SEGURIDAD EN REDES:

FIREWALLS

Grupo : Mirta Migone
Laura Acosta

Profesores : Lic. Javier Diaz
Ing. Luis Marrone
Lic. Miguel Luengo

• INTRODUCCIÓN

El tema de la seguridad no es exclusivo del área de la informática sino que, por el contrario, lo rigen principios aplicables a diferentes ambientes, siendo la informática uno de sus tantos casos particulares. Cuando nos referimos al ambiente informático y nos planteamos los problemas de tener bien conservados los trabajos, asegurar la integridad de la información, y la reserva y privacidad de los datos ante extraños, es que se habla de seguridad de redes y de la implementación de sistemas de seguridad. El objetivo de cualquier sistema de seguridad es minimizar las posibilidades de que se metan intrusos, la rápida detección de los mismos, y - si es necesario - la reconstrucción del estado de los datos al momento de la violación.

Parte I: FUNDAMENTOS TEÓRICOS

PARTE I - FUNDAMENTOS TEÓRICOS

PASOS PARA LLEGAR A UN SISTEMA DE SEGURIDAD EFECTIVO

Para la definición/construcción de un sistema de seguridad efectivo primero corresponde hacer un análisis de riesgos, el cual implica determinar qué es lo que se necesita proteger, de qué se lo quiere proteger, y cómo protegerlo. El punto esencial es listar todas los componentes que se podrían afectar por un problema de seguridad. En la RFC 1244 - o Pfleger (1989) - se sugiere una lista de categorías que se tomó adaptada del libro "Internet y Seguridad en Redes":

(1) Hardware: CPUs, plaquetas, teclados, terminales, estaciones, PCs, impresoras, disk drives, líneas de comunicación, terminal servers, routers.

(2) Software: programas fuente, programas objeto, utilitarios, programas de diagnóstico, sistemas operativos, programas de comunicación.

(3) Datos: durante la ejecución, almacenados on-line, archivados off-line, backups, logs de auditoría, bases de datos, en tránsito sobre los medios de comunicación.

(4) Gente: usuarios, administradores, mantenedores de hardware.

(5) Documentación: sobre programas, hardware, sistemas, procedimientos administrativos locales.

(6) Suministros: papel, formularios, cintas, medios magnéticos.

Luego se enumeran los niveles de riesgo y se los clasifica por grado de severidad. Durante el proceso se toman decisiones evaluando la conveniencia de aplicar seguridad y el costo de la misma.

Cuando, al enumerar los componentes a resguardar, hablamos de:

- Los **datos**, se tienen que observar tres características que necesitan estar protegidas: Disponibilidad (medida de cuán importante es tenerlo disponible todo el tiempo), integridad (medida de cuán importante es que un archivo o los datos del mismo sean consistentes) y confidencialidad (se aplica a los recursos a los cuáles se desea restringir el acceso). La información debería protegerse contra las acciones ilegales tanto a nivel de recursos físicos como a nivel de integridad (es decir, protegerla de modificaciones no autorizadas) y disponibilidad (garantizar que desde el exterior no se pueda evitar el acceso al provocar una saturación de la red con tráfico) de los datos.
- Los **recursos**, se tendría que prevenir el libre uso de los mismos en cualquier momento por parte de los intrusos, cuando argumentan que hay espacio libre y tiempo de máquina en exceso.
- La **reputación**, se debe considerar que una violación (break-in) en el sitio genera falta de confianza en la organización de parte de la gente, y a la vez, puede pasar a ser un lugar en donde se deja piratería de software, pornografía, etc., aun cuando no sea nuestra culpa.

En resumen, antes del desarrollo del mecanismo de seguridad, se debería considerar:

I. Qué recursos se están tratando de proteger.

II. Determinar las medidas necesarias de seguridad específicas del host. Protección por password, encriptado de archivo, firewall, etc.

Determinar de quién debe defenderse a los sistemas de computación.

