto interesante a destacar es la posibilidad de maximizar este ataque usando varias direcciones en un mismo ataque, por ejemplo mediante el uso de ordenadores *zombies* o *botnets* [www59][www60][www61]. De esta forma si los temporizadores se agrupan por la dirección de origen de la conexión obligamos al consumo de más recursos extra para mantener el control de las conexiones TCP existentes en el sistema atacado.

Como ya hemos visto, no es necesario el uso de una gran cantidad de ancho de banda para perpetrar este tipo de ataques. En el caso de tener varias máquinas bajo nuestro control, la utilidad *fantaip* de la suite *unicornscan* nos lo permite, podemos simultáneamente realizar miles de ataques a servidores lo que dado la efectividad del *sockstress* pudo ser una catástrofe sin precedentes en Internet.

Finalmente remarcar en este punto que este tipo de ataque es independiente del sistema operativo, arquitectura y plataforma. De esta forma cualquier dispositivo conectado a Internet (desde servidores hasta *smartphones*) que permitan conexiones TCP es susceptible de ser atacado.

1.5.1 *Sockstress* en IPv6

El protocolo IPv6 [www62][www63][www64] es el reemplazo natural del protocolo IP que utilizamos actualmente[6] en Internet denominado IPv4. Como ya hemos visto anteriormente el protocolo IP es el encargado de proporcionar el mecanismo básico de transporte en Internet. Fue desarrollado principalmente en la década de los 70, publicándose como RFC en 1980 [www65], y se actualizó con unos ligeros cambios en 1981 [www66].

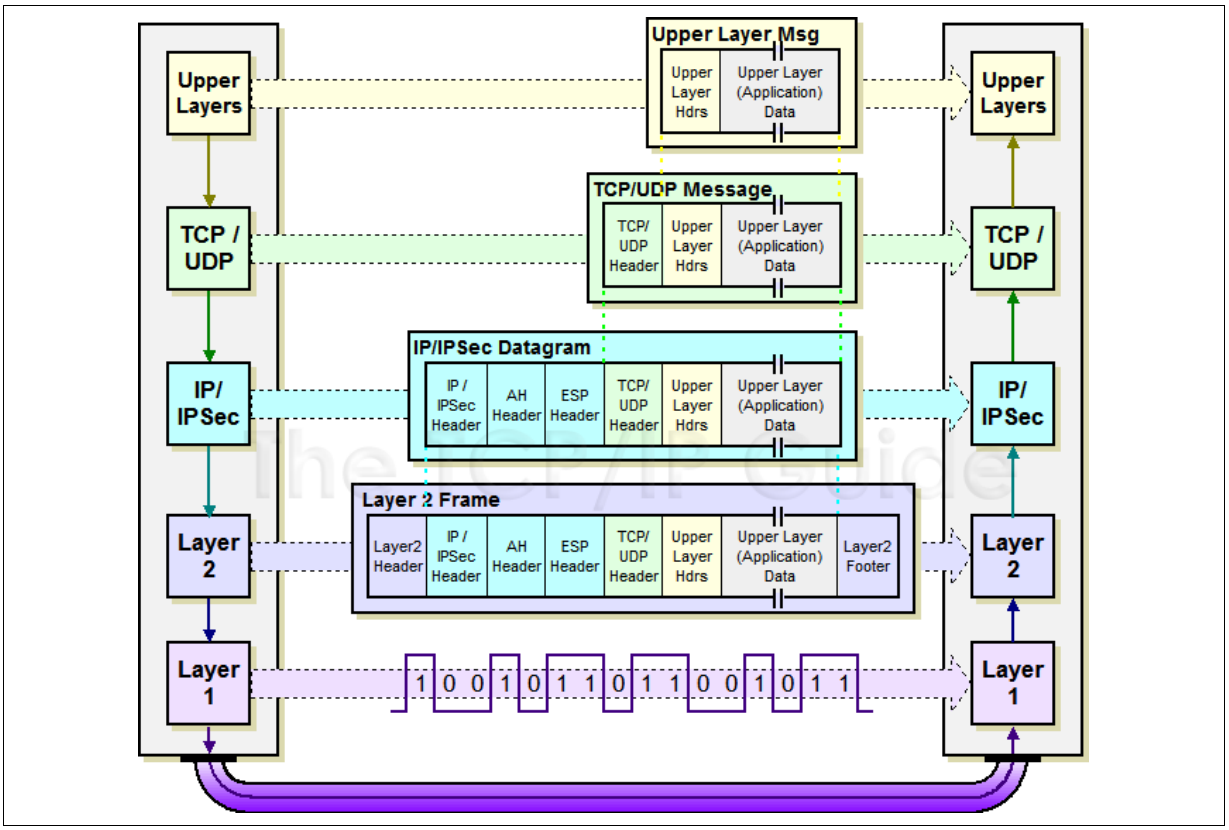


Fig. - 10: Esquema del IPv6 [img7].

El desarrollo de Internet y las necesidades actuales han dejado obsoletas muchas de las previsiones realizadas para IPv4. Además, la aparición de nuevos servicios y expectativas de crecimiento futuras tampoco son soportadas en la versión actual. Podemos citar los problemas de direcciones IP disponibles, 32 bits en el caso de IPv4 y 128 bits en el caso de IPv6, o la falta de mecanismos de seguridad (denominadas IPSec en IPv6) como puntos fuertes que justifican esta nueva revisión del protocolo IP.

Debido a que *sockstress* es un tipo de ataque orientado directamente a la naturaleza del propio TCP, la substitución del protocolo IP no elimina la problemática anterior. Es más, en la entrevista radiofónica comentaron que debido a la complejidad del IPv6 (ver figura 7) y sus nuevas funcionalidades, los recursos necesarios para implementarlo son muy superiores a los de IPv4. De esta forma las posibilidades de explotar este tipo de ataques al propio protocolo TCP6 aumentan enormemente.

[6] Las estimaciones no parecen sugerir que substituirá a IPv4 antes del año 2011.

2. Impacto social

2.1 Cronología de la divulgación de la vulnerabilidad *sockstress*

En este apartado realizaremos una reconstrucción tan fiel como sea posible de la secuencia temporal que desencadenó la divulgación pública de *sockstress*. Iniciaremos el recorrido en el año 2005 cuando Jack C. Louis intuye esta vulnerabilidad. Rápidamente pasaremos al momento del primer aviso realizado por Robert E. Lee en Agosto de 2008 y su confirmación por el CERT-fi. Detallaremos la secuencia de acontecimientos durante el año 2009 y finalizaremos el recorrido en el momento de la publicación de este documento. Cabe destacar y agradecer la ayuda prestada por varios de los protagonistas como Robert, Fyodor, Jason Ross o Dan Kaminsky que muy amablemente contestaron todas mis consultas y emails.

