

PORT SCANNING

ESCANEANDO ORDENADORES

REMOTOS: TIPOS DE SCANEOS

Para utilizar programas/herramientas de "escaneo" como el NMAP necesitamos conocer la forma en que se establecen las conexiones y adentrarnos en su funcionamiento. Este artículo te ilustrará sobre el tema.

1- Avanzando en el análisis de sistemas

Ya hemos comentado la importancia que tiene el análisis previo de un sistema de cara a saltarnos su seguridad, hoy en día existen numerosas técnicas de 'escaneo' de puertos y la mayoría ya han sido implementadas; Un ejemplo claro es NMap del que ya se habló en la revista.



Muy importante

MUY IMPORTANTE: Para cuando leas este artículo, en la Web de PC PASO A PASO (www.hackxcrack.com) podrás descargarte el artículo "Técnicas del Port Scannig y uso del NMAP", que fue publicado en el número 9 de PC PASO A PASO.

Ambos artículos se complementan y nuestra intención es que puedas comprender este texto aunque no compreses en su momento el número 9 de PC PASO A PASO.

Quizás seas de los que piensa que un 'simple escaneo de puertos' no es la mejor forma para entrar en un sistema... aunque sea simplemente uno de los pasos previos evidentemente te equivocas, imaginemos por ejemplo que un usuario aburrido se dedica a jugar con el 'Intesné' escaneando la red y tiene una lista con 250.000 host's con interesante información acerca de su sistema operativo, puertos abiertos, etc.

Ahora imagina que se publica una vulnerabilidad para algún demonio (proceso/programa) de algún sistema operativo como IRIX, FreeBSD o determinada versión de Linux. Nuestro usuario coge su lista y busca coincidencias, ¡acaba de encontrar un centenar de máquinas vulnerables en las que obtener privilegios de administrador! ¿A que no te haría gracia estar en la lista de nuestro amigo?

Por ese motivo resulta interesante conocer mejor el funcionamiento de diferentes técnicas de "port scanning" y de como usar la información obtenida para explotar las debilidades de un sistema así como la forma de protegernos de estos ataques en la medida en que esto sea posible ;-)

2- Algunas nociones fundamentales

Es interesante conocer algunas cosas acerca del protocolo TCP de cara a comprender varias cosas que leerás a continuación y por ello volveremos a mencionar algunos de los conceptos de artículos anteriores.

El protocolo de control de transmisión (TCP) es orientado a conexión y esto implica que se genera un circuito (virtual) entre dos host's cuya comunicación se considera fiable, TCP asegura unas condiciones óptimas para el circuito establecido y utiliza lo que se denomina "acuse de recibo" para garantizar que los datos lleguen correctamente a su destino. Cuando queremos establecer una

conexión ambas partes deberán estar de acuerdo en participar o dicha conexión no se podrá realizar.

SEGMENTO TCP

La cabecera de un segmento tcp (20 bytes normalmente) es la siguiente:

1 puerto origen		1 puerto destino	
2 número de secuencia			
3 número de secuencia ack			
4 long. cabecera	10 reservado	5 flags	6 ventana
7 checksum		8 puntero urgente	
9 opciones			
11 DATOS			

Explicaré muy brevemente los campos ;) **

**** Seguramente no comprenderás muchos de los conceptos que a continuación se detallarán, no te preocupes demasiado, la idea es que empiecen a "sonarte" un montón de nombres "raros" y tener una visión global del asunto. Para comprender a la perfección una conexión TCP/IP necesitarías programar sockets en lenguaje C, algo que no tardaremos mucho en publicar... por el momento no te agobies e intenta simplemente tener una visión general. Para practicar todo lo que a partir de ahora se explicará puedes emplear el NMAP.**

1. Puerto origen y destino, bastante explícito :P
2. Número de secuencia, al intentar establecer una conexión los host's que intervienen en el proceso eligen un número aleatorio para empezar a contabilizar bytes que viajarán en los segmentos de datos de la conexión. Los sucesivos segmentos que se envíen llevarán como número de secuencia el número aleatorio elegido al principio más el número de bytes enviados hasta ese momento.