Determinar la probabilidad de una amenaza.

Implementar medidas para proteger el recurso de red.

III. Considerar el presupuesto de la organización cuando se planea una Seguridad Internet.

IV. Diseñar una Política de Seguridad que describa las incumbencias de la seguridad de red de su organización. Esta política debería tener en cuenta lo siguiente:

Planeamiento de la Seguridad de Red.

Política de Seguridad de los Sitios.

Análisis de Riesgo. Este ítem involucra determinar lo siguiente:

Lo que se necesita proteger.

De qué se lo necesita proteger.

Cómo protegerlo.

Estimativa del riesgo de perder el recurso.

Estimativa de la importancia del recurso.

V. Considerar los siguientes factores para determinar quién garantizará el acceso a los servicios sobre las redes:

¿El acceso a los servicios se garantizará desde un punto central?

¿Qué métodos se usarán para crear cuentas y terminar el acceso?

VI. Diseñar e Implementar Reglas de Filtrado de Paquetes.

VII. Asegurar que el Firewall tiene las siguientes propiedades:

Todo el tráfico del interior al exterior, así como el del exterior al interior debe pasar a través del firewall.

Permitir que sólo el tráfico autorizado según fue definido por la política de seguridad de la organización sea el que pase a través del firewall.

Asegurar que el firewall es inmune a la penetración.

VIII. Educar a los usuarios respecto de la protección de la password:

Educar a los usuarios para que no usen passwords que son fáciles de adivinar.

Asegurar que las longitudes de password son adecuadas.

Ejecutar un adivinador de password.

Requerir el uso de un generador de password.

Siempre usar una mezcla de caracteres en mayúscula y minúscula.

Siempre usar al menos uno o dos caracteres no-alfanuméricos.

Nunca usar palabras del diccionario o escritas al revés.

Nunca usar una porción o variación de un nombre propio, dirección o algo que pueda identificarlo obviamente (al usuario).

1. Tipos de Amenazas

Una vez identificados los recursos que necesitan protección, proceder a detectar las posibles *amenazas* que puedan recibir, principalmente cuando se está conectado a la Internet. Resulta muy útil examinarlas porque de esa manera se pueden determinar las pérdidas potenciales que existen. A continuación se enumeran algunas de las consideradas clásicas:

- **Acceso no autorizado a los recursos y/o a la información (*Intrusión*).**
  Son los ataques más comunes. Dentro de este grupo se encuentra el uso de la cuenta de otro usuario (hijacking) para acceder a la red (por robo o adivinación de la contraseña) y a sus recursos, hasta los ataques de ingeniería social. La gravedad depende del sitio y de la naturaleza de la pérdida potencial. El firewall lo previene, por ejemplo, bloqueando todas las formas de llegar a un sistema sin necesidad de saber un nombre de cuenta o password, reduciendo el número de cuentas accesibles. A veces se lo configura para usar passwords one-time. Además proporciona un lugar controlado como para registrar los intentos de ingreso al sistema que pueden ayudar a detectar trabajos de adivinación. También ayuda a evitar, por ejemplo, los ataques de canal de comando que pueden ocurrir en SMTP porque restringe el número de máquinas disponibles para abrir canales de comando; a la vez, brinda un servidor seguro sobre esas máquinas.
- **o de Información (*Information Theft*).**
  eralmente estos ataques explotan servicios de Internet que se utilizan para obtener información, induciéndolos a que proporcionen más información que para la que están preparados, o para que se la den a la gente equivocada. Muchos servicios Internet están diseñados para usarse sobre redes locales, y no tienen el tipo o grado de seguridad que les permitiría un uso seguro a través de la Internet. Para el robo de información se trata de acceder a las computadoras buscando nombres de usuario y passwords. Este trabajo se hace con los programas llamados sniffers. Conseguir información más específica respecto de un sitio requiere de una dedicación y paciencia extrema, o la certeza de que la información que se quiere pasará a través de un lugar dado en un momento dado.
  Un firewall configurado adecuadamente protegerá de la gente que trata de obtener más información de la que se quiere dar. Pero una vez que se entregó información a través de la Internet, es muy difícil protegerla de los usuarios a los que no se quiere que llegue, y ya no están bajo el alcance del firewall.
- **Denegación de Servicio (*Denial of Service*).**
  En esta categoría se encuentran los que impiden completamente el uso de las computadoras del sitio. Se lo puede producir haciendo "*flooding*" (inundando el sistema o red con mensajes, procesos o pedidos de red al punto que se tiene una pérdida en productividad), hasta llegar a métodos más inteligentes como deshabilitar servicios, re-rutearlos, o reemplazarlos. O la red puede quedar fraccionada por la inhabilitación de un