En el año 2003 Robert funda una pequeña empresa de seguridad informática denominada Dyad. Poco tiempo después Jack decide unirse a Robert debido a su afinidad por los temas de seguridad en redes, lo que le permitiría trabajar en una de sus grandes pasiones. Fruto de esta colaboración nació la utilidad *unicornscan* [www71] que permitía a Jack analizar distintos aspectos de configuración en redes IP. Un tiempo después la empresa fue adquirida por Outpost24 para completar su catálogo de servicios en seguridad.

A inicios de 2005 Jack ya había observado comportamientos extraños utilizando configuraciones concretas de su herramienta *unicorn*. Para profundizar en este aspecto decide desarrollar una herramienta denominada *sockstress* que se encarga de realizar conexiones para verificar los servicios existentes en servidores. Tras varias pruebas llega a la conclusión de que en determinadas circunstancias los temporizadores de los sistemas operativos pueden colapsarse al recibir varias conexiones TCP.

Pese a ser conscientes del problema, comenta Robert, prefirieron no divulgar en ese momento 'su descubrimiento' hasta analizar en profundidad el tema y decidir una estrategia común. Este ataque afecta a toda Internet y por tanto decidieron ir con pies de plomo hasta estar realmente seguros de todos los aspectos técnicos y sus implicaciones. Como no había ninguna urgencia y el trabajo del día a día muchas veces nos desborda, este tema quedó latente unos años si bien intermitentemente seguían realizando pruebas y optimizando la herramienta *sockstress*.

El día 20 de Agosto de 2008 Robert, director del área de seguridad de la empresa Outpost24, anuncia [www85] la presentación pública de la vulnerabilidad *sockstress* en la conferencia de seguridad en tecnologías de la información Sec-T [www86] de Estocolmo. Sorprendentemente este anuncio tiene nula repercusión en los medios especializados y pasa prácticamente desapercibida. Robert comenta que durante las semanas posteriores a este anuncio ningún fabricante, programador o periodista contactó con él.

El día 11 de Septiembre, casualidad de fecha o no, se realizó una presentación mediante transparencias [www87][www88] y una demostración en vivo del colapso de los servicios TCP [www89]. Pese a no revelar aspectos técnicos importantes cabe destacar que utilizaron una herramienta que ya tenían totalmente desarrollada para realizar los diferentes ataques de forma sencilla. Pese a la prueba evidente de su efectividad en la demostración realizada, únicamente un grupo bastante reducido de expertos que asistían a la conferencia se interesaron por este tema.

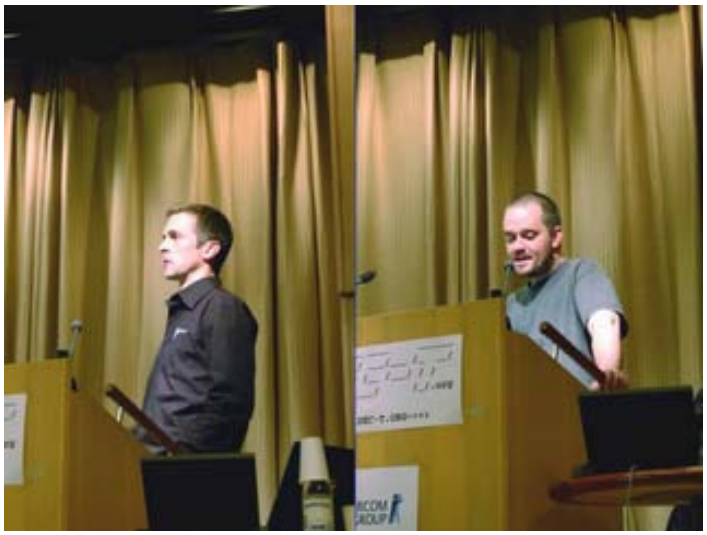


Fig. - 11: Imagen de la conferencia SEC-T del 11 de Septiembre de 2008 [img8].

El 12 de Septiembre tras la presentación en sociedad de la vulnerabilidad *sockstress*, Robert recoge en su blog las primeras reacciones de la comunidad [www90]. Es a partir de este momento cuando la comunidad de expertos en seguridad informática empiezan a especular con los posibles detalles técnicos y el alcance real de esta vulnerabilidad. Las webs y foros de "hackers" reales e imaginarios (*script kiddies*) genera cientos de entradas con comentarios de todo tipo intentando ser los primeros en adivinar los detalles hasta el momento secretos del nuevo ataque definitivo al TCP (*TCP killer*).

En esas mismas fechas Jack y Robert tuvieron contacto con el experto en seguridad Felix Lindner (conocido como FX) y Dan Kaminsky [www106]. En estas conversaciones informales discutieron varios aspectos técnicos sobre el alcance de esta vulnerabilidad y la gestión de la divulgación de la información.

El 30 de Septiembre de 2008 Brenno de Winter [www91] realizó una entrevista a Robert y Jack [www92] que cambiará definitivamente el curso de los acontecimientos. En esta interesantísima entrevista cuyo *podcast* se encuentra disponible [www50][www51] el ataque *sockstress* queda totalmente enmarcado dejando visibles, a ojos de experto, varios aspectos concretos de esta vulnerabilidad. Es realmente en este punto dónde toda Internet, y no únicamente la comunidad de expertos en seguridad, se hace eco de este peligro [www93].

Robert comentó posteriormente que la idea de la entrevista con Brenno surgió espontáneamente tras un viaje a Ámsterdam. En ningún momento fueron conscientes de que el *podcast* se haría público. Creyeron que se trataba de una grabación informal que se realizaba con el objetivo de tomar notas para un artículo. La transcripción de la entrevista, curiosamente, no se encuentra disponible actualmente [www108].

El 1 de Octubre de 2008 la noticia salta a la portada del popular portal de noticias *slashdot* [www94] y miles de páginas webs y foros de todo el mundo la reproducen. A partir de este momento el nivel de paranoia aumenta considerablemente entre la comunidad de expertos en seguridad y aparecen cientos de "consejos" basados en especulaciones destinados a minimizar el impacto de un posible ataque [www79][www95].

Ante la magnitud alcanzada por los acontecimientos Robert se ve obligado a realizar una aclaración en su blog personal [www96] para reconducir el tema y evitar echar más leña al fuego. Por otro lado, a estas alturas, ni las administraciones públicas ni los fabricantes directamente afectados habían contactado con Robert o Jack.

