INAP
INSTITUTO NACIONAL DE ADMINISTRACIÓN PÚBLICA

Curso *online*: **Seguridad en Redes WAN e Internet**

Módulo 6 Arquitecturas de seguridad y configuración de cortafuegos

Autores: Daniel Díaz y Andrés Marín

Índice de contenidos

Capítulo 1 INTRODUCCIÓN A LOS CORTAFUEGOS

Los cortafuegos son los elementos principales de muchos diseños de redes seguras. En ellos se confía para poder definir y separar las zonas confiables por las que los datos pueden distribuirse sin miedo de que lleguen a manos indebidas, y en las que los usuarios pueden ir de un sitio a otro sin tener que identificarse a cada momento. Una zona es confiable si podemos fiarnos de quienes están dentro, bien porque está físicamente segregada del resto del mundo, bien porque el nexo de unión con otras redes establece una política fiable de control de acceso.

Normalmente se quiere conectar una red interna (*Intranet)* a la red pública mundial (*Internet*) pero evitando la falta de limitaciones y controles implícitos en Internet. Para lo cual se suele establecer un único nexo de unión entre las dos redes con un estrecho control. Un punto de paso obligado permite evitar que nada se escape de una red a otra red, y que nadie la atraviese sin la debida autorización.

Los cortafuegos (*firewalls)* son dispositivos que controlan el flujo de tráfico entre dos o más redes, permiten o bloquean el tráfico en base a una serie de reglas emanadas de las políticas de seguridad de la organización. Su complejidad reside en las reglas que admiten y en como toman las decisiones en base a dichas reglas.

Un cortafuegos rara vez se basa en una sola técnica: por lo general es una combinación de técnicas desarrolladas para solucionar diferentes problemas.

Los cortafuegos pueden hacer mucho por la seguridad de un sitio, de hecho pueden:

- Ser fuente de decisiones de seguridad, permiten concentrar las medidas de seguridad en el punto de inspección.
- Permitir que sólo pasen los servicios que cumplan con las reglas establecidas para ello.
- Almacenar de manera eficiente la relación con Internet. Un cortafuegos puede registrar lo que ocurre entre la red protegida y la red externa.

- Limitar los riesgos. A veces se utiliza un cortafuegos para mantener separada una sección de una red local de otra, y así evitar que los problemas que impactan en una sección se extiendan a través de toda la red.

Un cortafuegos nos protege contra las amenazas de la red, pero ciertas amenazas están fuera de su control. Por ejemplo, no puede protegernos contra:

- Ataques internos.
- Conexiones que no pasan por él.
- Amenazas desconocidas. Ningún cortafuegos puede defenderse de manera automática contra cada nueva amenaza que surge, no se puede instalar un cortafuegos y esperar que nos proteja para siempre
- Virus. Aunque los cortafuegos revisan todo el tránsito que entra para determinar si puede pasar a la red interna, la revisión puede no ser suficiente para encontrar un virus en los datos. La protección contra virus en un cortafuegos no es muy práctico.

Los cortafuegos son cada vez más necesarios en nuestras redes, pero todos los expertos recomiendan que no se usen en lugar de otras herramientas, sino junto a ellas. Actualmente, los equipos de usuario también deben utilizar cortafuegos personales, que pueden ayudar al usuario a detectar comportamientos anómalos de una aplicación que por primera vez requiere que se le de acceso a través del cortafuegos.

Los cortafuegos generalmente nos ayudan a establecer una seguridad del perímetro de la red, pero no serían necesarios si los usuarios solo tuvieran abiertos los puertos necesarios, tuvieran los servicios actualizados, no utilizaran contraseñas débiles, etc.

Hay distintas tipos de cortafuegos:

1.1. Tipos de cortafuegos

Se pueden clasificar atendiendo a dónde están instalados:

- En las máquinas finales de los usuarios: cortafuegos personales (host firewallls)
- En los routers: lo que entendemos normalmente por cortafuegos que realizan filtrado básico de paquetes.
- En equipos dedicados (dedicated security appliances): ofrecen muchas funcionalidades de seguridad como inspección profunda de paquetes (DPI), detección y prevención de intrusos (IDS/IPS), gestión de VPNs, antiphishing, antimalware, etc. Algunos los denominan sistemas unificados de gestión de amenazas (unified threat management systems).
- En máquinas virtuales o hipervisores: cortafuegos virtuales.

También los podemos clasificar atendiendo al tipo de filtrado que implementan:

- Filtrado estático de paquetes.
- Filtrado con estado (stateful).
- Filtrado de circuitos.
- Filtrado por aplicación.
- Otros filtrados más avanzados (heurísticos, AI, etc.).

Habitualmente un router o un conmutador además de encaminar paquetes, es capaz de filtrarlos por distintos parámetros dependiendo del nivel en el que actúa (puede actuar en todos):

1. Físico: por el interfaz que entra.
2. Enlace: por las direcciones MAC y el tipo de ethernet.
3. Red: versión y direcciones IP, tipo de protocolo transportado.
4. Transporte: puertos TCP/UDP, flags de TCP.
5. Aplicación: HTTP URL, contenidos de mensaje, etc.

Los cortafuegos no están estadarizados, son un *hack*. Cada fabricante diseña sus funcionalidades, que pueden romper aplicaciones, puesto que la regla común es “si no lo entiendo lo tiro”, en ocasiones alteran el contenido y las direcciones de los paquetes, y por supuesto hay muchos más desarrolladores de aplicaciones que de cortafuegos.

Sin embargo se han convertido en un estandar de facto, y por regla general todos los clientes inician las conexiones (para que el cortafuegos deje establecer la conexión) y muchas aplicaciones saben cómo atravesar (*traverse*) cortafuegos y sistemas de reescritura de puertos (PAT/NAT). Por ejemplo el modo pasivo de FTP, STUN/TURN. Hablaremos de esto más adelante.

El filtrado dinámico de paquetes permite, analizando el contenido de algunos paquetes, detectar el establecimiento de una sesión (por ejemplo TLS o FTP) y activar una ventana temporal que autoriza o no el tránsito directo de los paquetes asociados.

Los cortafuegos a nivel de aplicación evalúan los paquetes en la capa de aplicación antes de permitir una conexión manteniendo un riguroso control del estado de todas las conexiones y el número de secuencia de los paquetes. Adicionalmente, este tipo de cortafuegos suele prestar, servicios de autenticación de usuarios.

La operación de los cortafuegos a nivel de red se asemeja a la de un guardia urbano que en un cruce decide qué coches pueden pasar y cuáles no, dependiendo de su origen y su destino. A la vez que cada paquete se encamina hacia su destino, se filtran los paquetes no deseados.

La práctica totalidad de los cortafuegos a nivel de aplicación, suelen prestar servicios de *proxy* (intermediario). Tanto es así que a menudo se identifican biunívocamente unos con otros. Un

proxy es un servicio específico que controla el tráfico de un determinado protocolo (como HTTP, FTP, DNS, LDAP, SMTP, *Telnet*, etc.), proporcionando un control de acceso adicional y un registro detallado de sucesos respecto al mismo.

Con la instalación de intermediarios (*proxies)* también conocidos como pasarelas (*application gateways)*, el cliente, situado en un lado del cortafuegos, habla con un servidor situado en el propio cortafuegos. Este servidor le identifica, registra sus peticiones y, si lo considera oportuno, encamina estas hacia el verdadero servidor situado al otro lado del cortafuegos. La contestación regresa por el mismo camino, quedando igualmente registrada.

1.2. Reglas de filtrado de paquetes

Los cortafuegos permiten definir conjuntos de reglas para especificar que acción deben tomar con cada paquete:

1. El conjunto está ordenado y las reglas se aplican por orden, por tanto la posición de cada regla dentro del conjunto es muy significativa pues si un paquete encaja en una regla el cortafuegos la aplica y pasa al paquete siguiente. Aquí acaba la tarea y no se siguen aplicando reglas.

2. Las reglas conllevan distintos tipos de acciones. Las más comunes son DENY (tirar el paquete) o ALLOW (permitir). Pero también podemos encontrar reglas dependiendo del tipo de cortafuegos (recordemos que nos son estándar):

    - REJECT: enviar mensaje de vuelta TCP RST o ICMP Port Unreachable
    - LOG/ALERT: guardar el paquete para inspección futura o alguna acción inmediata
    - TRACK CONNECTION: permitir y recordar esta conexión (en cortafuegos con estado)
    - PORT FORWARD: redirige el paquete a otro puerto (en proxies transparentes).

Una regla de filtrado de paquetes es un mecanismo que, en base a la información del encabezado del paquete, permite controlar qué información puede fluir desde y hacia una red. Los filtros permiten o bloquean los paquetes, en general mientras se enrutan de una red a otra. Para realizar el filtrado de paquetes se deben establecer un conjunto de reglas que especifiquen qué tipos de paquetes van a permitirse y qué tipos van a bloquearse.

El tipo de router utilizado en un cortafuegos para filtrado de paquetes se conoce como enrutador de protección *(screening router)*. Se trata del tipo más básico de cortafuegos. Analizan el tráfico de red fundamentalmente en la capa 3, teniendo en cuenta, a veces, alguna de las características del tráfico generado en las capas 1, 2 y/o 4. Los elementos con que se cuenta a la hora de decidir si un paquete es válido o no son los siguientes:

- La dirección de origen desde donde, supuestamente, viene el paquete
- La dirección del *host* de destino del paquete

- El tipo de tráfico o protocolo: TCP, UDP o ICMP
- Los puertos de origen y destino
- La interfaz física del cortafuegos a través de la que llega el paquete y por la que habría de darle salida

La información se obtiene en parte de los encabezados de cada paquete y en parte por su ubicación: el enrutador conoce la interfaz por la que llega y debe salir el paquete. Un enrutador de protección además de determinar si puede o no enrutar un paquete a su destino, también determina si debe hacerlo o no. El "debe" o "no debe" lo determina la política de seguridad del sitio, por ello el enrutador de protección se ha de configurar para hacerla cumplir. La política de seguridad se debe plasmar en una serie de reglas. Ejemplos de reglas a implantar en un enrutador podrían ser:

- Bloquear todas las conexiones hacia o desde ciertos sistemas de los que se desconfía.
- Bloquear todas las conexiones que entran de la red externa, excepto las conexiones SMTP (para poder recibir correo electrónico) y HTTP al servidor web.
- Bloquear servicios peligrosos, como TFTP, RPC *rlogin*, *rsh*, *rcp*, etc. permitir en su lugar los servicios "s" (slogin, ssh, scp, etc.)
- No permitir el uso de *Telnet* en los accesos a nuestra red desde el exterior.

El filtrado de paquetes puede permitir o negar un servicio, pero no puede identificar a los usuarios ni a los archivos, ni tampoco puede proteger operaciones individuales dentro de un servicio. En los firewalls de nivel de aplicación esto es teóricamente posible, utilizando procedimientos como los denominados DPI (Deep Packet Inspection), esto es inspección más profunda de los paquetes. Sin embargo el hecho de que la mayoría del tráfico viene cifrado a nivel de aplicación hacen que esto no sea posible.

Para configurar un enrutador con filtrado de paquetes, primero se debe decidir qué servicios se quieren permitir o negar, y después convertir las decisiones en reglas sobre los paquetes.

También hay que tener en cuenta que, por lo general, los protocolos son bidireccionales; casi siempre incluyen un extremo que envía una solicitud o comando, y otro que envía una respuesta. Cuando se diseñen las reglas para el filtrado de paquetes, se debe tener en cuenta que los paquetes viajan en ambas direcciones. Por ejemplo, no sirve de nada permitir paquetes *Telnet* de salida que enviarán comandos desde el teclado a un anfitrión remoto, si no se permite que regresen los paquetes para esa conexión que traen el despliegue en pantalla.

Por el contrario, tampoco sirve de nada bloquear media conexión. Muchos ataques pueden llevarse a cabo si los atacantes pueden introducir paquetes en nuestra red, aunque no puedan obtener respuestas. Esto es posible porque las respuestas pueden ser lo suficientemente predecibles para permitir que los atacantes lleven a cabo su parte de la conversación sin tener

que ver las respuestas. Por ejemplo, aunque el atacante no pueda ver directamente el archivo /etc/passwd, quizá puedan ejecutar un comando para que le envíe una copia.

Las reglas de filtrado, 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.

La forma de procesar un paquete difiere según el modelo, el fabricante o el modo de actuación configurado y define en gran medida la permisividad del cortafuegos:

- Para el filtrado de paquetes, los sistemas más normales y restrictivos, utilizan dos listas de reglas: una de permitidas y otra de denegadas. Estos sistemas exigen que el paquete pase con éxito por ambas listas, es decir, que no sea expresamente denegado en la una y sea expresamente autorizado en la segunda.
- En otros sistemas existe una única lista de reglas y el paquete es procesado según la primera regla de la tabla que defina como tratarlo.
- Por último, también encontramos diferencias en qué hacer cuando no se encuentra ninguna regla válida: algunos productos aceptan el paquete y otros lo rechazan; de cualquier forma, para evitar problemas, lo mejor es insertar siempre una regla por defecto al final de la lista.

En la siguiente tabla tenemos un ejemplo de una de estas últimas listas de reglas en la que el cortafuegos posee la dirección IP 192.168.1.1:

R.	**Direcc. de origen**	**Puerto de origen**	**Direcc.de destino**	**Puerto de destino**	**Acción**	**Descripción**
1	Cualquiera	Cualquiera	192.168.1.0	>1023	Aceptar	Permite que pasen los paquetes de retorno de una conexión originada en la red interna
2	192.168.1.1	Cualquiera	Cualquiera	Cualquiera	Rechazar	Previene conexiones directas del cortafuegos con otro *host*
3	Cualquiera	Cualquiera	192.168.1.1	Cualquiera	Rechazar	Previene de accesos directos desde *hosts* externos al cortafuegos
4	192.168.1.0	Cualquiera	Cualquiera	Cualquiera	Aceptar	Permite acceso al exterior sin restricciones a los usuario internos
5	Cualquiera	Cualquiera	192.168.1.2	SMTP	Aceptar	Permite a los usuarios

						externos enviar e-mail
6	Cualquiera	Cualquiera	192.168.1.3	HTTP	Aceptar	Permite a los usuarios externos acceder al servidor web interno
7	Cualquiera	Cualquiera	Cualquiera	Cualquiera	Rechazar	Cualquier regla que no haya sido explícitamente definida es explícitamente denegada

Ejemplo de Lista de Reglas de un Cortafuegos por Filtrado de Paquetes

Una regla para bloquear los paquetes que entran con direcciones fuente falsificadas (direcciones de nuestra red interna), podría ser:

➢ **Regla**	➢ **Sentido**	**Dirección fuente**	**Dirección destino**	➢ **Acción**
A	Entrada	Interna	Cualquiera	Rechazar

1.2.1. VENTAJAS Y DESVENTAJAS DEL FILTRADO DE PAQUETES

Las principales bondades de este tipo de cortafuegos están en su rapidez, transparencia y flexibilidad. Proporcionan un alto rendimiento y escalabilidad a bajo coste, y son muy útiles para bloquear ataques moderados de Denegación de Servicio, por ello se siguen implantando como servicios integrados en algunos *routers* y dispositivos hardware de balanceo de carga de gama media-alta. Las principales ventajas del uso del filtrado de paquetes son:

- Un enrutador de protección puede ayudar a proteger toda una red

Consideremos como ejemplo un servicio inseguro como *Telnet*: si se deshabilitan todos los servidores de telnet pero queda alguno en alguna máquina (porque se reinstala por error), como *Telnet* no es aceptado por el enrutador, todas las máquinas estarán protegidas.

- El filtrado de paquetes no necesita conocimiento o cooperación del usuario

- Algunas políticas no pueden aplicarse por medio de filtrado de paquetes

No permiten especificar ciertas reglas. Por ejemplo, no se puede aplicar restricciones a usuarios específicos. De modo similar, los paquetes muestran hacia qué puerto van pero no hacia qué aplicación; cuando se aplican restricciones a los protocolos de nivel más alto, se hace por número de puerto con la esperanza de que ningún otro programa esté ejecutándose en el puerto asignado a ese protocolo. Intrusos maliciosos podrían alterar fácilmente esta clase de control.

- No disponen de un sistema de monitorización sofisticado

Muchas veces el administrador no puede determinar si el *router* está siendo atacado o si su seguridad ha sido comprometida

En conclusión, el filtrado de paquetes no es efectivo como única medida de seguridad, pero si es muy práctico como primera barrera, en la que se bloquean ciertos ataques, se filtran protocolos no deseados y se pasan los paquetes restantes a otro cortafuegos que examine los protocolos de las capas más altas.

1.2.2. Cortafuegos con estado

Hoy en día con el precio de la memoria, la inmensa mayoría de los cortafuegos pueden guardar estado. También se les conoce como *Stateful Inspection Firewall o Circuit Level Firewall*, son básicamente cortafuegos de filtrado de paquetes en los que, además, se valida que el paquete corresponde a una petición de nueva conexión o que pertenece a un circuito virtual (o sesión) ya establecido entre un *host* externo y otro interno.

Cuando una aplicación crea una sesión TCP con un *host* remoto, se establece un puerto en el sistema "originario" de la conexión con objeto de recibir allí los datos provenientes del sistema remoto. De acuerdo a las especificaciones de TCP, este puerto del *host* cliente estará comprendido entre 1.023 y 16.384. En el sistema remoto se establecerá, así mismo, un puerto que será siempre menor que 1024.