3. El número de secuencia ack (o acknowledge number) es lo que permite validar los segmentos que van llegando a un host, con este fin se coloca en el campo el número de secuencia del segmento incrementado en 1, dicho byte espera ser recibido en el siguiente envío. **

**** Los puntos 2 y 3 es, para que nos entendamos, la forma que tiene el protocolo TCP de no perder ningún paquete. Si estamos enviando a un compañero un archivo Word de un par de megas, este se corta en pequeños trocitos y el receptor debe recibirlos todos, no puede perderse un solo paquete. Vale, ya se que explicarlo así es muy poco técnico, pero quiero que se entienda.**

4. La longitud de la cabecera en múltiplos de 32 bits. **

**** Hemos dicho que el archivo Word es cortado en paquetitos ¿verdad? Bueno, pues cada paquetito es como una carta. La CABECERA de un paquete TCP es como el sobre de una carta, es donde figuran los datos del remitente (el que envía la carta) y del destinatario (el que debe recibir la carta). Dentro del sobre ¿qué encontramos?, pues lo importante, el contenido, los DATOS, lo que quieren comunicarnos, que nos ha tocado la lotería y cosas de ese tipo ;P Pues bien, un paquete TCP es exactamente igual que una carta, tenemos una CABECERA (el sobre) y unos DATOS (en este caso un cachito del archivo WORD que estamos enviando).**

5. Ahora lo que nos interesa especialmente, el campo 'flags' (banderas):

Tenemos varios flags para disfrute personal y son URG|ACK|PSH|RST|SYN|FIN ahora te explico algo más acerca de ellos, como verás no tienen ningún misterio :)

URG, indica que el segmento transporta datos urgentes.

ACK, indica que el número de secuencia ack de la cabecera es válido.

PSH, fuerza el envío inmediato de los datos recibidos al nivel de aplicación, que serían las aplicaciones finales a nivel de usuario.

RST, indica que la comunicación debe reiniciarse.

SYN, durante la conexión indica el proceso de sincronización.

FIN, indica que el host desea finalizar la conexión.

6. El tamaño de la ventana es el número de bytes que esperan ser recibidos sin necesidad de ser validados por parte de un host y puede variar durante la conexión activa.

7. La integridad de los datos y la cabecera tcp se pueden verificar mediante el checksum o suma de verificación.

8. El puntero urgente (que se utiliza junto al flag URG) indica el número de secuencia del último byte que se considera parte de los datos fuera de banda (urgentes).

9. Opciones TCP como el tamaño máximo de segmento, el 'window scale' y otras que de momento no vamos a ver aquí.

10,11. Un campo reservado y de datos respectivamente.

¿COMO SE ESTABLECE UNA CONEXIÓN ORDENADA TCP?

El proceso se denomina three-way handshake (o saludo en tres fases)



Es posible que no comprendas el esquema pero tranquilidad que ahora te lo explico paso a paso ;-) seguro que lo entenderás mejor:

El proceso para recibir una petición de conexión implica estar preparado para llevar a cabo una serie de llamadas a funciones como lo son bind (), listen () o connect () de las que hablaremos en el texto de forma genérica.

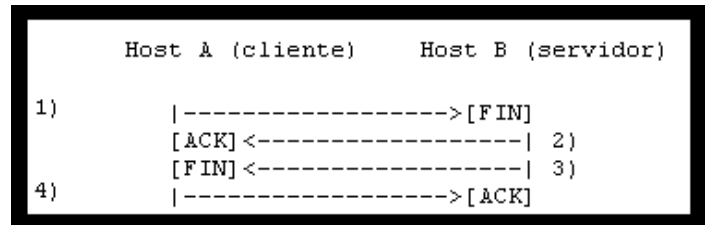
1) Cuando el host A desea establecer una conexión mediante un circuito TCP envía un segmento al host B, el segmento llevará el flag SYN levantado indicando el proceso de sincronización y no suele llevar ningún tipo de datos salvo

la cabecera IP y TCP así como las posibles opciones TCP si las hay. El host A genera una llamada a connect () para el propósito.