El 2 de Octubre de 2008 se desencadenan finalmente los anuncios oficiales en el CERT de Finlandia Fi-Cert [www52][www97][www98] y en la empresa Outpost24 dónde trabajan Robert y Jack [www47]. Este mismo día se inicia una agria discusión entre el famoso experto en seguridad informática Gordon Lyon, conocido como Fyodor y autor de la famosísima herramienta *nmap* [www100], y los padres del *sockstress*. En un artículo actualmente retirado por el mismo Gordon[7] [www101] *Fyodor* especulaba sobre esta nueva vulnerabilidad y su posible importancia. Ese mismo día Robert le contestaba sobre sus conjeturas en su blog [www103].

En esta misma fecha empieza el goteo de artículos de opinión [www104][www105] [www106] dónde se cuestiona la importancia y la gestión en la divulgación de *sockstress*, realizándose paralelismos con la divulgación del fallo de seguridad en el servicio de DNS que anunció hace dos años Dan Kaminsky [www106][www107]. La gestión de la divulgación de vulnerabilidades será analizada en profundidad más adelante en el punto 2.2.

El 5 de Octubre, tras varios días de intensas discusiones en Internet sobre cómo están gestionando la divulgación de información Robert y Jack, parece imponerse la opinión de que esta gestión está siendo como mínimo irresponsable [www2][www102]. En respuesta a las numerosas críticas recibidas, Robert escribe en su blog [www110] explicando los motivos por los cuales decidieron conceder la entrevista que posteriormente se distribuyó como *podcast*.

Los días van transcurriendo entre miles de entradas, comentarios y artículos de opinión. Sin embargo se aprecia claramente como la discusión técnica queda arrinconada para centrarse en la gestión de la vulnerabilidad. El eterno debate sobre la divulgación total, parcial o responsable vuelve a incendiar Internet. Robert reconoció posteriormente que la idea era trabajar inicialmente con los fabricantes para buscar una solución y posteriormente tras publicar las actualizaciones publicitar los detalles técnicos. Sin embargo los proveedores, organismos oficiales y fabricantes seguían en este punto ignorando el potencial peligro.

El día 9 de Octubre Robert concede una entrevista [www111] a la revista online *SCMagazine*, el podcast está disponible en [www113], dónde comenta brevemente todo los aspectos surgidos alrededor de *sockstress* y procura tranquilizar a la comunidad de usuarios de Internet. Incluso podemos encontrar referencias a esta vulnerabilidad en diarios españoles como El Mundo [www80].

El 16 de Octubre de 2008 aparece una nueva web que responde a la inquietante dirección de http://www.sockstress.com [www112] que obligó a Robert a realizar en su blog una aclaración indicando que no tenían relación alguna. Por estas mismas fechas dos empresas contactaron con él para pedirle explicaciones sobre la web, y aunque intentó hablar con la persona que registró este dominio no hubo manera de conseguirlo. Varios meses después, finalmente, pudo comunicarse con el propietario del dominio y constató que no había malas intenciones tras esta propuesta.

[7] Pese a no estar disponible el articulo original podemos encontrar citas y referencias en [www102]

Robert comentó al propietario del dominio que finalmente no se publicaría el código de la herramienta *sockstress* y por tanto éste desapareció en Enero de 2010. Como dato anecdótico cabe destacar por esas fechas incluso la aparición de ofertas de "trabajo" que solicitaban programadores para realizar una herramienta que implementara este ataque por menos de 250$ [www117].

El 17 de Octubre Jack y Robert realizan una nueva presentación sobre *sockstress* en la conferencia de seguridad T2 en Finlandia [www118]. A estas alturas la expectación de esta presentación se centró en lo efectivo que era el ataque y la demostración práctica utilizando ordenadores portátiles que colapsaba varios sistemas operativos distintos. Cabe destacar también que en esta misma fecha CISCO publicó su respuesta oficial a esta vulnerabilidad [www19].

Una vez más sorprende el tiempo de reacción de los fabricantes ante determinadas amenazas. Hemos indicado el ejemplo de CISCO por ser una empresa suficientemente importante como para servir de referencia ya que gran parte de su catálogo de productos dependen directamente de la seguridad en las redes IP.

Fig. - 12: Imagen de la conferencia T2 del 17 de Octubre de 2008 [img9].

El día 20 de Octubre de 2008 se producen varios acontecimientos importantes relativos a la distribución por canales oficiales. El *National Vulnerability Database*, correspondiente al CERT norteamericano, publica el anuncio de esta vulnerabilidad así como su impacto en las diferentes plataformas existentes [www53]. Debido al peso que este organismo tiene en los fabricantes de software y hardware podríamos señalar este punto como el de la confirmación oficial de la vulnerabilidad *sockstress*.

A pesar de haber sido ya recogido por otros CERT anteriormente no cabe duda, desgraciadamente, que el impacto en las grandes compañías y organismos oficiales de tecnologías de la información había sido pobre. Este aviso obliga a las compañías que quieran seguir prestando servicio en Estados Unidos a asegurar que sus productos no son vulnerables a este ataque, lo que precipita las actualizaciones y parches pertinentes.

Robert realiza una entrada en su blog ese mismo día comentando sus impresiones tras la presentación en la conferencia de seguridad T2 [www121]. Como suele ocurrir en estos casos el interés viene centrado en la demostración que sigue siendo exitosa contra todos los sistemas operativos. Por otro lado la discusión entre Fyodor, Jack y Robert vuelve a enmarañarse produciendo una vuelta de tuerca más que finaliza con unas aclaraciones que realiza Robert en su blog [www120].

Durante este periodo Jack visitó los Estados Unidos para trabajar con los ingenieros de CISCO y de varios CERT. El objetivo era reproducir el ataque en entornos controlados para analizarlo y tratar de encontrar las mejores soluciones o minimizar su impacto.

A partir de este punto la repercusión pública de la vulnerabilidad *sockstress* decae exponencialmente. Los distintos fabricantes van sacando sus actualizaciones con cuentagotas y los expertos en seguridad se centran en otras vulnerabilidades, virus y *exploits*. El periodo vacacional asociado a las navidades también ayuda a correr un tupido velo de indiferencia que termina por relegar esta vulnerabilidad al armario de grandes catástrofes de Internet.

Finalizaremos el recorrido cronológico de esta vulnerabilidad el día 9 de Septiembre de 2009. En esta fecha se realizan las actualizaciones en los avisos de los distintos CERT con las soluciones que han proporcionado los distintos fabricantes finalizando el ciclo de vida oficial de la vulnerabilidad *sockstress* [www52][www53][www78][www129].