Los cortafuegos con filtrado de paquetes deben permitir tráfico entrante en todos los puertos superiores (1.023 hasta 16.384) para permitir los datos de retorno de las conexiones salientes. Esto crea un gran riesgo de intrusiones. Los cortafuegos con inspección de estado resuelven eficazmente este problema construyendo una tabla con información correspondiente a todas las sesiones TCP abiertas y los puertos que utilizan para recibir los datos y no permitiendo el tráfico entrante a ningún paquete que no corresponda con ninguna de estas sesiones y puertos.

Los cortafuegos con inspección de estado examinan el establecimiento de cada conexión para asegurarse de que es legítima y está permitida. Los paquetes no son remitidos a su destino hasta que el establecimiento de la conexión ha sido correctamente completado y verificado.

El cortafuegos mantiene una tabla de conexiones válidas (en las que se incluye información del estado de cada sesión) y deja pasar los paquetes que contienen información correspondiente a una entrada válida en dicha tabla de circuitos virtuales. Una vez que la conexión finaliza la entrada en la tabla es eliminada y el circuito virtual entre los dos *hosts* es cerrado.

Las tablas de estado de circuitos virtuales suelen contener, por cada conexión, la siguiente información:

- Un identificador de sesión único asignado a cada conexión establecida
- El estado de la conexión: negociándose (*Handshake*), establecida o cerrándose
- El número de secuencia del último paquete

- La dirección IP origen de los datos
- La dirección IP destino de los datos
- El interfaz físico de la red, si procede, a través de la que los paquetes llegan
- El interfaz físico de red, si procede, a través de la que los paquetes salen

Usando esta información y analizando las cabeceras de los paquetes, el cortafuegos es capaz de determinar cuándo un paquete es válido y cuando no lo es. Los campos de la cabecera de TCP que habitualmente inspeccionan los cortafuegos son Puertos origen y destino, número de secuencia y número de *acknowledgement*.

Sobre esta base se realizan otro tipo de verificaciones para, por ejemplo, asegurarnos que no ha habido suplantación (*spoofing*), que no existen paquetes mal formados etc. También son comunes en ellos la implantación de sistemas de translación de direcciones, NAT, que ocultan eficazmente el interior de nuestra red a intrusos externos.

1.2.3. Consideraciones para configurar un cortafuegos con inspección de estado

Como complemento a lo anterior, a continuación se comentan las características de los protocolos IP, TCP, UDP y ICMP que son relevantes para el filtrado de paquetes.

El campo de banderas TCP contiene el bit ACK (de confirmación de recepción). Al examinar este bit ACK, un enrutador puede determinar si un paquete específico es el que inicia una conexión TCP, si el bit ACK no está encendido, o es un paquete subsiguiente si el bit ACK está encendido. El bit ACK se activa siempre que un extremo de la conexión ha recibido información del otro extremo (confirma la información recibida). Por lo tanto, el bit ACK se activa en todos los paquetes que van en cualquier dirección, excepto en el primer paquete del cliente al servidor.

Reconocer los paquetes TCP de “inicio de conexión” permite aplicar una política para que los clientes internos se conecten a servidores externos pero que evite que clientes externos se conecten a servidores internos. Esto se logrará permitiendo que los paquetes TCP de inicio de conexión (los que no tienen encendido el bit ACK) sólo salgan de los clientes internos a servidores externos, pero no se permitirá su entrada de clientes externos a servidores internos.

El cuerpo de un paquete IP puede contener un paquete UDP en lugar de un paquete TCP. Cada paquete (datagrama) UDP es independiente. Los paquetes UDP no son parte de un “circuito virtual”, como los de TCP.

Algunos productos de filtrado de paquetes, utilizan tablas en memoria para apuntar los paquetes UDP salientes que han visto. Así, pueden permitir que únicamente regresen los paquetes de respuesta correspondientes a través del mecanismo de filtrado. Para que un paquete entrante se tome como una respuesta debe ser del anfitrión y puerto que envió el paquete de salida. Las reglas creadas para permitir las respuestas son de tiempo limitado; vencen después de un lapso determinado de tiempo.

ICMP se usa para mensajes de estado y control de IP. Muchos sistemas para filtrado de paquetes permiten filtrar paquetes ICMP basados en el campo tipo del mensaje.

Si se va a eliminar un paquete, el enrutador puede o no enviar un código de error de ICMP indicando lo que pasó. Enviar como respuesta un código de error de ICMP tiene el objeto de prevenir a la máquina emisora para que no vuelva a enviar el paquete; por lo tanto, ahorra algo de tráfico en la red y algún tiempo al usuario en el extremo remoto.

Desde el punto de vista del filtrado de paquetes, el problema de la fragmentación IP es que sólo el primer fragmento contendrá en el encabezado la información para los protocolos del nivel superior, como TCP, que el sistema para filtrado de paquetes necesita para decidir si permite o no todo el paquete. Normalmente la regla que se aplica es permitir el paso de fragmentos no iniciales, y hacer el filtrado sólo en el primer fragmento de un paquete. Esto es seguro ya que, si el filtrado decide eliminar el paquete original, sin importar cuántos reciba. Si no se puede reconstruir el paquete original, el paquete reunido en forma parcial no será aceptado.

1.3. SISTEMAS *PROXY*

Los servidores *proxy* son programas de aplicación o servidores especializados que se ejecutan en un cortafuegos anfitrión: ya sea un anfitrión con doble acceso con una interface en la red interna y la otra en la red externa, o algún otro anfitrión bastión que tenga acceso a Internet y sea accesible desde las máquinas internas. Estos programas toman las solicitudes de los usuarios para servicios de Internet y, si son acordes con la política de seguridad del sitio, los envían a los servidores reales. Los servidores *proxy* proporcionan conexiones sustitutas y actúan como compuertas a los servicios; por esta razón se conocen a veces como *compuertas a nivel aplicación*, y a las máquinas donde se ejecutan pasarelas de aplicación.

Los sistemas *proxy* evitan la frustración del usuario y las inseguridades de un anfitrión con doble acceso, automatizando la interacción con tal anfitrión. El usuario tiene la ilusión de que trata directamente (o casi en forma directa) con el servidor que está en Internet, con un mínimo de interacción directa con el anfitrión con doble acceso.

Un servicio *proxy* requiere dos componentes: un servidor *proxy* y un cliente *proxy*. En esta situación, el *servidor proxy* se ejecuta en el anfitrión con doble acceso. Un cliente *proxy* es una versión especial de un programa cliente común (por ejemplo, un cliente *Telnet* o FTP) que se comunica con el servidor *proxy* en lugar de hacerlo con el "verdadero" servidor que está en Internet, con frecuencia, los programas cliente comunes pueden usarse como clientes *proxy*.

El servidor *proxy* evalúa solicitudes del cliente *proxy* y decide, en base a un conjunto de reglas, cuáles aprobar y cuáles negar. Si aprueba una solicitud, el servidor *proxy* contacta con el verdadero servidor en nombre del cliente (de ahí viene el término *proxy*, representante), y procede a transmitir las solicitudes del cliente *proxy* al verdadero servidor, y las respuestas del verdadero servidor al cliente *proxy*.

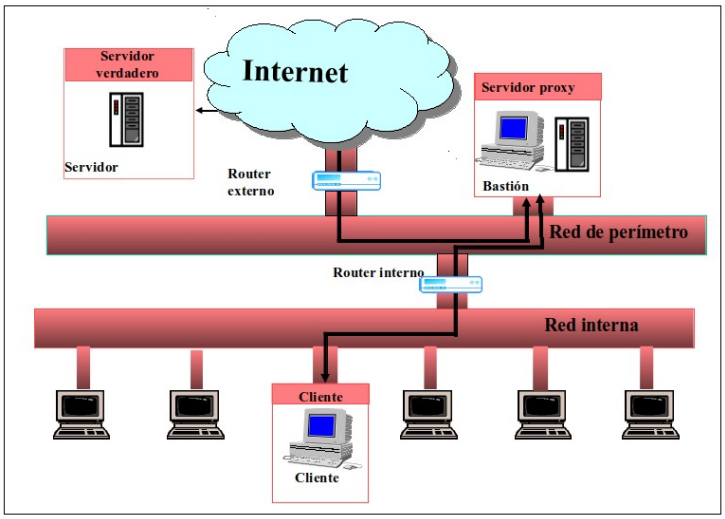
Un servidor *proxy* es una solución software, no una arquitectura de cortafuegos en sí. Se pueden utilizar servidores *proxy* junto con cualquiera de las arquitecturas para cortafuegos descritas más adelante.

Los servidores *proxy* manejan toda la comunicación entre usuarios y servicios de Internet de una forma transparente. Para el usuario, un servidor *proxy* presenta la ilusión de que trata directamente con el verdadero servidor. Al verdadero servidor, el servidor *proxy* presenta la ilusión de que trata directamente con un usuario en el anfitrión *proxy*. La figura VII-3 ilustra la diferencia entre la realidad y la ilusión con los sistemas *proxy*.

Los sistemas *proxy* sólo son efectivos cuando se utilizan junto con algún método para restringir el tráfico a nivel IP entre los clientes y los verdaderos servidores, como un enrutador de protección o un anfitrión con doble acceso. Si hay conexión a nivel IP entre los clientes y los verdaderos servidores, los clientes pueden saltarse el sistema *proxy* (y supuestamente también puede hacerlo alguien del exterior). Un sistema *proxy* proporciona acceso a Internet a un solo anfitrión, o a un número muy pequeño de anfitriones, aunque parece que lo proporciona a todos.

Normalmente un servidor *proxy* se monta en el cortafuegos (bastión) siendo el flujo de paquetes el que se indica en la figura:

Servicio proxy genérico



Los routers pueden obligar a utilizar

1.3.1. VENTAJAS Y DESVENTAJAS DEL USO DE *PROXY*

La principal ventaja de los servicios *proxy* es que permiten monitorizar y contabilizar la actividad del sistema de comunicaciones. 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.

El servidor *proxy* no sólo envía las solicitudes de los usuarios a los verdaderos servicios de Internet. También puede controlar lo que hacen los usuarios, ya que puede tomar decisiones sobre las solicitudes que procesa.

El uso del *proxy* también presenta algunas desventajas:

- Los servicios proxy aparecen con posterioridad a los servicios no proxy

Aunque los programas *proxy* están ampliamente disponibles para los servicios más antiguos y simples, es más difícil encontrar software para servicios nuevos o poco comunes.

- Los servicios *proxy* por lo general requieren de modificaciones a los clientes, a los procedimientos o a ambos

Excepto en contadas ocasiones los servidores *proxy* necesitan modificar los clientes y/o los procedimientos. Cualquier tipo de modificación acarrea problemas, las personas no pueden utilizar las herramientas con sus instrucciones normales. Las aplicaciones proxy funcionan peor que las aplicaciones no proxy.

- Los servidores *proxy* no funcionan para algunos servicios
- Los servicios *proxy* no nos protegen de todas las debilidades de los protocolos

El uso de *proxy* requiere determinar que operaciones son seguras en un protocolo. No todos los protocolos proporcionan formas fáciles de hacer esto.

1.3.2. TIPOS DE SERVIDORES *PROXY*

Hay dos tipos de *Proxy* : a nivel aplicación y a nivel circuito

- Un *proxy* a nivel aplicación es el que sabe sobre la aplicación específica para la cual está proporcionando servicios; comprende e interpreta los comandos en el protocolo de la aplicación. Un ejemplo de este tipo es *Sendmail o Apache o NGINX* que implementan un protocolo de recibir y enviar.
- Un *proxy* a nivel circuito es el que crea un circuito entre el cliente y el servidor sin interpretar el protocolo de la aplicación. Un ejemplo son las modernas compuertas *proxy* híbridas que parecen como *proxy* para el exterior pero como enrutador con filtrado para el interior. Un simple *netcat* puede ser una forma sencilla de proxy a nivel de circuito.

Para poder establecer una conexión *proxy*, se debe saber a dónde dirigirla, un anfitrión *proxy* sólo puede recibir conexiones que van hacia él y también hay que indicarle donde hacer la conexión hacia fuera. Un *proxy* a nivel de aplicación puede obtener esa información del protocolo de aplicación (ya sea utilizando las características de diseño, o interpretando datos proporcionados por el usuario). Un *proxy* a nivel de circuito no puede interpretar el protocolo

de aplicación y necesita que le proporcionen la información a través de otros medios (por ejemplo mediante un cliente modificado que le dé al servidor la dirección de destino).

Aunque “a nivel aplicación” y “a nivel circuito”, son términos utilizados, a menudo se suelen utilizar los términos “dedicados” y “genéricos”. Un servidor *proxy* dedicado es el que sirve a un solo protocolo; un servidor *proxy* genérico es el que sirve a varios protocolos. En la práctica, los servidores *proxy* dedicados son a nivel aplicación; los servidores *proxy* genéricos son a nivel circuito.

Hay otro tipo de *proxy*, denominados inteligentes, un servidor *proxy* inteligente hace más cosas que simplemente transmitir peticiones. Por ejemplo, la mayoría de proxies HTTP (apache, squid, varnish polipo, etc.) ofrecen caché de datos; para que las distintas peticiones para los mismos datos no salgan a través de Internet. Es más fácil que sea inteligente un servidor *proxy* a nivel aplicación que uno a nivel circuito que tiene habilidades limitadas.

1.4. SERVICIOS ADICIONALES PROPORCIONADOS POR LOS CORTAFUEGOS

Un valor añadido sobre los cortafuegos actuales son los servicios adicionales que disponen y que facilitan las labores de protección y administración de la red. Se trata de servicios en algunos casos hechos a medida, y en otros habituales de otros dispositivos pero que, en cualquier caso, representan un punto importante a la hora de decantarnos por una u otra implementación.

1.4.1. Translación de Direcciones de Red (NAT)

Los servicios de NAT (*Network Address Translation)* resuelven dos de los principales problemas de seguridad e infraestructura de las redes actuales. En primer lugar, constituyen una herramienta muy efectiva para esconder las direcciones de red reales de nuestra red interna. En segundo lugar, y debido a la reducción del espacio de direcciones IP disponibles, muchas organizaciones usan NAT para permitir la salida a Internet de sus equipos de red interna con un mínimo de direcciones legalmente válidas. Existen tres estrategias diferentes a la hora de implementar NAT:

- Translación de Direcciones de Red Estática

En este esquema de NAT cada sistema interno de la red privada tiene su propia dirección IP exterior. Con este sistema se logra esconder el esquema interno de nuestra red, pero no la reducción de direcciones IP válidas de acceso al exterior. Los cortafuegos que incluyen esta característica usan para ello una simple tabla de correspondencia entre unas direcciones y otras.

- Translación de Direcciones de Red Oculta

Con este esquema todos los sistemas de la red interna comparten la misma dirección IP externa. Reviste dos importantes inconvenientes: es imposible poner a disposición de los usuarios externos ningún recurso de la red interna, y obliga al cortafuegos a usar su propia dirección externa como sustituta de la dirección de todos los equipos que protege, con lo cual

implícitamente estamos revelando la dirección del mismo y lo hacemos susceptible de ser atacado directamente, además de restarle flexibilidad al sistema.

- Translación de Puertos

El sistema de translación de Puertos (PAT) resuelve los dos problemas vistos en el esquema anterior convirtiéndolo en la mejor forma de implementar NAT. En primer lugar no es necesario usar la dirección externa del cortafuegos, sino que podemos crear otra dirección virtual para este propósito. En segundo lugar, es posible hacer accesibles recursos internos a los usuarios del exterior.

El cortafuegos usa el puerto del cliente para identificar cada conexión entrante y construye a tal efecto una tabla de translaciones como la siguiente:

Dirección IP interna	**Puerto cliente interno**	**PAT**
192.168.1.108	1028	3313
192.168.1.112	1039	3314
192.168.1.102	1400	3315
192.168.1.101	1515	3316
192.168.1.115	1027	3317
192.168.1.120	1026	3318

La translación de puertos se realiza de forma secuencial en algunos sistemas (como el ejemplo anterior) y aleatoria, dentro de un rango de puertos válidos, en otros.

1.4.2. Protocolo de configuración dinámica de Hosts (DHCP)

DHCP, (*Dynamic Host Configuration Protocol)* es un servicio de asignación automática de direcciones IP con importantes y evidentes ventajas administrativas a la hora de mantener redes de tamaño medio/amplio que muchos cortafuegos (sobre todo los que trabajan en las capas 2,3 y/o 4) incluyen como valor añadido. Además de configurar la dirección de red, permite configurar el encaminamiento, servidor DNS, servidores de impresoras, etc.

1.4.3. Redes Privadas virtuales (VPN)

Uno de los servicios adicionales más valorados de los cortafuegos actúales es la posibilidad de construcción de Redes Privadas Virtuales VPN que permiten extender a las comunicaciones externas la seguridad del interior de nuestra red.

Una VPN se construye en la cúspide de la pila de protocolos ya existentes en la red usando protocolos adicionales, cifrados fuertes y mecanismos de control de integridad, sustitución o repetición de la información transmitida.

En el módulo anterior mencionamos diferentes formas de construir una VPN: IPSec o L2TP del IETF o el PPTP de Microsoft.

El motivo por el que se coloca el servidor de VPN en el cortafuegos es evidente: colocarlo detrás de él haría que el tráfico cifrado entrante y saliente generado por el servidor VPN no pudiese ser inspeccionado totalmente y hubiera que obviar funciones como las de autenticación, *logging*, escaneo de virus, etc sobre todo este tráfico. Colocando el servidor VPN detrás del cortafuegos lo hacemos vulnerable a ataques directos.