router, etc. En general no son ataques sino problemas que para resolverlos hay que desconectar temporalmente algún servicio. Los firewalls no se diseñan para tratar con este tipo de problema.

2. Tipos de Atacantes

Los tipos de ataques arriba mencionados son realizados por distintos tipos de atacantes que se pueden agrupar groseramente en las siguientes categorías:

- **joyriders**: Son personas que solamente están buscando divertirse.
- **vándalos:** Son personas que concretamente buscan hacer daño.
- **score keepers**: Son personas que buscan prestigio acumulando cantidad y calidad de los sistemas quebrados.
- **espías (industriales y otros):** Son personas que roban "cosas" que se pueden convertir en dinero en forma inmediata o en el futuro.

3. Estrategias o Principios de Seguridad

Ante la decisión de implementar un mecanismo de seguridad, se recomienda seguir las *estrategias o principios de seguridad* que se enumeran a continuación:

- **Privilegio mínimo**: Se refiere a dar a cada objeto (usuario, programa, sistema) los accesos mínimos para poder ejecutar la tarea asignada, y sólo eso. El ejemplo típico es el host bastión. Aplicar este principio tiene dos problemas: uno es que hay protocolos o aplicaciones (programas) que no lo tienen incluido como característica propia en su diseño, y al querer agregarlo se complica su buen desempeño. El segundo problema puede estar al implementarse algo menor que el mínimo privilegio.
- **Defensa en profundidad**: Cuando se referencia a este principio, se habla de instalar múltiples mecanismos de seguridad que se complementan unos a otros; de esta manera se prevé que al fallar uno, no se comprometa toda la seguridad del lugar. Si se cuenta con una única estrategia elaborada y sofisticada, y se consigue vencerla, ya no hay mecanismos de reserva para proteger el recurso en cuestión. Basándose en esto es que no se pretende que los firewalls sean una solución completa a la amplia variedad de problemas (conocidos y por conocerse) de seguridad de la Internet. Como un ejemplo tenemos la seguridad de red - donde se incluye al firewall -, la seguridad de host y la administración de los sistemas, y la educación de los usuarios.
- **Punto de choque**: La idea de este principio es tener definido un canal de ataque estrecho, que resulta más fácil de monitorear y, así, controlar mejor los posibles ataques. Dentro de la seguridad de red el punto de choque sería el firewall instalado entre la Internet y la red de la organización.