Transcurrido más de un año desde el anuncio realizado por Jack y Robert, queda el gusto agridulce de cómo pudo haber sido y cómo fue. Si bien Robert asegura que mayoritariamente repetiría los mismos pasos, eran los mejores que se podían realizar en esas circunstancias, reconoce que los primeros meses fueron muy duros. Ahora Jack y Robert gozan de una reputación y conocimiento del entorno que –reconoce Robert– mejoraría significativamente la gestión de una nueva vulnerabilidad.

Fig. - 13: Jack C. Louis, 1977-2009 [img10].

Jack C. Louis falleció en un accidente doméstico el 15 de Marzo de 2009. Desde estas líneas nos sumamos a las numerosas muestras en su recuerdo [www83][www122][www123].

2.2 La gestión de vulnerabilidades

En este apartado se reflexionará sobre la gestión de vulnerabilidades desde un punto de vista más filosófico que técnico. Por otro lado y para no aburrir con más bibliografía puesto que en los apartados anteriores ya existen numerosas referencias, únicamente señalaremos a los exponentes más importantes de las diferentes corrientes.

De forma rápida y sencilla podemos definir que la seguridad en la información y los servicios viene determinada por tres aspectos: confidencialidad, integridad y disponibilidad (ver figura 14). Generalmente este triunvirato se referencia con sus siglas en inglés "CIA" o como la triada de la seguridad de la información [www126][www127][www128].

- **Confidencialidad** (*Confidentiality*): La información (al igual que los servicios) ha ser accesible únicamente a las personas, entidades o servicios debidamente autorizados y en el grado adecuado a su acreditación. Es decir, una persona autorizada debe tener acceso a la información que necesita y no a más (ni a menos).

  Este aspecto de la seguridad es el encargado de cubrir los accesos no autorizados a la información y proteger el sistema de divulgaciones internas o externas.

- **Integridad** (*Integrity*): La información, al igual que los servicios, debe ser confiables e íntegra. De esta forma la información debe protegerse tanto de los cambios y modificaciones realizados por agentes no autorizados como por los realizados accidental o deliberadamente por fuentes autorizadas. Además la información ha de ser confiable, de forma que existan procesos de validación de esta.

  La integridad de la información extiende la protección más lejos de un "simple" control de modificaciones. La confiabilidad en la información significa que nos fiamos de nuestros datos, algo que no es tan obvio como parece. Si el soporte informático corrompe los datos es un fallo de integridad. Además si por ejemplo tenemos una base de datos con direcciones de clientes y muchos tienen un código postal erróneo o incompleto, ¡es un fallo de integridad!.

  Disponibilidad (*Availability*): Los servicios y la información deben estar disponibles cuando son necesarios. Este aspecto hace referencia al acceso genérico, no a los requisitos, sistemas o mecanismos que puedan implementarlo (alta disponibilidad, redundancia…).

  Los ataques de denegación de servicio (DOS) [www6], como sucede en el caso de la vulnerabilidad *sockstress*, se centran únicamente en este punto. Un caso atípico pero muy ilustrativo de fallo en la disponibilidad se produjo el año 2002 en Noruega. En un centro cultural falleció el investigador jefe encargado de la custodia de la base de datos y con él la clave de acceso [www129][www130]. Esta base de datos contenía más de 11.000 libros y manuscritos originales con los fondos culturales más importantes de la institución.

  Como si de una película se tratase, los responsables pusieron la base de datos a disposición de toda la Internet solicitando su ayuda. Finalmente Joachim Eriksson, un programador de una empresa de videojuegos, adivinó el password.

Pese a la obviedad de su importancia este requisito suele ser el menos apreciado. Los escándalos por fallos en la confidencialidad o integridad en los datos o servicios son muy significativos, mientras que los problemas de disponibilidad de estos parecen menos importantes. Tal vez psicológicamente toleramos mejor no poder acceder a nuestra información cuando deberíamos que el hecho de que haya sido comprometida o alterada de alguna forma.

Seguramente la creencia inconsciente de que aún está ahí nos tranquiliza, puesto que es únicamente cuestión de tiempo conseguir el acceso. Por otro lado, generalmente cuando la amenaza o circunstancia que imposibilita el acceso a nuestra información desaparece recuperamos la normalidad, a diferencia de los fallos de integridad o confidencialidad. No podemos conseguir que la gente olvide algo que ha visto.

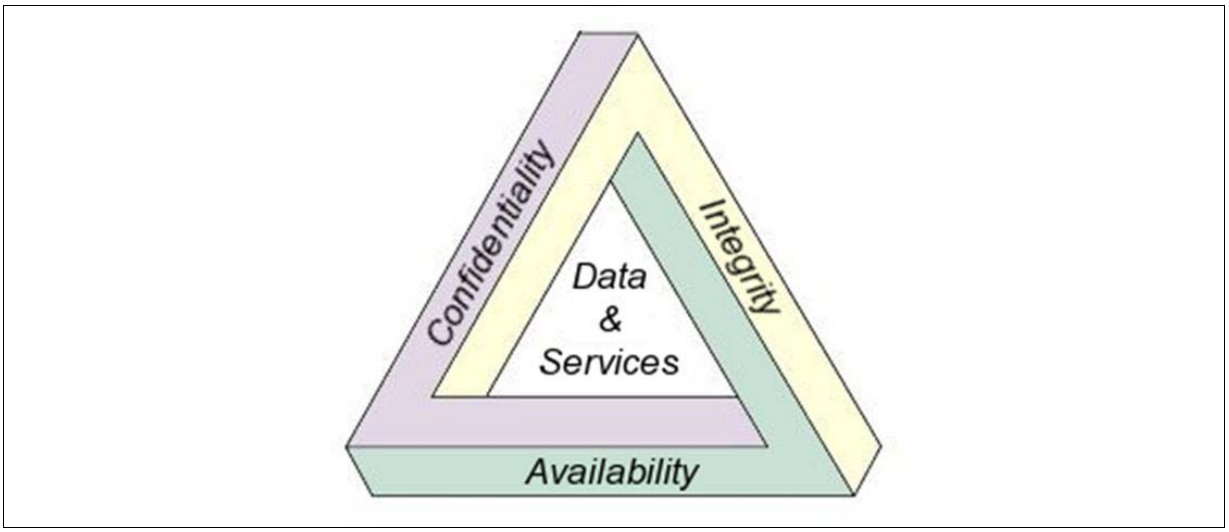


Fig. - 14: Gráfico de la seguridad en la información [img11].