2) Ahora el host B enviará la validación para el segmento anterior y activará el flag ACK y en el número de secuencia ack colocará el número de secuencia del segmento recibido + 1, como el proceso de sincronización no ha terminado el flag SYN sigue levantado.

3) El host A sabe que el host B ha validado la petición al recibir el segmento pero ahora el host B está esperando a que se valide su segmento. El host A envía un segmento colocando el valor oportuno en el número de secuencia ack y activando el flag ACK, el flag SYN no viaja levantado esta vez. En este punto la conexión se ha completado con éxito.

¿COMO SE FINALIZA UNA CONEXIÓN ORDENADA TCP?



Ahora te explico que significa eso, no te impacientes, ya sabes que aquí siempre lo explicamos todo para que no se nos escape ningún detalle :)

1) El host A genera una llamada a close() para finalizar la conexión y se envía un segmento con el flag FIN levantado.

2) El host B valida cualquier información previa enviando un segmento con el flag ACK levantado.

3) Para que el host A pueda cerrar la conexión el host B debe enviar un segmento con el flag FIN activado.

4) Se valida el segmento anterior y la conexión se cierra con éxito.

**** Todo esto ocurre de forma transparente al usuario cada vez que establecemos una conexión con un pc remoto (por ejemplo cuando hacemos un simple PING o nos conectamos a una Web con el navegador). De**

hecho suceden muchas más cosas, pero nos conformamos con que captes el concepto general de que establecer una conexión es un proceso de 3 pasos y cerrarla es un proceso de cuatro pasos y que los "flags" (BANDERAS) tienen mucho que decir en este proceso :)

3- Las técnicas de "port scanning"

Existen muchas técnicas para descubrir qué puertos tiene abiertos un host y determinar que servicios están corriendo e incluso bajo qué privilegios, pero lo más importante es hacerlo de forma sigilosa sin que la actividad quede registrada en los 'logs' de nuestra víctima y evitar que los sistemas de detección de intrusos (IDS) nos apunten como origen del escaneo. Te aseguro que a muchos usuarios no les hará gracia que les toques los... puertos :P

Vamos a ver detalladamente varias de las técnicas de escaneo ya conocidas, algunas son una maravilla :) y otras no tanto pero en cualquier caso debes conocerlas para poder adaptar mejor el escaneo a tus necesidades jeje. Ya empezaste a leer acerca de ellas cuando tratamos en la revista el NMap pero ahora veremos con más detalle esas y otras técnicas como el 'Idle Scan' :)

- TCP CONNECT (), mediante esta técnica no necesitarás ningún tipo de privilegio especial como sucederá con otros tipos de escaneo como TCP SYN, y además es una técnica muy rápida puesto que podemos efectuar varias conexiones en paralelo. Su funcionamiento (la mayoría ya lo habéis deducido) es el siguiente: intentamos establecer una conexión con un puerto determinado si el puerto está abierto la llamada a connect () tiene éxito y se nos retornará el valor oportuno en caso contrario la llamada fallará. El problema reside en que los intentos de conexión se registrarán automáticamente en el sistema y en principio no nos interesa en absoluto dejar huellas de ningún tipo.

- TCP SYN (), hace un momento te acabo de explicar como se establece una conexión completa, mandamos un paquete con el flag SYN tal y como se haría normalmente al intentar establecer una conexión y ahora la máquina remota nos responde amablemente con SYN|ACK hasta aquí va todo perfecto pero ahora nosotros para llevar la contraria NO vamos a responder con ACK para establecer la conexión,

en lugar de eso la abortaremos mediante un paquete con RST, ¿y eso por qué? Te preguntarás, la respuesta es muy sencilla, si ya sabemos que hay alguien al otro lado (tras recibir SYN|ACK) para que seguir intentando conectar ¿? así evitamos establecer la conexión completa y es poco probable que nuestro escaneo quede registrado que es de lo que se trata ;) este escaneo probablemente pasará desapercibido pero recuerda, eso sí, que se necesitan privilegios de administrador para usar esta técnica y que el sistema pueda montar ese tipo de paquetes, a este tipo de escaneo se le conoce también como escaneo "medio abierto".