El principal problema de las VPN es elevado coste de recursos que supone el cifrado completo de las comunicaciones, lo cual reduce considerablemente el ancho de banda efectivo. Una solución para mitigar este problema es usar una tarjeta cifradora por hardware, que suelen reducir aproximadamente un 50% el tiempo necesario en realizar el cifrado.

1.4.4. Modeladores del ancho de banda o reguladores

Estos dispositivos denominados *Bandwidth Shapers o Throttlers*, están adquiriendo un auge asombros en los últimos tiempos y son muchas las formas en las que nos los encontramos: Programas o elementos específicos e servicios de valor añadido en *routers* o cortafuegos.

Un modelador del ancho de banda se emplaza entre la red interna y la salida a Internet (el mismo lugar del cortafuegos, de ahí su inclusión en los mismos) y puede ser comparado con un "guardia de tráfico". Mediante reglas se definen distintas colas, cada una de las cuales alberga un tipo distinto de tráfico: *e-mails*, transferencias de ficheros, tráfico http, archivos musicales o de vídeo, etc. Cada una de las colas de tráfico posee una prioridad distinta, de forma que podemos poner en primer lugar aquellas que correspondan al tráfico más crítico para nuestra organización.

Aunque pueda parecer que estos métodos introducen más retardo que desahogo en el tráfico de la red no es así en absoluto: los *Shapers* analizan, al igual que los cortafuegos con inspección de estado sólo los primeros paquetes de una conexión y una vez que esta es identificada la asignan a una cola de tráfico en particular hasta que esta finaliza.

1.4.5. Inspección de Contenidos

Uno de los servicios más interesantes que, a nivel de aplicación, ofrecen los cortafuegos, es realizar una inspección de contenidos en el tráfico HTTP y SMTP que permita incluir diferentes elementos como videos HTML o Flash, códigos, inspección de virus (binarios o de macro), inspección del contenido de ciertos formatos ampliamente introducidos (zip, docx, xlsx, pptx, odt, etc.), bloqueo de contenidos en base a URL's, direcciones IP y/o palabras clave, o bloqueo de comandos específicos de determinadas aplicaciones.

1.4.6. Autenticación de usuarios

Otro servicio básico en los cortafuegos a nivel de aplicación es la autenticación de usuarios que en los dispositivos a nivel de red debe limitarse a la dirección IP de procedencia de la petición,

con el consiguiente riesgo de suplantación, mientras que en estos pueden habilitarse servicios clásicos de combinación *login/password*.

1.4.7. Alta Disponibilidad y Balanceo de Carga

Uno de los principales inconvenientes de los cortafuegos es la disminución del rendimiento que provocan. Los cortafuegos empresariales de gama alta suelen ofrecer una solución para paliar este problema al mismo tiempo que ofrecen redundancia mediante el balanceo de carga entre dos o más dispositivos cortafuegos. Logramos, de esta forma, mejorar el problema del rendimiento y ofrecer alta disponibilidad y tolerancia a fallos en nuestra política de seguridad.

1.4.8. Integración con Sistemas de Detección de Intrusos (IDS's)

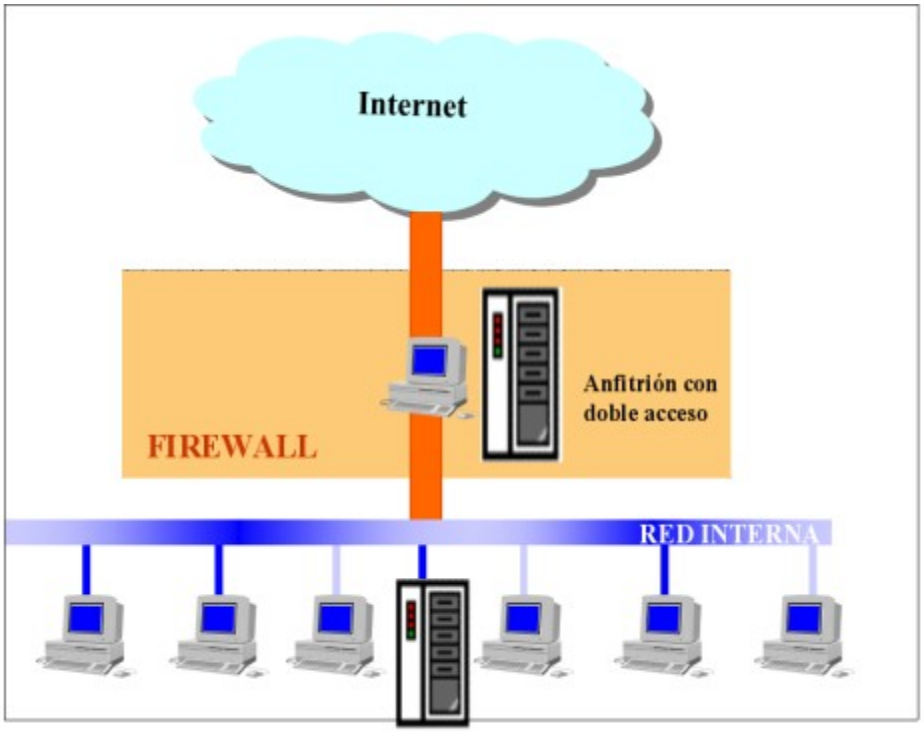
Los sistemas de detección de intrusos son herramientas o dispositivos que nos permiten inspeccionar nuestro sistema, generar alertas y conocer cuando alguien ha tratado de penetrar en nuestro sistema o lo ha conseguido. Existen dos tipos de sistemas IDS los de *hosts* y los de redes. Los IDS de *hosts* se basan en el análisis de las estadísticas de uso o el uso indebido de ciertos recursos (comandos, archivos, etc.) del sistema. Los IDS de red buscan patrones sospechosos en los paquetes TCP, malformaciones en la estructura de los mismos, etc.

Capítulo 2 ARQUITECTURAS DE LOS CORTAFUEGOS

2.1. ARQUITECTURAS DE ANFITRIÓN CON DOBLE ACCESO

Una arquitectura de anfitrión con doble acceso se construye alrededor de un ordenador anfitrión que tiene por lo menos dos interfaces de red. Tal anfitrión podría actuar como enrutador; enrutando paquetes IP de una red a otra. Sin embargo, al implementar esta arquitectura se desactiva la función de enrutamiento. Tanto los sistemas de dentro del cortafuegos como los de fuera pueden comunicarse con el anfitrión con doble acceso, pero estos sistemas no pueden comunicarse entre sí de manera directa. El tráfico IP entre ellos está bloqueado en su totalidad.

Los anfitriones con doble acceso pueden proporcionar un alto nivel de control. Sin embargo, es necesario mucho trabajo para aprovechar de manera consistente las ventajas potenciales de anfitrión con doble acceso.



Arquitectura de anfitrión con doble acceso

Un anfitrión con doble acceso sólo puede proporcionar servicios tipo *proxy*, o permitir que los usuarios inicien una sesión directa con él. Pero esto último puede activar servicios considerados inseguros, por ello, muchos usuarios no consideran conveniente emplear un anfitrión con doble acceso iniciando una sesión con él antes de salir a Internet.

La arquitectura de red para un cortafuegos de anfitrión con doble acceso es bastante sencilla: el anfitrión se coloca entre, y se conecta a, Internet y la red interna. La figura VII-5 muestra esta arquitectura.

2.2. ARQUITECTURA DE ANFITRIÓN DE PROTECCIÓN

Mientras una arquitectura de anfitrión con doble acceso proporciona servicios desde un anfitrión conectado a varias redes (pero con el enrutamiento desactivado), una arquitectura de anfitrión de protección proporciona servicios en un anfitrión conectado sólo a la red interna, utilizando un enrutador independiente para el filtrado de paquetes. La figura VII-6 muestra una versión sencilla de una arquitectura de anfitrión de protección.

El anfitrión bastión está colocado en la red interna. El filtrado de paquetes en el enrutador de protección está configurado de tal manera que el anfitrión bastión es el único sistema en la red interna con el que los anfitriones en Internet pueden abrir conexiones (por ejemplo, para entregar correo electrónico). Aun así, sólo están permitidas ciertos tipos de conexiones. Cualquier sistema externo que intente tener acceso a los sistemas o servicios internos tendrá que conectarse con este anfitrión. Por lo tanto, el anfitrión bastión debe mantener un alto nivel de seguridad.

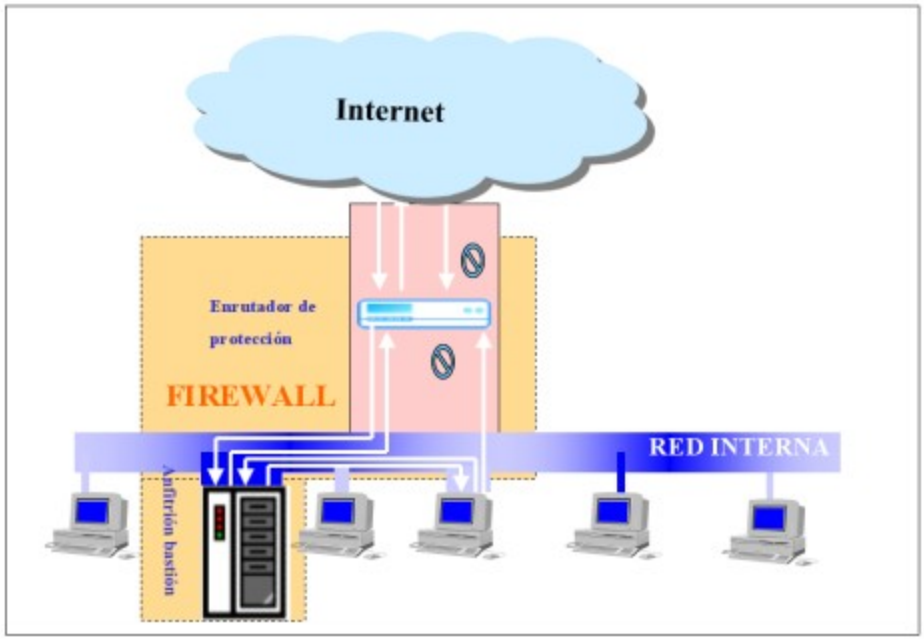
El filtrado de paquetes también permite que el anfitrión bastión abra conexiones permitidas al mundo exterior (lo “permitido” lo determina la política de seguridad específica del sitio).

En el enrutador de protección el filtrado de paquetes se puede configurar para:

- Permitir que otros anfitriones internos abran conexiones con anfitriones en Internet para ciertos servicios (permitiendo esos servicios a través del filtrado de paquetes).
- No permitir las conexiones de anfitriones internos (obligándolos a utilizar servicios *proxy* a través del anfitrión bastión).

Se pueden mezclar estos enfoques para diferentes servicios; algunos pueden permitirse directamente por medio del filtrado de paquetes; otros pueden permitirse sólo de manera indirecta por medio de *proxies*. Todo depende de la política específica que el sitio intente cumplir.

Debido a que esta arquitectura permite que los paquetes se muevan de Internet a las redes internas, puede parecer más arriesgada que la arquitectura de anfitrión con doble acceso, diseñada para que ningún paquete externo alcance la red interna. Sin embargo, en la práctica, la arquitectura de anfitrión con doble acceso también es propensa a fallar permitiendo que, en realidad, pasen los paquetes de la red externa a la interna. Además, es más fácil defender un enrutador, que proporciona un conjunto muy limitado de servicios, que defender un anfitrión. Para casi todos los propósitos, la arquitectura de anfitrión de protección proporciona más seguridad y facilidad de uso que la arquitectura de anfitrión con doble acceso.



Arquitectura de anfitrión de protección

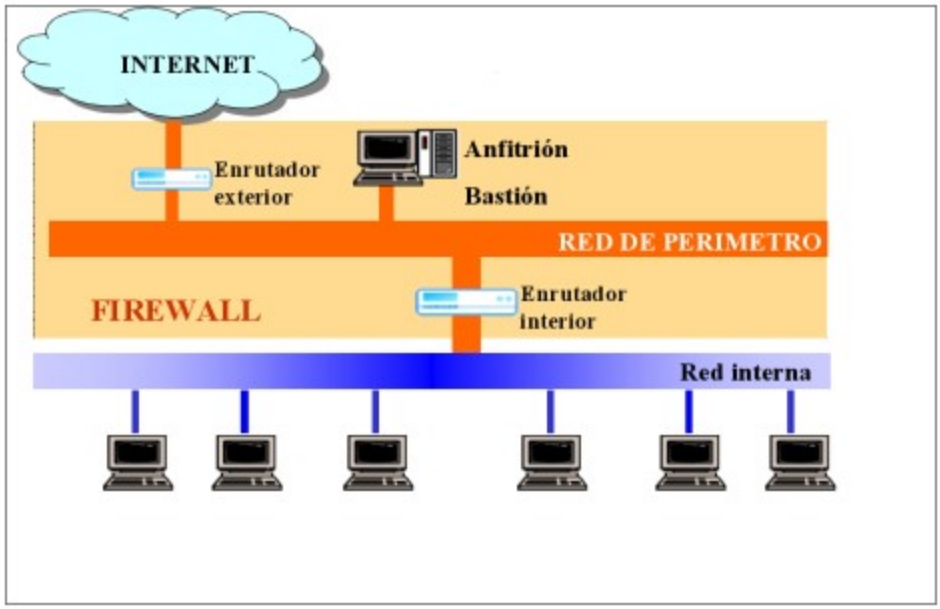
La principal desventaja de esta arquitectura es que si un atacante logra penetrar el anfitrión bastión, ya no hay ningún control de seguridad de red entre ese anfitrión y el resto de los anfitriones internos. El enrutador también presenta un solo punto de falla; si éste es vulnerado, toda la red estará a merced de un atacante.

2.3. ARQUITECTURA DE SUBRED DE PROTECCIÓN

La arquitectura de subred de protección, también conocida como red de perímetro DMZ (*De-Militarized Zone)*, agrega una capa adicional de seguridad a la arquitectura de anfitrión de protección al añadir una red de perímetro que aísla aún más la red interna de Internet. Esta arquitectura, es con diferencia la más utilizada e implantada hoy en día.

En una arquitectura de anfitrión de protección, la red interna está totalmente abierta para ser atacada desde el anfitrión bastión. Al aislar al anfitrión bastión en una red de perímetro, se puede reducir el impacto de una entrada forzada.

En la forma más sencilla de arquitectura de subred de protección, hay dos enrutadores de protección, cada uno conectado a la red de perímetro. Uno colocado entre la red de perímetro y la red interna, y el otro entre la red de perímetro y la red externa (por lo general, Internet). En la red de perímetro, que constituye el sistema cortafuegos, se incluye el anfitrión bastión y también se suelen incluir sistemas que requieran un acceso controlado, como baterías de *modems* o el servidor de correo, que son los únicos elementos visibles desde fuera de nuestra red. Para entrar a la red interna con este tipo de arquitectura, un atacante tendría que penetrar ambos enrutadores. Aunque el atacante lograra de alguna forma penetrar al anfitrión bastión, aún tendría que pasar al enrutador interior.



Algunos sitios van tan lejos como para crear una serie de capas de redes perímetro entre el mundo exterior y su red interna. Los servicios menos confiables y más vulnerables se colocan en las redes de perímetro exteriores, más lejos de la red interior. La idea es que al atacante que penetre a una máquina en la red de perímetro exterior, le costará más trabajo atacar con éxito las máquinas internas debido a las capas adicionales de seguridad entre el perímetro exterior y la red interna. Sin embargo, esto sólo es cierto si las distintas capas tienen sentido; si los sistemas de filtrado entre cada capa permiten lo mismo entre todas las capas, las capas adicionales no proporcionan mayor seguridad.

La figura muestra una posible configuración de cortafuegos con una arquitectura de subred de protección.

Arquitectura de subred de protección (utilizando dos enrutadores)

2.3.1. Red de perímetro

La red de perímetro es una capa adicional de seguridad, ubicada entre la red externa y la red interna protegida. Si un atacante entra con éxito a los límites externos del cortafuegos, la red de perímetro ofrece una capa adicional de protección entre el atacante y los sistemas internos.

Con una red de perímetro, si alguien entra a un anfitrión bastión, podrá curiosear sólo su tráfico. Todo el tráfico en la red de perímetro debe ser de o para el anfitrión bastión, o hacia o desde Internet.

Debido a que el tráfico estrictamente interno (tráfico entre dos anfitriones internos que se supone es importante) no pasa por la red de perímetro, aunque el anfitrión bastión esté en

peligro los atacantes no podrán verlo. Parte del trabajo de diseñar un cortafuegos consiste en asegurar que este tráfico no sea, en sí, lo suficientemente confidencial como para que leerlo comprometa su sitio.