Pese a que el triunvirato CIA de la seguridad informática es un modelo comúnmente aceptado en las tecnologías de la información y comunicación (TIC), no es el único. Se han planteado diferentes alternativas, como el hexágono de Donn B. Parker (*Parkerian hexad*) [www131][www132], que añade otros conceptos como los de posesión de control o utilidad.

Pese al gran esfuerzo realizado por las instituciones gubernamentales, los fabricantes o desarrolladores y los expertos en seguridad TIC, la gestión de vulnerabilidades sigue sin estar a la altura de las necesidades. En este documento ya hemos analizado la secuencia de acontecimientos seguida en el caso *sockstress* y una vez más, la gestión ha sido como mínimo muy discutible.

La falta de criterios comunes y procedimientos estándar para gestionar las vulnerabilidades en los productos y sistemas sigue siendo una constante. Ciertamente el carácter universal del internet dificulta la implantación de metodologías comunes y la creación de un marco legislativo. Sin embargo en otros aspectos sociales sí ha sido posible.

Un ejemplo de estos intentos fue la propuesta denominada "*Responsible Vulnerability Disclosure Process*" [www133] presentada en año 2002 por Steven Christey y Chris Wysopal al IETF. Tras un cierto debate en las listas no se alcanzó ningún consenso, por lo que con el paso del tiempo este documento caducó [www134].

La opinión mayoritaria de los expertos en seguridad, al contrario que en el caso de la vulnerabilidad del sistema de nombres DNS [www106], es que Jack y Robert no gestionaron adecuadamente este tema. Ya hemos visto en el apartado anterior cómo realizaron inicialmente una presentación que incluyó una demostración contra varios sistemas que fueron colapsados, asegurando que afectaba a cualquier sistema operativo existente. Gran parte de la comunidad, Dan Kaminsky entre ellos, consideró este exhibicionismo gratuito y alarmista, generando cientos de comentarios y artículos con feroces críticas.

Por otro lado la reacción de los fabricantes, incluso después de ser publicada en el CERT de Finlandia, tampoco ha estado a la altura. Ciertamente las grandes empresas han de evaluar el impacto y coste que supone la creación del parche o actualización que lo subsane, pero parece que los ataques de denegación de servicio no se encuentran entre sus prioridades.

El debate existente desde hace ya muchos años queda enmarcado en dos posiciones encontradas que son irreconciliables. La divulgación total (*full disclosure*) que promulga compartir el conocimiento con toda la comunidad para buscar soluciones inmediatamente y la divulgación responsable (*responsible disclosure*) que implica proporcionar un periodo de gracia al fabricante antes de publicitarla.

La **divulgación responsable** [www135][www136][www137] aboga por establecer un control estricto sobre el flujo de información que se genera ante el descubrimiento de nuevas vulnerabilidades o fallos en los programas y servicios. Es imposible controlar en que momento o quien puede descubrir un fallo de seguridad, por lo tanto regular las acciones que puede realizar permite salvaguardar al sistema y a los clientes. Este control se justifica por la necesidad de evitar chantajes o usos ilícitos de esta información por parte de los descubridores de la vulnerabilidad.

Este modelo de gestión topa con grandes problemas reales puesto que Internet no conoce fronteras y las diferentes legislaciones locales pueden limitar las posibles acciones que podemos realizar. Por otro lado pese a la existencia de distintos CERT en prácticamente todos los países, los protocolos de comunicación y gestión de vulnerabilidades dejan mucho que desear. Los manuales de buenas prácticas establecen que una vez comunicado al fabricante este debe proporcionar una solución en 30 días. Sin embargo ningún mecanismo obliga a las empresas a proporcionar una solución, cada compañía es libre de considerar importante o no cualquier vulnerabilidad. Asimismo tampoco existe la obligación de establecer un período máximo de resolución de incidencias.

Ciertamente el descubrimiento de vulnerabilidades en programas y servicios de Internet ha sido utilizado, algunas veces con los medios de comunicación como cómplices, como una hoguera de las vanidades. Los distintos egos de expertos en seguridad o hackers amateurs compiten para ver quien descubre el problema más grande y peligroso que tiene mayor repercusión mediática. Encontrar fallos de seguridad en productos de Microsoft se ha convertido en algunos ámbitos más en un juego que en un interés altruista por mejorar la seguridad. Cabe destacar también al abuso o extorsión que a veces se ha producido utilizando fallos de seguridad, pero este ámbito compete directamente al campo judicial.

La **divulgación total o abierta** [www138][www139][www140] señala la necesidad de publicar las vulnerabilidades tan rápido como sean descubiertas. El doble objetivo que se pretende es el de informar a los usuarios inmediatamente del peligro real para que adopten las medidas que consideren así como forzar a los fabricantes a solventarlo rápidamente.

Cataloged vulnerabilities

Year	Total vulnerabilities cataloged	From direct reports
Q1-Q3, 2008	6,058	310
2007	7,236	357
2006	8,064	345
2005	5,990	213
2004	3,780	170
2003	3,784	191
2002	4,129	343
2001	2,437	153
2000	1,090	-
1999	417	-
1998	262	-
1997	311	-
1996	345	-
1995	171	-
Totals	**44,074**	

Fig. - 15: Histórico de vulnerabilidades elaborado por el CERT [img12].

Por un lado debemos asumir que es indiscutible el descubrimiento de fallos de seguridad en todos los sistemas. Igualmente es imposible controlar en que momento y que persona descubre este fallo, por tanto carece de sentido el uso de políticas oscurantistas o represoras. En la figura 15 podemos observar con datos el aumento de vulnerabilidades desde 1995 al tercer trimestre de 2008.

Los fabricantes y proveedores de servicios prestan cada vez más atención a las relaciones públicas con sus clientes. Generalmente no se discuten las inversiones de millones de euros en publicidad o campañas de mejora de la imagen para mantener y captar nuevas cuotas de mercado. La divulgación total de vulnerabilidades aprovecha este interés para forzar a los fabricantes a mejorar la seguridad en sus productos y solventar rápidamente los problemas.

La publicación de vulnerabilidades argumenta que divulgar el problema permite eliminar el uso privilegiado de esta información. De esta forma los hackers malintencionados que pretenden sacar beneficio o empresas irresponsables que únicamente solventan fallos de seguridad si llegan a la opinión pública tienen más difícil perpetuar sus malas artes. Además, esta política permite a los usuarios tomar las contramedidas adecuadas en sus sistemas hasta que el fabricante haya solventado este fallo.

Por otro lado intentar controlar o censurar la publicación de fallos de seguridad históricamente ha demostrado que no es útil. La resolución de fallos de seguridad es un proceso muy costoso para las empresas que no crea excesivo valor, puesto que el producto ya ha sido vendido y este soporte no se cobra al cliente.