- TCP FIN, aunque el escaneo TCP SYN no suele dejar rastro en los logs del sistema y su detección puede ser algo complicada hay algunos entornos sensibles al escaneo SYN como aquellos en los que hay filtros de paquetes o "firewalls" y por otro lado existen utilidades que registrarán los intentos de conexión SYN. Para estos casos puede resultar de interés echar mano al escaneo sigiloso FIN (o stealth scan), se fundamenta en el hecho de que los puertos abiertos ignoran los paquetes recibidos con el flag FIN (vacío) y los cerrados responderán con RST. Como se dijo en su momento al presentar Nmap los sistemas de Micro\$oft entre otros no son susceptibles a este tipo de ataques pues no siguen las pautas establecidas, quien lo diría xD ¿verdad?

- ACK scan, para esos "ambientes hostiles" con la presencia de cortafuegos puede interesar "nmapear" las reglas de filtrado. Mediante esta técnica podemos enviar un paquete con el flag ACK con los números de secuencia y ack de forma incorrecta o aleatoria de manera que recibamos un paquete de control ("ICMP unreachable", inalcanzable) o no se reciba respuesta, en tal caso el puerto se encuentra filtrado y en caso contrario la conexión será reiniciada y las conexiones al puerto no estarían filtradas. Como habrás deducido, no puedes usar este tipo de escaneo para averiguar los puertos abiertos. Si deseas además encontrar los puertos abiertos y además clasifica los puertos según se encuentren filtrados o no, existe otra técnica conocida como 'Window Scan' que además aprovecha anomalías el tamaño de la ventana TCP por parte de varios sistemas operativos para determinar si el puerto está abierto.

- UDP scan (User Datagram Protocol scan), antes de nada aclarar que se trata de un escaneo mediante un protocolo "no orientado a conexión" de manera que no existe

un circuito virtual entre host's ni tampoco garantía de entrega alguna de los paquetes enviados. Se utiliza cuando queremos averiguar que puertos udp (un puerto puede ser tcp y udp al mismo tiempo pues difieren en el tipo de protocolo pese a tener el mismo número identificador) están abiertos, si tras mandar el paquete se recibe un mensaje de control informando del error el puerto está cerrado y en caso contrario, no recibir nada, se encuentra abierto. Esta técnica es muy poco precisa ya que si por ejemplo se filtran las conexiones udp aparecerán como abiertos puertos que están bloqueados. Además el número de conexiones udp suele estar limitado por los sistemas de manera que el escaneo puede ser bastante lento, y he dicho que suele limitarse porque existe cierto sistema operativo cuyo nombre no recuerdo ;) que no atiende a las especificaciones establecidas por los RFC's de manera que el escaneo irá mucho más rápido en los sistemas de la compañía Microsoft. Por el mismo motivo el escaneo denominado 'Null Scan' o escaneo nulo que se basa en no levantar ninguna bandera tampoco funcionará al usarlo contra una máquina Window\$:(

- Xmas Scan: muy similar al escaneo nulo esta técnica envía un paquete con todas las banderas levantadas. Si el puerto está abierto debe ignorar el paquete.

