Curso de Seguridad Infórmatica

Creada por V. Ana González Hernández

Curso de Seguridad Informática

Módulo 7. Firewalls

Índice

*Nota de la autora de los 7 primeros temas curso: Escribir un curso de seguridad informática en una semana, es una tarea titánica casi imposible, aún así he decidido enfrentarme a ella para hacer posible este curso. Aunque me encantaría escribir todo con palabras propias, me resulta imposible debido a la falta de tiempo, por lo que copiaré varios textos de los cuales dejaré las referencias para la hacer posible la explicación de todo el contenido que el curso comprende.

Creada por V. Ana González Hernández

7.1.1 Un poco de historia ¿De donde vienen los firewalls?

ORIGEN DE LA PALABRA FIREWALLS

El concepto de firewall o muro de fuego, proviene de la mecánica automotriz, donde se lo considera una lámina protectora / separadora entre el habitáculo de un vehículo y las partes combustibles del motor, que protege a sus pasajeros en caso de incendio.

Análogamente, un firewall, en un sentido más informático, es un sistema capaz de separar el habitáculo de nuestra red, o sea, el área interna de la misma, del posible incendio de crackers que se produciría en ese gran motor que es internet.

El primer documento publicado para la tecnología firewall data de 1988, cuando el equipo de ingenieros Digital Equipment Corporation (DEC) desarrolló los sistemas de filtro conocidos como cortafuegos de filtrado de paquetes. Este sistema, bastante básico, fue la primera generación de lo que se convertiría en una característica más técnica y evolucionada de la seguridad de Internet. En AT&T Bell, Bill Cheswick y Steve Bellovin, continuaban sus investigaciones en el filtrado de paquetes y desarrollaron un modelo de trabajo para su propia empresa, con base en su arquitectura original de la primera generación.

Durante 1989 y 1990, tres colegas de los laboratorios AT&T Bell, Dave Presetto, Janardan Sharma, y Nigam Kshitij, desarrollaron la segunda generación de servidores de seguridad. Esta segunda generación de cortafuegos tiene en cuenta, además, la colocación de cada paquete individual dentro de una serie de paquetes. Esta tecnología se conoce generalmente como la inspección de estado de paquetes, ya que mantiene registros de todas las conexiones que pasan por el cortafuegos, siendo capaz de determinar si un paquete indica el inicio de una nueva conexión, es parte de una conexión existente, o es un paquete erróneo. Este tipo de cortafuegos pueden ayudar a prevenir ataques contra conexiones en curso o ciertos ataques de denegación de servicio.

Tercera generación - cortafuegos de aplicación

Son aquellos que actúan sobre la capa de aplicación del modelo OSI. La clave de un cortafuegos de aplicación es que puede entender ciertas aplicaciones y protocolos (por ejemplo: protocolo de transferencia de ficheros, DNS o navegación web), y permite detectar si un protocolo no deseado se coló a través de un puerto no estándar o si se está abusando de un protocolo de forma perjudicial.

Un cortafuegos de aplicación es mucho más seguro y fiable cuando se compara con un cortafuegos de filtrado de paquetes, ya que repercute en las siete capas del modelo de referencia OSI. En esencia es similar a un cortafuegos de filtrado de paquetes, con la diferencia de que también podemos filtrar el contenido del paquete. El mejor ejemplo de cortafuegos de aplicación es ISA (Internet Security and Acceleration)

Acontecimientos posteriores

En 1992, Bob Braden y DeSchon Annette, de la Universidad del Sur de California (USC), dan forma al concepto de cortafuegos. Su producto, conocido como "Visas", fue el primer sistema con una interfaz gráfica con colores e iconos, fácilmente implementable y compatible con sistemas operativos como Windows de Microsoft o MacOS de Apple.

En 1994, una compañía israelí llamada Check Point Software Technologies lo patentó como software denominándolo FireWall-1.

La funcionalidad existente de inspección profunda de paquetes en los actuales cortafuegos puede ser compartida por los sistemas de prevención de intrusiones (IPS).

Actualmente, el Grupo de Trabajo de Comunicación Middlebox de la Internet Engineering Task Force (IETF) está trabajando en la estandarización de protocolos para la gestión de cortafuegos.

Otro de los ejes de desarrollo consiste en integrar la identidad de los usuarios dentro del conjunto de reglas del cortafuegos. Algunos cortafuegos proporcionan características tales como unir a las identidades de usuario con las direcciones IP o MAC. Otros, como el cortafuegos NuFW, proporcionan características de identificación real solicitando la firma del usuario para cada conexión.

¿Qué es un firewall?

Un firewall es un sistema o un grupo de sistemas que implementan una política de control de acceso entre dos o más redes. Podemos imaginarlo como compuesto por dos grandes módulos; uno destinado a bloquear los accesos y el otro a permitirlos.

¿Por qué un firewall?

Básicamente la razón para la instalación de un firewall es casi siempre la misma: proteger una red privada contra intrusos dentro de un esquema de conectividad a Internet.

En la mayoría de los casos, el propósito es prevenir el acceso de usuarios no autorizados a los recursos computacionales en una red privada y a menudo prevenir el tráfico no autorizado de información propietaria hacia el exterior.

Objetivo de un Firewalls

Un firewall sirve para múltiples propósitos, entre otros podemos anotar los siguientes:

- Restricción de entrada de usuarios a puntos cuidadosamente controlados de la red interna.

- Prevención ante los intrusos que tratan de ganar espacio hacia el interior de la red y los otros esquemas de defensas establecidos.

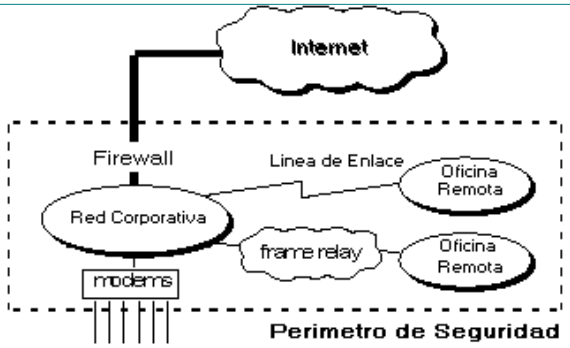
- Restricción de uso de servicios tanto a usuarios internos como externos.

- Determinar cuáles de los servicios de red pueden ser accedidos dentro de ésta por los que están fuera, es decir, quién puede entrar a utilizar los recursos de red pertenecientes a la organización.

Todo el tráfico que viene de la Internet o sale de la red corporativa interna pasa por el firewall de tal forma que él decide si es aceptable o no.

El firewall determina cual de los servicios de red pueden ser accedidos dentro de esta por los que están fuera, es decir quien puede entrar para utilizar los recursos de red pertenecientes a la organización.

Para que un firewall sea efectivo, todo trafico de información a través del Internet deberá pasar a través del mismo donde podrá ser inspeccionada la información. El firewall podrá únicamente autorizar el paso del trafico, y el mismo podrá ser inmune a la penetración, desafortunadamente, este sistema no puede ofrecer protección alguna una vez que el agresor lo traspasa o permanece entorno a este.



El firewall permite al administrador de la red definir un "choke point" (envudo), manteniendo al margen los usuarios no-autorizados fuera de la red, prohibiendo potencialmente la entrada o salida al vulnerar los servicios de la red, y proporcionar la protección para varios tipos de ataques posibles.
Uno de los beneficios clave de un firewall es que ayuda a simplificar los trabajos de administración.
El firewall ofrece un punto donde la seguridad puede ser monitoreada y si aparece alguna actividad sospechosa , este generara una alarma ante la posibilidad de que ocurra un ataque, o suceda algún problema en el transito de los datos.

Tipos básicos de firewalls

Conceptualmente hay dos tipos de firewalls, nivel de red y nivel de aplicación aunque los firwalls pueden ser físicos o lógicos, esto quiere decir que pueden ser máquinas reales compradas para tal propósito incluso routers con estas características o pueden ser aplicaciones que forman parte de nuestro sistema operativo o que instalamos para este fin.

-Firewall de red: Los firewalls de nivel de red toman sus acciones en función del origen, la dirección de destino y el port en cada paquete IP. Los modernos firewalls de este tipo se han sofisticado y mantienen información respecto del estado de las conexiones que están activas a través de él, etc.. Este tipo de firewall tienden a ser muy rápidos y son transparentes al usuario.

-Firewall de aplicación: Los firewalls de nivel de aplicación por lo general son hosts corriendo proxy servers, que no permiten el tráfico directo entre redes, manteniendo una elaborada auditoría y logeo del tráfico que pasa a través de él. Este tipo de firewall puede ser utilizado para realizar las tareas relativas al NAT, debido a que como las comunicaciones van de un lado hacia el otro se puede enmascarar la ubicación original. Este tipo tiende a proveer una auditoría más detallada y un mayor grado de seguridad que los de nivel de red.

Creada por V. Ana González Hernández

7.1.2 POLÍTICAS DE DISEÑO DE FIREWALLS

Las políticas de accesos en un Firewalls se deben diseñar poniendo principal atención en sus limitaciones y capacidades pero también pensando en las amenazas y vulnerabilidades presentes en una red externa insegura. Conocer los puntos a proteger es el primer paso a la hora de establecer normas de seguridad. También es importante definir los usuarios contra los que se debe proteger cada recurso, ya que las medidas diferirán notablemente en función de esos usuarios.
Generalmente se plantean algunas preguntas fundamentales que debe responder cualquier política de seguridad:

- ¿Qué se debe proteger?. Se deberían proteger todos los elementos de la red interna (hardware, software, datos, etc.).
- ¿De quién protegerse?. De cualquier intento de acceso no autorizado desde el exterior y contra ciertos ataques desde el interior que puedan preverse y prevenir.

Sin embargo, podemos definir niveles de confianza, permitiendo selectivamente el acceso de determinados usuarios externos a determinados servicios o denegando cualquier tipo de acceso a otros.

- ¿Cómo protegerse?. Esta es la pregunta más difícil y está orientada a establecer el nivel de monitorización, control y respuesta deseado en la organización. Puede optarse por alguno de los siguientes paradigmas o estrategias:

Paradigmas de seguridad

- Se permite cualquier servicio excepto aquellos expresamente prohibidos.
- Se prohíbe cualquier servicio excepto aquellos expresamente permitidos. La más recomendada y utilizada aunque algunas veces suele acarrear problemas por usuarios descontentos que no pueden acceder a tal cual servicio.

Estrategias de seguridad

- Paranoica: se controla todo, no se permite nada.
- Prudente: se controla y se conoce todo lo que sucede.
- Permisiva: se controla pero se permite demasiado.
- Promiscua: no se controla (o se hace poco) y se permite todo.

¿Cuánto costará?. Estimando en función de lo que se desea proteger se debe decidir cuanto es conveniente invertir.

Creada por V. Ana González Hernández

7.1.3 Tipos de Firewall por funcionalidad:

De acuerdo a las técnicas de filtrado de tráfico que se implementan, los firewalls pueden clasificarse en diferentes tipos:

Filtrado de paquetes stateless (sin estado):

Es la forma más básica de filtrado de tráfico. Usualmente se aplica en dispositivos de capa de red e implementa conjuntos de reglas estáticas que examinan los encabezados de cada paquete para permitir o denegar el tráfico, sin ninguna relación con los flujos de tráfico precedentes. Trabajan bien cuando el objetivo filtra aplicaciones basadas en TCP que no utilizan negociación dinámica de puertos.

Filtrado de paquetes statefull (con estado):

Es un método de filtrado de paquetes que trabaja a nivel de flujo o conexión, con ocasionales intervenciones a nivel de la aplicación.
Mantienen una tabla de estado que hace seguimiento de las sesiones que atraviesan el firewall y en función de ella hace inspección de cada paquete que atraviesa el dispositivo.
El mecanismo asume que si se permite el inicio de la conexión, cualquier conexión adicional que requiera esa aplicación será permitida.
Es un mecanismo confiable para filtrar tráfico de red entre dominios de seguridad.

Filtrado de paquetes stateful con inspección (SPI (Stateful Packet Inspection)) y control de aplicaciones:

Se trata de firewalls stateful que incorporan motores de análisis de tráfico que suman servicios adicionales que reciben la denominación de AIC (Application Inspection and Control) o DPI (Deep Packet Inspection).
Estos sistemas reensamblan en memoria las sesiones de capa de transporte para realizar inspección de protocolos de capa de aplicación y decodifican los protocolos de capa de aplicación para permitir filtrado de protocolos y contenidos. Pueden verificar los protocolos de capa de aplicación para eliminar paquetes que no se conformen con el funcionamiento estándar del protocolo.

Sistemas de prevención de intrusos en la red: NIPS (Network Intrusion Prevention Systems):

También conocidos como IDS/IPS o IDPS (Intrusion Detection Prevention System), los vamos a tratar en detalle en el siguiente tema. Son sistemas que analizan el tráfico de la red con el propósito de bloquear tráfico malicioso conocido.
Se asienta en una base de datos de ataques que debe ser actualizada periódicamente.
Son mecanismos permisivos y usualmente no pueden detectar amenazas nuevas a menos que hayan sido incluidas en las actualizaciones.

Gateways de aplicaciones (proxies):

Es un sistema de software diseñado para actuar como intermediario y reenviar requerimientos de capa de aplicación y respuestas entre los clientes y los servidores. En términos de control de acceso, permite un filtrado y seguimiento muy granular tanto de las solicitudes como de las respuestas.
Brindan opciones de control de acceso confiables para los protocolos soportados. Sin embargo, hay que tener presente que no hay proxies disponibles para todas las aplicaciones corporativas y no se aplican a aplicaciones de tiempo real.

7.2 Tipos de arquitecturas de firewalls

7.2.1 Firewall de filtrado de paquetes

Se utilizan Routers con filtros y reglas basadas en políticas de control de acceso. El Router es el encargado de filtrar los paquetes (un Choke) basados en cualquiera de los siguientes criterios:

1. Protocolos utilizados.
2. Dirección IP de origen y de destino.
3. Puerto TCP–UDP de origen y de destino.

Estos criterios permiten gran flexibilidad en el tratamiento del tráfico. Restringiendo las comunicaciones entre dos computadoras (mediante las direcciones IP) se permite determinar entre cuales máquinas la comunicación está permitida.
El filtrado de paquetes mediante puertos y protocolos permite establecer que servicios estarán disponibles al usuario y por que puertos.
Se puede permitir navegar en la WWW (puerto 80 abierto) pero no acceder a la transferencia de archivos vía FTP (puerto 21 cerrado).
Debido a su funcionamiento y estructura basada en el filtrado de direcciones y puertos este tipo de Firewalls trabajan en los niveles de Transporte y de Red del Modelo OSI y están conectados a ambos perímetros (interior y exterior) de la red.
Tienen la ventaja de ser económicos, tienen un alto nivel de desempeño y son transparentes para los usuarios conectados a la red. Sin embargo presenta debilidades como:

1. No protege las capas superiores a nivel OSI.
2. Las necesidades aplicativas son difíciles de traducir como filtros de protocolos y puertos.
3. No son capaces de esconder la topología de redes privadas, por lo que exponen la red al mundo exterior.
4. Sus capacidades de auditoría suelen ser limitadas, al igual que su capacidad de registro de actividades.
5. No soportan políticas de seguridad complejas como autentificación de usuarios y control de accesos con horarios prefijados.

Cualquier router ip utiliza reglas de filtrado para reducir la carga de la red, por ejemplo, se descartan paquetes cuyo ttl (time to live o tiempo de vida) ha llegado a cero, paquetes con un control de errores erróneos, o simplemente tramas de broadcast.
Además de estas aplicaciones, el filtrado de paquetes se puede utilizar para implementar diferentes políticas de seguridad en una red.
El objetivo principal de todas ellas suele ser evitar el acceso no autorizado entre dos redes, pero manteniendo intactos los accesos autorizados.
Su funcionamiento es habitualmente muy simple: se analiza la cabecera de cada paquete, y en función de una serie de reglas establecidas de antemano la trama es bloqueada o se le permite seguir su camino.
Estas reglas suelen contemplar campos como el protocolo utilizado (tcp, udp, icmp. . .), las direcciones fuente y destino, y el puerto destino, lo cual ya nos dice que el firewall ha de ser capaz de trabajar en los niveles de red (para discriminar en función de las direcciones origen y destino) y de transporte (para hacerlo en función de los puertos usados). Además de la información de cabecera de las tramas, algunas implementaciones de filtrado permiten especificar reglas basadas en la interfaz del router por donde se ha de reenviar el paquete, y también en la interfaz por donde ha llegado hasta nosotros.
¿Cómo se especifican tales reglas?
Generalmente se expresan como una simple tabla de condiciones y acciones que se consulta en orden hasta encontrar una regla que permita tomar una decisión sobre el bloqueo o el reenvío de la trama, adicionalmente, ciertas

implementaciones permiten indicar si el bloqueo de un paquete se notificaría la máquina origen mediante un mensaje icmp.