Muchas vulnerabilidades conocidas han necesitado meses o incluso años [www142] para ser solventadas. Incluso existen casos en que únicamente tras ser utilizadas por un ataque han sido solventadas por el fabricante. Podemos citar cientos de ejemplos de empresas como Adobe y sus múltiples vulnerabilidades en Acrobat Reader o Microsoft con Internet Explorer.

Permitir que sean las propias empresas las que decidan cuando y cómo se han de solventar las vulnerabilidades confiando en su criterio, más que ingenuo es temerario. La realidad ha demostrado, según el soporte legal de cada país, cómo la práctica más común de los fabricantes es la amenaza con la denuncia a los tribunales, la indiferencia o el desprestigio de los descubridores del problema.

Ciertamente la divulgación total tampoco se ha postulado como la panacea para solventar los problemas de seguridad. Ejemplos como el *sockstress* o aspectos como la necesidad de obligar a los usuarios a actualizar sus productos periódicamente son aspectos que claramente se han de mejorar. Sin embargo, probablemente, son pasos en la dirección correcta.

La cuestión a debatir, en mi opinión, no ha de ser la de buscar culpables. Hemos de buscar **responsables**. La RAE define responsable [www125] como "*Obligado a responder de algo o por alguien.*" o "Dicho *de una persona: Que pone cuidado y atención en lo que hace o decide*."

Es meridianamente claro que no es posible controlar quien descubre vulnerabilidades y por tanto el descubrimiento de vulnerabilidades. Pueden ser expertos en el tema, estudiantes, aprendices de hacker,… por azar o como conclusión de sesudos estudios. No hay norma y por tanto no se debería intentar poner puertas al campo. De hecho intentar regular lo imposible únicamente consigue crear frustración y excepciones a la regla, lo que acaba invalidándola.

Por otro lado tenemos que es muy difícil predecir si las decisiones tomadas inicialmente producirán el efecto previsto. En el caso analizado anteriormente hemos observado como Robert y Jack pensaron que realizando una presentación parcial y comunicando su vulnerabilidad al CERT de Finlandia seria suficiente. Desgraciadamente el comportamiento tanto de los fabricantes como de los expertos en seguridad no fue el esperado.

Una solución razonable seria copiar los comportamientos "naturales" que tenemos en el mundo real. Desde niños en las escuelas, clase y programas infantiles de la televisión nos inculcan que si existe un problema o indicio sospechoso hay que avisar a la autoridad para que se haga cargo de la situación. En el caso de Internet esta actitud se registra a veces de forma automática como cuando se detectan webs fraudulentas o de contenido pedófilo. No hay duda de cómo proceder y a quien avisar.

En el caso de las vulnerabilidades, todos tenemos nuestra pizca de orgullo, este paso no minimiza el papel de la persona que ha dado el aviso, si no que estandariza el proceso de su resolución. En el caso de las tecnologías de la información no siempre es tan obvio, si usted descubriera un importante fallo de seguridad en una aplicación o servicio de Internet ¿a quien avisaría? ¿qué haría primero? ¿y sus amigos y parientes, sabrían que hacer?.

De esta forma la primera decisión, y por tanto la más importante, ha de ser a quien se ha de comunicar el "descubrimiento". Para que todos sepamos cómo proceder adecuadamente es necesario intensificar las acciones de formación en aspectos de seguridad informática y realizar seminarios y campañas publicitarias de alcance nacional. La seguridad en TIC nos afecta ya a todos los ciudadanos y no puede seguir siendo un reducto de expertos en seguridad, oscuros profetas o aprendices de hacker.

Fig. - 16: Logotipo de las entidades CERT [img13].

En el caso de Internet esta figura vendría representada por los CERT, aunque desgraciadamente la realidad es que su gestión no es la que cabría esperar. Necesariamente estos centros han de contar con personal especializado y canales de comunicación reconocidos por todos los usuarios. El soporte real de las administraciones, proveedores de servicios y fabricantes es crítico para obtener una gestión adecuada de la seguridad en Internet. **La seguridad (informática) es un tema demasiado serio como para dejarlo al libre albedrío.**

La verdad es que había planeado finalizar este documento con la frase anterior, sin embargo creo interesante escribir unas líneas más. Durante estas semanas (febrero de 2010) casi todos los medios de comunicación han recogido la noticia de un sofisticado ataque procedente de China que han sufrido entre otras compañías Gmail, Yahoo, Symantec y Adobe. La conocida como "**operación Aurora**"[8].

Aprovechando una serie de vulnerabilidades de Internet Explorer no publicitadas (*0-day*) se instalaba malware en ordenadores de distintos trabajadores y usuarios. Entre otras operaciones este malware robó propiedad intelectual de las empresas y examinó sus correos buscando información de disidentes chinos.

La *gracia* del asunto es que estos atacantes pudieron perpetrar sus ataques utilizando facilidades (o puertas traseras) que el gobierno norteamericano exige a las empresas tecnológicas. Muchos gobiernos democráticos exigen *facilidades* de acceso a los productos y servicios de muchas compañías TIC con el objetivo de poder utilizarlos en caso necesario. Obviamente la asunción de que únicamente ellos utilizarían estos "servicios" es muy reveladora de cómo la seguridad es percibida por muchas entidades gubernamentales. **La seguridad (informática) es un tema demasiado serio como para dejarlo únicamente a criterio del gobierno.**

[8] http://www.cnn.com/2010/OPINION/01/23/schneier.google.hacking/index.html

3. Agradecimientos

Para ser honestos, una vez más, debo reconocer que este documento es más largo de lo que en un principio planifiqué. También reconozco que he tenido que dedicar bastante más tiempo del que disponía inicialmente a documentarme, leer y sobre todo entender las infinitas referencias del tema.

De esta forma, si usted ha soportado las veintiséis páginas anteriores y ha llegado hasta este punto, mi agradecimiento sincero es para usted. También quiero agradecer a Dan, Gordon, Jason y especialmente a Robert su tiempo, ayuda y paciencia en la reconstrucción de la secuencia temporal. Finalmente gracias a David y Arnau por sus acertados comentarios.

I would like to thank Dan Kaminsky, Fyodor, Jason Ross and mainly ***Robert E. Lee*** *for their time and comments. Without their help, sockstress timeline rebuild wasn't possible.*

4. Copyright

Bibliografía

[Tanembaum87] "*Operating systems: design and implementation*", Andrew S. Tanembaum. Prentice Hall.

[Stevens94] "*TCP/IP Illustrated, Volume I*", W. Richard Steven. Addison-Wesley.