- Idle scan, esta es para mí una de las técnicas más interesantes y que realmente interesa poder usar siempre que se pueda ya que es altamente sigilosa y no dejamos ni rastro de nuestra IP :-) esto es así debido básicamente a que no será el origen del escaneo... o mejor dicho no para la máquina víctima de nuestro Idle scan jeje. Con lo leído hasta ahora y para tu total comprensión del Idle Scan lo único nuevo que deberías saber es que los paquetes que se envían llevan un 'identificador de fragmento' que se suele incrementar cada vez que se envía uno. Esta técnica de escaneo utiliza host's intermedios o los llamados "zombies" para escanear un objetivo, ahora te explico la forma de hacerlo:

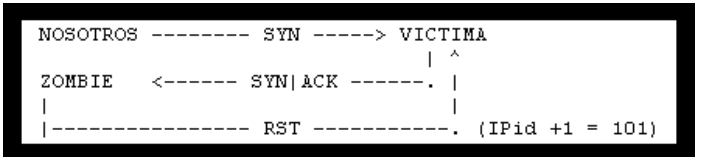
Elegir un host zombie para determinar su identificación IP (IPid), ahora enviamos un paquete para probar un puerto en nuestra víctima desde el host zombie (ya deberías saber que podemos falsear la dirección IP tal y como se comentó en el artículo del NMap, pero nos interesa obtener los resultados ahora). El resultado puede ser que el puerto este abierto o cerrado (--no me digas) en el primer caso NOSOTROS

enviamos un paquete falso como si fuésemos el host ZOMBIE de manera que la respuesta del host VICTIMA irá para el host tal y como muestra el ejemplo, supondremos que el IPid del zombie es ahora 100 e ignoraremos a que puerto se dirige junto con otros datos irrelevantes para hacer el ejemplo un poco más inteligible:

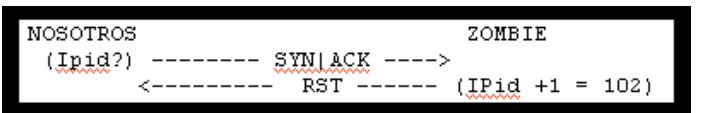
Lo primero es obtener el IPid del zombie escogido (100),



Tras la primera consulta procedemos,

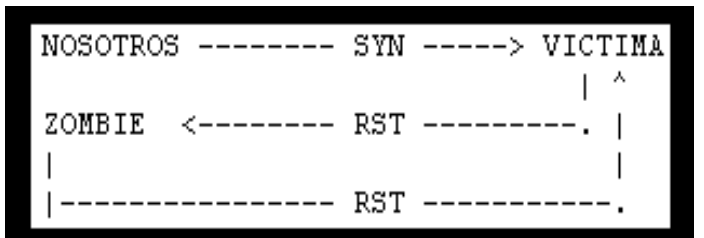


Chequeamos el IPid para ver si el puerto se encuentra abierto,



¿Qué ha pasado arriba?

Pues lo que ha ocurrido es que ahora el IPid del zombie se ha incrementado en 1 tras rebotar al zombie nuestra petición de conexión. Ahora tu que eres una persona muy atenta observas de nuevo el IPid de nuestro amigo el zombie xD y observas que efectivamente vale $100 + 2 = 102$ y eso significa que el puerto está abierto :-) pero como no siempre tendremos esa suerte ahora veremos que ocurre si el puerto está cerrado, tomaremos los mismos datos previos pero con esa diferencia respecto al estado del puerto:



¿Y ahora que ha ocurrido?

Pues muy sencillo :-), al rebotar al zombie la petición de conexión la víctima (a la que también se le puede llamar host remoto :P) nos dice "¿SYN@#~?¿eso qué es?" Como le suena raro y curiosamente el puerto está cerrado nos reinicia la conexión mediante RST. Ahora miramos de nuevo el IPid de nuestro zombie y nos damos cuenta de que solo ha incrementado en una unidad desde la última vez por lo que el puerto estaba cerrado. Si te fijas la IP que recibe la víctima es siempre la de la máquina zombie ;-), muy bien pues ahora imagina que nuestra víctima tiene habilitado un cortafuegos que tiene "permiso" para dejar pasar los paquetes cuya IP coincida con la de nuestro zombie (sucede por ejemplo en las relaciones de confianza), iacabas de saltarte el cortafuegos por la cara!. Espero que hayas comprendido bien el funcionamiento de esta potente técnica de escaneo que como ves es muy efectiva. El principal problema es encontrar host's que tengan números IPid predecibles como serían las dedicadas a impresión puesto que tienen poca actividad o tráfico para llevar a buen término esta técnica.