- **Enlace más débil**: Este es un principio fundamental de la seguridad en general. Determina que "la medida de la fortaleza de una cadena lo da el vínculo más débil". Por lo tanto, la idea es minimizar la cantidad de puntos débiles y controlar los que no se pueden eliminar.
- **Situación de "fail-safe" (o de falla segura)**: En esta estrategia de seguridad se establece que si un sistema tiene que fallar, falle de modo tal que se le deniegue el acceso al atacante. Cuando se aplica este principio en el contexto de seguridad de redes se debe elegir cuál es la postura del sitio. Hay dos posturas fundamentales que se pueden tomar con respecto a la política y decisiones de seguridad: *denegar por defecto* y *permitir por defecto*. En la primera se especifica sólo lo que se permite y se prohibe lo demás. En este caso es necesario examinar los servicios solicitados por los usuarios, considerar cómo se los puede proveer en forma segura y, finalmente, permitir sólo a los que se le comprende el funcionamiento luego de haber evaluado su necesidad. En la segunda se especifica sólo lo que se prohibe y se permite todo lo demás. Esta postura no responde al principio de "falla segura" porque se deberían conocer todos los riesgos posibles, cosa que es prácticamente imposible. Además, el responsable del firewall tiene que vigilar y educar constantemente a los usuarios para minimizar los problemas de seguridad.
- **Participación Universal**: Con este principio se requiere tener cooperación de los usuarios del sitio. Se necesita que todos informen cualquier suceso extraño relacionado con la seguridad y que se respete el hecho de que las passwords sean personales y secretas.
- **Diversidad de la defensa**: Establecer esta estrategia implica usar sistemas de seguridad de diferentes proveedores y, a la vez, que la configuración la realicen diferentes grupos de personas. Así se trata de evitar que se repitan los mismos agujeros de seguridad al trabajar diferentes personas con diferentes formas de pensar, y diferentes productos con diferentes posibles fallas implícitas.
- **Simplicidad:** Este principio apunta a que las cosas simples son más fáciles de mantener y a que la complejidad proporciona recovecos en donde se pueden esconder "cosas".

4. Políticas de Seguridad

El nivel de seguridad de la red, la funcionalidad que ofrece, y la facilidad de uso que tiene, la determinan las decisiones que a este respecto toma o deja de tomar el administrador. Pero antes de tomar estas decisiones se deben señalar cuáles son los objetivos de seguridad. Hasta que no se determinen cuáles son, no se puede hacer un uso efectivo de algún grupo de herramientas de seguridad porque simplemente no se sabrá qué chequear y qué restricciones imponer.

Antes de implantar un proyecto de seguridad, se tiene que tener en claro la política a seguir y los riesgos de ésta. Se debe establecer quiénes tendrán acceso a cada parte de la información, las reglas individuales a seguir, definir y difundir a todos los empleados la importancia que tiene la información para la empresa, qué se puede discutir con otros

empleados, si se pueden usar computadoras personales, módem, etc., y establecer las acciones a tomar ante las posibles transgresiones.

En resumen, una política de seguridad es la formalización de las reglas que deben seguir las personas a las que se les dio acceso a cierta tecnología y/o a cierta información.

El propósito principal de una política de seguridad es informar a los usuarios, grupos de trabajo y directivos, de sus requerimientos obligatorios para proteger los recursos y la información que tiene su organización en las redes. La política debería especificar los mecanismos a través de los cuáles se pueden alcanzar estos requerimientos. Otro propósito es proveer una línea de base a partir de la cual adquirir, configurar y auditar sistemas de computación y redes para concordar con la política. Por eso no tiene sentido el intento de usar un conjunto de herramientas de seguridad en la ausencia de al menos una política de seguridad implícita.

Cuando la implementación de una política de seguridad de red impide que los usuarios cumplan con sus tareas en forma efectiva, se puede llegar a tener consecuencias no convenientes; por ejemplo, encontrarse las formas de ignorarla, convirtiéndola en algo inútil. Por eso una buena política (o política efectiva) es aquella a la que todos los usuarios y administradores de la red pueden aceptar, y están dispuestos a reforzar.

Llegado el momento de determinar los objetivos, en mayor medida estarán influenciados por las siguientes opciones:

(1) Servicios ofrecidos versus seguridad provista.

Cada servicio ofrecido a los usuarios trae sus propios riesgos de seguridad (implícitos).

Para algunos servicios el riesgo supera el beneficio del servicio y el administrador puede elegir eliminar el servicio antes que tratar de brindarle en forma segura.

(2) Facilidad de uso versus seguridad.