[Tanenbaum03] "*Sistemas operativos modernos*", Andrew Tanembaum. Prentice Hall.

[www1] http://www.hispasec.com/unaaldia/3632

[www2] http://www.grc.com/sn/notes-164.htm

[www3] http://sockstress.com/

[www4] http://www.ietf.org/rfc/rfc0791.txt

[www5] http://en.wikipedia.org/wiki/Internet_Protocol

[www6] http://gabriel.verdejo.alvarez.googlepages.com/DEA-es-1LosprotocolosTCPIP.pdf

[www7] http://www.darpa.mil/

[www8] http://ipref.wordpress.com/2008/06/03/la-pila-de-protocolos-de-tcpip/

[www9] http://en.wikipedia.org/wiki/OSI_model

[www10] http://www.itu.int/rec/T-REC-X/en

[www11] http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Internet-Protocols.html

[www12] http://www.protocols.com/pbook/tcpip1.htm

[www13] http://tools.ietf.org/rfc/rfc792.txt

[www14] http://insecure.org/sploits/ping-o-death.html

[www15] http://en.wikipedia.org/wiki/Ping_flood

[www16] http://www.fistconference.org/files/ataquededenegaciondeservicios.pdf

[www17] http://tools.ietf.org/rfc/rfc768.txt

[www18] http://www.ietf.org/rfc/rfc0793.txt

[www19] http://en.wikipedia.org/wiki/Transmission_Control_Protocol#Connection_establishment

[www20] http://www.inetdaemon.com/tutorials/internet/tcp/3-way_handshake.shtml

[www21] http://www.tcpipguide.com/free/t_TCPConnectionEstablishmentProcessTheThreeWayHandsh.htm

[www22] http://www.netbook.cs.purdue.edu/animations/TCP%203-way%20handshake.html

[www23] http://www.cs.purdue.edu/homes/park/cs536-e2e-3.pdf

[www24] http://en.wikipedia.org/wiki/Network_address_translation

[www25] http://www.ietf.org/rfc/rfc1631.txt

[www26] http://en.wikipedia.org/wiki/Proxy_server

[www27] http://www.freeproxy.info/en/free_proxy/faq/index.htm

[www28] http://www.iana.org/assignments/port-numbers

[www29] http://gabriel.verdejo.alvarez.googlepages.com/DEA-es-3IDS.pdf

[www30] http://en.wikipedia.org/wiki/Kevin_Mitnick

[www31] http://mitnicksecurity.com/

[www32] http://cr.yp.to/syncookies.html

[www33] http://en.wikipedia.org/wiki/SYN_cookies

[www34] http://www.usenix.org/events/bsdcon/full_papers/lemon/lemon_html/node9.html

[www35] http://www.unixtimestamp.com/index.php

[www36] http://en.wikipedia.org/wiki/Timestamp

[www37] http://en.wikipedia.org/wiki/Modular_arithmetic

[www38] http://en.wikipedia.org/wiki/Maximum_segment_size

[www39] http://en.wikipedia.org/wiki/Tarpit_(networking)

[www40] http://support.microsoft.com/kb/842851

[www41] http://staff.washington.edu/dittrich/misc/ddos/

[www42] http://gabriel.verdejo.alvarez.googlepages.com/DEA-es-2DOS-DDOS.pdf

[www43] http://en.wikipedia.org/wiki/Denial-of-service_attack
[www44] http://www.linfo.org/kernel.html
[www45] http://www.kernel.org/
[www46] http://en.wikipedia.org/wiki/Kernel_(computing)
[www47] http://www.outpost24.com/news/news-2008-10-02.html
[www48] http://blog.robertlee.name/
[www49] http://www.outpost24.com/
[www50] http://debeveiligingsupdate.nl/audio/bevupd_0003.mp3
[www51] http://media.grc.com/mp3/Whole_SockStress_Mono_64kbps.mp3
[www52] https://www.cert.fi/haavoittuvuudet/2008/tcp-vulnerabilities.html
[www53] http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-4609
[www54] http://www.ieee802.org/3/
[www55] http://en.wikipedia.org/wiki/Ethernet
[www56] http://www.cisco.com/en/US/tech/tk827/tk369/technologies_white_paper09186a00800d6979.shtml
[www57] http://www.protocolbase.net/protocols/protocol_IP%20Fragment.php
[www58] http://en.wikipedia.org/wiki/IP_fragmentation
[www59] http://www.itu.int/ITU-D/cyb/cybersecurity/projects/botnet.html
[www60] http://www.symantec.com/norton/theme.jsp?themeid=botnet
[www61] http://en.wikipedia.org/wiki/Botnet
[www62] http://www.ipv6.org/
[www63] http://www.ietf.org/rfc/rfc2460.txt
[www64] http://en.wikipedia.org/wiki/IPv6
[www65] http://tools.ietf.org/html/rfc760
[www66] http://tools.ietf.org/html/rfc791
[www67] http://msdn.microsoft.com/en-us/library/ms740548(VS.85).aspx
[www68] http://en.wikipedia.org/wiki/Raw_socket
[www69] http://www.wireshark.org/faq.html
[www70] http://en.wikipedia.org/wiki/Promiscuous_mode
[www71] http://www.unicornscan.org/
[www72] http://www.tenouk.com/Module42a.html
[www73] http://www.cs.utah.edu/~swalton/listings/sockets/programs/
[www74] http://www.wireshark.org/
[www75] http://www.tcpdump.org/
[www76] http://nmap.org/
[www77] http://insecure.org/stf/tcpdos/outpost24-sect-sockstress.ppt
[www78] https://www.cert.fi/haavoittuvuudet/2008/tcpvulnerabilitiesstatement.html
[www79] http://tools.cisco.com/security/center/viewAlert.x?alertId=16773
[www80] http://www.elmundo.es/navegante/2008/10/07/tecnologia/1223366908.html
[www81] http://blog.robertlee.name/2008/04/unicornscan-broken-tsc-alternate-timing.html
[www82] http://blog.robertlee.name/2008/04/hack-in-box-dubai.html
[www83] http://blogs.hackerscenter.com/2009/03/unicornscan-author-jack-c-louis-dies-in.html
[www84] http://blog.robertlee.name/2008/04/discretion-in-wrong-hands.html
[www85] http://blog.robertlee.name/2008/08/updates.html
[www86] http://www.sec-t.org/2008/Agenda.html
[www87] http://sec-t.org/2008/files/SEC-T2008.zip
[www88] http://www.grc.com/sn/files/Sec-T_Powerpoint_Presentation.zip
[www89] http://blog.nordenfelt.com/2008/09/12/sec-t-day-1/
[www90] http://blog.robertlee.name/2008/09/sec-t-sockstress-recap.html
[www91] http://debeveiligingsupdate.nl/?s=outpost24
[www92] http://blog.robertlee.name/2008/09/sockstress-podcast-interview.html