Hasta aquí hemos visto con detalle los tipos de escaneo más importantes y más utilizados. Te recuerdo que explicamos en un artículo anterior la herramienta que aplica estas y otras técnicas de "port scanning" que fue NMap, permíteme que te sugiera dicha utilidad si no te apetece programar tu propio escáner de puertos ahora que ya sabes como funciona el asunto :)

4- UN PASO MÁS: DETECCIÓN REMOTA DE SO'S

Ni que decir tiene la importancia que tiene saber a que sistema operativo nos enfrentamos ya que resulta fundamental de cara al análisis y la penetración de un sistema así como la explotación remota debido a fallas conocidas (e incluso desconocidas :P) en alguno de los servicios, etc. Existen varias formas y utilidades que nos permitirán obtener una huella digital (o fingerprint) de nuestra víctima que nos sirva para discriminar un sistema operativo del resto, ahora veremos algunas.

PRIMER CONTACTO

Si ya has realizado conexiones directamente desde la consola ya sea para mirar el correo, usar el FTP y/o cualquier otra cosa que se pueda hacer mediante telnet te habrás fijado de que en muchos casos se nos está dando una valiosa información de forma totalmente gratuita por parte del demonio que corre en ese puerto referente a ellos mismos y en muchos casos la plataforma sobre la que corren :) pero no siempre lo tendremos tan fácil por lo que ahora veremos otras técnicas bastante más "sofisticadas" jeje.

```
linux $ ftp ftp.netscape.com
Connected to ftp.gftp.netscape.com.
220-15
220 ftpnscp.newaol.com FTP server (SunOS 5.8) ready.
Name (ftp.netscape.com:linux): anonymous
500 'AUTH SSL': command not understood.
SSL not available
331 Guest login ok, send your complete e-mail address as password.
Password:
230-The response ` ' is not valid.
230-Next time please use your e-mail address as password.
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> syst
215 UNIX Type: L8 Version: SUNOS
ftp>
```

```
linux $ telnet 131.215.48.89 21
Trying 131.215.48.89...
...
220 secant FTP server (Version wu-2.6.1(2) Thu Nov 29 15:02:58
PST 2001) ready.
ftp> syst
215 UNIX Type: L8
```

```
linux $ telnet 216.127.72.117
...
Red Hat Linux release 7.2 (Enigma)
Kernel 2.4.9-34 on an i686
login: ^D
```

ANALIZANDO LA PILA TCP/IP

A este método se le suele llamar "Stack FingerPrinting" y permite obtener conclusiones analizando la pila de protocolos TCP/IP.

Algunas formas de discriminar según el sistema operativo conocidas se basan en:

- Flag de fragmentación, en casos concretos se establece por parte de ciertos SO's el flag DF (deshabilitar la fragmentación) de la cabecera IP.
- Números de secuencia, se intentan encontrar patrones en los números iniciales escogidos al responder a una solicitud de conexión como por ejemplo incrementar un valor X cada cierto tiempo o elegir valores totalmente aleatorios, devolver el mismo número enviado, etc.
- Limitación ICMP, el sistema operativo Linux siguiendo las normas y recomendaciones establecidas limita el número de

mensajes del tipo ICMP unreachable (inalcanzable) a un máximo de 20 por segundo pero no todos los sistemas operativos siguen las reglas establecidas tal y como he comentado antes ;)

- Flag no determinado, enviamos SYN y uno de los 6 bits reservados para ver si se mantienen estos bits levantados en la respuesta ya que algunos sistemas cerraran la conexión tras considerar el error, y como sobre gustos no hay nada escrito... :P
- Tipo de servicio, se puede comprobar uno de los campos del mensaje de control de errores ICMP de tipo "unreachable" para ver si su valor es 0 u otro ya que lo normal es que su valor sea 0 pese haber sistemas, como es el caso de Linux, en que se devuelve un 192 decimal (0xC0).
- Opciones TCP, se pueden observar los valores devueltos, su orden o si son soportadas o no ciertas opciones como lo son el window scale, o el MSS (tamaño máximo de segmento).
- Fragmentos solapados, dependiendo de la implementación del sistema para el tratamiento de los fragmentos solapados, el nuevo fragmento sobrescribe o es sobrescrito por el fragmento anterior, hay que utilizar todos los fragmentos para reconstruir el paquete original completo.