El sistema más fácil de usar es el que no requiere passwords de usuario, pero en este caso no hay seguridad. Al solicitarse passwords, se hace más seguro al sistema aunque se agrega un poco de complejidad. Un nivel más complejo dando una seguridad superior es el de las passwords únicas generadas por el sistema.

(3) Costo de seguridad versus riesgo de pérdida.

Hay muchos costos diferentes de seguridad: monetarios (es decir, el costo de comprar hardware y software de seguridad como firewalls y generadores de passwords únicas - "one-time"), performance (porque encriptar y desencriptar lleva tiempo), y facilidad de uso (ya mencionado anteriormente). También hay muchos niveles de riesgo, dados por los diferentes tipos de ataques de los que se puede ser víctima. Cada tipo de costo debe pesarse ("estimarse") contra cada tipo de pérdida.

Una buena política de seguridad tiene las siguientes características:

(1) Debe ser implementable a través de procedimientos de administración del sistema, publicación de indicaciones de uso aceptables, u otros métodos apropiados.

(2) Debe ser reforzable con herramientas de seguridad en donde sea apropiado, y con sanciones donde la prevención real no es técnicamente posible.
(3) Debe definir claramente las áreas de seguridad para los usuarios, administradores, y gerencia.

Los componentes de una buena política de seguridad incluyen:

(1) Una Guía de Compras de Tecnologías de Computación que especifique las características de seguridad requeridas, o preferidas. Estas deberían suplir/suplementar los principios y políticas adquiridas existentes.
(2) Una Política Privada que defina las expectativas razonables de privacidad observando temas tales como monitoreo del correo electrónico, logging de teclado, y acceso a los archivos de usuario.
(3) Una Política de Acceso que define los privilegios y derechos de acceso para proteger las posesiones de la pérdida o divulgación mediante la especificación de guías de uso aceptables para los usuarios, equipo de operaciones, y gerencia. Debería proveer indicaciones para los vínculos externos, comunicaciones de datos, conexión de dispositivos a una red, y agregado de nuevo software a los sistemas. También debería especificar cualquiera de los mensajes de notificación requeridos (p.ej., los mensajes de conexión deberían proveer advertencias acerca del uso autorizado y monitoreo en línea, y no decir simplemente "Hola").
(4) Una Política de Rendición de Cuentas que defina las responsabilidades del usuario, equipo de operaciones, y gerencia. Debería especificar una capacidad de auditoría, y proveer pautas de manejo de incidentes (es decir, qué hacer y a quién contactar si se detecta una posible intrusión).
(5) Una Política de Autenticación que establece confianza a través de una política de passwords efectiva, y determinando pautas para autenticación en lugares remotos y el uso de dispositivos de autenticación (p.ej., one-time passwords y los dispositivos que las generan).
(6) Una sentencia de Disponibilidad que determina las expectativas de los usuarios para la disponibilidad de los recursos. Debería remarcar los temas de redundancia y recuperación, así como especificar las horas operativas y los períodos de mantenimiento time-down. También debería incluir contactar información para sistema de reporte y fallas de red.
(7) Un Sistema de Tecnología de Información y Política de Manejo de Red que describa cómo la gente de mantenimiento interna y externa está autorizada para manejar y acceder a la tecnología. Un tema importante para indicar aquí es si se permite el mantenimiento remoto y cómo se controla tal acceso. Otra área de consideración es el outsourcing y cómo se lo maneja.
(8) Una Política de Informe de Violaciones que indique qué tipo de violaciones (p.ej., privacidad y seguridad, interna y externa) deben informarse y a quién se le debe hacer el mismo.
(9) Información de Soporte que suministra a los usuarios, equipo, y gerencia el contacto con la

información para cada tipo de violación de política; pautas sobre como manejar consultas del exterior acerca de un incidente de seguridad, o información que puede considerarse confidencial o propietaria; y referencias cruzadas para los procedimientos de seguridad e información relacionada, tales como políticas de la compañía y leyes y regulaciones gubernamentales.