[www93] http://www.darkreading.com/blog/archives/2008/09/things_are_abre.html
[www94] http://it.slashdot.org/article.pl?sid=08/10/01/0127245
[www95] http://searchsecurity.techtarget.com/news/article/0,289142,sid14_gci1332898,00.html
[www96] http://blog.robertlee.name/2008/10/clearing-up-some-factual-inaccuracies.html
[www97] https://www.cert.fi
[www98] http://blog.robertlee.name/2008/10/cert-fi-statement-on-issues.html
[www99] http://insecure.org/stf/tcp-dos-attack-explained.html
[www100] http://insecure.org/fyodor/
[www101] http://insecure.org/stf/tcp-dos-attack-explained.html
[www102] http://blogs.techrepublic.com.com/networking/?p=679
[www103] http://blog.robertlee.name/2008/10/conjecture-speculation.html
[www104] http://news.cnet.com/8301-1009_3-10056759-83.html?tag=mncol
[www105] http://arstechnica.com/security/news/2008/10/researchers-disclose-deadly-cross-platform-tcpip-flaws.ars
[www106] http://gabriel.verdejo.alvarez.googlepages.com/ARTICULOS-LavulnerabilidadKaminskyde.pdf
[www107] http://unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html
[www108] http://www.curbrisk.com/security-blog/outpost24-tcp-denial-of-service-vulnerability-interview-transcript.html
[www109] http://www.informationweek.com/blog/main/archives/2008/10/tcp_flaw_an_abj.html;jsessionid=T2XVKSKZTPEO3QE1GHRSKHWATMY32JVN
[www110] http://blog.robertlee.name/2008/10/for-those-with-limited-time.html
[www111] http://blog.robertlee.name/2008/10/sc-magazine-sockstress-podcast.html
[www112] http://www.elmundo.es/navegante/2008/10/07/tecnologia/1223366908.html
[www113] http://podcasts.scmagazine.com/episode/2_2008-10-08T13_33_13-07_00.mp3?dl=1
[www114] http://maliciousattacker.blogspot.com/2008/10/sockstress-released-lookout-internet.html
[www115] http://www.sockstress.com/releases/sockstress-0.1.0.tgz
[www116] http://blog.robertlee.name/2008/10/very-strange-website.html
[www117] http://www.freelance-projects.info/net/build-a-sockstress-application-by-taylormade/
[www118] http://www.t2.fi/schedule/2008/#speech8
[www119] http://www.cisco.com/warp/public/707/cisco-sr-20081017-tcp.shtml
[www120] http://blog.robertlee.name/2008/10/more-detailed-response-to-gordons-post.html
[www121] http://blog.robertlee.name/2008/10/t2-sockstress-talk-recap.html
[www122] http://blog.robertlee.name/2009/03/jack-c-louis-loss-of-dear-friend.html
[www123] http://www.facebook.com/home.php#/group.php?sid=2ba33c753ccca44aa94e60fbf066fe62&gid=302985395000
[www124] http://blog.robertlee.name/2009/09/sockstress-tcp-dos-cert-fi-advisory.html
[www125] http://buscon.rae.es/draeI/SrvltGUIBusUsual?TIPO_HTML=2&LEMA=responsable
[www126] http://privacy.med.miami.edu/glossary/xd_confidentiality_integrity_availability.htm
[www127] http://www.guidetocissp.com/2007/02/cbk-1-security-management-practices.html
[www128] http://en.wikipedia.org/wiki/Information_security
[www129] http://news.cnet.com/Hackers-unlocking-Norways-history/2100-1002_3-934060.html
[www130] http://news.cnet.com/Swedish-programmer-cracks-password/2100-1002_3-934653.html
[www131] http://en.wikipedia.org/wiki/Parkerian_Hexad
[www132] http://www.schneier.com/blog/archives/2006/08/updating_the_tr.html
[www133] http://www.wiretrip.net/rfp/txt/ietf-draft.txt
[www134] https://datatracker.ietf.org/drafts/draft-christey-wysopal-vuln-disclosure/
[www135] http://www.csoonline.com/article/440110/The_Vulnerability_Disclosure_Game_Are_We_More_Secure_
[www136] http://ips2.blogs.com/matts_blog/2004/09/why_full_disclo.html
[www137] http://www.sans.org/reading_room/whitepapers/threats/how_do_we_define_responsible_disclosure_932?show=932.php&cat=threats
[www138] http://www.csoonline.com/article/216205/Schneier_Full_Disclosure_of_Security_Vulnerabilities_a_Damned_Good_Idea_
[www139] http://www.wildernesscoast.org/bib/disclosure-by-date.html
[www140] http://en.wikipedia.org/wiki/Full_disclosure
[www141] http://www.csoonline.com/article/221113/Software_Vulnerability_Disclosure_The_Chilling_Effect
[www142] http://www.computerworld.com/s/article/9146820/Microsoft_confirms_17_year_old_Windows_bug

Imágenes y Figuras

[img1] http://en.wikipedia.org/wiki/File:UDP_encapsulation.svg
[img2] http://i.technet.microsoft.com/cc758065.985bd715-8119-46f8-b27f-817452bd88ab(en-us).gif
[img3] http://ee.fudan.edu.cn/~qylu/comp_net/socket2/wskfaq/articles/bitmaps/state-diagram-small.gif
[img4] http://tjliu.myweb.hinet.net/COA_CH_12.files/image032.jpg
[img5] http://openlearn.open.ac.uk/file.php/2542/T822_1_017i.jpg
[img6] http://www.checkpoint.com/defense/advisories/public/announcement/090809-tcpip-dos-sockstress.html
[img7] http://www.tcpipguide.com/free/diagrams/ipsectransport.png
[img8] http://3.bp.blogspot.com/_h6XFAnDWBZA/SM5WTC9mxDI/AAAAAAAAACU/56IBiAlEhJ8/s320/Picture+1.png
[img9] http://radian.org/notebook/wp-content/uploads/2008/10/sockstress.jpg
[img10] http://3.bp.blogspot.com/_h6XFAnDWBZA/ScQc6jMiz0I/AAAAAAAAAG4/QiZt0a35cmg/s400/IMG_0230.JPG
[img11] http://www.yousecure.co.uk/sitebuildercontent/sitebuilderpictures/CIA.jpg
[img12] http://www.cert.org/stats/
[img13] http://www.region1hls-wa.org/CERT/CERT_home.htm