2.3.2. Anfitrión bastión

Con la arquitectura de subred de protección, se enlaza un anfitrión bastión (o anfitriones) a la red de perímetro; este anfitrión es el principal punto de contacto para las conexiones que entran desde el mundo exterior; por ejemplo:

- Sesiones de correo electrónico (SMTP) para entregar mensajes al sitio
- Conexiones FTP que entran al servidor FTP anónimo del sitio
- Consultas que entran al servicio de nombres de dominio (DNS) del sitio, etcétera

Los servicios que salen (de clientes internos a servidores en Internet) se manejan de alguna de las formas siguientes:

- Configurando el filtrado de paquetes en los enrutadores exteriores e interiores para que los clientes internos tengan acceso a los servidores externos de forma directa
- Instalando los servidores *proxy* para que se ejecuten en el anfitrión bastión para permitir a los clientes internos acceder a los servidores externos de manera directa.
- Configurando el filtrado de paquetes a fin de que los clientes internos se comuniquen con los servidores *proxy* en el anfitrión bastión y viceversa, para prohibir las comunicaciones directas entre clientes internos y el mundo exterior.

La mayoría de las tareas del anfitrión bastión son para actuar como servidor *proxy* para varios servicios, ya sea ejecutando software servidor *proxy* especializado para protocolos específicos (como HTTP o FTP), o ejecutando servidores estándar para protocolos basados en *proxy* (como SMTP).

2.3.3. Enrutador interior

El enrutador interior (a veces llamado enrutador de choque) protege la red interna tanto de Internet como de la red de perímetro.

El enrutador interior realiza la mayor parte del filtrado de paquetes para el cortafuegos. Permite que servicios seleccionados salgan de la red interna hacia Internet. Estos servicios son los que un sitio puede soportar y proporcionar con seguridad utilizando el filtrado de paquetes en lugar de servicios *proxy*. Cada sitio debe establecer su propia definición de qué significa “seguro”, tendrá que considerar sus propias necesidades, capacidades y limitaciones; no hay una respuesta única sobre los servicios a permitir para todos los sitios.

El objetivo del enrutador interior es limitar los servicios entre el anfitrión bastión y la red interna para reducir el número de máquinas (y de servicios en esas máquinas) que pueden ser atacadas desde el anfitrión bastión, en caso de verse comprometido. Para alcanzar este objetivo se debe:

- Limitar los servicios permitidos entre el anfitrión bastión y la red interna sólo a los que en realidad se necesiten, como HTTP, SMTP o DNS
- Limitar los servicios, hasta donde sea posible, permitiéndoles ir o venir de anfitriones internos determinados; por ejemplo, SMTP podría estar limitado sólo a conexiones entre el anfitrión bastión y el servidor o servidores de correo interno.
- Poner mucha atención en la seguridad de estos anfitriones internos y de los servicios restantes que pueden ser contactados por el anfitrión bastión, pues serán los que perseguirá el atacante si logra entrar en el anfitrión bastión.

2.3.4. Enrutador exterior

En teoría, el enrutador exterior (a veces llamado enrutador de acceso) protege tanto la red de perímetro como la red interna de Internet. En la práctica los enrutadores exteriores tienden a permitir casi cualquier cosa que salga de la red de perímetro y, en general, realizan muy poco filtrado de paquetes. Las reglas de filtrado para proteger las máquinas internas tienen que ser, en esencia, las mismas tanto para el enrutador interior como para el exterior; si hay un error en las reglas que permita el acceso a un atacante, es probable que el error esté presente en ambos enrutadores.

Las únicas reglas para el filtrado de paquetes realmente especiales en el enrutador exterior son las que protegen las máquinas de la red de perímetro (esto es, los anfitriones bastión y el enrutador interno).

El resto de reglas que se podrían establecer en el enrutador exterior son duplicados de las del enrutador interior. Son reglas que evitan que el tráfico inseguro pase entre los anfitriones internos e Internet. Para soportar los servicios *proxy*, el enrutador interior debe permitir que los anfitriones internos envíen paquetes de algunos protocolos hacia el anfitrión bastión, el enrutador exterior debería dejar pasar esos protocolos siempre y cuando vengan del anfitrión bastión. Estas reglas son deseables para dar un nivel adicional de seguridad, pero en teoría sólo bloquean paquetes que no pueden existir porque ya han sido bloqueados por el enrutador interior. Si en realidad existen, es porque el enrutador interior falló o alguien ha conectado un anfitrión inesperado a la red de perímetro.

Una de las tareas que el enrutador exterior puede realizar con éxito, y que no se puede hacer fácilmente en otro lugar, es el bloqueo de cualquier paquete que entra de Internet que tiene direcciones fuentes falsificadas. Tales paquetes dicen venir desde la red interna, pero en realidad entran de Internet.

El enrutador interior puede hacer esto, pero no determinar si los paquetes que deben provenir de la red perímetro son falsos. Aunque en la red perímetro no debe haber nada totalmente confiable, esta zona debe ser más confiable que el universo exterior. Poder falsificar paquetes desde ella, da al atacante casi todas las ventajas para comprometer al anfitrión bastión. El enrutador externo se encuentra en un límite más claro. El enrutador interior tampoco puede proteger los sistemas en la red de perímetro contra paquetes falsificados.

2.4. Variaciones en las arquitecturas de cortafuegoss

Además de las arquitecturas de cortafuegos estándar ya comentadas, para adaptarse mejor a la política de seguridad, al hardware y al presupuesto de cada sitio hay una gran flexibilidad en la configuración y combinación de los componentes de un cortafuegos.

Hoy en día es habitual encontrarse con arquitecturas en las cuáles el cortafuegos tiene tres interfaces de red, uno a la red interna, otro a los servidores públicos de la DMZ y otro al router externo. La ventaja de esta configuración es que aunque alguno de los servidores públicos (o proxies) sea penetrado, el atacante tiene que seguir pasando a través del proxy.

Aunque pueda desafortunadamente tampoco ofrece ninguna ventaja si el atacante engaña a un usuario (dentro de la red interna) y penetra su máquina. En estos casos puede ser de alguna ayuda el uso de un cortafuegos de nivel de aplicación, y los más comunes son los Web Application Firewalls, que intentan buscar las amenazas más comunes en las respuestas de los servidores web externos antes de devolverlas al usuario correspondiente.

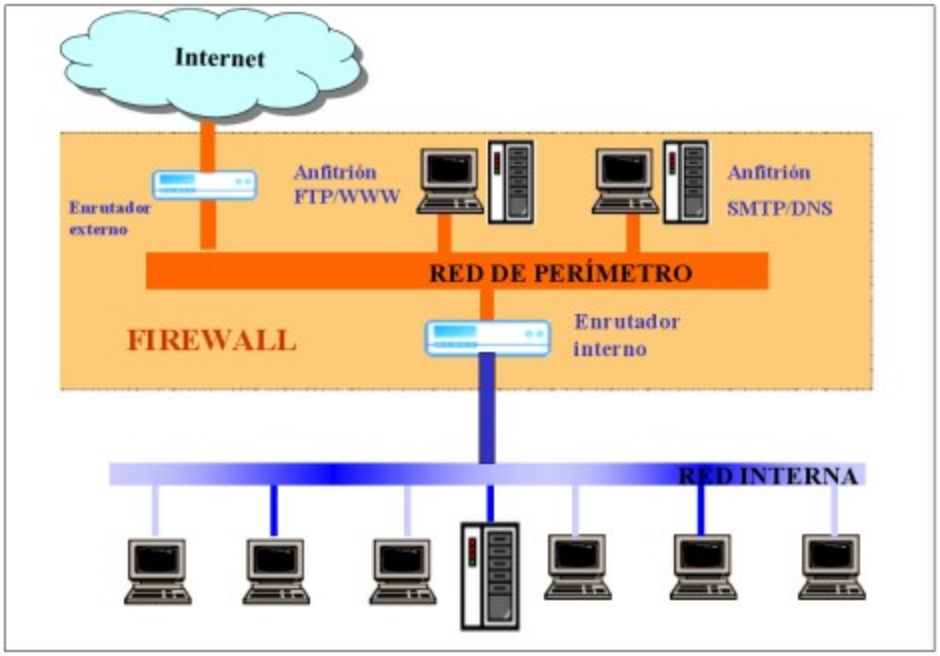
2.4.1. UTILIZACIÓN DE MÚLTIPLES ANFITRIONES BASTIONES

Las razones por las que se recomienda esta configuración, ver figura VII-8, son el rendimiento, la redundancia y la necesidad de separar datos o servidores por cuestiones de seguridad:

Para que el rendimiento de los usuarios propios no se vea mermado por las actividades de usuarios externos, se puede decidir que un anfitrión bastión gestione los servicios importantes para los usuarios internos (como SMTP, *proxy*, etc.), mientras que el otro anfitrión maneje los servicios que proporciona Internet (por ejemplo, FTP anónimo).También, por razones de rendimiento, se pueden instalar varios anfitriones bastión con los mismos servicios, aunque esta solución presenta el problema del balanceo de cargas.

Para obtener redundancia el cortafuegos se puede configurar con varios anfitriones bastión, de este modo, si uno falla o está sobrecargado los servicios pueden ser proporcionados por otro. Por ejemplo, se podría configurar y designar varios anfitriones bastión como servidores DNS para un dominio (por medio de registros NS de DNS que especifiquen los servidores de nombre para un dominio) o como servidores SMTP (por medio de registros MX [Intercambio de correo] de DNS, que especifican qué servidores aceptarán correo para un anfitrión o dominio determinados) o ambos.

Por razones de seguridad y para evitar que los conjuntos de datos de los servicios se interfieran entre sí, se podría, por ejemplo, proporcionar un servidor *http* para que lo usen los clientes de la empresa en Internet, y otro para que lo use el público en general. Al proporcionar dos servidores, se pueden ofrecer diferentes datos a los clientes y es posible mejorar el rendimiento, pues permite usar una máquina menos cargada y más potente. También se podrían ejecutar los servidores *http* y el FTP anónimo en máquinas independientes, a fin de eliminar la posibilidad de que se pueda utilizar un servidor para poner en riesgo el otro



Múltiples anfitriones bastiones

2.4.2. Fusión de enrutadores y el anfitrión bastión

Si se tiene un enrutador con suficiente capacidad y flexibilidad, los enrutadores interior y exterior se pueden fusionar en uno solo.

Se necesita un enrutador que permita especificar tanto los filtros de entrada como los de salida en cada interface. En una interface del enrutador se mantiene la red de perímetro, y en la otra interfaz se efectúa la conexión a la red interna. Cierta parte del tráfico fluiría de manera directa entre la red interna e Internet (el tráfico permitido por las reglas establecidas en el enrutador para el filtrado de paquetes) y la otra parte fluiría entre la red de perímetro e Internet, o la red de perímetro y la red interna (el tráfico manejado por los servidores *proxy*).

Esta arquitectura, como la de anfitrión de protección, hace vulnerable el sitio porque sólo hay un enrutador.

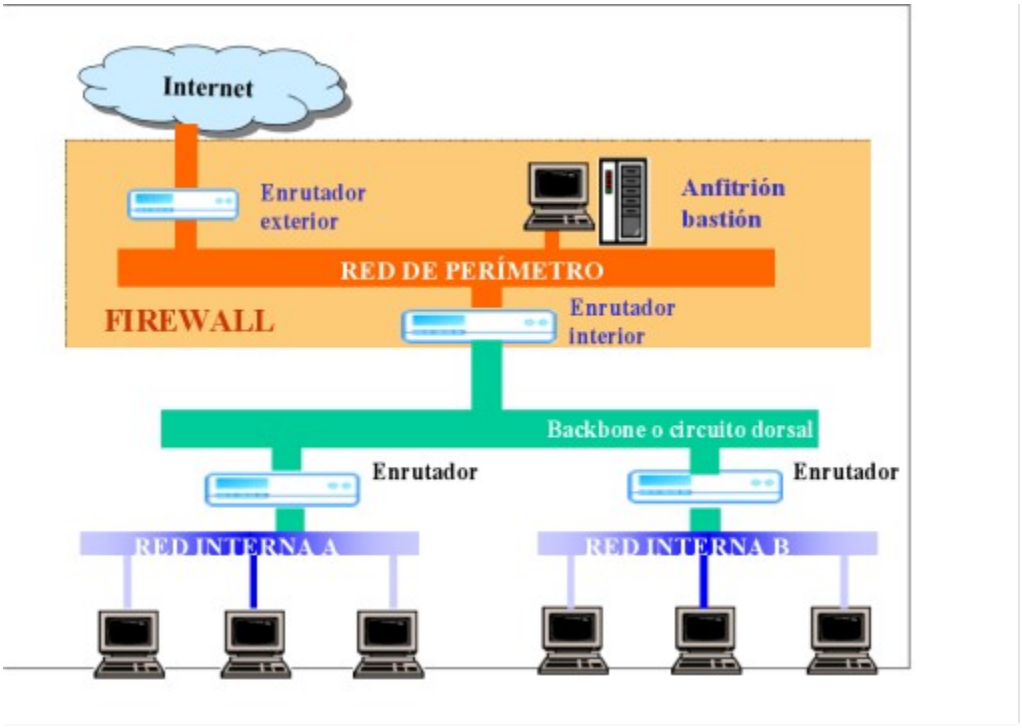
Quizá haya casos en que se utilice una sola máquina con doble acceso como anfitrión bastión y como enrutador exterior. Se puede usar un anfitrión con doble acceso para enrutar tráfico y aunque no dará el mismo rendimiento o la flexibilidad que un enrutador dedicado, será suficiente para una sola conexión de bajo ancho de banda.

2.4.3. Utilizar múltiples enrutadores interiores

El problema básico de esta configuración es que el software de enrutamiento en un sistema interno podría decidir que la vía más rápida hacia otro sistema interno es a través de la red de perímetro. Si hay suerte, este enfoque simplemente no funcionará porque estará bloqueado por el filtrado de paquetes de uno de los enrutadores. Si no hay suerte, funcionará y habrá tráfico sensible, totalmente interno, fluyendo a través de la red de perímetro, donde pueden observarlo si alguien logra entrar al anfitrión bastión.

Usar varios enrutadores interiores para conectar la red de perímetro a varias partes de la red interna puede causar muchos problemas y, en general, es mala idea.

Una razón para instalar varios enrutadores interiores es si se tienen varias redes internas, y existen razones técnicas, organizativas o políticas para no compartir un solo enrutador. Aunque la forma más sencilla de acomodar estas redes sería darles interfaces separadas en un solo enrutador, si hay demasiadas redes para un solo enrutador, o si compartir un enrutador es inaceptable por otras razones, se puede instalar un *backbone* interno (red dorsal interna) y conectarlo a la red de perímetro con un solo enrutador, como muestra la figura de abajo:



Múltiples redes internas (arquitectura backbone)

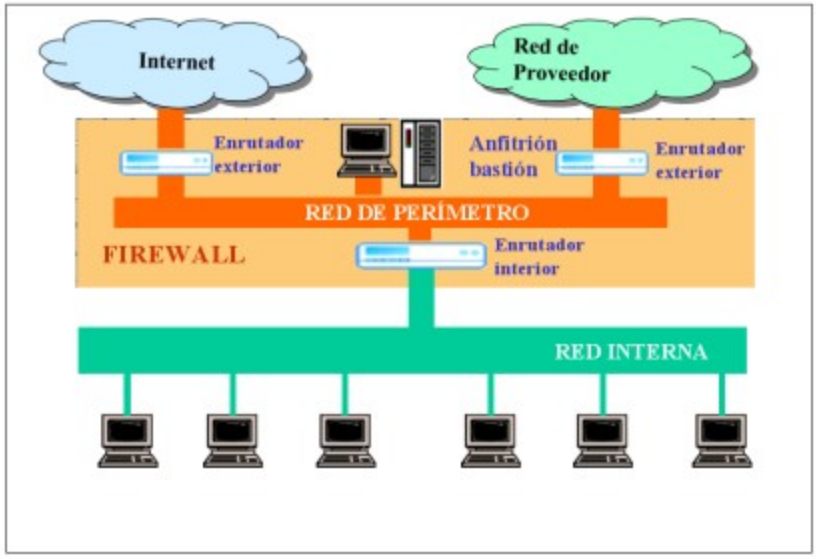
2.4.4. Utilización de varios enrutadores exteriores

Existen ciertos casos en que tiene sentido conectar varios enrutadores exteriores a la misma red de perímetro. Por ejemplo cuando:

- Para tener redundancia, se tienen varias conexiones con Internet, a través de diferentes proveedores de servicio (multihoming).
- Se tiene una conexión a Internet, más otras conexiones a otros sitios.

Anexar varios enrutadores exteriores que van a la misma red externa (por ejemplo, dos proveedores de Internet distintos) no es un problema de seguridad significativo. Aunque esta solución permite tener diferentes conjuntos de filtros, sería mejor tener un enrutador exterior con múltiples interfaces de red exterior.

Las cosas son más complejas si las conexiones son a diferentes lugares; por ejemplo, una a Internet y otra a un sitio con el que se está colaborando y para el que se necesita más ancho de banda. Este tipo de arquitectura tiene sentido para evitar que un atacante que entrara al anfitrión bastión de la red de perímetro curiosease el tráfico sensible entre su sitio y el colaborador. En caso de ser así, convendría instalar varias redes de perímetro en lugar de varios enrutadores exteriores en una sola red de perímetro.