Siempre hemos de tener presente el orden de análisis de las tablas para poder implementar la política de seguridad de una forma correcta, cuanto más complejas sean las reglas y su orden de análisis, más difícil será para el administrador comprenderlas.
Independientemente del formato, la forma de generar las tablas dependerá obviamente del sistema sobre el que trabajemos, por lo que es indispensable consultar su documentación. Algunos ejemplos particulares – pero aplicables a otros sistemas – pueden encontrarse en http://www.fwtk.org/fwtk/docs/overview.pdf

Por ejemplo, imaginemos una hipotética tabla de reglas de filtrado de la siguiente forma:

Origen	Destino	Tipo	Puerto	Acción
158.43.0.0	*	*	*	Deny
*	195.53.22.0	*	*	Deny
158.42.0.0	*	*	*	Allow
*	193.22.34.0	*	*	Allow

Si al cortafuegos donde está definida la política anterior llegara un paquete proveniente de una máquina de la red 158.43.0.0 se bloqueará su paso, sin importar el destino de la trama, de la misma forma, todo el tráfico hacia la red 195.53.22.0 también se detendrá.

Pero, **¿qué sucedería si llega un paquete de un sistema de la red 158.42.0.0 hacia 193.22.34.0?**
Una de las reglas nos indica que dejemos pasar todo el tráfico proveniente de 158.42.0.0, pero la siguiente nos dice que si el destino es 193.22.34.0 lo bloqueemos sin importar el origen.
En este caso depende de nuestra implementación particular y el orden de análisis que siga.

Si se comprueban las reglas desde el principio, el paquete atravesaría el cortafuegos, ya que al analizar la tercera entrada se finalizarían las comprobaciones. Si operamos al revés, el paquete se bloqueará porque leemos antes la ultima regla.

Como podemos ver, ni siquiera en nuestra tabla – muy simple – las cosas son obvias, por lo que si extendemos el ejemplo a un firewall real podemos hacernos una idea de hasta que punto hemos de ser cuidadosos con el orden de las entradas de nuestra tabla.
¿Qué sucederá si, con la tabla del ejemplo anterior, llega un paquete que no cumple ninguna de nuestras reglas?
El sentido común nos dice que por seguridad se debería bloquear, pero esto no siempre sucede así diferentes implementaciones ejecutan diferentes acciones en este caso.
Algunas deniegan el paso por defecto, otras aplican el contrario de la ultima regla especificada (es decir, si la última entrada era un Allow se niega el paso de la trama, y si era un Deny se permite), otras dejan pasar este tipo de tramas. . .
De cualquier forma, para evitar problemas cuando uno de estos datagramas llega al cortafuegos, lo mejor es insertar siempre una regla por defecto al final de nuestra lista – recordemos una vez más la cuestión del orden – con la acción que deseemos realizar por defecto; si por ejemplo deseamos bloquear el resto del tráfico que llega al firewall con la tabla anterior, y suponiendo que las entradas se analizan en el orden habitual, podríamos añadir a nuestra tabla la siguiente regla:

Origen	Destino	Tipo	Puerto	Acción
*	*	*	*	Deny

La especificación incorrecta de estas reglas constituye uno de los problemas de seguridad habituales en los firewalls de filtrado de paquetes, no obstante, el mayor problema es que un sistema de filtrado de paquetes es incapaz de analizar (y por tanto verificar) datos situados por encima del nivel de red Osi.
A esto se le añade el hecho de que si utilizamos un simple router como filtro, las capacidades de registro de información del mismo suelen ser bastante limitadas, por lo que en ocasiones es difícil la detección de un ataque.

Se puede considerar un mecanismo de prevención más que de detección. Para intentar solucionar estas – y otras vulnerabilidades – es recomendable utilizar aplicaciones software capaces de filtrar las conexiones a servicios, a dichas aplicaciones se les denomina proxies de aplicación, y las vamos a comentar en el punto siguiente.

7.2.2 PROXY –GATEWAYS DE APLICACIONES

Para evitar las debilidades asociadas al filtrado de paquetes, los desarrolladores crearon software de aplicación encargados de filtrar las conexiones. Estas aplicaciones son conocidas como Servidores Proxy y la máquina donde se ejecuta recibe el nombre de Gateway de Aplicación (pasarela de aplicación) o Bastión Host.
El Proxy, instalado sobre el Nodo Bastión, actúa de intermediario entre el cliente y el servidor real de la aplicación, siendo transparente a ambas partes.
Cuando un usuario desea un servicio, lo hace a través del Proxy. Este, realiza el pedido al servidor real devuelve los resultados al cliente. Su función fue la de analizar el tráfico de red en busca de contenido que viole la seguridad de la misma. Gráficamente:

Los servicios proxy poseen una serie de ventajas de cara a incrementar nuestra seguridad.

En primer lugar, permiten únicamente la utilización de servicios para los que existe un proxy, por lo que si en nuestra

organización la pasarela de aplicación contiene únicamente proxies para telnet, http y ftp, el resto de servicios no estarán disponibles para nadie.

Una segunda ventaja es que en la pasarela es posible filtrar protocolos basándose en algo más que la cabecera de las tramas, lo que hace posible por ejemplo tener habilitado un servicio como ftp pero con órdenes restringidas (podríamos bloquear todos los comandos put para que nadie pueda subir ficheros a un servidor).

Además, los gateways de aplicación, permiten un grado de ocultación de la estructura de la red protegida (por ejemplo, la pasarela es el único sistema cuyo nombre está disponible hacia el exterior), facilitaría la autenticación y la auditoría del tráfico sospechoso antes de que alcance el host destino y, quizás más importante, simplifica enormemente las reglas de filtrado implementadas en el router (que como hemos dicho antes pueden convertirse en la fuente de muchos problemas de seguridad):

- Sólo hemos de permitir el tráfico hacia la pasarela, bloqueando el resto.

El principal inconveniente que encontramos a la hora de instalar una pasarela de aplicación es que cada servicio que deseemos ofrecer necesita su propio proxy, además se trata de un elemento que frecuentemente es más caro que un simple filtro de paquetes, y su rendimiento es mucho menor (por ejemplo, puede llegar a limitar el ancho de banda efectivo de la red, si el análisis de cada trama es costoso). En el caso de protocolos cliente–servidor (como telnet) se añade la desventaja de que necesitamos dos pasos para conectar hacia la zona segura o hacia el resto de la red, incluso algunas implementaciones necesitan clientes modificados para funcionar correctamente.

Una variante de las pasarelas de aplicación la constituyen las pasarelas de nivel de circuito

(Circuit- level Gateways) sistemas capaces de redirigir conexiones (reenviando tramas) pero que no pueden procesar o filtrar paquetes en base al protocolo utilizado, se limitan simplemente a autenticar al usuario (a su conexión) antes de establecer el circuito virtual entre sistemas.
La principal ventaja de este tipo de pasarelas es que proveen de servicios a un amplio rango de protocolos;
no obstante, necesitan software especial que tenga las llamadas al sistema clásicas sustituidas por funciones de librerías seguras, como socks.

Monitorización de la actividad

Monitorizar la actividad de nuestro cortafuegos es algo indispensable para la seguridad de todo el perímetro protegido.
La monitorización nos facilitará información sobre los intentos de ataque que estemos sufriendo (origen, franjas horarias, tipos de acceso. . .), así como la existencia de tramas que aunque no supongan un ataque a priori sí que son al menos sospechosas.

¿Qué información debemos registrar?

Además de los registros estándar (los que incluyen estadísticas de tipos de paquetes recibidos, frecuencias, o direcciones fuente y destino se recomienda auditar información de la conexión (origen y destino, nombre de usuario – recordemos el servicio ident – hora y duración), intentos de uso de protocolos denegados, intentos de falsificación de dirección por parte de máquinas internas al perímetro de seguridad (paquetes que llegan desde la red externa con la dirección de un equipo interno) y tramas recibidas desde routers desconocidos.

Evidentemente, todos esos registros han de ser leídos con frecuencia (puedes tener un sin fin de alertas en el firewall, pero si no ves los logs del mismo ni te darás cuenta), y el administrador de la red ha de tomar medidas si se detectan actividades sospechosas, si la cantidad de logs generada es considerable nos puede interesar el uso de herramientas

que filtren dicha información.

Un excelente mecanismo para incrementar mucho nuestra seguridad puede ser la sustitución de servicios reales en el cortafuegos por programas trampa.
La idea es sencilla: se trata de pequeñas aplicaciones que simulan un determinado servicio, de forma que un posible atacante piense que dicho servicio está habilitado y prosiga su 'ataque', pero que realmente nos están enviando toda la información posible sobre el intruso.
Este tipo de programas, una especie de troyano, suele tener una finalidad múltiple: aparte de detectar y notificar ataques, el atacante permanece entretenido intentando un ataque que cree factible, lo que por un lado nos beneficia directamente – esa persona no intenta otro ataque quizás más peligroso – y por otro nos permite entretener al intruso ante una posible traza de su conexión.

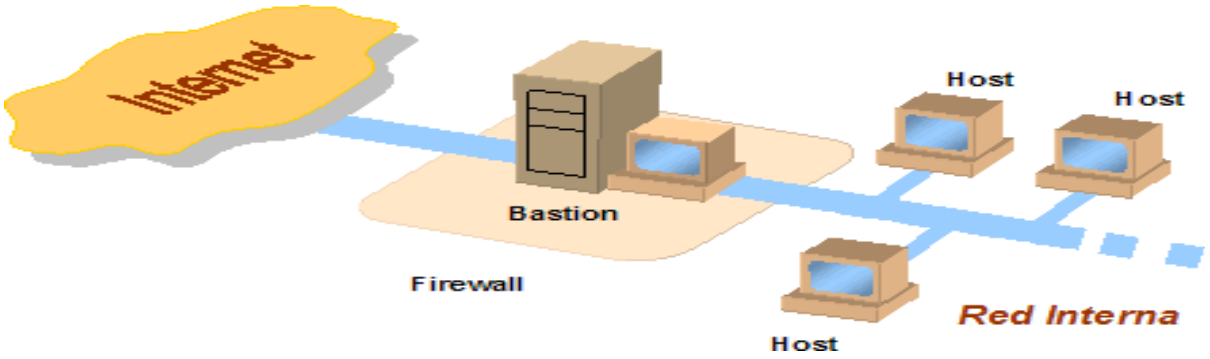
Evidentemente, nos estamos arriesgando a que nuestro atacante descubra el mecanismo y lance ataques más peligrosos, pero como el nivel de conocimientos de los atacantes de redes habituales en general no es muy elevado, este mecanismo nos permite descubrir posibles exploits utilizados por los instrusos, observar a qué tipo de atacantes nos enfrentamos, e incluso divertirnos con ellos.
En la Universidad Politécnica de Valencia existen algunos sistemas con este tipo de trampas, y realmente es curioso observar cómo algunos intrusos siguen intentando aprovechar bugs que fueron descubiertos – y solucionados – hace más de cuatro años (ejemplos típicos aquí son phf y algunos problemas de sendmail).

En un artículo clásico a la hora de hablar de seguridad, se muestra como Bill Cheswick, un experto en seguridad de los laboratorios AT&T estadounidenses, es capaz de analizar detenidamente gracias a estos programas las actividades de un intruso que intenta acceder al gateway de una compañía.

7.2.3 DUAL-HOMEDHOST

Son dispositivos que están conectados a ambos perímetros (interior y exterior) y no dejan pasar paquetes IP (como sucede en el caso del Filtrado de Paquetes), por lo que se dice que actúan con el "**IP–Forwarding desactivado**"(el ip forwarding es el reenvío de direcciones a través de la red).
Un usuario interior que desee hacer uso de un servicio exterior, deberá conectarse primero al Firewall, donde el Proxy atenderá su petición, y en función de la configuración impuesta en dicho Firewall, se conectará al servicio exterior solicitado y hará de puente entre este y el usuario interior.
Es decir que se utilizan dos conexiones. Uno desde la máquina interior hasta el Firewall y el otro desde este hasta la máquina que albergue el servicio exterior.

Este segundo modelo de cortafuegos está formado por simples máquinas Unix equipadas con dos o más tarjetas de red y denominadas anfitriones de dos bases (dual–homed hosts) o multibase (multi–homed hosts), y en las que una de las tarjetas se suele conectar a la red interna a proteger y la otra a la red externa a la organización.
En esta configuración el choke y el bastión coinciden en el mismo equipo: la máquina Unix.

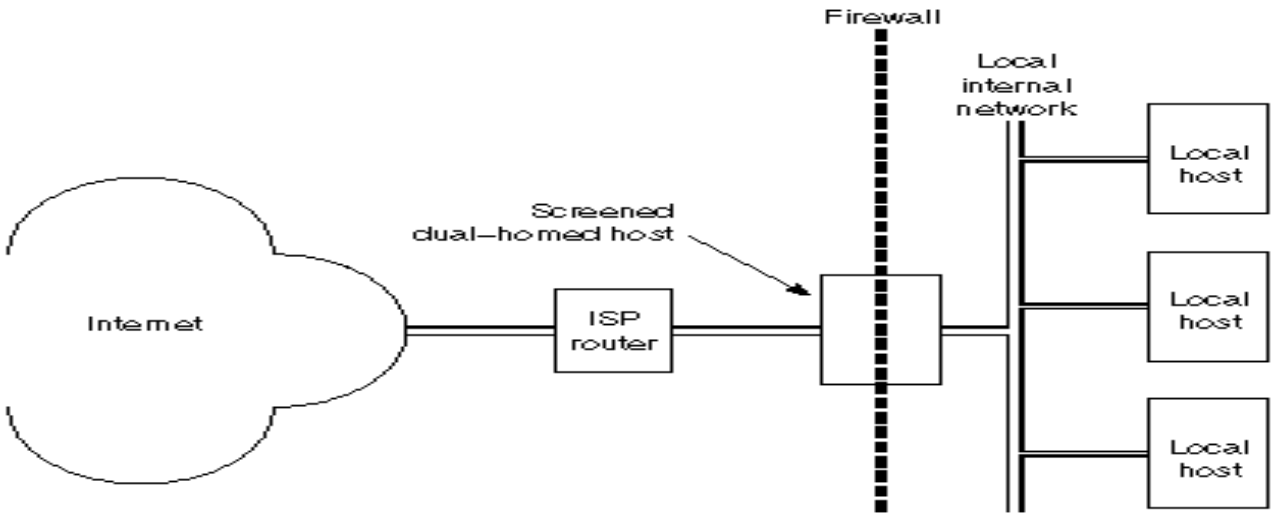


Creada por V. Ana González Hernández

El sistema ha de ejecutar al menos un servidor proxy para cada uno de los servicios que deseemos pasar a través del cortafuegos, y también es necesario que el IP Forwarding esté deshabilitado en el equipo, aunque una máquina con dos tarjetas puede actuar como un router, para aislar el tráfico entre la red interna y la externa es necesario que el choke no enrute paquetes entre ellas.
Así los sistemas externos 'verán' al host a través de una de las tarjetas y los internos a través de la otra, pero entre las dos partes no puede existir ningún tipo de tráfico que no pase por el cortafuegos: todo el intercambio de datos entre las redes se ha de realizar bien a través de servidores proxy situados en el host bastión o bien permitiendo a los usuarios conectar directamente al mismo.

La segunda de estas aproximaciones es sin duda poco recomendable, ya que un usuario que consiga aumentar su nivel de privilegios en el sistema puede romper toda la protección del cortafuegos, por ejemplo reactivando el IP Forwarding, además – esto ya no relativo a la seguridad sino a la funcionalidad del sistema – suele ser incómodo para los usuarios tener que acceder a una máquina que haga de puente entre ellos e Internet. De esta forma, la ubicación de proxies es lo más recomendable, pero puede ser problemático el configurar cierto tipo de servicios o protocolos que no se diseñaron teniendo en cuenta la existencia de un proxy entre los dos extremos de una conexión.

7.2.4 SCREENED HOST
En este caso se combina un Router con un host bastión y el principal nivel de seguridad proviene del filtrado de paquetes. En el bastión, el único sistema accesible desde el exterior, se ejecuta el Proxy de aplicaciones y en el Choke se filtran los paquetes considerados peligrosos y sólo se permiten un número reducido de servicios.