Hay dos filosofías subyacentes diametralmente opuestas que pueden adoptarse cuando se define un plan de seguridad. Ambas alternativas son modelos legítimos de adoptar, y la elección entre ellas dependerá del sitio y sus necesidades de seguridad.

La primera opción es desconectar todos los servicios y entonces seleccionar los que se van a habilitar, analizando caso por caso a medida que sean necesarios. Esto puede hacerse a nivel de host o de red según sea apropiado. Este modelo de "**Denegar todo**", generalmente es más seguro que el dado a continuación. Su implementación correcta lleva más trabajo a la vez que un mejor conocimiento de los servicios. Permitir sólo los servicios conocidos implica un mejor análisis de un servicio/protocolo en particular y el diseño de un mecanismo de seguridad adecuado al nivel de seguridad del sitio.

El otro modelo, que llamamos modelo "**Permitir todo**", es más fácil de implementar, pero generalmente es menos seguro que el anterior. Simplemente habilita todos los servicios, generalmente la opción por defecto a nivel de host, y permite que todos los protocolos viajen a través de los límites de la red, usualmente el default a nivel del router. A medida que se hacen evidentes los agujeros de seguridad, se los restringe o emparcha o bien a nivel de host o bien a nivel de red.

Cada uno de estos modelos se puede aplicar a diferentes porciones del sitio, dependiendo de los requerimientos de funcionalidad, control administrativo, política del sitio, etc. Por ejemplo, la política puede ser usar el modelo "permitir todo" cuando se instalan estaciones para uso general, pero adoptar el modelo "denegar todo" cuando se instalan servidores de información, por ejemplo, uno de e-mail. Igualmente, una política "permitir todo" puede adoptarse para tráfico entre LANs internas al sitio, pero una política "denegar todo" puede adoptarse entre el sitio (la organización) y la Internet.

Se recomienda tener cuidado cuando se mezclan filosofías como en los ejemplos anteriores. En muchas organizaciones se elige la opción de una zona externa resistente y una zona media más "blanda".

La creación de una política de seguridad - a nivel computación - pretende garantizar que los esfuerzos gastados en seguridad sean menores que los de recuperación ante una amenaza.

Una vez establecida la política de seguridad, se la debería comunicar claramente a los usuarios, equipo, y gerencia. Una parte importante del proceso es tener la firma de todo el personal indicando que han leído, entendido y acordado respetar la política. Finalmente, es

recomendable revisarla cada cierto tiempo para ver si está soportando exitosamente sus necesidades de seguridad.

Se debe tener en cuenta que un gran porcentaje del éxito de un esquema de seguridad depende del mantenimiento y la difusión del mismo.

Las políticas que se adopten deben ser definidas de forma sencilla y clara puesto que, al querer hacerlas complicadas, después resulten imposibles de mantener.

Se debe monitorear periódicamente tratando de encontrar algún intento de violación de la seguridad. Este monitoreo se debe hacer en distintos horarios, para despistar al posible intruso. Si se produce alguna falla, se la tiene que estudiar en el momento para saber qué es lo que esta sucediendo y definir qué acción tomar ante posibles intentos de violación: si recuperar la información de un backup, si es un tipo de ataque nuevo que no se había tenido en cuenta, o si proviene de la red interna y tratar entonces de descubrir quién fue.

Es conveniente suscribirse a listas o foros que traten los temas de seguridad y estar en contacto con los proveedores de los productos para recibir las actualizaciones de los mismos.

SEGURIDAD CON FIREWALLS

Los problemas de seguridad se originan desde el momento en que una red de redes (por ejemplo, la Internet) puede conectar distintas organizaciones, y algunas de éstas resultan no ser confiables. Es muy importante definir la seguridad en un ambiente de redes, pero a la vez muy difícil debido a que se tienen que conocer la arquitectura, los detalles de hardware y los protocolos de red involucrados.