Arquitectura que utiliza varios enrutadores exteriores

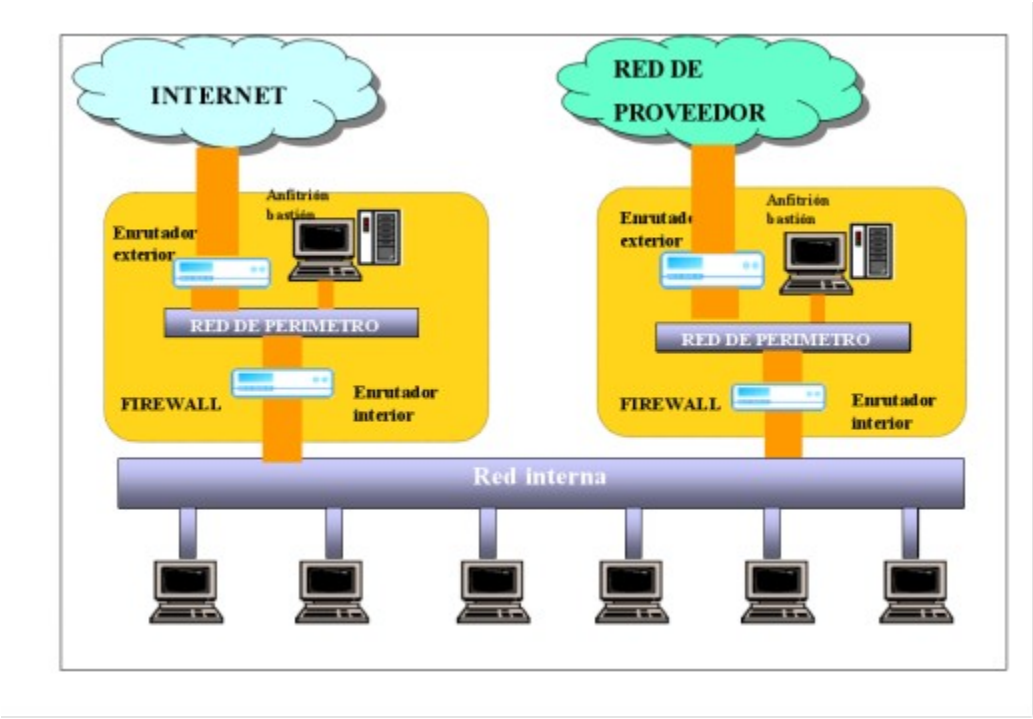
2.4.5. Utilización de múltiples redes de perímetro

En ciertas situaciones tiene sentido que la configuración incluya múltiples redes de perímetro.

No tiene mucho sentido pagar por dos conexiones a Internet y luego ejecutar ambas a través del mismo enrutador o enrutadores o dos enrutadores a través de una única red de perímetro.

Poner dos enrutadores exteriores, dos redes de perímetro y dos enrutadores interiores asegura que no haya un único punto de falla entre las redes internas y las externas.

También se podrían colocar dos o más redes de perímetro, a fin de transmitir datos moderadamente confidenciales a través de una red perimetral, y una conexión a Internet a través de la otra. En este caso, hasta puede conectar ambas redes de perímetro al mismo enrutador interior.



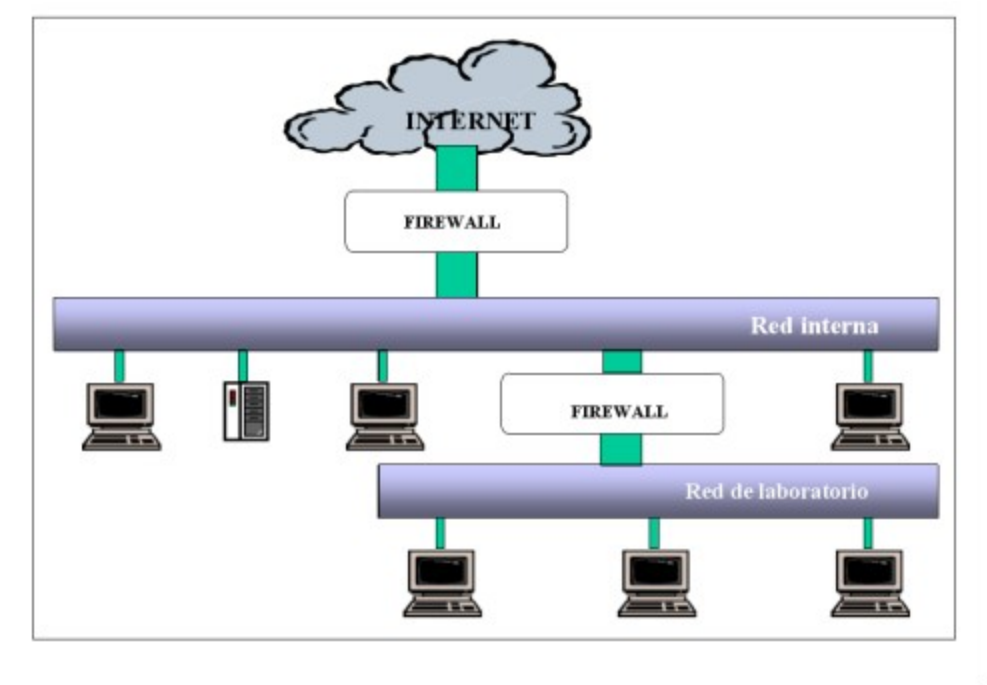
Tener múltiples redes de perímetro es menos arriesgado que tener múltiples enrutadores interiores compartiendo la misma red interna, pero aun así el mantenimiento es un problema. Teniendo múltiples enrutadores internos se pueden presentar varios puntos de riesgo. Esos enrutadores deben vigilarse con cuidado para que continúen aplicando las políticas de seguridad apropiadas.

2.5. Cortafuegos internos

Con frecuencia tiene sentido mantener una parte de una organización separada de otra. No todos los usuarios necesitan los mismos servicios o información, y con frecuencia la seguridad es más crítica en unas unidades que en otras. La mayoría de las organizaciones tienen aplicaciones o procesos que deben ser especialmente seguros. En las universidades, pueden ser proyectos de investigación específicos, o las oficinas de registro de alumnos; en compañías comerciales, pueden ser nuevos productos bajo desarrollo; en casi cualquier lugar, las máquinas de contabilidad y finanzas necesitan protección adicional. Algunos trabajos del gobierno clasificados también requieren protección extra.

Muchas de las herramientas y técnicas que se utilizan para construir cortafuegos para Internet también son útiles para construir estos cortafuegos internos.

Esta tendencia se lleva al extremo hoy en día y es habitual que haya configurado un cortafuegos dentro de cada máquina que se conecta a la red.



Arquitectura para cortafuegos con una red de laboratorio

Capítulo 3 CONFIGURACIÓN DE CORTAFUEGOS

3.1. CARACTERISTICAS PARA CONFIGURAR EL SERVICIO DE CORREO ELECTRÓNICO

El correo electrónico es uno de los servicios más vulnerables. Los servidores de correo son blancos tentadores para los atacantes, porque aceptan datos arbitrarios de cualquier anfitrión externo.

Un sistema de correo tiene tres partes, cada una de estas partes es vulnerable por razones distintas:

- Un servidor acepta correo de anfitriones externos o lo envía a ellos. El servidor acepta comandos directamente (relacionados con la entrega de correo) de anfitriones externos, si el servidor no es seguro, puede dar al atacante todo el acceso que el mismo tiene. A*taques en el canal de comandos*
- Un agente de entrega coloca el correo en el buzón correcto en el anfitrión local. El agente de entrega necesita permisos especiales porque necesita escribir dentro del

buzón de cada usuario. Aunque el agente de entrega no necesita comunicarse con el mundo externo, si de alguna forma se le puede alterar el intruso obtiene un acceso muy amplio

- Un agente de usuario deja que el receptor lea el correo y componga correo de salida. El agente de usuario se ejecuta como un usuario y no se comunica con el mundo exterior; sin embargo, con frecuencia puede ejecutar otros programas arbitrarios en respuesta a los datos recibidos.

El agente de entrega y el agente de usuario son vulnerables a peligros en el contenido de los mensajes de correo; *ataques manejados por datos*.

Para protegernos contra ataques en el canal de comandos, un cortafuegos puede limitar el número de máquinas en las que los atacantes pueden abrir canales de comandos y entonces proporcionar un servidor seguro en esas máquinas. SMTP ha ido mejorando (ESMTP) con muchas extensiones, entre otras las dedicadas a seguridad que pueden limitar el acceso al servidor de SMTP únicamente a usuarios/máquinas basado en autenticación fuerte o con canales seguros. Estas evoluciones mejoran mucho la seguridad pero aun así es aconsejable reforzarlas mediante el cortafuegos.

No hay mucho que pueda hacer un servidor contra ataques manejados por datos; tiene que permitirse el paso de los datos, o no se podrá recibir correo. La mejor postura es ejecutar agentes de entrega y agentes de usuario actualizados y educar a los usuarios.

Existe una tendencia generalizada a ofrecer acceso al correo a través de un interfaz web. Esto no sustituye a los protocolos que aquí se explican que los tendrá que implementar el servidor web a través de una pasarela.

3.1.1. PROTOCOLO SIMPLE DE TRANSFERENCIA DE CORREO *(SMTP)*

En Internet, el intercambio de correo electrónico entre servidores de correo se maneja con *SMTP*. En el anfitrión bastión, un servidor *SMTP* acepta correo y examina la dirección destino para decidir si entrega el correo localmente o lo transmite a otra máquina.

SMTP es un sistema de *guardar y enviar*, y tales sistemas se combinan bien con las aplicaciones de cortafuegos, en particular con las que utilizan servicios *proxy*. La mayoría de los sitios envían las conexiones *SMTP* a un anfitrión bastión que ejecuta un servidor *SMTP* seguro, que es el *proxy*. Por ello no se permitirá que los anfitriones externos se pongan en contacto con anfitriones internos por medio de *SMTP*. Sólo anfitriones configurados especialmente pueden aceptar conexiones *SMTP* en forma segura.

SMTP es un servicio basado en TCP; los receptores *SMTP* utilizan el puerto 25. Los remitentes de *SMTP* emplean un puerto seleccionado al azar por encima del 1023.

Las principales características para configurar *SMTP* se pueden resumir en:

- Utilizar las características de guardar y enviar de *SMTP* para enviar todo el correo que entra y sale a través del anfitrión bastión

- Utilizar el filtrado de paquetes para limitar las conexiones *SMTP* de los anfitriones externos sólo al anfitrión bastión
- Utilizar el filtrado de paquetes para limitar las conexiones *SMTP* del anfitrión bastión a un servidor interno o conjunto de servidores específicos
- Permitir que cualquier sistema interno envíe correo de salida al anfitrión bastión
- Mantener al día los parches en los agentes de entrega y agentes de usuario
- Educar a los usuarios en relación con los engaños basados en el correo, así como darles instrucciones sobre la ejecución de determinados programas o para cambiar sus contraseñas a cadenas específicas

Generalmente, los filtros a establecer son para permitir *SMTP* de entrada y salida sólo entre anfitriones externos y el anfitrión bastión, y entre el anfitrión bastión y los servidores de correo internos.

3.1.2. PROTOCOLOS DE ENTREGA DE CORREO ELECTRÓNICO: *POP E IMAP*

SMTP se utiliza para intercambiar correo entre servidores. Para que los usuarios accedan a su correo como un archivo en la máquina donde fue entregado por *SMTP*, se suele utilizar un protocolo independiente.

POP e IMAP son protocolos cliente-servidor que sirven para manejar buzones electrónicos de usuario.

Con POP, el buzón de un usuario (el verdadero archivo donde el correo electrónico del usuario se guarda para que lo vea después) está en un servidor, no en la máquina personal del usuario. Cuando el usuario quiere su correo electrónico, accede a su buzón utilizando un programa cliente en su propia máquina empleando el protocolo POP. POP echa un cerrojo al buzón y no permite el acceso concurrente al buzón a ninguna otra aplicación, ni siquiera para recibir mensajes nuevos. POP permite conocer los mensajes que hay en el servidor y su longitud, transferir mensajes, marcarlos para su borrado y cerrar la sesión borrando los mensajes marcados.

IMAP es un protocolo más evolucionado que POP, soporta la gestión y suscripción a múltiples buzones en el servidor, soporta MIME (es capaz de referenciar partes y tipos de un mensaje), permite buscar, bajar, marcar y mover mensajes entre buzones. IMAP se concibió como un protocolo que permitiera la gestión de correo desde múltiples clientes (no tiene el problema del cerrojo del buzón único que tiene POP). IMAP se puede usar en modo on-line, off-line y desconectado, de forma que los usuarios nómadas pueden reconciliar sus cambios entre conexiones al servidor.

Con el uso de *POP* a través de Internet se presentan dos cuestiones de seguridad importantes:

- Primera, evitar que los clientes y servidores *POP* estándar envíen la contraseña *POP* del usuario a través de Internet en forma clara, de modo que cualquiera que espíe la conexión

puede captar y volver a utilizarla después con todos los privilegios del usuario (no sólo su correo electrónico); en la mayoría de los casos, la contraseña *POP* es la misma que la contraseña de usuario de inicio de sesión. Existen variantes de *POP* más seguras que soportan Kerberos (con frecuencia llamado KPOP) y contraseñas que no se pueden volver a utilizar para autenticar (con frecuencia llamado APOP), pero estas variantes seguras no están soportadas ampliamente.

- Segunda, la confidencialidad, el correo que se acceda a través de Internet por medido de *POP* estará visible a quien espíe las sesiones de *POP*, por eso hay organizaciones que no permiten acceder al correo interno a través de Internet.

POP es un servicio basado en TCP. Los servidores *POP* para la versión actual del protocolo *POP*, denominada *POP*3, emplean el puerto 110. Los clientes *POP* ocupan puertos por encima del 1023. Las reglas a establecer para el filtrado de paquetes podrían ser las siguientes:

Dirección	**Dirección fuente**	**Dirección destino**	**Protocolo**	**Puerto fuente**	**Puerto destino**	**ACK**	**Notas**
Dentro	Externo	Interno	TCP	>1023	110	[a]	POP entrante C-S
Fuera	Interno	Externo	TCP	110	>1023	Sí	POP saliente S-C
Fuera	Interno	Externo	TCP	>1023	110	[a]	POP saliente C-S
Dentro	Externo	Interno	TCP	110	>1023	Si	POP entrante S-C

[a] ACK no está encendido en el primer paquete de este tipo (establecer conexión) pero lo estará en los demás

C-S (de cliente a servidor), S-C (de servidor a cliente)

Una conexión *POP* de salida permitiría a los usuarios de un sitio obtener su correo desde otros sitios, la cual habría que permitir si existiera cierta demanda. *POP* a través de Internet es tan poco común que quizá a nadie le interesa *POP* de salida.

Las conexiones *POP* entrantes son las que permiten a las personas que están en otros sitios leer el correo entregado para ellas en su sitio. Si se requiere esta funcionalidad las conexiones *POP* se deben limitar a un servidor *POP* ejecutándose en un solo anfitrión, lo cual limitará el número de cuentas vulnerables y la información que pasa a través de Internet.

Las principales características para configurar *POP* se pueden resumir en:

- No permitir que los usuarios transfieran el correo de su sitio a través de Internet por medio de *POP*, a no ser que se pueda hacer sin dar a conocer contraseñas o que no preocupe la confidencialidad del correo en sí o se tenga un canal cifrado para transferirlo.

- Si hay usuarios que desean transferir correo desde otros sitios por medio de *POP*, se debe hacer filtrado de paquetes, tal vez limitándolo a conexiones provenientes de sitios específicos o a anfitriones específicos de la red interna.

La **inseguridad** del protocolo POP, que solo ofrece autenticación segura con APOP y KPOP, ha llevado a una solución muy extendida: situarlo detrás de un servidor (wrapper) de SSL/TLS. De esta forma las sesiones POP se pueden beneficiar de la seguridad de la sesión negociada e incluir confidencialidad e integridad. POP3S se presta por TCP en el puerto 995.

Las mismas consideraciones caben con IMAP, que también se diseñó como un protocolo inseguro. IMAP se definió desde su versión 4 con capacidades de seguridad de forma que permite el uso de avanzados APIs como Servicio Genérico de Seguridad (GSS), o STARTTLS. IMAP se presta sobre TCP en el puerto 220, o sobre SSL/TLS en el puerto 993.

3.2. CARACTERÍSTICAS PARA CONFIGURAR HTTP

HTTP va sobre TCP en su puerto 80, habitualmente. Los proxy-caches de HTTP a menudo los encontramos en el puerto 8080. Cuando se quiere dar un servicio seguro, se utiliza HTTP sobre un canal seguro SSL/TLS en el puerto 443, en ocasiones también en el 8443.

3.2.1. CARACTERÍSTICAS DE *HTTP* PARA USARLO COMO *PROXY*

La mayoría de los clientes http soportan proxy transparente al usuario.

El uso del servidor http como servidor proxy puede proporcionar un beneficio adicional, al utilizar caché localmente con las páginas web obtenidas de Internet, que permite mejorar el rendimiento del cliente y reducir los requerimientos de ancho de banda de la red. Esto lo hace asegurándose que las páginas web más usuales se obtengan solamente una vez en un sitio; la segunda vez y subsiguientes se obtiene la copia de la página del caché local, en lugar de una copia nueva del servidor original desde Internet.

Aunque hay varias alternativas en relación donde colocar el servidor web la más común es colocarlo en la red perímetro.

3.2.2. LA SEGURIDAD EN RELACIÓN CON *HTTP*

Los servidores son susceptibles de recibir ataques de clientes maliciosos, típicamente explotando vulnerabilidades conocidas de desbordamiento de buffer (buffer overflow), o de vulnerabilidades en módulos o bibliotecas que carga el servidor (por ejemplo php, jdk, etc.)