Un paso más en términos de seguridad de los cortafuegos es la arquitectura screened host o choke- gate, que combina un router con un host bastión, y donde el principal nivel de seguridad proviene del filtrado de paquetes (es decir, el router es la primera y más importante línea de defensa). En la máquina bastión, único sistema accesible desde el exterior, se ejecutan los proxies de las aplicaciones, mientras que el choke se encarga de filtrar los paquetes que se puedan considerar peligrosos para la seguridad de la red interna, permitiendo únicamente la comunicación con un reducido número de servicios.

Pero, ¿dónde situar el sistema bastión, en la red interna o en el exterior del router? La mayoría de autores recomiendan situar el router entre la red exterior y el host bastión, pero otros defienden justo lo contrario: situar el bastión en la red exterior no provoca aparentemente una degradación de la seguridad, y además ayuda al administrador a comprender la necesidad de un elevado nivel de fiabilidad en esta máquina, ya que está sujeta a ataques externos y no tiene por qué ser un host fiable, de cualquier forma, la 'no degradación' de la seguridad mediante esta aproximación es más que discutible, ya que habitualmente es más fácil de proteger un router que una máquina con un operativo de propósito general, como Unix, que además por definición ha de ofrecer ciertos servicios, no tenemos más que fijarnos en el número de problemas de seguridad que afectan a por ejemplo a IOS (el sistema operativo de los routers Cisco), muy reducido frente a los que afectan a diferentes flavours de Unix.
En todo caso, aparte de por estos matices, asumiremos la primera opción por considerarla mayoritaria entre los expertos en seguridad informática, así cuando una máquina de la red interna desea comunicarse con el exterior existen dos posibilidades:

• El choke permite la salida de algunos servicios a todas o a parte de las máquinas internas a través de un simple filtrado de paquetes.

• El choke prohíbe todo el tráfico entre máquinas de la red interna y el exterior, permitiendo sólo la salida de ciertos servicios que provienen de la máquina bastión y que han sido autorizados por la política de seguridad de la organización.

Así estamos obligando a los usuarios a que las conexiones con el exterior se realicen a través de los servidores proxy situados en el bastión.

La primera aproximación entraña un mayor nivel de complejidad a la hora de configurar las listas de control de acceso del router, mientras que si elegimos la segunda la dificultad está en configurar los servidores proxy

Creada por V. Ana González Hernández

(recordemos que no todas las aplicaciones soportan bien estos mecanismos) en el host bastión. Desde el punto de vista de la seguridad es más recomendable la segunda opción, ya que la probabilidad de dejar escapar tráfico no deseado es menor.
Por supuesto, en función de la política de seguridad que definamos en nuestro entorno, se pueden combinar ambas aproximaciones, por ejemplo permitiendo el tráfico entre las máquinas internas y el exterior de ciertos protocolos difíciles de encaminar a través de un proxy o sencillamente que no entra en mucho riesgo para nuestra seguridad (típicamente, ntp, dns. . .), y obligando para el resto de servicios a utilizar el host bastión.

La arquitectura screened host puede parecer a primera vista más peligrosa que la basada en una simple máquina con varias interfaces de red, en primer lugar, tenemos no uno sino dos sistemas accesibles desde el exterior, por lo que ambos han de ser configurados con las máximas medidas de seguridad.
Además, la mayor complejidad de diseño hace más fácil la presencia de errores que puedan desembocar en una violación de la política implantada, mientras que con un host con dos tarjetas nos aseguramos de que únicamente aquellos servicios con un proxy configurado podrán generar tráfico entre la red externa y la interna (a no ser que por error activemos el IP Forwarding).

Sin embargo, aunque estos problemas son reales, se solventan tomando las precauciones necesarias a la hora de diseñar e implantar el cortafuegos y definiendo una política de seguridad correcta.
De cualquier forma, en la práctica esta arquitectura de cortafuegos está cada vez más en desuso debido a que presenta dos puntos únicos de fallo, el choke y el bastión: si un atacante consigue controlar cualquiera de ellos, tiene acceso a toda la red protegida; por tanto, es más popular, y recomendable, una arquitectura screened subnet, de la que vamos a hablar a continuación.

7.2.5 SCREENED SUBNET
En este diseño se intenta aislar la máquina más atacada y vulnerable del Firewall, el Nodo Bastión.
Para ello se establece una Zona Desmilitarizada (DMZ) de forma tal que sin un intruso accede a esta máquina no consiga el acceso total a la subred protegida.
En este esquema se utilizan dos Routers: uno exterior y otro interior. El Router exterior tiene la misión de bloquear el tráfico no deseado en ambos sentidos: hacia la red interna y hacia la red externa. El Router interior hace lo mismo con la red interna y la DMZ (zona entre el Router externo y el interno).

Es posible definir varias niveles de DMZ agregando más Routers, pero destacando que las reglas aplicadas a cada uno deben ser distintas ya que en caso contrario los niveles se simplificarían a uno solo.

Como puede apreciarse la Zona Desmilitarizada aísla físicamente los servicios internos, separándolos de los servicios públicos. Además, no existe una conexión directa entre la red interna y la externa.
Los sistemas Dual–Homed Host y Screnned pueden ser complicados de configurar y comprobar, lo que puede dar lugar, paradójicamente, a importantes agujeros de seguridad en toda la red.
En cambio, si se encuentran bien configurados y administrados pueden brindar un alto grado de protección y ciertas ventajas:

1. Ocultamiento de la información: los sistemas externos no deben conocer el nombre de los sistemas internos. El Gateway de aplicaciones es el único autorizado a conectarse con el exterior y el encargado de bloquear la información no solicitada o sospechosa.

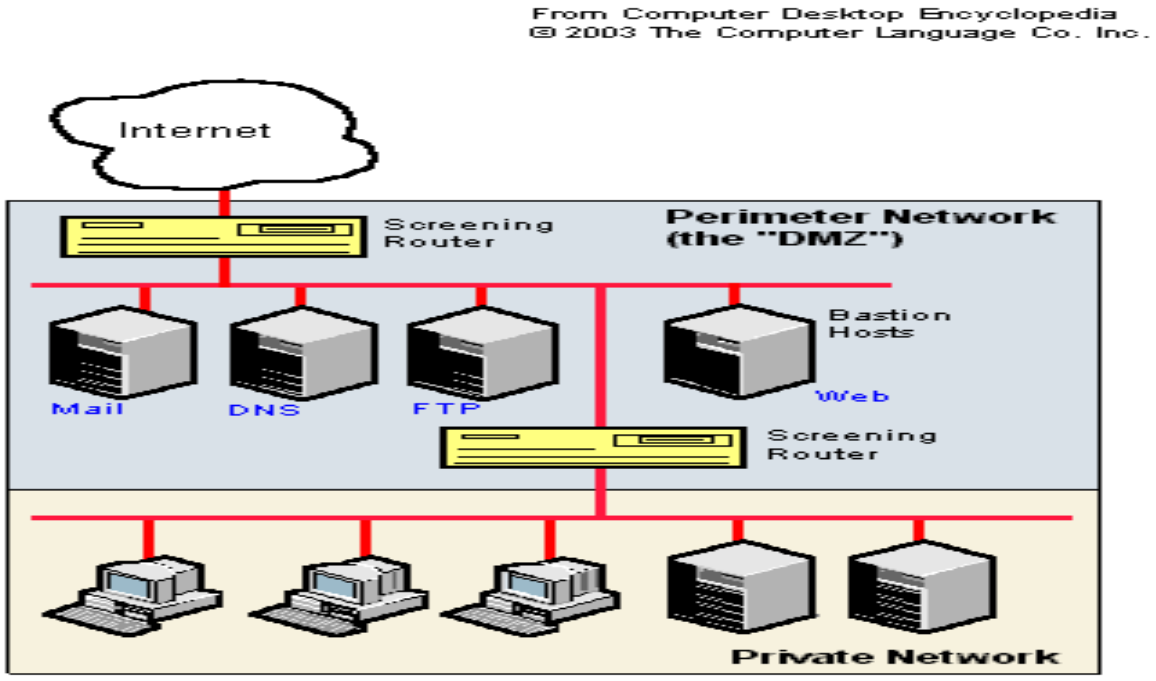
2. Registro de actividades y autenticación robusta: El Gateway requiere de autenticación cuando se realiza un pedido de datos externos. El registro de actividades se realiza en base a estas solicitudes.

3. Reglas de filtrado menos complejas: Las reglas del filtrado de los paquetes por parte del Router serán menos

compleja dado a que él sólo debe atender las solicitudes del Gateway.
Así mismo tiene la desventaja de ser intrusivos y no transparentes para el usuario ya que generalmente este debe instalar algún tipo de aplicación especializada para lograr la comunicación. Se suma a esto que generalmente son más lentos porque deben revisar todo el tráfico de la red.

La arquitectura Screened Subnet, también conocida como red perimétrica o De-Militarized Zone (DMZ) es con diferencia la más utilizada e implantada hoy en día ya que añade un nivel de seguridad en las arquitecturas de cortafuegos situando una subred (DMZ) entre las redes externa e interna, de forma que se consiguen reducir los efectos de un ataque exitoso al host bastión: como hemos venido comentando, en los modelos anteriores toda la seguridad se centraba en el bastión1 , de forma que si la seguridad del mismo se veía comprometida, la amenaza se extendía automáticamente al resto de la red. Como la máquina bastión es un objetivo interesante para muchos instrusos, la arquitectura DMZ intenta aislarla en una red perimétrica de forma que un intruso que accede a esta máquina no consiga un acceso total a la subred protegida.

Screened subnet es la arquitectura más segura, pero también la más compleja; se utilizan dos routers, denominados exterior e interior, conectados ambos a la red perimétrica como se muestra a continuación:



En esta red perimétrica, que constituye el sistema cortafuegos, se incluye el host bastión y también se podrá incluir sistemas que requieran un acceso controlado, como baterías de módems o el servidor de correo, que serán los únicos elementos visibles desde fuera de nuestra red.

El router exterior tiene como misión bloquear el tráfico no deseado en ambos sentidos (hacia la red perimétrica y hacia la red externa), mientras que el interior hace lo mismo pero con el tráfico entre la red interna y la perimétrica, así un atacante habrá de romper la seguridad de ambos routers para acceder a la red protegida, incluso es posible implementar una zona desmilitarizada con un único router que posea tres o más interfaces de red, pero en este caso si se compromete este único elemento se rompe toda nuestra seguridad, frente al caso general en que hay que comprometer ambos, tanto el externo como el interno.

También podemos, si necesitamos mayores niveles niveles de seguridad, definir varias redes perimétricas en serie, situando los servicios que requieran de menor fiabilidad en las redes más externas: así el atacante habrá de saltar por todas y cada una de ellas para acceder a nuestros equipos; evidentemente, si en cada red perimétrica se siguen las mismas reglas de filtrado, niveles adicionales no proporcionan mayor seguridad.

Esta arquitectura de cortafuegos elimina los puntos únicos de fallo presentes en las anteriores:

Antes de llegar al bastión (por definición, el sistema más vulnerable) un atacante ha de saltarse las medidas de seguridad impuestas por el enrutador externo. Si lo consigue, como hemos aislado la máquina bastión en una subred estamos reduciendo el impacto de un atacante que logre controlarlo, ya que antes de llegar a la red interna ha de comprometer también al segundo router; en este caso extremo (si un intruso logra comprometer el segundo router), la arquitectura DMZ no es mejor que un screened host.
Por supuesto, en cualquiera de los tres casos (compromiso del router externo, del host bastión, o del router interno) las actividades de un intruso pueden violar nuestra seguridad, pero de forma parcial, por ejemplo, simplemente accediendo al primer enrutador puede aislar toda nuestra organización del exterior, creando una negación de servicio importante, pero esto suele ser menos grave que si lograra acceso a la red protegida.

Excepto en el primero, compuesto únicamente por un choke
Aunque, como hemos dicho antes, la arquitectura DMZ es la que mayores niveles de seguridad puede proporcionar, no se trata de la panacea de los cortafuegos. Evidentemente existen problemas relacionados con este modelo: por ejemplo, se puede utilizar el firewall para que los servicios fiables pasen directamente sin acceder al bastión, lo que puede dar lugar a un incumplimiento de la política de la organización. Un segundo problema, quizás más grave, es que la mayor parte de la seguridad reside en los routers utilizados; como hemos dicho antes las reglas de filtrado sobre estos elementos pueden ser complicadas de configurar y comprobar, lo que puede dar lugar a errores que abran importantes brechas de seguridad en nuestro sistema.

Algo que puede incrementar en gran medida nuestra seguridad y al mismo tiempo facilitar la administración de los cortafuegos es utilizar un bastión diferente para cada protocolo o servicio en lugar de uno sólo, sin embargo, esta arquitectura presenta el grave inconveniente de la cantidad de máquinas necesarias para implementar el firewall, lo que impide que muchas organizaciones la puedan adoptar. Una variante más barata puede consistir en utilizar un único bastión pero servidores proxy diferentes para cada servicio ofertado.

Cada día es más habitual en todo tipo de organizaciones dividir su red en diferentes subredes, esto es especialmente aplicable en entornos de I+D o empresas medianas, donde con frecuencia se han de conectar campus o sucursales separadas geográficamente, edificios o laboratorios diferentes, etc.
En esta situación es recomendable incrementar los niveles de seguridad de las zonas más comprometidas (por ejemplo, un servidor donde se almacenen expedientes o datos administrativos del personal) insertando cortafuegos internos entre estas zonas y el resto de la red.
Aparte de incrementar la seguridad, firewalls internos son especialmente recomendables en zonas de la red desde la que no se permite a priori la conexión con Internet, como laboratorios de prácticas: un simple PC con Linux o FreeBSD que deniegue cualquier conexión con el exterior del campus va a ser suficiente para evitar que los usuarios se dediquen a conectar a páginas web o chats desde equipos no destinados a estos usos.

Concretamente en el caso de redes de universidades será muy interesante filtrar las conexiones a irc o a muds, ya sea a nivel de aulas o laboratorios o a nivel de todo el campus, denegando en el router de salida de la red hacia INet cualquier tráfico a los puertos 6667, 8888 y similares, aunque realmente esto no evitaría que todos los usuarios siguieran jugando desde los equipos de la universidad – por ejemplo a través de un servidor que disponga de conexión

en otros puertos –, sí conseguirá que la mayor parte de ellos dejara de hacerlo.

RESTRICCIONES EN EL FIREWALL

La parte más importante de las tareas que realizan los Firewalls, la de permitir o denegar determinados servicios, se hacen en función de los distintos usuarios y su ubicación:

1. Usuarios internos con permiso de salida para servicios restringidos: permite especificar una serie de redes y direcciones a los que denomina Trusted (validados). Estos usuarios, cuando provengan del interior, van a poder acceder a determinados servicios externos que se han definido.

2. Usuarios externos con permiso de entrada desde el exterior: este es el caso más sensible a la hora de vigilarse. Suele tratarse de usuarios externos que por algún motivo deben acceder para consultar servicios de la red interna. También es habitual utilizar estos accesos por parte de terceros para prestar servicios al perímetro interior de la red. Sería conveniente que estas cuentas sean activadas y desactivadas bajo demanda y únicamente el tiempo real que sean necesarias.

ACCESS CONTROL LISTS (ACL)

Las Listas de Control de Accesos proveen de un nivel de seguridad adicional a los clásicos provistos por los Sistemas Operativos. Estas listas permiten definir permisos a usuarios y grupos concretos. Por ejemplo pueden definirse sobre un Proxy una lista de todos los usuarios (o grupos de ellos) a quien se le permite el acceso a Internet, FTP, etc. También podrán definirse otras características como limitaciones de anchos de banda y horarios.

Beneficios de un firewall

Administra los accesos posibles del Internet a la red privada.

Protege a los servidores propios del sistema de ataques de otros servidores en Internet.

Permite al administrador de la red definir un "choke point" (embudo), manteniendo al margen los usuarios no-autorizados, prohibiendo potencialmente la entrada o salida al vulnerar los servicios de la red.

Ofrece un punto donde la seguridad puede ser monitoreada.

Ofrece un punto de reunión para la organización. Si una de sus metas es proporcionar y entregar servicios información a consumidores, el firewall es ideal para desplegar servidores WWW y FTP.

Limitación de un Firewalls

Puede únicamente autorizar el paso del trafico, y él mismo podrá ser inmune a la penetración. Desafortunadamente, este sistema no puede ofrecer protección alguna una vez que el agresor lo traspasa o permanece en torno a éste.