Estos son algunos de los métodos que existen y que se suelen utilizar para determinar con MUCHA precisión que SO corre en nuestra víctima pero no son los únicos y posiblemente aparecerán algunos otros más por lo que no es el objetivo de este artículo profundizar en todos ellos ahora, quizás más adelante en próximos artículos ;-)

Por supuesto existen potentes utilidades como QUESO (Qué Sistema Operativo) que puedes obtener de <http://www.apostols.org> con el objetivo de detectar el SO de un host remotamente. QUESO utiliza varios paquetes con ACK=0 y un número de secuencia aleatorio de la siguiente forma:

- 1 SYN
- 2 SYN|ACK
- 6 PSH
- 3 FIN
- 4 SYN|FIN
- 5 FIN|ACK
- 7 SYN|otros flags no estandarizados.

QUESO es un programa bien diseñado mediante un fichero aparte que contiene las respuestas que se esperan según el tipo de paquete enviado y que se utiliza para contrastar los resultados obtenidos para afirmar con bastante seguridad que SO corre en nuestro objetivo, en fin una maravilla que ya deberías tener :)

5- PortSentry

Mediante esta interesante utilidad que te puedes descargar de su sitio web oficial en <http://www.psionic.com> (pero suele venir el paquete correspondiente en muchas de las distribuciones de Linux) capaz de detectar y responder en tiempo real a un escaneo de puertos contra una máquina incluso puede detectar escaneos ocultos! aunque no en todos los casos evidentemente ;-), PortSentry se pone a la escucha en los puertos no utilizados que se indican en /etc/port Sentry.conf (por lo general) y guarda las direcciones IP origen del escaneo guardándolas a continuación en /etc/host.deny entonces se puede dejar en un host inválido como 999.999.999.999 mediante TCPWrappers o utilidades similares. Evidentemente necesitarás algo más para proteger tu sistema como un cortafuegos y una política de seguridad adecuada... :P

La línea de comandos básica para PortSentry teniendo en cuenta que estamos bajo Linux como super usuario es:

`root # portsentry -stcp`, detectar escaneos ocultos TCP.

`root # portsentry -atcp`, como el anterior pero en modo avanzado.

`root # portsentry -tcp`, detección básica TCP con atadura al puerto (con binding o ligado al puerto).

`root # portsentry -udp`, detección básica UDP con atadura al puerto.

`root # portsentry -sudp`, puede detectar escaneos UDP ocultos.

`root # portsentry -audp`, como -sudp pero en modo avanzado.

Puede que hablemos de esta y otras utilidades relacionadas en un futuro pero por ahora esto es más que suficiente :) para empezar.

6- Medidas de seguridad básicas

Como has podido observar existen numerosas formas de escanear una red y obtener información sensible acerca de la misma por lo que se recomienda instalar sistemas IDS que registren cualquier actividad sospechosa así como un buen cortafuegos (con una buena política restrictiva) y disponer de un buen 'syslog'. Además lo ideal sería guardar todos nuestros "logs" en un servidor remoto junto con sumas de verificación u otros sistemas que nos aseguren que no se han modificado nuestros registros. Existen muchas aplicaciones para proteger nuestro sistema del exterior pero ninguna medida es suficiente como para proteger al 100% nuestro sistema, eso es lo único que es realmente seguro si tu computadora está conectada a la red de redes :P y lo seguiremos demostrando en los siguientes artículos ;-)