Los clientes pueden ser atacados por servidores maliciosos, que pueden intentar desde acceder a información de contexto/sesión presente en el cliente (navegador), instalar programas maliciosos, o hacerles vícitimas de ataques más sofisticados como los de cross-site scripting, en los que participa más de un servidor y más de uno puede ser malicioso.

Configurar los clientes *http* con cuidado y advertir a los usuarios de que no deben cambiar la configuración ni instalar extensiones, etc. basándose en consejos externos y nunca por correo electrónico. Establecer procedimientos para revisar dicha configuración de forma remota y si es

posible automatizarlo. Si los requerimientos de seguridad son muy altos, podemos instalar un agente que envíen alertas cada vez que un usuario cambia cualquier aspecto de la configuración.

Si nuestro presupuesto lo permite, debemos considerar la adquisición de un WAF. Aún así si un atacante conoce y explota un buffer overflow del navegador de nuestros usuarios, que el WAF no conoce, el atacante podrá ejecutar código con el nivel de privilegios del usuario.

3.3. CARACTERÍSTICAS PARA CONFIGURAR *SSH*

SSH es un servicio de ejecución remota de comandos basado en TCP.

Por lo general los servidores SSH utilizan el puerto 22 y los clientes puertos por encima del 1023.

SSH permite establecer una sesión en un servidor remoto y además facilita el establecimiento de túneles para transportar otros servicios, por ejemplo la interfaz gráfica (XWindows o el que sea), una conexión con un servidor de base de datos, etc. Hay clientes y servidores de SSH para la inmensa mayoría de los sistemas operativos más populares, y muchos de ellos son software libre.

Las principales recomendaciones para configurar SSH se resumen en:

- Realizar una gestión consciente de las claves públicas de las máquinas.
- Cada vez que se actualice la clave, actualizarla en los sistemas implicados.
- Difundir y publicar el historial de claves públicas actualizado.
- Limitar el acceso privilegiado (root) a la máquina, no permitirlo con contraseña a no ser que sea indispensable.
- No permitir versiones antiguas del protocolo (ssh1).
- No permitir a todos los usuarios, por ejemplo limitarlo a usuarios físicos que lo soliciten.
- Forzar el uso de claves públicas largas (1024/2048) y desautorizar el uso de contraseñas.
- Protegerlo detrás del firewall.

El uso de claves públicas en SSH (SFTP/SCP) es manifiestamente más sencillo que la utilización de contraseñas.

El único punto débil es que las credenciales típicamente se almacenan en la máquina del usuario.

Hoy en día los usuarios a menudo acceden a los servicios utilizan máquinas diferentes, si queremos garantizar el acceso a SSH desde cualquiera de ellas, debería de actualizarse en el servidor la clave pública del usuario en cada máquina y añadirla a la lista de máquinas autorizadas (authorized_keys) en su cuenta del servidor de SSH, y el administrador debería de encargarse de mantener actualizada la lista de máquinas conocidas (known_hosts) en todas las máquinas.

Las implementaciones SSH vienen acompañadas de una herramienta que permite la generación de claves de usuario.

En el caso concreto de openssh, es ssh-keygen.

Veamos un ejemplo de utilización de ssh y generación de claves:

```
amarin@host:~/tmp/ssh$ ssh-keygen -b 2048  -t rsa

Generating public/private rsa key pair.
Enter file in which to save the key (/home/amarin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in
/home/amarin/tmp/ssh/id_rsa.
Your public key has been saved in
/home/amarin/tmp/ssh/id_rsa.pub.
The key fingerprint is:
c2:e8:11:f4:63:99:2e:ff:f7:9c:93:c9:3c:52:e5:af amarin@host


amarin@host:~$ cat /home/amarin/ssh/id_rsa.pub

ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAyWRhtTB2ugfTLPYbeqw...=
amarin@host

amarin@host:~$ cat /home/amarin/ssh/id_rsa

-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,F7C650B545DEBC91

JAEFw012ofBNfaFbLiaOJfOTwQbsyfzqRnGJyhRtcjo4P4il+v/mcJkQV1R5gqiu
Itr
...
3I26DcyG/fzfd4E+5GGsAz0z/CPSpzUzgzBMkWFbDEqVlkVeOHFYrQ==
-----END RSA PRIVATE KEY-----
```

Una vez generada la nueva pareja de claves del usuario, hay que actualizarla en las claves autorizadas de los servidores a los que normalmente acceda, en los ficheros authorized_keys de su cuenta. Una vez hecho esto, podemos entrar en el servidor utilizando la pareja de claves pública/privada que hemos creado. Lo podemos ver en detalle si activamos la opción de depuración con -v.:~$ **slogin -v host**

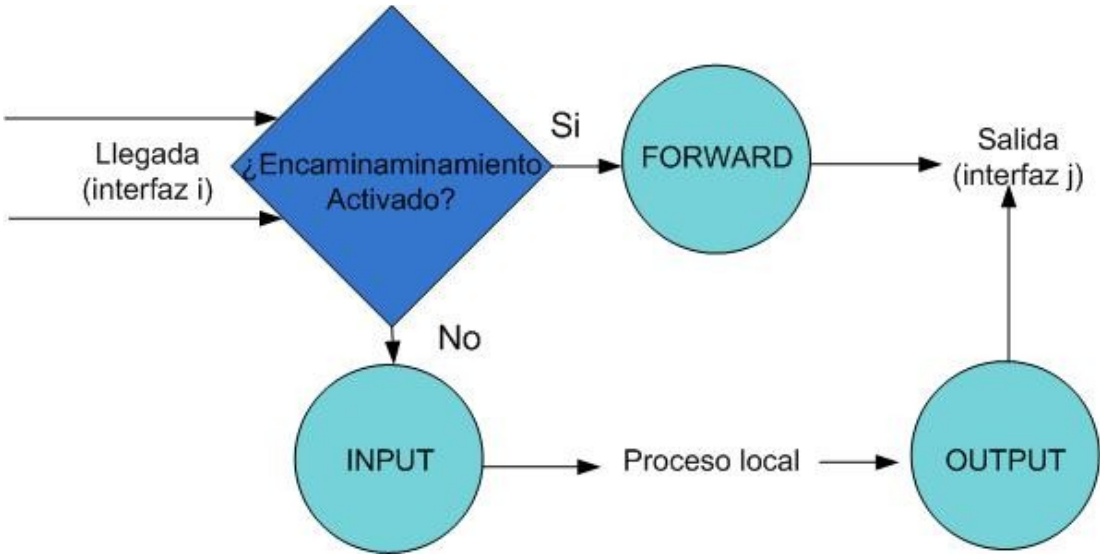
3.4. CONFIGURACIÓN DE REGLAS DE FILTRADO CON IPTABLES

Un filtro de paquetes es un software que examina las cabeceras de los paquetes y decide si debe descartarlo (DROP), aceptarlo (ACCEPT), u otras manipulaciones (QUEUE, RETURN). Los filtros nos permiten controlar y vigilar el tráfico y asegurar niveles de seguridad en la máquina.

IPTables es una potente herramienta para gestionar y especificar el filtrado de paquetes y las traducciones NAT disponible en cualquier ditribución linux. IPTables utiliza por debajo netfilter. Desde los kernels 2.4, netfilter es el sucesor de ipchains (2.2) y de ipfwadm (2.0).

IPTables utiliza listas de reglas en la tabla de filtros, a estas listas se les denomina cadenas. Cuando un paquete entra en una cadena, se va comparando con las reglas de la cadena. Si encaja con la regla, se tomará la política de la regla, si no encaja con ninguna, se aplicará la política de la cadena (generalmente DROP).

El núcleo comienza con tres cadenas por defecto: INPUT, FORWARD y OUTPUT. Cuando se recibe un paquete en una interfaz, y el paquete va destinado a la propia máquina, se aplica la cadena INPUT, si el núcleo tiene activada la función de reenvío y el paquete va destinado a otra interfaz de red, se aplica la cadena FORWARD. Si el paquete se recibe de un proceso local de la máquina, se le aplica el filtro OUTPUT, gráficamente se representa así:



Donde el interfaz i puede ser igual o distinto al j.

IPTables permite diversas operaciones sobre cadenas, cómo crear una nueva cadena (-N), borrar una cadena vacía (-X), vaciar de reglas una cadena (-F), listar las reglas (-L), etc. También permite manipular las reglas de una cadena: añadir una nueva regla a una cadena (-A), borrar una regla de un sitio en particular de una cadena (-D), etc.

Al definir las reglas, IPTables permite especificar la dirección fuente, destino, servicio, protocolo, etc. Y para los distintos servicios y protocolos hay definidas extensiones, pues IPTables es extensible.

Veamos un caso prático para filtrar el uso de POP3:

```
iptables -N RULE_POP

iptables -A OUTPUT -p tcp -m tcp --dport 110 -m state --state NEW -j RULE_POP

iptables -A INPUT -p tcp -m tcp --dport 110 -m state --state NEW -j RULE_POP

iptables -A FORWARD -p tcp -m tcp --dport 110 -m state --state NEW -j RULE_POP

iptables -A RULE_POP -j ACCEPT
```

Ahora bien, si lo que queremos es denegar el uso de POP3 y solo permitir el uso de POP con SSL, cambiaríamos en la última regla anterior ACCEPT por DROP y añadiríamos las reglas:

```
iptables -N RULE_POP3S

iptables -A OUTPUT -p tcp -m tcp --dport 995 -m state --state NEW -j RULE_POP3S

iptables -A INPUT -p tcp -m tcp --dport 995 -m state --state NEW -j RULE_POP3S

iptables -A FORWARD -p tcp -m tcp --dport 995 -m state --state NEW -j RULE_POP3S

iptables -A RULE_POP3S -j LOG --log-level info --log-prefix "POP3S: aceptada "

iptables -A RULE_POP3S -j ACCEPT
```

Veamos un caso prático para permitir el uso de SSH. Comenzamos permitiendo el acceso a la máquina por ssh:

```
iptables -A INPUT -i eth0 -p tcp –dport 22 -j ACCEPT
```

A continuación consideremos el caso muy común ataques de diccionario de SSH, es decir, vamos a protegernos de que una o más máquinas estén intentando probar con combinaciones de usuarios/contraseñas para entrar en nuestra máquina por SSH. Específicamente lo que vamos a hacer es poner un contador de forma que si en menos de 10 minutos tenemos más de 15 intentos desde el mismo origen, tiramos el paquete.

Primero borramos y creamos una nueva cadena que llamaremos ATSSH:

```
iptables -X ATSSH

iptables -N ATSSH
```

Especificamos una regla con el límite de intentos:

```
iptables -A ATSSH -m recent ! --rcheck --seconds 600 --hitcount 15 --rttl --name SSH --rsource -j RETURN

iptables -A ATSSH -j LOG --log-level debug --log-prefix "Parado ataque a SSH:  "
```

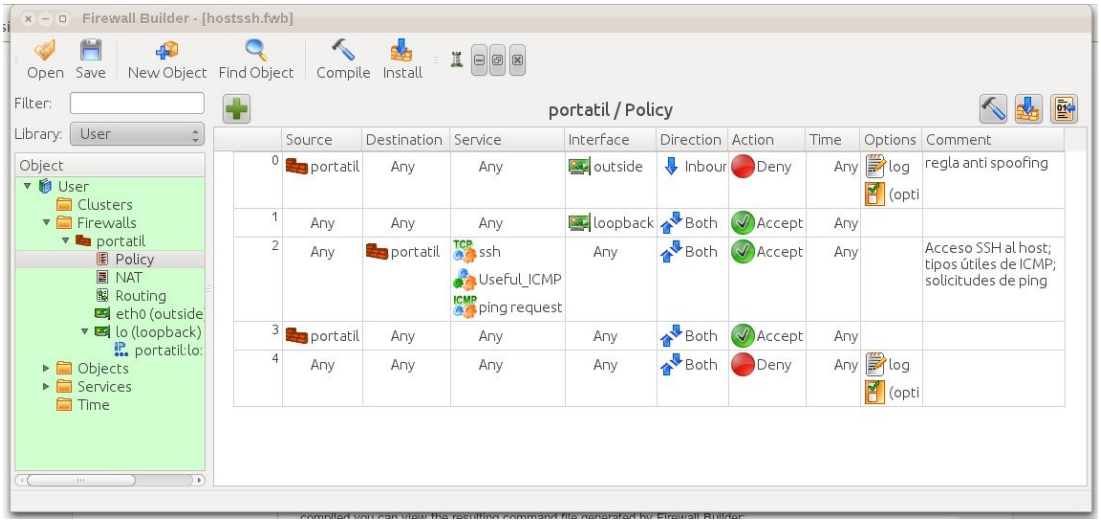
```
iptables -A ATSSH -p tcp -j DROP
```

Por último añadimos mediante una regla en la cadena INPUT la cadena que acabamos de definir (ATSSH):

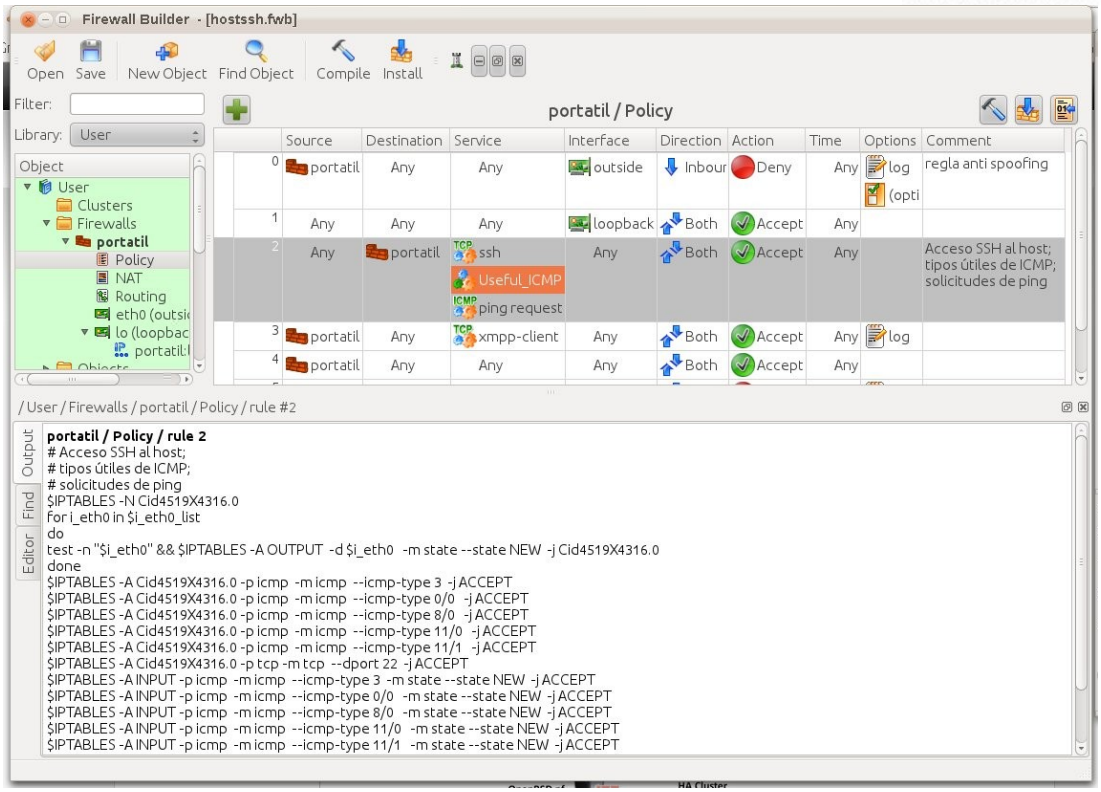
```
iptables -A INPUT -i eth0 -p tcp --dport 22 -m state --state NEW -m recent --name SSH --set
--rsource -j ATSSH
```

Seguir la especificación de cadenas y reglas, puede llegar a ser tedioso e incluso complicado. Existen herramientas que ofrecen un interfaz de usuario más amigable que la línea de comandos de iptables, por ejemplo fwbuilder, turtle firewall, o ufw.

Fwbuilder, es una herramienta GUI muy popular que permite gestionar los firewalls de distintas máquinas: examinarlos, cambiarlos, y cargarlos desde la máquina del administrador. Tiene gran cantidad de plantillas para diversos cortafuegos, entre ellos los basados en iptables. Además ofrece una biblioteca de protocolos, aplicaciones y utilidades que puede agilizar la definición y comprobación de reglas. A continuación mostramos una captura de pantalla de la configuración de una máquina que puede ser accedida solo a través de SSH:



También podemos seleccionar una regla y pedirle que la compile para examinar el resultado:



3.5. CARACTERÍSTICAS PARA CONFIGURAR EL *DNS*

DNS es un sistema de gestión de los dominios de nombres en Internet. Su funcionamiento es similar a una base de datos distribuida que traduce los nombres de anfitrión a direcciones IP, y direcciones IP a nombres de anfitrión. DNS también permite gestionar otros tipos de datos sobre los dominios de nombres en Internet.

DNS define los dominios de nombres en Internet de frma jerárquica en árbol, y las separa por etiquetas y un carácter de separación, el punto (.), por ejemplo www.google.es.

DNS establece una separación entre las administraciones de las distintas zonas de Internet. Por ejemplo uc3m.es denota la zona correspondiente a la autoridad de la Universidad Carlos III de Madrid.