No puede proteger contra aquellos ataques que se efectúen fuera de su punto de operación.

No puede proteger de las amenazas a que está sometido por traidores o usuarios inconscientes.

No puede prohibir que los traidores o espías corporativos copien datos sensitivos y los substraigan de la empresa.

No puede proteger contra los ataques de la "Ingeniería Social", por ejemplo un Hacker que pretende ser un supervisor o un nuevo empleado despistado.

No puede proteger contra los ataques posibles a la red interna por virus informativos a través de archivos y software.

Y hasta aquí la teoría del tema 7, a partir de aquí os muestro como se configuran diferentes reglas en diferentes firewalls pero esta parte ya no es obligatoria y como hay niveles muy dispares queda reflejado para quien le resulte interesante lo entienda y se sienta con ganas o curiosidad para ello, decir que la parte de iptables es especialemente interesante.

7.3 Parte práctica o firewalls de aplicación que podemos curiosear

No hablaré del firewall de windows en detalle pero os diré que todas las opciones que muestra se resumen en:

Activar o desactivar el firewall de windows,

Permitir un programa a través de él,

Bloquear todas las conexiones entrantes, incluidas las de la lista de aplicaciones permitidas,

Notificarme cuando Firewall de Windows bloquee una nueva aplicación y sólo como configuración avanza permite la modificación de las reglas.

Como sistema propietario que es ofrece su propio soporte así que a los os interese este firewall en concreto podéis dirigiros a http://windows.microsoft.com/es-es/windows-8/windows-firewall-from-start-to-finish

Para el resto, pasemos a ver y configurar nuestras reglas para ver de verdad como funciona un firewall:

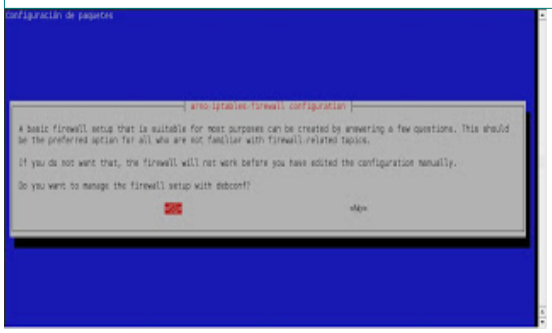
Arno iptables

Arno es un firewall basado en iptables de fácil manejo e instalación, veamóslo paso a paso:

1.Nos logueamos en una terminal gráfica o texto (CTRL+ALT+F1...F6) como usuario root e instalamos el paquete "arno-iptables-firewall":

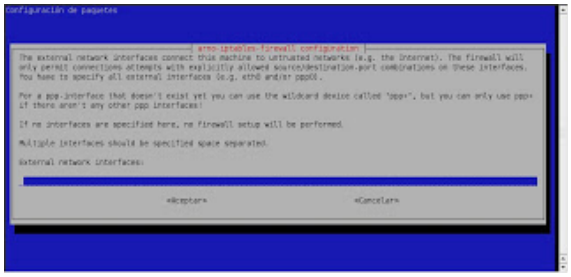
```
$su -password:
#aptitude install arno-iptables-firewall

Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado extendido
Inicializando el estado de los paquetes... Hecho
Leyendo las descripciones de las tareas... Hecho
Construir la base de datos de etiquetas... Hecho
Se instalarán automáticamente los siguientes paquetes NUEVOS:
gawk
Se instalarán los siguiente paquetes NUEVOS:
arno-iptables-firewall gawk
0 paquetes actualizados, 2 nuevos instalados, 0 para eliminar y 0 sin actualizar.
Necesito descargar 792kB de ficheros. Después de desempaquetar se usarán 2511kB.
¿Quiere continuar? [Y/n/?] y
escribiendo información de estado extendido... Hecho
Des:1 http://ftp.rediris.es etch/main gawk 1:3.1.5.dfsg-4 [694kB]
Des:2 http://ftp.rediris.es etch/main arno-iptables-firewall 1.8.8.c-1 [97,7kB]
Descargados 792kB en 44s (17,8kB/s).
```

2. Una vez instalado, nos aparecerá la siguiente ventana que nos pregunta si queremos configurar el paquete mediante debconf.



RESPUESTA: yes [ENTER]

3. En la siguiente ventana debemos especificar la interfaz de red por la que salimos a internet. Como me conecto a través de un modem/router y sólo tengo una tarjeta de red, la interface es "eth0".

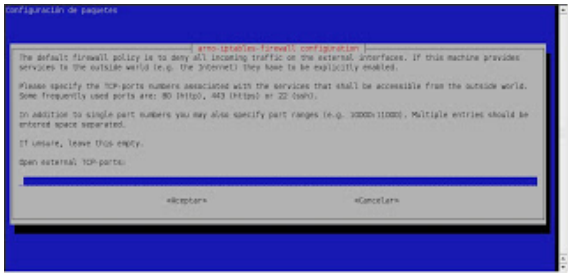


RESPUESTA: eth0 para configurar la interfaz de red ethernet y wlan0 para configurar la interfaz wifi[ENTER]

4. Ahora debemos especificar que puertos TCP necesitamos tener abiertos en nuestro firewall.

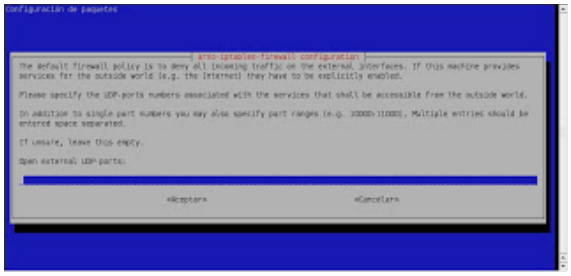
EJEMPLO:
- Amule: Abrir los puertos 4661 TCP, 4664 UDP
- Servidor SSH: abrir el puerto 22

NOTA:
El firewall de nuestro router debe tener abiertos también éstos puertos.



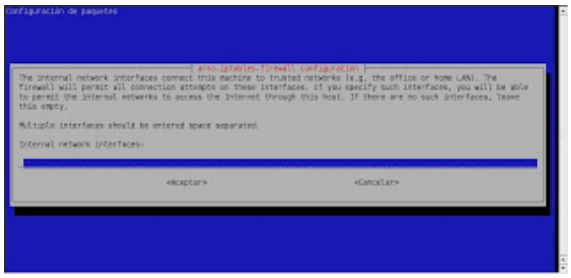
RESPUESTA: 4661 22 [ENTER]

5. Nos pedirá que puertos UDP queremos abrir:



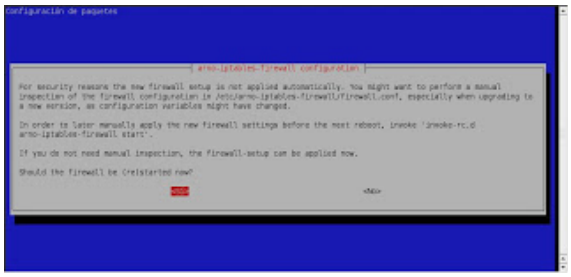
RESPUESTA: 4664 [ENTER]

6. La ventana siguiente sólo deberemos configurarla en el caso de que tengamos varias tarjetas de red (eth0, eth1...) y una de ellas, esté configurada hacia nuestra red local a la cual queremos permitirle cualquier conexión a nuestro equipo.



RESPUESTA: dejar_en_blanco [ENTER]

7. Finalmente iniciamos nuestro firewall.



RESPUESTA: yes [ENTER]

Escribiendo información de estado extendido... Hecho
Preconfigurando paquetes ...
Seleccionando el paquete arno-iptables-firewall previamente no seleccionado.
(Leyendo la base de datos ...
88957 ficheros y directorios instalados actualmente.)
Desempaquetando arno-iptables-firewall (de .../arno-iptables-firewall_1.8.8.c-1_all.deb) ...
Configurando arno-iptables-firewall (1.8.8.c-1) ...

```
Arno's Iptables Firewall Script v1.8.8c
--------------------------------------------------------------------------------
Sanity checks passed...OK
Detected IPTABLES module... Loading additional IPTABLES modules:
All IPTABLES modules loaded!
Configuring /proc/.... settings:
Enabling anti-spoof with rp_filter
Enabling SYN-flood protection via SYN-cookies
Disabling the logging of martians
Disabling the acception of ICMP-redirect messages
Setting the max. amount of simultaneous connections to 16384
Enabling protection against source routed packets
Setting default conntrack timeouts
Enabling reduction of the DoS'ing ability
Setting Default TTL=64
Disabling ECN (Explicit Congestion Notification)
Enabling support for dynamic IP's
Flushing route table
/proc/ setup done...
Flushing rules in the filter table
Setting default (secure) policies
Using loglevel "info" for syslogd

Setting up firewall rules:
--------------------------------------------------------------------------------
Accepting packets from the local loopback device
Enabling setting the maximum packet size via MSS
Enabling mangling TOS
Logging of stealth scans (nmap probes etc.) enabled
Logging of packets with bad TCP-flags enabled
Logging of INVALID packets disabled
Logging of fragmented packets enabled
Logging of access from reserved addresses enabled
Setting up anti-spoof rules
Reading custom IPTABLES rules from /etc/arno-iptables-firewall/custom-rules
Loading (user) plugins
Setting up INPUT policy for the external net (INET):
Enabling support for a DHCP assigned IP on external interface(s): eth0
Logging of explicitly blocked hosts enabled
Logging of denied local output connections enabled
Packets will NOT be checked for private source addresses
Allowing the whole world to connect to TCP port(s): 4661 22
Allowing the whole world to connect to UDP port(s): 4664
Denying the whole world to send ICMP-requests(ping)
Logging of dropped ICMP-request(ping) packets enabled
Logging of dropped other ICMP packets enabled
Logging of possible stealth scans enabled
Logging of (other) connection attempts to PRIVILEGED TCP ports enabled
Logging of (other) connection attempts to PRIVILEGED UDP ports enabled
```

Logging of (other) connection attempts to UNPRIVILEGED TCP ports enabled
Logging of (other) connection attempts to UNPRIVILEGED UDP ports enabled
Logging of other IP protocols (non TCP/UDP/ICMP) connection attempts enabled
Logging of ICMP flooding enabled
Applying INET policy to external (INET) interface: eth0 (without an external subnet specified)
Security is ENFORCED for external interface(s) in the FORWARD chain

Apr 02 23:36:20 All firewall rules applied.

Ya tendremos nuestro firewall configurado.
Podemos volver a ejecutar la configuración del firewall con el siguiente comando:

#dpkg-reconfigure arno-iptables-firewall

Si lo que queremos es deshabilitar temporalmente el cortafuegos:

#/etc/init.d/arno-iptables-firewall stop

Para habilitarlo de nuevo:
#/etc/init.d/arno-iptables-firewall start

Este es el cortafuegos más sencillo que yo conozca que existe para linux, viene con las reglas preconfiguradas pero aún así podemos verlas.

Ahora veamos algo más manual:

IPTABLES

Qué es iptables

IPtables es un sistema de firewall vinculado al kernel de linux que se ha extendido enormemente a partir del kernel 2.4 de este sistema operativo. Al igual que el anterior sistema ipchains, un firewall de iptables no es como un servidor que lo iniciamos o detenemos o que se pueda caer por un error de programación(esto es una pequeña mentira, ha tenido alguna vulnerabilidad que permite DoS, pero nunca tendrá tanto peligro como las aplicaciones que escuchan en determinado puerto TCP): iptables esta integrado con el kernel, es parte del sistema operativo.

¿Cómo se pone en marcha? Realmente lo que se hace es aplicar reglas. Para ellos se ejecuta el comando iptables, con el que añadimos, borramos, o creamos reglas. Por ello un firewall de iptables no es sino un simple script de shell en el que se van ejecutando las reglas de firewall.

Notas: bueno, para los más geeks, Vale, se puede implementar un script de inicio en /etc/rc.d/INIT.d (o /etc/INIT.d) con el que hagamos que iptables se "inicie o pare" como un servidor más.

Lo podemos hacer nosotros o es probable que venga en la distribución (como en redhat por ejemplo). También se pueden salvar las reglas aplicadas con el comando iptables-save en un fichero y gestionar ese fichero con una aplicación o front-end desde la X o desde webmin.

Vale, tenemos una máquina linux con soporte para iptables, tiene reglas aplicadas y empiezan a llegar/salir/pasar

paquetes. No nos liemos: olvidemos cuantas tarjetas de red hay, que direcciones ip tiene la máquina y olvidemos si el paquete entra o sale. Las reglas de firewall están a nivel de kernel, y al kernel lo que le llega es un paquete (digamos, un marrón ;)) y tiene que decidir que hacer con él.

El kernel lo que hace es, dependiendo si el paquete es para la propia maquina o para otra maquina, consultar las reglas de firewall y decidir que hacer con el paquete según mande el firewall. Este es el camino que seguiría un paquete en el kernel:

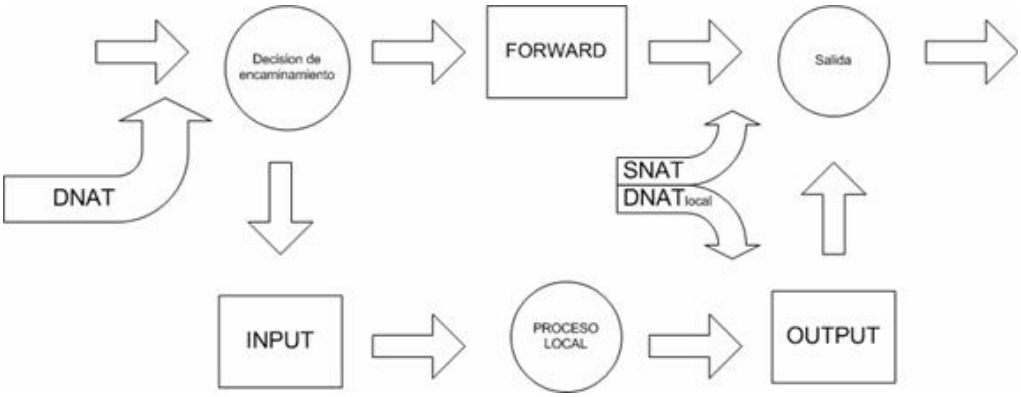


Figura 5: cuando un paquete u otra comunicación llega al kernel con iptables se sigue este camino

Como se ve en el gráfico, básicamente se mira si el paquete esta destinado a la propia maquina o si va a otra. Para los paquetes (o datagramas, según el protocolo) que van a la propia maquina se aplican las reglas INPUT y OUTPUT, y para filtrar paquetes que van a otras redes o maquinas se aplican simplemente reglas FORWARD.
INPUT,OUTPUT y FORWARD son los tres tipos de reglas de filtrado.

Pero antes de aplicar esas reglas es posible aplicar reglas de NAT: estas se usan para hacer redirecciones de puertos o cambios en las IPs de origen y destino. Veremos ejemplos.
E incluso antes de las reglas de NAT se pueden meter reglas de tipo MANGLE, destinadas a modificar los paquetes; son reglas poco conocidas y es probable que no las usen.
Por tanto tenemos tres tipos de reglas en iptables:
- MANGLE
- NAT: reglas PREROUTING, POSTROUTING
- FILTER: reglas INPUT, OUTPUT, FORWARD.

Al grano: creando un firewall con iptables

Empezemos a ver configuraciones de firewall con iptables, empezando desde la más básica a las más complejas, en las que se establece la denegación como política por defecto.

Nota: se recomienda encarecidamente ir practicando estas reglas en alguna maquina linux disponible, y especialmente hacer uso de la herramienta iptraf para depurar y comprobar el funcionamiento de iptables. Con iptraf podemos comprobar si las conexiones TCP/IP se llegan a establecer o no.

Una conexión tcp/ip empieza con el three-way-handshake tal como hemos visto en el tema 3:

- La maquina que desea conectarse a otra envía un paquete con flan SYN
- Si la otra maquina acepta, envía un SYN/ACK
- Entonces la máquina establece la conexión.

Si el firewall esta denegando la conexión, con iptraf veremos que la maquina origen solo manda paquetes con el flags (de SYN), y que del otro lado no sale nada.

Saber usar iptraf nos ayudará mucho.

3.1 Proteger la propia máquina

Muy bien, tenemos una máquina linux pinchada en internet y queremos protegerla con su propio firewall.