El tema de la seguridad en redes es muy amplio porque puede enfocarse desde distintos aspectos. Por ejemplo, si quisiéramos proteger los datos cuando viajan a través de la red, se podrían contar con diferentes mecanismos de encriptación y/o encapsulamiento. Pero si quisiéramos controlar los accesos de los usuarios a ciertas aplicaciones, servicios o recursos, deberíamos utilizar alguno de los diferentes mecanismos de autenticación que podrían ir desde el típico Username/password hasta métodos más sofisticados como el Kerberos. Y si buscáramos controlar los accesos desde una red externa hacia nuestra red interna de manera que ésta última sea segura, se podría necesitar la implementación de un firewall.

Por eso, cuando se trata el tema de los firewalls, se está considerando a un mecanismo de seguridad que trabaja controlando los accesos a nivel de host y de red.

1. ¿Qué es un Firewall?

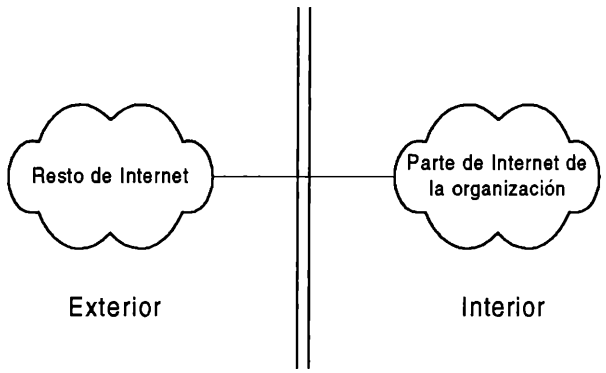
Un Firewall es un mecanismo o grupo de mecanismos usados para controlar y observar los accesos entre dos redes con el objetivo de protegerla.

Tiene múltiples propósitos:

- ◊ Restringir la entrada de los usuarios por lugares cuidadosamente controlados.
- ◊ Prevenir los ataques desde el exterior.
- ◊ Restringir los permisos de los usuarios.
- ◊ Forzar a que todo el tráfico de adentro hacia afuera, así como el de afuera hacia adentro de la red deba pasar a través del firewall.
- ◊ Permitir que pase por el firewall sólo el tráfico autorizado según se definió en la política de seguridad de la organización.

La forma de implementarlo puede variar, pero en principio se lo puede ver como un par de mecanismos: uno para bloquear el tráfico y otro para permitirlo. Algunos ponen un mayor énfasis sobre el bloqueo del tráfico, mientras que otros enfatizan permitir el tráfico. Frecuentemente se lo instala en el punto donde la red interna se conecta con la Internet (o red externa). Todo tráfico externo (o desde Internet) hacia la red interna pasa a través del firewall; así se puede determinar si dicho trafico es aceptable de acuerdo a las políticas de seguridad que se hayan establecido.

Lógicamente, un firewall es un separador, un analizador, un limitador. La implementación física varía de acuerdo al sitio. A menudo, es un conjunto de componentes de hardware: un router, un host, o una combinación de routers, computadoras y redes con el software apropiado. Rara vez es un solo objeto físico aunque los productos comerciales más nuevos tratan de poner todo en una misma "caja". Usualmente está compuesto por múltiples partes y alguna de esas partes puede realizar además otras tareas diferentes a las del firewall en sí. La conexión de Internet casi siempre es parte del firewall.



Además, tiene otros usos. Por ejemplo, se puede usar para separar LANs internas a fin de agrupar las partes de un sitio por funcionalidad operacional o porque tienen distintas necesidades de seguridad. Sin embargo, no debería ser la única solución disponible en el modelo de seguridad porque provee seguridad de perímetro y no apunta a los temas de seguridad en sistemas individuales.

2. ¿Qué puede hacer un Firewall?

Un firewall es un *punto de decisiones de seguridad*. Todo el tráfico de entrada y salida pasa a través de este único punto de estrangulación ("choke point") donde pueden imponerse varios sistemas de seguridad y auditoría; además es el lugar donde se conecta la red con Internet.