Cada zona corresponde a un dominio, en el que puede haber otros subdominios. Si se delega la autoridad de los subdominios a otra autoridad, entonces se crea una nueva zona. Por ejemplo, it.uc3m.es denota la zona del departamento de ingeniería telemática dentro de la misma universidad, sin embargo der.uc3m.es es un subdominio dentro del dominio de la universidad, porque no se ha pedido la delegación.

La zona jerárquicamente más alta es la que cubren los servidores raíz de Internet (root-servers) y corresponde a la etiqueta después del punto final (etiqueta vacía). A continuación se encuentran las etiquetas de dominios más altos o top-level domains (.com, .edu, .gov, .mil) y las geográficas (.es, .de, .fr, uk). El ICANN es la autoridad que se encarga de los nombres en

Internet, creación de TLD, etc. Esto comporta asignación de nombres y llevar las tareas de gestión asociadas al seguimiento de zonas asignadas, el ICANN cuenta con agentes de registro que llevan estas tareas de forma comercial. El Internic es un instrumento del ICANN para el gobierno de nombres que nos facilita la consulta de dominios y datos asociados con los técnicos responsables (whois).

DNS funciona de la siguiente manera: cuando una aplicación necesita conocer algún dato sobre un dominio/nombre, construye una consulta que envía a su servidor DNS. Si éste tiene la información guardada de preguntas anteriores la devolverá directamente, y en caso contrario reenviará la consulta al servidor de nombres sufijo cuya dirección conozca. Por esta razón todos los servidores DNS están obligados a conocer las direcciones de todos los servidores raíz. Una vez obtenida la respuesta se devolverá al que la originó (y los servidores intermedios la almacenarán en sus cachés durante el tiempo marcado por el administrador de la zona.

3.5.1. CARATERÍSTICAS DEL *DNS* PARA EL FILTRADO DE PAQUETES

Hay dos tipos de actividades *DNS* de red: búsquedas y transferencias de zona.

- Las búsquedas se producen cuando un cliente *DNS* (o un servidor *DNS* que actúa por parte de un cliente) consulta a un servidor *DNS* sobre cierta información; por ejemplo, la dirección IP de un nombre de anfitrión específico, el nombre de anfitrión de una dirección IP determinada, el servidor de nombres para un dominio específico, o el servidor de correo para un anfitrión.

- Las transferencias de zona se producen cuando un servidor *DNS* (el servidor secundario) solicita de otro servidor *DNS* (el servidor principal) todo lo que sabe sobre una pieza determinada del árbol de nombres de *DNS* (la zona)

Por razones de rendimiento, las búsquedas de *DNS* normalmente se ejecutan mediante UDP. Si el resultado no cabe en un solo datagrama UDP, la búsqueda se volverá a hacer empleando TCP. Un servidor *DNS* utiliza el puerto 53 para todas sus actividades UDP y TCP. Ocupa un puerto al azar por encima del 1023 tanto para UDP como para TCP. Por lo tanto, se puede diferenciar entre:

- Una solicitud de cliente a servidor: el puerto fuente está por encima del 1023, el puerto destino es el 53.

- Una respuesta de servidor a cliente: el puerto fuente es el 53, el puerto destino está por encima del 1023.

- Una solicitud o respuesta de servidor a servidor: con UDP, tanto el puerto fuente como el destino son el 53; con TCP el servidor que solicita utilizará un puerto por encima del 1023.

Las transferencias de zona *DNS* se hacen mediante TCP. La conexión se inicia desde un puerto al azar por encima del 1023 en el servidor secundario (que solicita los datos) al puerto 53 en el servidor principal (que envía los datos solicitados por el secundario). Un servidor secundario también debe hacer una solicitud *DNS* normal de un servidor principal para decidir cuándo hacer una transferencia de zona

DNS está estructurado para que los servidores siempre actúen como *proxy* para los clientes. También es posible utilizar una característica de *DNS* llamada *forwarding* (reenvío), para que un servidor *DNS* sea realmente un *proxy* para otro servidor.

3.5.2. PROBLEMAS DE SEGURIDAD DEL *DNS*

A continuación se describen alguno de los problemas de seguridad que plantea el uso de *DNS*.

- ***Respuestas falsas a solicitudes DNS***

El primer problema de seguridad con *DNS* es que muchos servidores y clientes *DNS* pueden ser engañados por un atacante para crear información falsa. Muchos clientes y servidores no revisan si las respuestas que obtienen se relacionan con las preguntas que realmente hicieron, o si vienen del servidor al que le preguntaron. Por ello los servidores pueden responder a solicitudes con información falsa. Por ejemplo, un atacante podría cargar la caché de su servidor con información que dice que su dirección IP corresponde al nombre de un anfitrión en el cual se confía para darle acceso sin contraseña por medio de *rlogin*. Las versiones recientes de *DNS* para UNIX (BIND 4.9 y posteriores) revisan para ver si hay respuestas falsas.

- ***Datos desiguales de nombre de anfitrión y dirección IP en los árboles DNS***

En un caso como el que acabamos de describir, si se busca el nombre de anfitrión correspondiente a la dirección IP del atacante (búsqueda inversa), se obtiene el nombre de un anfitrión en el cual se confía. Si luego se busca la dirección IP de este nombre de anfitrión (búsqueda doble inversa), se podrá comprobar que la dirección IP no es igual a la que usa el atacante, lo que indica que algo sospechoso está sucediendo.

Cualquier programa que tome decisiones de autenticación o autorización basándose en la información del nombre de anfitrión que obtiene del *DNS* debe tener mucho cuidado de verificar los datos con este método de búsqueda inversa y búsqueda doble inversa.

No se debe realizar ninguna autenticación o autorización basada sólo en el nombre del anfitrión ni en la dirección IP; no hay forma de estar seguro de que un paquete viene de la dirección IP de donde pretende venir, a no ser que exista algún tipo de autenticación criptográfica dentro del paquete que sólo la verdadera fuente puede generar.

Algunas implementaciones de búsqueda doble inversa fallan con los anfitriones con múltiples direcciones, por ejemplo, anfitriones con doble acceso usados como *proxy*. Si ambas direcciones están registradas con el mismo nombre, una búsqueda *DNS* por nombre devolverá ambas, pero muchos programas sólo leerán la primera. Si resulta que la conexión venía de la segunda dirección, la comprobación doble inversa fallará aun cuando el anfitrión esté registrado correctamente.

- ***Revelar demasiada información a los atacantes***

Otro problema que se puede presentar con el uso de *DNS* es que se suministre información considerada confidencial. Algunas organizaciones consideran información confidencial los nombres de anfitrión internos y otra información complementaria. Esas organizaciones temen que los nombres de anfitrión puedan revelar nombres de proyectos o información confidencial

de productos, o el tipo de los anfitriones (lo cual puede facilitar un ataque). Por ejemplo, es fácil adivinar el tipo de un sistema si su nombre es “lab-windows” o “cisco-gw”.

La información de nombre de anfitrión más sencilla puede ser útil para un atacante que quiere infiltrarse en un sitio por medio de engaños, de manera física o electrónica, por ejemplo mediante un ataque de *ingeniería social*.

Además de los nombres de anfitriones internos, con frecuencia se coloca otra información complementaria dentro del *DNS*, la cual es útil localmente pero no se desea que la vea un atacante. Los registros de recursos HINFO y TXT de *DNS* son muy reveladores.

3.5.3. CONFIGURACIÓN DEL *DNS* PARA OCULTAR INFORMACIÓN

Aprovechando la capacidad del *DNS* de transmitir solicitudes, se puede dar a los anfitriones internos una vista sin restricciones de los datos DNS de los servidores internos y externos, mientras que a los anfitriones externos se restringe a una vista limitada (“sana”) de datos de *DNS* internos. La figura más abajo abajo muestra cómo configurar *DNS* para ocultar información.

A continuación se dan algunas recomendaciones para revelar sólo los datos que se quiere que la gente conozca.

3.5.3.1 *Configurar un servidor DNS ´falso´ en el anfitrión bastión para que lo utilice el mundo exterior*

El primer paso para ocultar información *DNS* al mundo exterior es configurar un servidor *DNS* falso en el anfitrión bastión. A este servidor deben apuntar los registros del servidor de nombres mantenido por el dominio padre, el servidor falso debe configurarse para que sea el servidor principal del conjunto de servidores autorizados; los otros servidores *DNS* son secundarios de este servidor principal.

Todo lo que sabe este servidor falso, en el anfitrión bastión, sobre su dominio es la información que se quiere revelar al mundo exterior, que, típicamente, sólo incluye los datos básicos del nombre del anfitrión y dirección IP de los siguientes anfitriones:

- Las máquinas que están en la red de perímetro (por ejemplo, las máquinas que conforman el cortafuegos).
- Cualquier máquina que alguien del mundo exterior necesita contactar directamente. Por ejemplo, un servidor interno de noticias de Usenet (*NNTP*) accesible desde el proveedor de servicios; o cualquier anfitrión accesible a través de Internet para usuarios de confianza.

Además, se deben publicar registros MX para cualquier nombre de dominio o anfitrión utilizado como parte de las direcciones de correo electrónico en los mensajes de correo electrónico y en la colocación de noticias de Usenet, para que las personas puedan responder a estos mensajes. Hay que tener presente que la contestación a los mensajes se puede demorar días, semanas, meses y hasta años después de que fueron enviados. Si un nombre de dominio o anfitrión determinado se ha utilizado como parte de una dirección de correo electrónico,

quizá se deba conservar un registro MX para ese anfitrión o dominio para siempre, o por lo menos hasta mucho después de que ya no exista, a fin de que las personas puedan responder a mensajes antiguos.

También se debe publicar información falsa para cualquier máquina que pueda contactar el mundo exterior en forma directa. Muchos servidores en Internet (por ejemplo, la mayoría de los servidores de *FTP* anónimo principales) insisten en conocer el nombre de anfitrión (no sólo la dirección IP) de cualquier máquina que se comunica con ellos, aunque no hagan nada con el nombre de anfitrión excepto registrarlo.

Como ya mencionamos, las máquinas que tienen direcciones IP y necesitan nombres de anfitriones para registrarlos realizan búsquedas inversas. Con una búsqueda inversa, el servidor empieza con la dirección IP remota de la conexión entrante y busca el nombre de anfitrión del que proviene esa conexión.

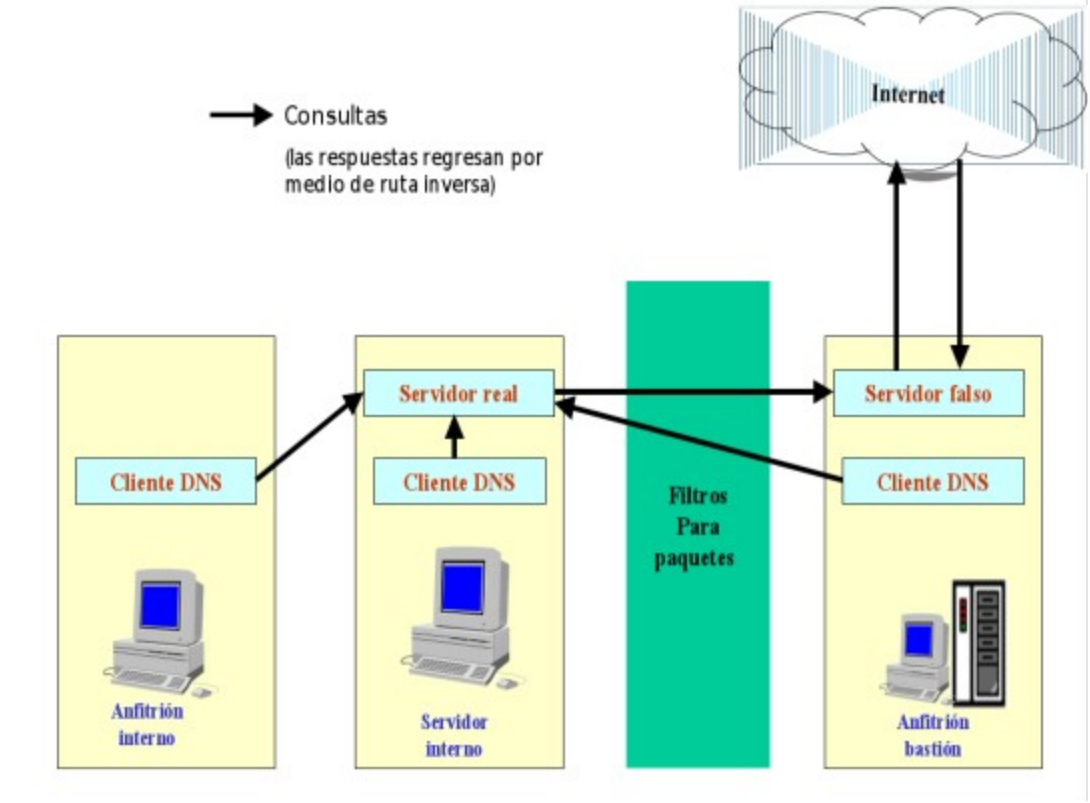
Si todo lo que querían estos servidores con las búsquedas inversas era un nombre para el registro, se podría simplemente crear un registro PTR comodín. Ese registro indicaría que un rango de direcciones pertenece a un anfitrión desconocido en un dominio determinado. Por ejemplo, podría tener una búsqueda para *.19.16.172.IN-ADDR.ARPA que devuelva *desconocido.algúnlado.net)*. Devolver esta información sería más o menos útil; por lo menos le indicaría al administrador del servidor de quien era la máquina (de *algúnlado.net)*. Cualquiera que tuviera un problema con la máquina podría efectuar un seguimiento a través de los contactos publicados para el dominio *algúnlado.net*.

Sin embargo, si sólo se hace esto existe un problema. Muchos servidores (en especial los servidores de *FTP* anónimo) hacen una búsqueda doble inversa, y no se comunican con el cliente a menos que la búsqueda tenga éxito.

Las personas que ejecutan servidores FPT anónimos que hacen búsquedas dobles inversas argumentan que quienes quieren los servicios tienen la responsabilidad de ser miembros de la comunidad de redes, y eso requiere que sean identificables. En realidad es cierto que quienes mantienen alguno de los servidores *FTP* anónimos más grandes y mejor conocidos están del lado de quienes favorecen la búsqueda doble inversa, y no proporcionarán servicio a no ser que ésta tenga éxito.

En la búsqueda doble inversa, un cliente *DNS*:

- Realiza una búsqueda inversa para traducir una dirección IP a un nombre de anfitrión.
- Hace una búsqueda normal en ese nombre de anfitrión para determinar su dirección IP nominal.
- Compara esta dirección IP nominal con la dirección IP original.



Utilización de un cortafuegos para ocultar información DNS

El servidor falso debe proporcionar datos falsos coherentes para todos los anfitriones de ese dominio cuyas direcciones IP serán vistas por el mundo exterior. Para cada dirección IP que le pertenezca, el servidor falso debe publicar un registro PTR con un nombre de anfitrión falso, así como un registro A correspondiente que relacione el nombre de anfitrión falso de nuevo a la dirección IP. Por ejemplo, para la dirección 172.16.1.2, podría publicar un registro PTR con el nombre *host-172-16-1-2.algúnlado.net* y un registro A correspondiente que relacione *host-172-16-1-2.algúnlado.net* de nuevo a la dirección IP correspondiente (172.16.1.2). Cuando se conecte con algún sistema remoto que intenta hacer una búsqueda inversa de su dirección IP (por ejemplo, 172.16.1.2) para determinar su nombre de anfitrión ese sistema obtendrá el nombre de anfitrión falso (por ejemplo, *host-172-16-1-2)*. Si luego el sistema intenta hacer una búsqueda doble inversa para traducir ese nombre de anfitrión a una dirección IP, le devolverá 172.16.1.2, que es igual a la dirección IP original luego cumple con la comprobación de coherencia.

Si se utiliza *proxy* estrictamente para conectar los anfitriones internos con el mundo exterior, no se necesita configurar información falsa para sus anfitriones internos; tan solo debe poner información para el anfitrión o anfitriones que ejecutan el servidor *proxy*. El mundo exterior

verá sólo la dirección de éste. Para una red grande, esto por sí solo puede hacer que valga la pena emplear el servidor *proxy* para *FTP*.