Lo único que tenemos que hacer es crear un script de shell en el que se van aplicando las reglas.
Los scripts de iptables pueden tener este aspecto:

Saludo a la afición (echo)
Borrado de las reglas aplicadas actualmente (flush)
Aplicación de políticas por defecto para INPUT, OUPUT, FORWARD
Listado de reglas iptables.

Ojo con el orden de las reglas!

```
#!/bin/sh
## SCRIPT de IPTABLES - ejemplo del manual de iptables
## Ejemplo de script para proteger la propia máquina
## Pello Xabier Altadill Izura
## www.pello.info - pello@pello.info

echo -n Aplicando Reglas de Firewall...

## FLUSH de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

## Establecemos politica por defecto
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

## Empezamos a filtrar

# El localhost se deja (por ejemplo conexiones locales a mysql)
/sbin/iptables -A INPUT -i lo -j ACCEPT

# A nuestra IP le dejamos todo
iptables -A INPUT -s 195.65.34.234 -j ACCEPT

# A un colega le dejamos entrar al mysql para que mantenga la BBDD
iptables -A INPUT -s 231.45.134.23 -p tcp --dport 3306 -j ACCEPT
```

```
# A un diseñador le dejamos usar el FTP
iptables -A INPUT -s 80.37.45.194 -p tcp -dport 20:21 -j ACCEPT

# El puerto 80 de www debe estar abierto, es un servidor web.
iptables -A INPUT -p tcp --dport 80 -j ACCEPT

# Y el resto, lo cerramos
iptables -A INPUT -p tcp --dport 20:21 -j DROP
iptables -A INPUT -p tcp --dport 3306 -j DROP
iptables -A INPUT -p tcp --dport 22 -j DROP
iptables -A INPUT -p tcp --dport 10000 -j DROP

echo " OK . Verifique que lo que se aplica con: iptables -L -n"

# Fin del script
```

Nota para freaks y geeks: siiii, que ya lo se, se puede mejorar este script usando variables, se puede poner el comando con el path completo, pero limítense a hacer copy-paste. Para el resto de mortales, no olvidarse de ponerle flags de ejecución: chmod +x firewall1.sh o chmod 750 firewall1.sh

En fin, ya se ve, un script de los más simple, con unas pocas reglas con las que cerramos puertos al público a los que no tienen porque tener acceso, salvo el 80.

Pero cualquiera con algo de ojo se habrá dado cuenta de que ni se filtra el UDP ni el ICMP. Apostaría cualquier cosa a que el sistema tiene algún puerto udp abierto, y además peligroso como el SNMP. Como he dicho anteriormente, en este tipo de firewall es recordable hacer un netstat para ver que puertos están en estado de escucha (abiertos), y salve que un rootkit nos haya modificado los binarios, netstat nos dará la información precisa que necesitamos. Hay gente que se decanta por hacerse un nmap así mismos.

Cuidado: dependiendo de cómo lo ejecutemos quizá no nos muestre todos los puertos, ya que suele mirar los bien conocidos.
Imaginemos que hemos dado un repaso a nuestro sistema, y ahora si que tenemos mejor identificados los puertos tcp y udp abiertos. Pero por si acaso nos curamos en salud y al final del script cerraremos el rango de puertos del 1 al 1024, los reservados tanto para tcp como udp.

```
#!/bin/sh
## SCRIPT de IPTABLES - ejemplo del manual de iptables
## Ejemplo de script para proteger la propia máquina
## Pello Xabier Altadill Izura
## www.pello.info - pello@pello.info

echo -n Aplicando Reglas de Firewall...

## FLUSH de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

## Establecemos política por defecto
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT
```

```
## Empezamos a filtrar

# El localhost se deja (por ejemplo conexiones locales a mysql)
/sbin/iptables -A INPUT -i lo -j ACCEPT

# A nuestra IP le dejamos todo
iptables -A INPUT -s 195.65.34.234 -j ACCEPT

# A un colega le dejamos entrar al mysql para que mantenga la BBDD
iptables -A INPUT -s 231.45.134.23 -p tcp --dport 3306 -j ACCEPT

# A un diseñador le dejamos usar el FTP
iptables -A INPUT -s 80.37.45.194 -p tcp -dport 20:21 -j ACCEPT

# El puerto 80 de www debe estar abierto, es un servidor web.
iptables -A INPUT -p tcp --dport 80 -j ACCEPT

# Cerramos rango de los puertos privilegiados. Cuidado con este tipo de
# barreras, antes hay que abrir a los que si tienen acceso.
iptables -A INPUT -p tcp --dport 1:1024 -j DROP
iptables -A INPUT -p udp --dport 1:1024 -j DROP

# Cerramos otros puertos que estan abiertos
iptables -A INPUT -p tcp --dport 3306 -j DROP
iptables -A INPUT -p tcp --dport 10000 -j DROP
iptables -A INPUT -p udp --dport 10000 -j DROP

echo " OK . Verifique que lo que se aplica con: iptables -L -n"

# Fin del script
```

¿Sencillo, no? Ahora basta con hacer copy-paste de estas reglas y aplicarlas y ajustarlas en su sistema (quizás uses PostgreSQL). Si tiene miedo de perder el control de una máquina remota, pruebe el script en una máquina local y asegúrese de que aplica lo que usted quiere. Funcionar va a funcionar seguro.

Firewall de una LAN con salida a internet

Ahora vamos a ver una configuración de firewall iptables para el típico caso de red local que necesita salida a internet.

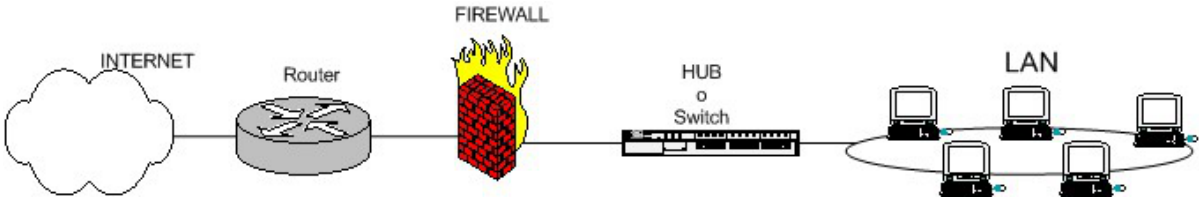


Figura 6: esquema de firewall típico entre red local e internet

¿Qué es lo que hace falta? Obviamente, una regla que haga NAT hacia fuera (enmascaramiento en iptables), con lo que se haría dos veces NAT en el firewall y en el router. Entre el router y el firewall lo normal es que haya una red privada (192.168.1.1 y 192.168.1.2 por ejemplo), aunque dependiendo de las necesidades puede que los dos tengan IP pública. El router se supone que hace un NAT completo hacia dentro (quizá salvo puerto 23), o sea que desde el exterior no se llega al router si no que de forma transparente se "choca" contra el firewall. Lo normal en este tipo de firewalls es poner la política por defecto de FORWARD en denegar (DROP), pero eso lo vemos más adelante. Veamos como sería este firewall-gateway:

```
#!/bin/sh
## SCRIPT de IPTABLES - ejemplo del manual de iptables
```

```
## Ejemplo de script para firewall entre red-local e internet
##
## Pello Xabier Altadill Izura
## www.pello.info - pello@pello.info

echo -n Aplicando Reglas de Firewall...

## FLUSH de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

## Establecemos politica por defecto
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

## Empezamos a filtrar
## Nota: eth0 es el interfaz conectado al router y eth1 a la LAN
# El localhost se deja (por ejemplo conexiones locales a mysql)
/sbin/iptables -A INPUT -i lo -j ACCEPT

# Al firewall tenemos acceso desde la red local
iptables -A INPUT -s 192.168.10.0/24 -i eth1 -j ACCEPT

# Ahora hacemos enmascaramiento de la red local
# y activamos el BIT DE FORWARDING (imprescindible!!!!!)
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -o eth0 -j MASQUERADE

# Con esto permitimos hacer forward de paquetes en el firewall, o sea
# que otras máquinas puedan salir a traves del firewall.
echo 1 > /proc/sys/net/ipv4/ip_forward

## Y ahora cerramos los accesos indeseados del exterior:
# Nota: 0.0.0.0/0 significa: cualquier red

# Cerramos el rango de puerto bien conocido
iptables -A INPUT -s 0.0.0.0/0 -p tcp -dport 1:1024 -j DROP
iptables -A INPUT -s 0.0.0.0/0 -p udp -dport 1:1024 -j DROP

# Cerramos un puerto de gestión: webmin
iptables -A INPUT -s 0.0.0.0/0 -p tcp -dport 10000 -j DROP

echo " OK . Verifique que lo que se aplica con: iptables -L -n"

# Fin del script
```

Firewall de una LAN con salida a internet con DMZ

Bueno, esto se va complicando. Imaginemos que tenemos una red parecida a la anterior pero ahora hacemos las cosas bien y colocamos ese servidor IIS en una DMZ:

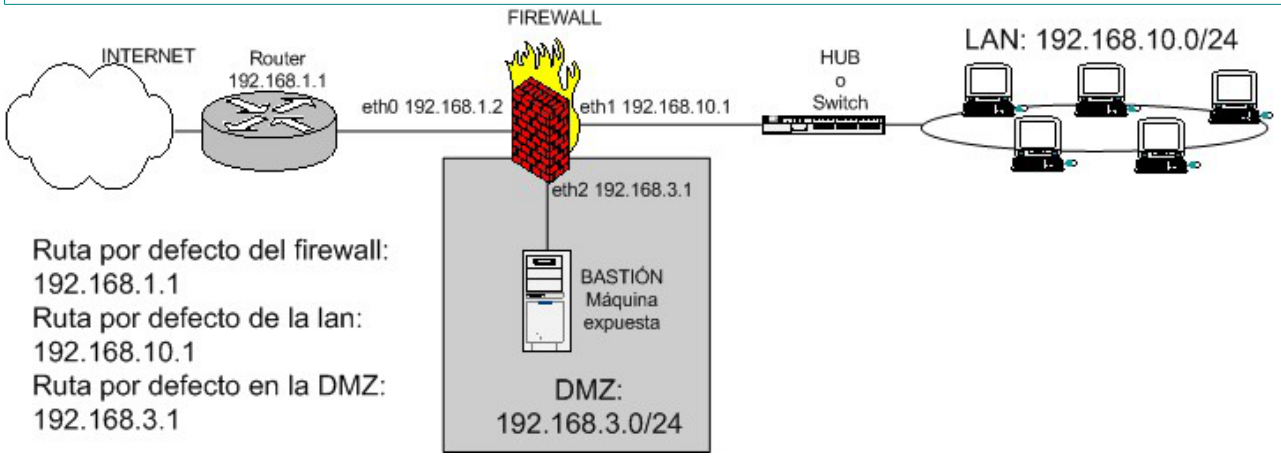


Figura 7: esquema de firewall entre red local e internet con zona DMZ para servidores expuestos

En este tipo de firewall hay que permitir:
- Acceso de la red local a internet.
- Acceso público al puerto tcp/80 y tcp/443 del servidor de la DMZ
- Acceso del servidor de la DMZ a una BBDD de la LAN
- Obviamente bloquear el resto de acceso de la DMZ hacia la LAN.

¿Qué tipo de reglas son las que hay que usar para filtrar el tráfico entre la DMZ y la LAN?

Solo pueden ser las FORWARD, ya que estamos filtrando entre distintas redes, no son paquetes destinados al propio firewall.

```
#!/bin/sh
## SCRIPT de IPTABLES - ejemplo del manual de iptables
## Ejemplo de script para firewall entre red-local e internet con DMZ
##
## Pello Xabier Altadill Izura
## www.pello.info - pello@pello.info

echo -n Aplicando Reglas de Firewall...

## FLUSH de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

## Establecemos política por defecto
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

## Empezamos a filtrar
## Nota: eth0 es el interfaz conectado al router y eth1 a la LAN
# Todo lo que venga por el exterior y vaya al puerto 80 lo redirigimos
# a una maquina interna
```

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j DNAT --to 192.168.3.2:80

# Los accesos de un ip determinada HTTPS se redirigen e esa
# maquina
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 443 -j DNAT --to 192.168.3.2:443

# El localhost se deja (por ejemplo conexiones locales a mysql)
/sbin/iptables -A INPUT -i lo -j ACCEPT

# Al firewall tenemos acceso desde la red local
iptables -A INPUT -s 192.168.10.0/24 -i eth1 -j ACCEPT

# Ahora hacemos enmascaramiento de la red local y de la DMZ
# para que puedan salir haca fuera
# y activamos el BIT DE FORWARDING (imprescindible!!!!!)
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -o eth0 -j MASQUERADE
iptables -t nat -A POSTROUTING -s 192.168.3.0/24 -o eth0 -j MASQUERADE

# Con esto permitimos hacer forward de paquetes en el firewall, o sea
# que otras máquinas puedan salir a traves del firewall.
echo 1 > /proc/sys/net/ipv4/ip_forward

## Permitimos el paso de la DMZ a una BBDD de la LAN:
iptables -A FORWARD -s 192.168.3.2 -d 192.168.10.5 -p tcp --dport 5432 -j ACCEPT

iptables -A FORWARD -s 192.168.10.5 -d 192.168.3.2 -p tcp --sport 5432 -j ACCEPT

## permitimos abrir el Terminal server de la DMZ desde la LAN
iptables -A FORWARD -s 192.168.10.0/24 -d 192.168.3.2 -p tcp --sport 1024:65535 --dport 3389 -j ACCEPT

# … hay que hacerlo en uno y otro sentido …
iptables -A FORWARD -s 192.168.3.2 -d 192.168.10.0/24 -p tcp --sport 3389 --dport 1024:65535 -j ACCEPT

# … por que luego:
# Cerramos el acceso de la DMZ a la LAN
iptables -A FORWARD -s 192.168.3.0/24 -d 192.168.10.0/24 -j DROP

## Cerramos el acceso de la DMZ al propio firewall
iptables -A INPUT -s 192.168.3.0/24 -i eth2 -j DROP

## Y ahora cerramos los accesos indeseados del exterior:
# Nota: 0.0.0.0/0 significa: cualquier red

# Cerramos el rango de puerto bien conocido
iptables -A INPUT -s 0.0.0.0/0 -p tcp -dport 1:1024 -j DROP
iptables -A INPUT -s 0.0.0.0/0 -p udp -dport 1:1024 -j DROP

# Cerramos un puerto de gestión: webmin
iptables -A INPUT -s 0.0.0.0/0 -p tcp -dport 10000 -j DROP

echo " OK . Verifique que lo que se aplica con: iptables -L -n"

# Fin del script
```

Vamos a ver: si las máquinas de la DMZ tienen una ip pública hay que tener muchísimo cuidado de no permitir el FORWARD por defecto. Si en la DMZ hay ip pública NO ES NECESARIO HACER REDIRECCIONES de puerto, sino que basta con rutar los paquetes para llegar hasta la DMZ. Este tipo de necesidades surgen cuando por ejemplo tenemos dos máquinas con servidor web (un apache y un IIS); ¿A cuál de las dos le redirigimos el puerto 80? No hay

manera de saberlo (No, con servidores virtuales tampoco, piénsalo), por eso se deben asignar IPs públicas o en su defecto usar puertos distintos.
Por tanto hay que proteger convenientemente toda la DMZ. Tampoco haría falta enmascarar la salida hacia el exterior de la DMZ, si tiene una ip pública ya tiene una pata puesta en internet; obviamente hay que decirle al router como llegar hasta esa ip pública. Así podría ser esta red:

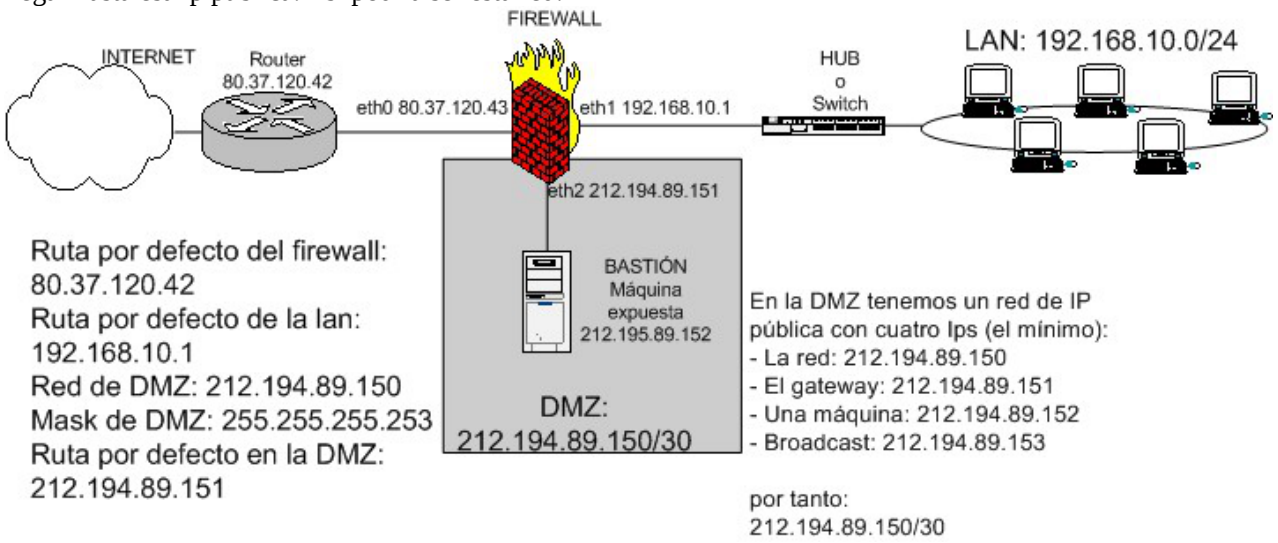


Figura 8: esquema de firewall entre red local e internet con zona DMZ para servidores expuestos usando IPs públicas

Y este podría ser un firewall adecuado:

```
#!/bin/sh
## SCRIPT de IPTABLES - ejemplo del manual de iptables
## Ejemplo de script para firewall entre red-local e internet con DMZ
## pero con IPs públicas.
## Pello Xabier Altadill Izura
## www.pello.info - pello@pello.info

echo -n Aplicando Reglas de Firewall...

## FLUSH de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

## Establecemos politica por defecto
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

## Empezamos a filtrar
## Nota: eth0 es el interfaz conectado al router y eth1 a la LAN

# El localhost se deja (por ejemplo conexiones locales a mysql)
```

```
/sbin/iptables -A INPUT -i lo -j ACCEPT

# Al firewall tenemos acceso desde la red local
iptables -A INPUT -s 192.168.10.0/24 -i eth1 -j ACCEPT

# Ahora hacemos enmascaramiento de la red local y de la DMZ
# para que puedan salir haca fuera
# y activamos el BIT DE FORWARDING (imprescindible!!!!!)
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -o eth0 -j MASQUERADE

# Con esto permitimos hacer forward de paquetes en el firewall, o sea
# que otras máquinas puedan salir a traves del firewall.
echo 1 > /proc/sys/net/ipv4/ip_forward

## Permitimos el acceso desde el exterior a los puertos 80 y 443 de DMZ
iptables -A FORWARD -d 212.194.89.152 -p tcp -dport 80 -j ACCEPT
iptables -A FORWARD -d 212.194.89.152 -p tcp -dport 443 -j ACCEPT
iptables -A FORWARD -d 212.194.89.150/30 -j DROP


## Permitimos el paso de la DMZ a una BBDD de la LAN:
iptables -A FORWARD -s 212.194.89.152 -d 192.168.10.5 -p tcp --dport 5432 -j ACCEPT

# en el otro sentido lo mismo
iptables -A FORWARD -s 192.168.10.5 -d 212.194.89.152 -p tcp --sport 5432 -j ACCEPT

## permitimos abrir el Terminal server de la DMZ desde la LAN
iptables -A FORWARD -s 192.168.10.0/24 -d 212.194.89.152 -p tcp --sport 1024:65535 --dport 3389 -j ACCEPT

# … hay que hacerlo en uno y otro sentido …
iptables -A FORWARD -s 212.194.89.152 -d 192.168.10.0/24 -p tcp --sport 3389 --dport 1024:65535 -j ACCEPT

# … por que luego:
# Cerramos el acceso de la DMZ a la LAN
iptables -A FORWARD -s 212.194.89.152 -d 192.168.10.0/24 -j DROP

## Cerramos el acceso de la DMZ al propio firewall
iptables -A INPUT -s 212.194.89.152 -i eth2 -j DROP

## Y ahora cerramos los accesos indeseados del exterior:
# Nota: 0.0.0.0/0 significa: cualquier red

# Cerramos el rango de puerto bien conocido
iptables -A INPUT -s 0.0.0.0/0 -p tcp -dport 1:1024 -j DROP
iptables -A INPUT -s 0.0.0.0/0 -p udp -dport 1:1024 -j DROP

# Cerramos un puerto de gestión: webmin
iptables -A INPUT -s 0.0.0.0/0 -p tcp -dport 10000 -j DROP

echo " OK . Verifique que lo que se aplica con: iptables -L -n"

# Fin del script
```

ATENCIÓN
Merece la pena pararse a explicar esta parte del firewall:

```
## permitimos abrir el Terminal server de la DMZ desde la LAN
iptables -A FORWARD -s 192.168.10.0/24 -d 212.194.89.152 -p tcp -sport 1024:65535 --dport 3389 -j ACCEPT
```

```
# … hay que hacerlo en uno y otro sentido …
iptables -A FORWARD -s 212.194.89.152 -d 192.168.10.0/24 -p tcp --sport 3389 --dport 1024:65535 -j ACCEPT

# … por que luego:
# Cerramos el acceso de la DMZ a la LAN
iptables -A FORWARD -s 212.194.89.152 -d 192.168.10.0/24 -j DROP
```

Lo que nos lleva a dos cuestiones:
¿Por qué hay que explicitar la abertura en uno y otro sentido? Porque la tercera regla cierra todo lo que va de la DMZ a la red local. Para abrir el puerto 3389 de tcp es imprescindible que un paquete de ida sea capaz de llegar hasta la DMZ y que a su vez pueda volver a la LAN. Esto de tener que especificar la abertura en uno y otro sentido será el pan de cada día en un iptables con política DROP por defecto: mejor protección pero más trabajo.

¿Por qué se explicita el puerto de origen/destino 1024:65535 en la primera y segunda regla? Imaginemos que un hacker logra acceso a la máquina de la DMZ. Si no especificamos el puerto de destino en esas dos reglas, el hacker puede abrir CUALQUIER puerto de la LAN siempre que pueda establecer como puerto origen suyo el tcp/3389, cosa fácil para un hacker que sepa algo de C o que tenga el programa pertinente a mano. De todas formas el hacker tendría que saber que existe ese tipo de reglas, si es listo probara con puertos de gestión o con puertos netbios. El problema es que se deja un vínculo con la LAN bien para administrarlo remotamente o para establecer relaciones de confianza y ahí es donde reside el peligro.

En las conexiones "legales" no se usa como puerto origen nada por debajo del 1024; cuando alguien se conecta a otro puerto en su extremo abre un puerto por encima del 1024. Especificándolo en la regla de firewall protegeremos un poco mejor la LAN, aunque los puertos por encima de 1024 estarán en peligro.

Firewall de una LAN con salida a internet y VPNS

En principio este caso no nos tendría que dar mayor problema, aunque la primera vez que lo montemos, el enmascaramiento nos jugará una mala pasada. Por eso conviene echar un vistazo en este caso.

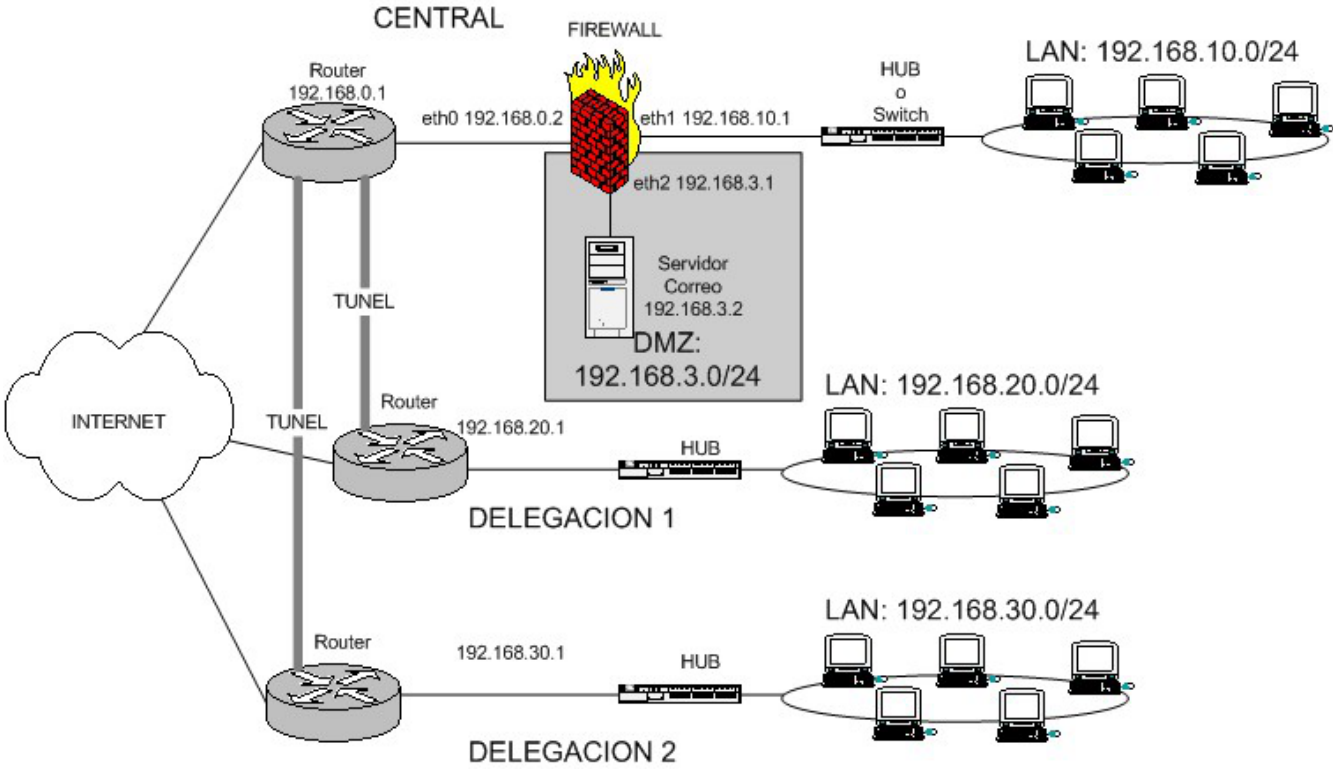


Figura 9: esquema de firewall entre red local e internet con zona DMZ y delegaciones que acceden a DMZ

Supongamos que entre los routers ya se ha establecido un tunel (con Ciscos se haria creando un interfaz Tunnel), y que si el firewall nos deja podríamos llegar de la central a las delegaciones y viceversa usando las IPs privadas. Vaya que se puede hacer un ping desde la central a 192.168.30.x y nos responde. Para ello es imprescindible que el router de la central tenga una ruta metida para llegar a 192.168.10.0/24 y por supuesto cada una ruta para cada delegación. Antes de meterse en el firewall hay que asegurar la visibilidad entre los routers y poder llegar a sus IPs privadas haciendo ping.

Supongamos también que en la central esta el servidor de correo que lógicamente debe tener el puerto 25 accesible desde internet, y debe ser accesible desde las delegaciones para puerto 25, 110 (pop3) o 143(imap). La salida a internet (web, ftp, etc..) cada uno la hace por su lado.

Veamos una posible configuración para este caso.

```
#!/bin/sh
## SCRIPT de IPTABLES - ejemplo del manual de iptables
## Ejemplo de script para firewall entre red-local e internet con DMZ
## y delegaciones. Las delegaciones deben tener acceso al correo de la DMZ
##
## Pello Xabier Altadill Izura
## www.pello.info - pello@pello.info

echo -n Aplicando Reglas de Firewall...

## FLUSH de reglas
```

```
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

## Establecemos politica por defecto
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

## Empezamos a filtrar
## Nota: eth0 es el interfaz conectado al router y eth1 a la LAN
# Todo lo que venga por el exterior y vaya al puerto 25 lo redirigimos
# a la maquina de la DMZ
iptables -t nat -A PREROUTING -i eth0 \
-p tcp --dport 25 -j DNAT --to 192.168.3.2:25

# Todo lo que venga por el interfaz del router(eth0) y vaya al 110
# siempre que sea una delegacion se acepta y redirije
iptables -t nat -A PREROUTING -s 192.168.20.0/24 -i eth0 \
-p tcp --dport 110 -j DNAT --to 192.168.3.2:110

iptables -t nat -A PREROUTING -s 192.168.30.0/24 -i eth0 \
-p tcp --dport 110 -j DNAT --to 192.168.3.2:110

# Todo lo que venga por el interfaz del router(eth0) y vaya al 110
# siempre que sea una delegacion se acepta y redirije
iptables -t nat -A PREROUTING -s 192.168.20.0/24 -i eth0 \
-p tcp --dport 143 -j DNAT --to 192.168.3.2:143

iptables -t nat -A PREROUTING -s 192.168.30.0/24 -i eth0 \
-p tcp --dport 143 -j DNAT --to 192.168.3.2:143


# El localhost se deja (por ejemplo conexiones locales a mysql)
iptables -A INPUT -i lo -j ACCEPT

# Al firewall tenemos acceso desde la red local
iptables -A INPUT -s 192.168.10.0/24 -i eth1 -j ACCEPT

# Ahora hacemos enmascaramiento de la red local y de la DMZ
# para que puedan salir haca fuera
# y activamos el BIT DE FORWARDING (imprescindible!!!!!)
# Cuidado con este enmascaramiento.
iptables -t nat -A POSTROUTING -s 192.168.10.0/24 -o eth0 -j MASQUERADE
iptables -t nat -A POSTROUTING -s 192.168.3.0/24 -o eth0 -j MASQUERADE

# Con esto permitimos hacer forward de paquetes en el firewall, o sea
# que otras máquinas puedan salir a traves del firewall.
echo 1 > /proc/sys/net/ipv4/ip_forward

# Para que desde la red local se salga hacia fuera hay que ENMASCARAR
# pero que pasa con las delegaciones también están fuera Y NO HAY QUE
```

```
# ENMASCARAR, debemos meter una regla FORWARD explicita para que no enmascare
# porque si no una petición de la LAN a otra delegación no se metería
# en el túnel.
iptables -A FORWARD -s 192.168.10.0/24 -d 192.168.20.0/24 -j ACCEPT
iptables -A FORWARD -s 192.168.20.0/24 -d 192.168.10.0/24 -j ACCEPT

iptables -A FORWARD -s 192.168.10.0/24 -d 192.168.30.0/24 -j ACCEPT
iptables -A FORWARD -s 192.168.30.0/24 -d 192.168.10.0/24 -j ACCEPT


# Abrimos el acceso para que se pueda acceder a la DMZ desde la LAN
# a puertos de correo

# En principio lo que va de LAN -> DMZ se acepta
iptables -A FORWARD -s 192.168.10.0/24 -d 192.168.3.0/24 -j ACCEPT

# Luedo desde la DMZ a la LAN solo se acepta 25,110,143
iptables -A FORWARD -s 192.168.3.0/24 -p tcp --sport 25 \
-d 192.168.10.0/24 -j ACCEPT

iptables -A FORWARD -s 192.168.3.0/24 -p tcp --sport 143 \
-d 192.168.10.0/24 -j ACCEPT

iptables -A FORWARD -s 192.168.3.0/24 -p tcp --sport 143 \
-d 192.168.10.0/24 -j ACCEPT

# Cerramos el acceso de la DMZ a la LAN
iptables -A FORWARD -s 192.168.3.0/24 -d 192.168.10.0/24 -j DROP

## Cerramos el acceso de la DMZ al propio firewall
iptables -A INPUT -s 192.168.3.0/24 -i eth2 -j DROP

## Y ahora cerramos los accesos indeseados del exterior:
# Nota: 0.0.0.0/0 significa: cualquier red

# Cerramos el rango de puerto bien conocido
iptables -A INPUT -s 0.0.0.0/0 -p tcp -dport 1:1024 -j DROP
iptables -A INPUT -s 0.0.0.0/0 -p udp -dport 1:1024 -j DROP

# Cerramos un puerto de gestión: webmin
iptables -A INPUT -s 0.0.0.0/0 -p tcp -dport 10000 -j DROP

echo " OK . Verifique que lo que se aplica con: iptables -L -n"