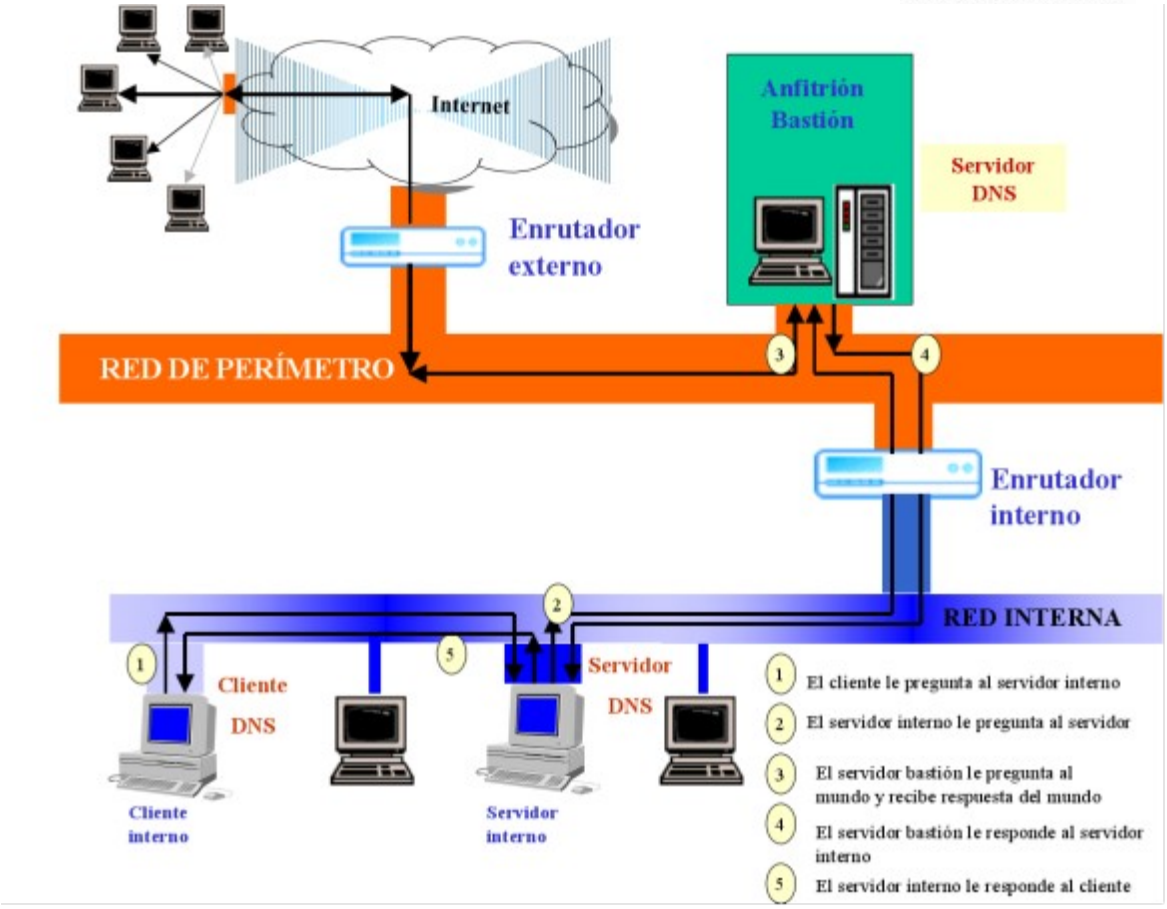
3.5.3.2 *Configurar un servidor DNS verdadero en un sistema interno para que lo utilicen los anfitriones internos*

Las máquinas internas deben usar la información *DNS* real sobre los anfitriones de una organización, no la información falsa presentada al mundo exterior. Esto se hace a través de una instalación estándar de servidor *DNS* en algún sistema interno. Aunque las máquinas internas quizá deseen saber también sobre las máquinas externas, por ejemplo, para traducir el nombre de anfitrión de un sitio *FTP* anónimo remoto a una dirección IP.

Una forma de lograr esto es proporcionar acceso a información *DNS* externa configurando el servidor *DNS* interno para consultar a los servidores *DNS* remotos directamente, conforme sea apropiado, a fin de resolver las solicitudes hechas por los clientes internos sobre anfitriones externos.

Por fortuna, el servidor *DNS* más común (el programa *named* de UNIX) proporciona una solución a este dilema: la instrucción *forwarders*. Esta instrucción le indica al servidor que si él mismo no conoce la información (ya sea de su propia información de zona o de su caché), debe transmitir la solicitud a un servidor específico y dejar que éste encuentre la respuesta, en lugar de intentar ponerse en contacto con servidores por toda Internet en un intento por determinar él mismo la respuesta.

La figura abajo muestra cómo funciona *DNS* con direccionamiento.

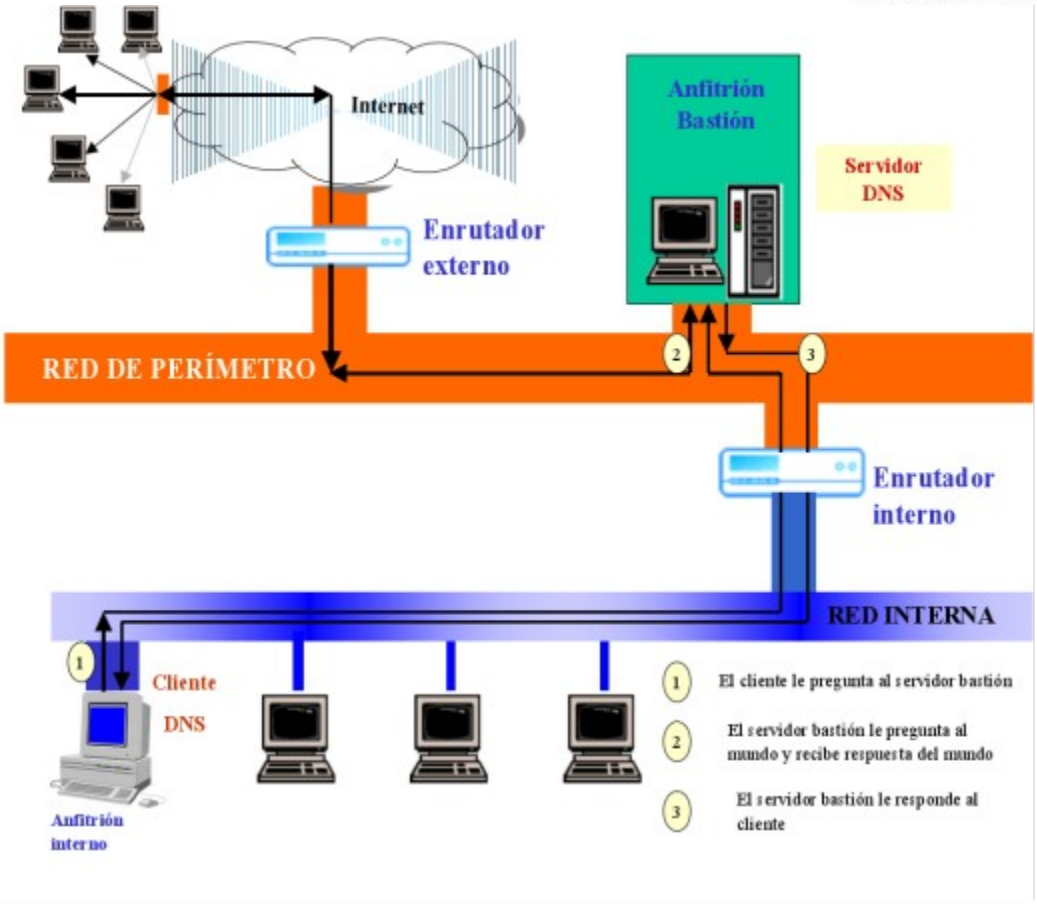


DNS con la utilización de direccionamiento (forwarding)

3.5.3.3 *Los clientes DNS internos consultan al servidor interno*

El siguiente paso es configurar los clientes *DNS* internos para que hagan todas sus indagaciones al servidor interno. En los sistemas UNIX, existen dos casos:

- Cuando el servidor interno recibe una indagación sobre un sistema interno, o sobre un sistema externo que está en su caché, contesta directamente.
- Cuando el servidor interno recibe una indagación sobre un sistema externo que no está en su caché, la envía al servidor del anfitrión bastión (a través de *forwarders*). El servidor del anfitrión bastión obtiene la respuesta de los servidores *DNS* apropiados en Internet y transmite la respuesta al servidor interno, el cual responde al cliente original y conserva la respuesta.



DNS sin forwarding

3.5.3.4 *Los clientes DNS bastión también consultan al servidor interno*

La clave de toda esta configuración para ocultar información es que los clientes *DNS* en el anfitrión bastión deben consultar al servidor interno para obtener información, no al servidor que está en el anfitrión bastión. De esta forma, los clientes *DNS* en el anfitrión bastión (como Sendmail, por ejemplo) pueden utilizar los nombres de anfitrión reales y continuar así para los anfitriones internos, pero los clientes que están en el mundo exterior no pueden tener acceso a los datos internos. De nuevo, existen dos casos:

- Cuando un cliente *DNS* en el anfitrión bastión pregunta sobre un sistema interno, obtiene la respuesta real directamente del servidor interno.
- Cuando un cliente *DNS* en el anfitrión bastión pregunta sobre un sistema externo, el servidor *DNS* interno transmite la solicitud al servidor *DNS* en el anfitrión bastión. Este último servidor obtiene la respuesta de los servidores *DNS* apropiados en Internet, y luego la transmite de nuevo al servidor interno. Este, a su vez, responde al cliente original en el anfitrión bastión.

Los clientes *DNS* en el anfitrión bastión podrían obtener información sobre anfitriones externos de forma más directa preguntándole al servidor *DNS* en el anfitrión bastión en lugar de hacerlo al que está en el anfitrión interno. Sin embargo, si hicieran esto, no podrían obtener la información interna "real", que sólo el servidor en el anfitrión interno tiene. Necesitan esta información, porque se comunican con los anfitriones internos así como con los externos.

3.5.3.5 *Qué debe permitir el sistema de filtrado de paquetes*

Para que funcione este esquema de direccionamiento con cualquier sistema de filtrado de paquetes entre el anfitrión bastión y los sistemas internos, el filtrado debe permitir:

- Solicitudes *DNS* del servidor interno al servidor del anfitrión bastión: los paquetes UDP del puerto 53 en el servidor interno al puerto 53 en el anfitrión bastión (regla A), y los paquetes TCP de los puertos por encima del 1023 en el servidor interno al puerto 53 en el anfitrión bastión (regla B)
- Respuesta a esas solicitudes del anfitrión bastión al servidor interno: los paquetes UDP del puerto 53 en el anfitrión bastión al puerto 53 en el servidor interno (regla C), y los paquetes TCP con el bit ACK encendido del puerto 53 en el anfitrión bastión a los puertos por encima del 1023 en el servidor interno (regla D)
- Solicitudes *DNS* de los clientes *DNS* del anfitrión bastión al servidor interno; los paquetes UDP y TCP de los puertos por encima del 1023 en el anfitrión bastión al puerto 53 en el servidor interno (reglas E y F)
- Respuestas del servidor interno a esos clientes *DNS* del anfitrión bastión: los paquetes UDP y los paquetes TCP con el bit ACK encendido del puerto 53 en el servidor interno a los puertos por encima del 1023 en el anfitrión bastión (G y H)

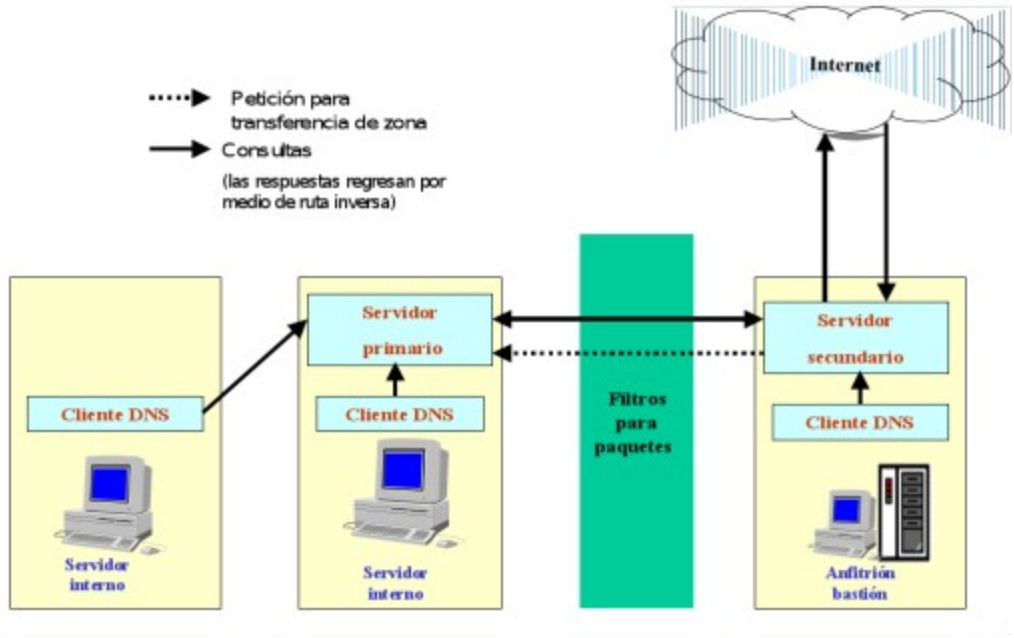
Regla	**Direc-ción**	**Dir. fuente**	**Dir. destino**	**Proto-colo**	**Puerto fuente**	**Puerto destino**	**ACK**	**Acción**
A	Externo	Servidor interno	Anfitrión bastión	UDP	53	53	[a]	Permitir
B	Externo	Servidor interno	Anfitrión bastión	TCP	>1023	53	Cualquiera	Permitir
C	Interno	Anfitrión bastión	Servidor interno	UDP	53	53	[a]	Permitir
D	Interno	Anfitrión bastión	Servidor interno	TCP	53	>1023	Si	Permitir

E	Interno	Anfitrión bastión	Servidor interno	TCP	>1023	53	Cualquiera	Permitir
F	Externo	Servidor interno	Anfitrión bastión	TCP	53	>1023	Si	Permitir

[a] Los paquetes UDP no tienen flag de ACK

3.5.4. CONFIGURACIÓN DEL *DNS* SIN OCULTAR INFORMACIÓN

El enfoque que hemos descrito antes no es la única posibilidad. Supongamos que se considera que no es necesario ocultar los datos del *DNS* interno al mundo exterior. En este caso la configuración *DNS* es similar a la anterior, pero más sencilla. La figura abajo muestra cómo funciona *DNS* sin ocultar información.



DNS sin ocultar información

Con este enfoque todavía debe haber un servidor *DNS* en el anfitrión bastión y un servidor *DNS* interno; sin embargo, uno de ellos puede ser un servidor secundario del otro. Por lo general es más fácil hacer al servidor *DNS* bastión dependiente (secundario) del servidor *DNS* interno y mantener los datos de *DNS* en el servidor interno. El servidor *DNS* interno aún se debe configurar para que transmita solicitudes al servidor *DNS* del anfitrión bastión, pero los clientes *DNS* del anfitrión bastión pueden configurarse para consultar al servidor del anfitrión bastión en lugar de al servidor interno.

El filtrado de paquetes entre el anfitrión bastión y el servidor interno debe permitir:

- Solicitudes *DNS* del servidor *DNS* interno al servidor *DNS* bastión: los paquetes UDP del puerto 53 en el servidor interno al puerto 53 en el anfitrión bastión y los paquetes TCP de los puertos por encima del 1023 en el servidor interno al puerto 53 en el anfitrión bastión.
- Respuestas del servidor *DNS* bastión al servidor *DNS* interno: los paquetes UDP del puerto 53 en el anfitrión bastión al puerto 53 en el servidor interno y los paquetes TCP con el bit ACK encendido del puerto 53 en el anfitrión bastión a los puertos por encima del 1023 en el servidor interno.

Si el anfitrión bastión también es un servidor secundario *DNS* y el anfitrión interno es el servidor *DNS* principal correspondiente, también se debe permitir lo siguiente:

- Solicitudes *DNS* del servidor *DNS* del anfitrión bastión al servidor *DNS* interno: los paquetes UDP del puerto 53 en el anfitrión bastión al puerto 53 en el servidor interno, y los paquetes TCP de los puertos por encima del 1023 en el anfitrión bastión al puerto 53 en el servidor interno
- Respuestas del servidor *DNS* interno de regreso al servidor *DNS* bastión: los paquetes UDP del puerto 53 en el servidor interno al puerto 53 en el anfitrión bastión, y los paquetes TCP con el bit ACK encendido del puerto 53 en el servidor interno a los puertos por encima del 1023 en el anfitrión bastión
- Peticiones para transferencia de zona *DNS* del anfitrión bastión al servidor interno: los paquetes TCP de los puertos por encima del 1023 en el anfitrión bastión al puerto 53 en el servidor interno
- Respuestas de transferencia de zona *DNS* del servidor interno al anfitrión bastión: los puertos TCP con el bit ACK encendido del puerto 53 al servidor interno a los puertos por encima del 1023 en el anfitrión bastión

3.5.5. RESUMEN DE RECOMENDACIONES PARA CONFIGURAR UN *DNS*

- Instalar un servidor *DNS* externo en un anfitrión bastión para que el mundo exterior acceda a él.
- No dar visibilidad al mundo exterior a los registros HINFO; o no utilizarlos o configurar *DNS* para ocultar información.
- Utilizar una implementación del BIND actualizada y búsquedas dobles inversas para evitar engaños
- Considerar la posibilidad de ocultar todos los datos de *DNS* internos y utilizar direccionamiento y registros falsos.

- Desactivar, usando el filtrado de paquetes, las transferencias de zona excepto para los servidores secundarios. Aunque se haya decidido no ocultar la información *DNS*, tal vez no haya una razón válida para que alguien, a no ser los servidores secundarios, realicen una transferencia de zona. No permitir las transferencias de zona se lo pone un poco más difícil a los atacantes.

3.6. CONCLUSIONES

El esquema descrito ofrece muchas ventajas, las desventajas potenciales son el costo y la complejidad y pensamos que estos factores son relativos y asumibles.

Los servicios Telenet y FTP, se deberían evitar y migrar a sus versiones seguras. Además el uso de FTP está en declive, debido al uso intensivo de HTTP para descarga de ficheros, o de otros programas P2P, que están más allá del alcance de este programa.

Si se necesita ahorrar dinero, sería factible construir una arquitectura de subred de protección usando un solo enrutador con tres interfaces. La solución sería un poco más compleja, pues habría que unir los dos grupos de filtrado anteriormente descritos.