# Fin del script
```

Se han remarcado en negrita las reglas FORWARD entre IPs privadas de delegaciones, ya que sin esas reglas y con el enmascaramiento de por medio no se podría acceder a las delegaciones. Cabe resaltar que entre delegaciones no hay visibilidad total, solamente la central vería a todas las demás, y las delegaciones solamente la central.
La delegaciones accederían al servidor de correo con una redirección, o sea que ellos se configurarían el servidor de correo como 192.168.10.1, mientras que desde la LAN se accedería directamente. Se puede hacer de distintas maneras.

Lo interesante sería poner ese firewall con DROP por defecto, se tratará de mostrar esa configuración al final.

Firewall puro y duro entre redes

En este caso olvidémonos de redes locales y de NAT. Aquí solo tendremos reglas de filtrado INPUT y FORWARD. Pongamos que tenemos el siguiente escenario:

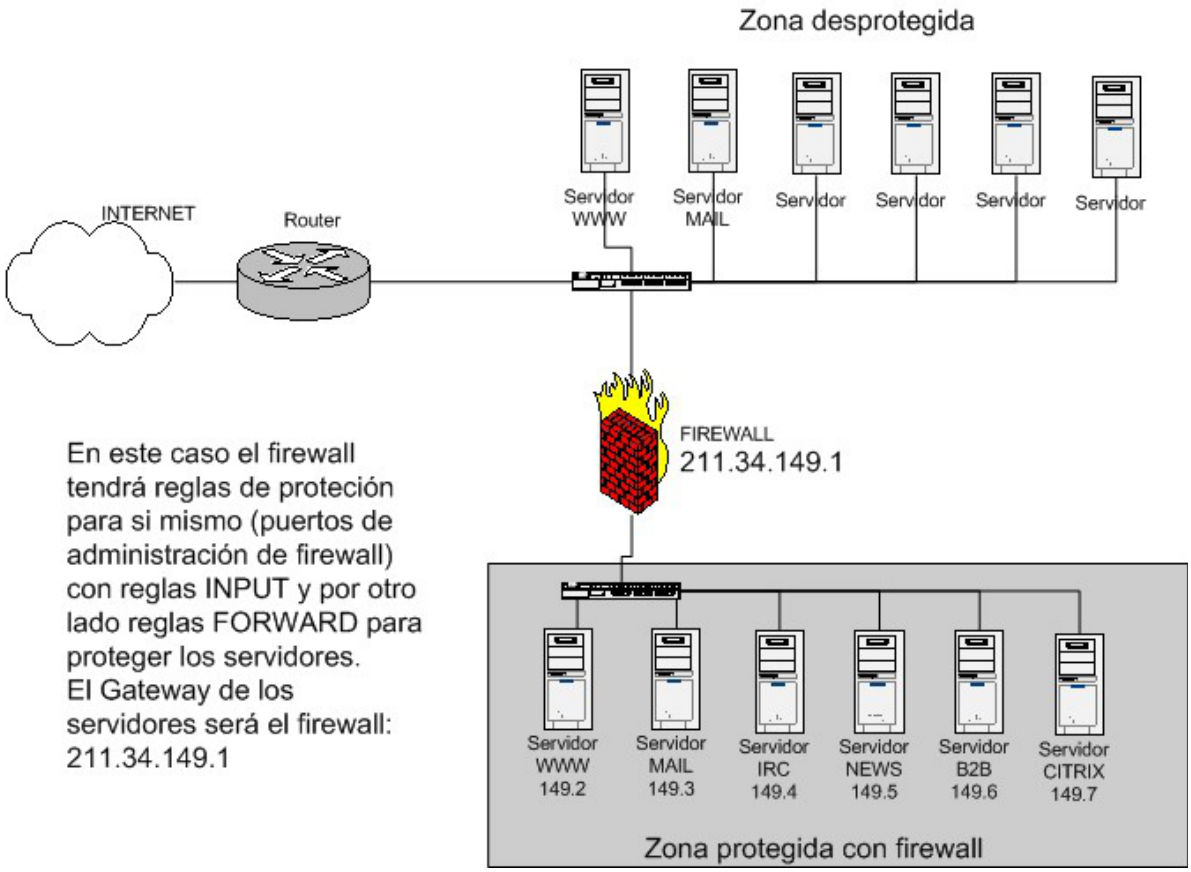


Figura 10: esquema de firewall entre redes, en la que solo se filtra y no se hace NAT

En el firewall debemos indicar una serie de reglas para proteger los equipos que están al otro lado de este dispositivo, todos ellos de la red 211.34.149.0/24
Cada uno de ellos da un servicio determinado, y puede estar gestionado desde distintas IPs, lo que significa que habrá que dar acceso a determinados puertos de gestión (22, 3389, etc..).
Este podría ser el aspecto del script del firewall:

```
#!/bin/sh
## SCRIPT de IPTABLES - ejemplo del manual de iptables
## Ejemplo de script para firewall entre redes.
## Pello Xabier Altadill Izura
## www.pello.info - pello@pello.info

echo -n Aplicando Reglas de Firewall...

## FLUSH de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F
```

```
## Establecemos politica por defecto
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

## Empezamos a filtrar
## Nota: eth0 es el interfaz conectado al router y eth1 a la LAN

# A nuestro firewall tenemos acceso total desde la nuestra IP
iptables -A INPUT -s 210.195.55.15 -j ACCEPT

# Para el resto no hay acceso al firewall
iptables -A INPUT -s 0.0.0.0/0 -j DROP

## Ahora podemos ir metiendo las reglas para cada servidor
## Como serán paquetes con destino a otras máquinas se aplica FORWARD

## Servidor WEB 211.34.149.2
# Acceso a puerto 80
iptables -A FORWARD -d 211.34.149.2 -p tcp --dport 80 -j ACCEPT

# Acceso a nuestra ip para gestionarlo
iptables -A FORWARD -s 210.195.55.15 -d 211.34.149.2 -p tcp --dport 22 -j ACCEPT

# El resto, cerrar
iptables -A FORWARD -d 211.34.149.2 -j DROP


## Servidor MAIL 211.34.149.3
# Acceso a puerto 25, 110 y 143
iptables -A FORWARD -d 211.34.149.3 -p tcp --dport 25 -j ACCEPT
iptables -A FORWARD -d 211.34.149.3 -p tcp --dport 110 -j ACCEPT
iptables -A FORWARD -d 211.34.149.3 -p tcp --dport 143 -j ACCEPT

# Acceso a gestion SNMP
iptables -A FORWARD -s 210.195.55.15 -d 211.34.149.3 -p udp --dport 169 -j ACCEPT

# Acceso a nuestra ip para gestionarlo
iptables -A FORWARD -s 210.195.55.15 -d 211.34.149.3 -p tcp --dport 22 -j ACCEPT

# El resto, cerrar
iptables -A FORWARD -d 211.34.149.3 -j DROP


## Servidor IRC 211.34.149.4
# Acceso a puertos IRC
iptables -A FORWARD -d 211.34.149.4 -p tcp --dport 6666:6668 -j ACCEPT

# Acceso a nuestra ip para gestionarlo
iptables -A FORWARD -s 210.195.55.15 -d 211.34.149.4 -p tcp --dport 22 -j ACCEPT

# El resto, cerrar
iptables -A FORWARD -d 211.34.149.4 -j DROP


## Servidor NEWS 211.34.149.5
# Acceso a puerto news
```

```
iptables -A FORWARD -d 211.34.149.5 -p tcp --dport news -j ACCEPT

# Acceso a nuestra ip para gestionarlo
iptables -A FORWARD -s 213.194.68.115 -d 211.34.149.5 -p tcp --dport 22 -j ACCEPT

# El resto, cerrar
iptables -A FORWARD -d 211.34.149.5 -j DROP


## Servidor B2B 211.34.149.6
# Acceso a puerto 443
iptables -A FORWARD -d 211.34.149.6 -p tcp --dport 443 -j ACCEPT

# Acceso a una ip para gestionarlo
iptables -A FORWARD -s 81.34.129.56 -d 211.34.149.6 -p tcp --dport 3389 -j ACCEPT

# El resto, cerrar
iptables -A FORWARD -d 211.34.149.6 -j DROP


## Servidor CITRIX 211.34.149.7
# Acceso a puerto 1494
iptables -A FORWARD -d 211.34.149.7 -p tcp --dport 1494 -j ACCEPT

# Acceso a una ip para gestionarlo
iptables -A FORWARD -s 195.55.234.2 -d 211.34.149.7 -p tcp --dport 3389 -j ACCEPT

# acceso a otro puerto quiza de BBDD
iptables -A FORWARD -s 195.55.234.2 -d 211.34.149.7 -p tcp --dport 1434 -j ACCEPT

# acceso a otro puerto quiza de BBDD
iptables -A FORWARD -s 195.55.234.2 -d 211.34.149.7 -p udp --dport 1433 -j ACCEPT

# El resto, cerrar
iptables -A FORWARD -d 211.34.149.7 -j DROP

echo " OK . Verifique que lo que se aplica con: iptables -L -n"

# Fin del script
```

Con esta firewall y sobretodo gracias a las reglas de DROP que metemos tras especificar lo que dejamos abiertos, protegeremos de manera eficaz todos lo puertos abiertos de las máquinas.

Firewall con política por defecto DROP

Aquí llega la sección para los auténticos administradores de pelo en pecho.
¿Qué supone el hecho de establecer como política por defecto la denegación?
" Se debe explicitar cada conexión permitida en los dos sentidos.
" Se debe conocer perfectamente qué debe estar abierto y qué no.
" Es muchos más difícil de mantener y si se hace conviene hacerlo desde el principio.
" No todo es más trabajo: también supone un firewall mucho más seguro.

En el ejemplo de la DMZ ya se presentaba esta situación en las reglas forward de una a otra red. Para ilustrar el

DROP por defecto, vamos a mostrar la configuración del ejemplo anterior de firewall entre redes pero con política por defecto DROP.

```
#!/bin/sh
## SCRIPT de IPTABLES - ejemplo del manual de iptables
## Ejemplo de script para firewall entre redes con DROP por defecto
## Pello Xabier Altadill Izura
## www.pello.info - pello@pello.info

echo -n Aplicando Reglas de Firewall...

## FLUSH de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

## Establecemos politica por defecto: DROP!!!
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

## Empezamos a filtrar
## Nota: eth0 es el interfaz conectado al router y eth1 a la LAN

# A nuestro firewall tenemos acceso total desde la nuestra IP
iptables -A INPUT -s 210.195.55.15 -j ACCEPT
iptables -A OUTPUT -d 210.195.55.15 -j ACCEPT

# Para el resto no hay acceso al firewall
# En principio esta de más, pero si rebajamos los permisos temporalmente
# nos cubre las espaldas
iptables -A INPUT -s 0.0.0.0/0 -j DROP

## Ahora podemos ir metiendo las reglas para cada servidor
## Como serán paquetes con destino a otras máquinas se aplica FORWARD

## Servidor WEB 211.34.149.2
# Acceso a puerto 80
iptables -A FORWARD -d 211.34.149.2 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -s 211.34.149.2 -p tcp --sport 80 -j ACCEPT

# Acceso a nuestra ip para gestionarlo
iptables -A FORWARD -s 210.195.55.15 -d 211.34.149.2 -p tcp --dport 22 -j ACCEPT

iptables -A FORWARD -s 211.34.149.2 -d 210.195.55.15 -p tcp --sport 22 -j ACCEPT

## Servidor MAIL 211.34.149.3
# Acceso a puerto 25, 110 y 143
iptables -A FORWARD -d 211.34.149.3 -p tcp --dport 25 -j ACCEPT
iptables -A FORWARD -s 211.34.149.3 -p tcp --sport 25 -j ACCEPT

iptables -A FORWARD -d 211.34.149.3 -p tcp --dport 110 -j ACCEPT
iptables -A FORWARD -s 211.34.149.3 -p tcp --sport 110 -j ACCEPT

iptables -A FORWARD -d 211.34.149.3 -p tcp --dport 143 -j ACCEPT
iptables -A FORWARD -s 211.34.149.3 -p tcp --sport 143 -j ACCEPT
```

```
# Acceso a gestion SNMP
iptables -A FORWARD -s 210.195.55.15 -d 211.34.149.3 -p udp --dport 169 -j ACCEPT

iptables -A FORWARD -s 211.34.149.3 -d 210.195.55.15 -p udp --sport 169 -j ACCEPT

# Acceso a nuestra ip para gestionarlo
iptables -A FORWARD -s 210.195.55.15 -d 211.34.149.3 -p tcp --dport 22 -j ACCEPT

iptables -A FORWARD -s 211.34.149.3 -d 210.195.55.15 -p tcp --sport 22 -j ACCEPT


## Servidor IRC 211.34.149.4
# Acceso a puertos IRC
iptables -A FORWARD -d 211.34.149.4 -p tcp --dport 6666:6668 -j ACCEPT
iptables -A FORWARD -s 211.34.149.4 -p tcp --sport 6666:6668 -j ACCEPT

# Acceso a nuestra ip para gestionarlo
iptables -A FORWARD -s 210.195.55.15 -d 211.34.149.4 -p tcp --dport 22 -j ACCEPT

iptables -A FORWARD -s 211.34.149.4 -d 210.195.55.15 -p tcp --sport 22 -j ACCEPT


## Servidor NEWS 211.34.149.5
# Acceso a puerto news
iptables -A FORWARD -d 211.34.149.5 -p tcp --dport news -j ACCEPT
iptables -A FORWARD -s 211.34.149.5 -p tcp --sport news -j ACCEPT

# Acceso a nuestra ip para gestionarlo
iptables -A FORWARD -s 213.194.68.115 -d 211.34.149.5 -p tcp --dport 22 -j ACCEPT

iptables -A FORWARD -s 211.34.149.5 -d 213.194.68.115 -p tcp --sport 22 -j ACCEPT

# El resto, cerrar
iptables -A FORWARD -d 211.34.149.5 -j DROP


## Servidor B2B 211.34.149.6
# Acceso a puerto 443
iptables -A FORWARD -d 211.34.149.6 -p tcp --dport 443 -j ACCEPT
iptables -A FORWARD -s 211.34.149.6 -p tcp --sport 443 -j ACCEPT

# Acceso a una ip para gestionarlo
iptables -A FORWARD -s 81.34.129.56 -d 211.34.149.6 -p tcp --dport 3389 -j ACCEPT

iptables -A FORWARD -s 211.34.149.6 -d 81.34.129.56 -p tcp --sport 3389 -j ACCEPT


## Servidor CITRIX 211.34.149.7
# Acceso a puerto 1494
iptables -A FORWARD -d 211.34.149.7 -p tcp --dport 1494 -j ACCEPT
iptables -A FORWARD -s 211.34.149.7 -p tcp --sport 1494 -j ACCEPT

# Acceso a una ip para gestionarlo
iptables -A FORWARD -s 195.55.234.2 -d 211.34.149.7 -p tcp --dport 3389 -j ACCEPT

iptables -A FORWARD -s 211.34.149.7 -d 195.55.234.2 -p tcp --sport 3389 -j ACCEPT

# acceso a otro puerto quiza de BBDD
```

```
iptables -A FORWARD -s 195.55.234.2 -d 211.34.149.7 -p tcp --dport 1434 -j ACCEPT

iptables -A FORWARD -s 211.34.149.7 -d 195.55.234.2 -p tcp --sport 1434 -j ACCEPT

# acceso a otro puerto quiza de BBDD
iptables -A FORWARD -s 195.55.234.2 -d 211.34.149.7 -p udp --dport 1433 -j ACCEPT

iptables -A FORWARD -s 211.34.149.7 -d 195.55.234.2 -p udp --sport 1433 -j ACCEPT

echo " OK . Verifique que lo que se aplica con: iptables -L -n"

# Fin del script
```

Ya esta, hemos levantado un verdadero muro entre internet y el conjunto de servidores que esta Tras el firewall. No se puede ni hacer un ping a las máquinas, salvo que se haya dado acceso total a una ip. Si quisieramos dar acceso al ping, pondríamos algo así:

Es más llevadero aplicar el DROP por defecto cuando el firewall es para la propia máquina. El primer escenario de esta manual trataba sobre este caso, ahora lo revisamos con la política por defecto drop.

```
#!/bin/sh
## SCRIPT de IPTABLES - ejemplo del manual de iptables
## Ejemplo de script para proteger la propia máquina
## con política por defecto DROP
## Pello Xabier Altadill Izura
## www.pello.info - pello@pello.info

echo -n Aplicando Reglas de Firewall...

## FLUSH de reglas
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

## Establecemos politica por defecto
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

## Empezamos a filtrar

# El localhost se deja (por ejemplo conexiones locales a mysql)
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# A nuestra IP le dejamos todo
iptables -A INPUT -s 195.65.34.234 -j ACCEPT
iptables -A OUTPUT -d 195.65.34.234 -j ACCEPT

# A un colega le dejamos entrar al mysql para que mantenga la BBDD
iptables -A INPUT -s 231.45.134.23 -p tcp --dport 3306 -j ACCEPT
iptables -A OUTPUT -d 231.45.134.23 -p tcp --sport 3306 -j ACCEPT

# A un diseñador le dejamos usar el FTP
iptables -A INPUT -s 80.37.45.194 -p tcp --dport 20:21 -j ACCEPT
iptables -A OUTPUT -d 80.37.45.194 -p tcp --sport 20:21 -j ACCEPT
```

```
# El puerto 80 de www debe estar abierto, es un servidor web.
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT


# Aquí están las reglas de cerrar. Como hemos comentado en la configuración
# anterior conviene tener esto escrito por si en algún momento se relaja el
# firewall y s cambia a de DROP a ACCEPT por defecto
# Cerramos rango de los puertos privilegiados. Cuidado con este tipo de
# barreras, antes hay que abrir a los que si tienen acceso.
iptables -A INPUT -p tcp --dport 1:1024
iptables -A INPUT -p udp --dport 1:1024

# Cerramos otros puertos que estan abiertos
iptables -A INPUT -p tcp --dport 3306 -j DROP
iptables -A INPUT -p tcp --dport 10000 -j DROP
iptables -A INPUT -p udp --dport 10000 -j DROP

echo " OK . Verifique que lo que se aplica con: iptables -L -n"

# Fin del script
```

Cómo depurar el funcionamiento del firewall

Programas útiles

IPTRAF. Sin duda alguna uno de los programas más prácticos para depurar el firewall es iptables, ya que con el podemos observar si la conexiones se establecen o no; es un programa de consola que es aconsejable controlar ya que muestra en tiempo real el tráfico que atraviesa nuestra máquina con todo lujo de detalles: origen/destino de ips y puertos, tráfico total o tráfico total según el interfaz de red, etc… Si vemos muchas conexiones simultaneas y nos perdemos, existe la posibilidad de aplicar filtros para captar solo aquello que nos interesa.

NMAP. La herramienta para escanear puertos por excelencia, rechace imitaciones. Es una herramienta de consola rápida, efectiva y con multitud de opciones. Podemos usarla desde máquinas ajenas a nuestra red para comprobar si realmente el firewall esta filtrando correctamente y en cierta manera para hacernos una idea de que "visión" pueden tener los hackers de nuestro sistema.

SHELL. En el propio script del firewall podemos añadir algunas opciones para descubrir fallos de sintaxis en las reglas. Claro, imaginemos que tenemos un firewall de 40 lineas y una de ellas falla cuando ejecutamos el script. ¿Cuál es? Es probable que el mensaje de error no aclare lo suficiente, por eso se puede añadir algo así al final de cada regla:

```
...
iptables -A INPUT -s 195.55.234.2 -j ACCEPT && echo " regla-21 ok"
iptables -A INPUT -s 213.62.89.145 -j ACCEPT && echo " regla-22 ok"
...
```

Si la regla se ejecuta bien mostrará el mensajito de ok.
Otra opción algo mas cutre sería ir eliminando o comentando reglas hasta dar con la regla que tiene la sintaxis incorrecta. Cabe reseñar que puede fallar una regla, pero a partir de ella el resto se ejecutan con normalidad.

Y, como manejamos luego todo esto?

Prácticamente todos los mensajes generados por el sistema se guardan por default en, esto es útil y necesario, pero de una manera muy sencilla podemos hacer que los eventos o mensajes generados por se guardan en su propio archivo de bitacora (log).

En este archivo de configuración se indica el modo en que los mensajes del sistema son bitacorizados a través de la utilleria que se instala y configura por default en todos los sistemas GNU/Linux y rara vez el usuario del sistema la modifica, pero en esta ocasión lo abriremos (con tu editor favorito) y haremos lo siguiente:

#> vi /etc/syslog.conf

Y añadimos al final la siguiente línea

kern.warning /var/log/iptables.log

se compone de varias líneas y cada línea es una regla que indica como bitacorizar los mensajes del sistema. Cada regla tiene dos campos, tal como el que añadimos al final. El primer campo (kern.warning) es el "selector" que se compone de dos partes separadas por el punto: facility.priority, 'facility' que en este caso es kern (kernel) representa el subsistema que produce el mensaje y 'priority' define la severidad del mensaje. Estas pueden ser en orden ascendente: debug, info, notice, warning, error, crit, alert, emerg.

Entonces 'warning' es el nivel 4 de prioridad e indica entonces que todos los mensajes provenientes del kernel que tengan un nivel de prioridad 4 o mayor se bitacorizarán en el archivo y se ignoran los de debug, info y notice que son del 3 hacía abajo y que generalmente son irrelevantes.

Es importante aclarar que los mensajes del sistema también seguirán guardándose en y otros que se tengan definidos en ya que lo que hicimos fue 'añadir' una línea más y no modificamos nada de lo que ya estaba ahí.

Reiniciando el servidor

Como cualquier otro programa servidor hay que reiniciarlo para que los cambios en el archivo de configuración surtan efecto.

#> /etc/rc.d/init.d/syslog restart o en varias distros
#> /etc/init.d/syslog restart
si tienes el comando service instalado
#> service syslog restart

Modificando el firewall

Si por ejemplo quieres bitacorizar todo lo que sea rechazado en la cadena INPUT:

iptables -A INPUT -j LOG -log--level 4
iptables -A INPUT DROP

Es decir, generalmente se usará el targer LOG antes de rechazar los paquetes que no queramos, y como notarás en el ejemplo usamos la opción que activará solo cuando el mensaje sea del tipo warning (4) o superior.

Un ejemplo más interesante es el siguiente:

iptables -A INPUT -s 192.168.10.10 -p tcp -m tcp --dport 22 -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 22 -j LOG --log-prefix 'INTENTO DE ACCESO A SSH ' --log-level 4

iptables -A INPUT -p tcp -m tcp --dport 22 -j REJECT
En este caso, la primera regla está aceptando a la ip 192.168.10.10, pero si es cualquier otra ip, entonces se bitacoriza el evento con el mensaje 'INTENTO DE ACCESO A SSH ' en /var/log/iptables.log

Esto es mucho más útil ya que posteriormente con un simple podremos observar solo los mensajes de este tipo:

#> grep 'INTENTO DE ACCESO A SSH' /var/log/iptables.log
Apr 13 10:43:39 equipolinux kernel: INTENTO DE ACCESO A SSH IN=eth1 OUT= MAC=00:50:fc:89:63:39:00:16:e6:82:cd:8a:08:00 SRC=192.168.10.16 DST=192.168.10.100 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=4601 DF PROTO=TCP SPT=1138 DPT=22 WINDOW=65535 RES=0x00 SYN URGP=0
Apr 13 10:44:10 equipolinux kernel: INTENTO DE ACCESO A SSH IN=eth1 OUT= MAC=00:50:fc:89:63:39:00:16:e6:82:cd:8a:08:00 SRC=192.168.10.16 DST=192.168.10.100 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=4602 DF PROTO=TCP SPT=1138 DPT=22 WINDOW=65535 RES=0x00 SYN URGP=0

Se puede apreciar que hay dos intentos de acceder al servidor ssh del equipolinux (192.168.10.100) desde el equipo 192.168.10.16 y con tan solo algunos segundos de diferencia, esto ya es sospechoso y habría que investigar quien está intentando conectarse y porque. (Claro estando en una LAN que controlemos)

Pues ahí lo tienes, puedes agregar a tus reglas de todos los targets LOG que consideres necesarios y revisarlos más útilmente en un archivo por separado. Espero te sea útil.

Ipchains

Configuración

Para ejecutar las reglas del script correctamente se deben tener dos datos: la dirección IP del interfaz por el que nos conectamos a Internet y el nombre de ese interfaz.

En este caso, la conexión es mediante PPP con asignación dinámica de IP (nos conectamos mediante Infovía Plus) por lo que recibimos la IP y el nombre del interfaz del programa PPP que al establecer la conexión nos pasa esa información en dos variables de entorno que son $IFNAME e $IPLOCAL (man pppd).

La ejecución del script es invocada en /etc/ppp/ip-up.local

Nota: Caso de tener conexión directa o IP fija, la llamada a rc.firewall debe realizarse en otro sitio (por ejemplo /etc/rc.d/rc.local) y estos datos, asignarse a mano. También habría que modificar esto en el caso de conexión mediante DHCP (ej. acceso a Internet por cable).

/etc/ppp/ip-up.local

Este fichero se encarga de realizar las acciones que podamos necesitar una vez que se establezca la conexión PPP. Por ejemplo, registrar conexiones, o en nuestro caso, establecer las reglas del cortafuegos. En nuestro caso invocaremos en él la llamada a un script /etc/rc.d/rc.firewall en el que incluiremos las reglas de filtrado de paquetes.

#!/bin/sh/etc/rc.d/rc.firewall

```
#!/bin/sh
#------------------------------------------------------------------
#    CONFIGURACIÓN DEL CORTAFUEGOS
#------------------------------------------------------------------
# Incluir la llamada a este script en /etc/ppp/ip-up.local
```

```
#------------------------------------------------------------------
# Variables de entorno activadas por pppd al establecerá la comunicación
#------------------------------------------------------------------
# IPLOCAL <- dirección IP del interface ppp
# IFNAME <- nombre del interface local


# Asignaciones locales

LOCALNET="192.168.0.0/24"
IPADDR=$IPLOCAL
TODAS="0.0.0.0/0"

# Nombres de Interfaz

PPP=$IFNAME
ETH="eth0"
LO="lo"

# Direcciones

LOOPBACK="127.0.0.1/32"
CLASE_A="10.0.0.0/8"
CLASE_B="172.16.0.0/16"
CLASE_C="192.168.0.0/16"
MULTICAST="240.0.0.0/3"
BROADCAST_0="0.0.0.0"
BROADCAST_1="255.255.255.255"

# Puertos conocidos

ROOT="0:1023" # Puertos reservados a root
NO_ROOT="1024:65535" # puertos no root
NFS="2049" # (TCP/UDP) NFS
OPENWINDOWS="2000" # (TCP) OpenWindows
XWINDOWS="6000:6001" # (TCP) X Window
PORTS="1020:1023" # Rango de puertos de SSH
PORTS="6667" # Puertos del servidor IRC

#--------------------------------
# Limpiamos las reglas anteriores
#--------------------------------
/sbin/ipchains -F

#-------------------------------------------
# Establecer la política por defecto
#      Permitir entrada
#      Permitir salida
#      Denegar IP Forward
#-------------------------------------------
/sbin/ipchains -P input  ACCEPT
/sbin/ipchains -P forward DENY
/sbin/ipchains -P output ACCEPT

#------------------------------------------------------------------------
```

```
#           Spoofing y direcciones ilegales
#--------------------------------------------------------------------------
# Evitar que entren paquetes de fuera indicando como dirección de origen
# nuestra IP

/sbin/ipchains -A input -i $PPP -s $IPADDR -j DENY

# Evitar que lleguen de fuera paquetes con origen o destino 127.0.0.1

/sbin/ipchains -A input -i $PPP -s $LOOPBACK -j DENY
/sbin/ipchains -A input -i $PPP -d $LOOPBACK -j DENY

# Evitar que lleguen de fuera paquetes con origen o destino de direcciones
# reservadas para redes privadas

/sbin/ipchains -A input -i $PPP -s $CLASE_A -j DENY
/sbin/ipchains -A input -i $PPP -d $CLASE_A -j DENY
/sbin/ipchains -A input -i $PPP -s $CLASE_B -j DENY
/sbin/ipchains -A input -i $PPP -d $CLASE_B -j DENY
/sbin/ipchains -A input -i $PPP -s $CLASE_C -j DENY
/sbin/ipchains -A input -i $PPP -d $CLASE_C -j DENY

# Evitar que salgan hacia el exterior paquetes cuyo destino sea nuestra
# propia IP

/sbin/ipchains -A output -i $PPP -d $IPADDR -j REJECT

# Evitar que salgan paquetes con origen o destino 127.0.0.1

/sbin/ipchains -A output -i $PPP -s $LOOPBACK -j REJECT
/sbin/ipchains -A output -i $PPP -d $LOOPBACK -j REJECT

# Evitar que salgan paquetes con origen o destino a direcciones reservadas
# para redes privadas

/sbin/ipchains -A output -i $PPP -s $CLASE_A -j REJECT
/sbin/ipchains -A output -i $PPP -d $CLASE_A -j REJECT
/sbin/ipchains -A output -i $PPP -s $CLASE_B -j REJECT
/sbin/ipchains -A output -i $PPP -d $CLASE_B -j REJECT
/sbin/ipchains -A output -i $PPP -s $CLASE_C -j REJECT
/sbin/ipchains -A output -i $PPP -d $CLASE_C -j REJECT

# Denegar paquetes de broadcast de fuera

/sbin/ipchains -A input -i $PPP -s $BROADCAST_1 -j DENY
/sbin/ipchains -A input -i $PPP -d $BROADCAST_0 -j DENY

#--------------------------------------------------------------------------
# Poner aquí los servicios explícitamente permitidos
# auth (identd) 113, ctcp (irc) (59)
#--------------------------------------------------------------------------
/sbin/ipchains -A input -p tcp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR 59 -j ACCEPT
/sbin/ipchains -A input -p tcp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR 113 -j ACCEPT

#--------------------------------------------------------------------------
# Rechazar acceso del exterior a servicios de nuestra máquina
# Echa un vistazo con "netstat -a" para ver qué puertos tienes abiertos por encima
```

```
# del 1023 y por tanto deberías cerrar
#--------------------------------------------------------------------------
/sbin/ipchains -A input -p tcp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR $ROOT -l -j DENY
/sbin/ipchains -A input -p udp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR $ROOT -l -j DENY

/sbin/ipchains -A input -p tcp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR $NFS -j DENY
/sbin/ipchains -A input -p udp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR $NFS -j DENY

/sbin/ipchains -A input -p tcp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR $OPENWINDOWS -j DENY
/sbin/ipchains -A input -p tcp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR $XWINDOWS -j DENY
/sbin/ipchains -A input -p tcp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR $SSH -j DENY
/sbin/ipchains -A input -p tcp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR $IRCD -j DENY

/sbin/ipchains -A input -p tcp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR 3128 -j DENY
/sbin/ipchains -A input -p tcp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR 3130 -j DENY
/sbin/ipchains -A input -p tcp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR 8080 -j DENY

/sbin/ipchains -A input -p tcp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR 1024 -j DENY
/sbin/ipchains -A input -p udp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR 1024 -j DENY
/sbin/ipchains -A input -p udp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR 1026 -j DENY
/sbin/ipchains -A input -p udp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR 1119 -j DENY
/sbin/ipchains -A input -p udp -i $PPP -s $TODAS $NO_ROOT -d $IPADDR 3401 -j DENY

#-----------------------------------------------
# Forward con enmascaramiento para la red local
#-----------------------------------------------
/sbin/ipchains -A forward -s $LOCALNET -j MASQ
```


Biografía y agradecimientos

http://www.monografias.com/trabajos14/firewalls/firewalls.shtml#ixzz2ZgT0Zmfa
http://www.monografias.com/trabajos3/firewalls/firewalls.shtml#arriba
http://roble.pntic.mec.es/~sgonzale/linux/cortafuegos.html
HERNÁNDEZ, Roberto. Firewalls: Seguridad en las redes e Internet. Boletín de Política Informática N° 2.
TESIS LICENCIATURA EN SISTEMAS AUTOR: A.S.S. BORGHELLO, CRISTIAN FABIAN
http://debiantotal.blogspot.com.es/2007/04/instalar-firewall-arno-iptables.html
http://www.pello.info/filez/firewall/iptables.html#3
http://www.linuxtotal.com.mx/?cont=info__tips_008 autor: sergio.gonzalez.duran@gmail.com
http://www.segu-info.com.ar/firewall/firewall.htm
Wikipedia

Agradecerles a estos autores sus contribuciones, la autoría de sus textos les pertenece completamente y me permito hacerles un guiño a mis amigos, a los viejos que siempre estuvieron ahí y a los nuevos, (ellos saben quienes son (Malka, Kalambre,Red,Uka,Mari,Fanta,Joker,Epsylon,), por leer mis temas, animarme en mis locuras, apoyarme y compartir conmigo conversaciones que no tienen muchos adeptos, pero sobretodo por acompañarme en el solitario camino hacia el conocimiento.

Creada por V. Ana González Hernández