Enfocar la seguridad de esta manera es más eficiente que extender por todos lados las decisiones de seguridad y tecnología. Aunque es costoso, muchos sitios encontraron que concentrar la seguridad de hardware y software en un firewall es menos caro y más efectivo que otras medidas de seguridad.

En general, la instalación de un firewall lleva a poder *exigir políticas de seguridad* y además, a imponerlas. Como algunos servicios que se utilizan en Internet son inseguros, mediante su instalación y siguiendo la política de seguridad de la red, se pueden permitir sólo algunos al establecer sus reglas de configuración. Se lo puede invocar para ayudar a reforzar políticas más complicadas dependiendo de las tecnologías que se elijan para implementarlo. Por ejemplo, ciertos sistemas con firewall sólo permiten transferir archivos de y hacia Internet, pero usando otro mecanismo de control, se puede determinar qué usuarios tienen esa capacidad.

Un firewall puede *registrar de manera eficiente la actividad de Internet* porque, como todo el tráfico pasa a través de éste, resulta ser un buen lugar para recolectar información sobre el uso del sistema y de la red, y así registrar qué ocurrió entre la red protegida y la red externa.

Además, se lo puede usar para *proteger secciones* dentro de una misma red interna. Un problema en una sección puede extenderse a toda la red, por lo tanto la existencia del firewall limita los daños que un problema de seguridad en una parte de la red podría causar a toda la red.

También provee una importante *función de logging y auditoría*; generalmente ofrece resúmenes al administrador acerca de qué tipo y cantidad de tráfico pasó a través de él, cuántos intentos hubo de quebrarlo, etc.

3. ¿Qué no puede hacer un Firewall?

Sí se puede decir que ofrece una excelente protección ante posibles amenazas externas, pero no es una solución completamente segura. Un firewall es vulnerable frente a los atentados que pudiera hacer la gente que está en la red interna, ya que ciertas amenazas están fuera de su control. Por ejemplo, no puede proteger contra las *acciones ilegales de los usuarios internos* tales como las copias de datos en un disco, tape, papel, etc., que después podrían mandarse fuera de la organización. Si el ataque es realmente dentro del firewall,

virtualmente puede notarlo. Los usuarios internos pueden robar datos, dañar hardware y software, y modificar programas cerca del firewall. Las amenazas internas requieren una seguridad interna, tal como un host de seguridad y educación a los usuarios.

Un firewall no puede proteger a *las conexiones que no pasan por él.* Por ejemplo, no tiene forma de prevenir intrusos que se comunican a través de un módem.

Tampoco puede enfrentar completamente las *nuevas amenazas* ya que está diseñado para proteger contra las amenazas conocidas. Por lo tanto, no puede defenderse automáticamente cuando sucede un ataque nuevo. Periódicamente la gente descubre nuevas formas de ataque, por lo que no se puede setear un firewall y creer que se está protegido para siempre.

Tampoco puede protegerse de los *virus* ni puede mantener los virus de una PC fuera de la red. Hay algunos que escanean todo el tráfico que pasa a través de él a la red interna; el escaneo es al menos de la dirección origen y destino. Con filtrados de paquetes o con proxys sofisticados, la protección de virus en firewalls no es algo muy práctico. Existen simplemente algunos tipos de virus y algunas formas de virus que pueden ocultarse en los datos. Detectar virus en un paquete de datos al azar pasando a través del firewall es muy dificultoso porque se requiere:

◊ Reconocer que el paquete es parte de un programa.
◊ Determinar cómo debería verse el programa.
◊ Determinar que el cambio es debido a un virus.

La forma más práctica de atacar el problema de los virus es a través de la protección del software y de la educación de los usuarios.

4. Tipos básicos de firewalls

Conceptualmente, hay dos tipos básicos de firewalls:

- A Nivel de Red
- A Nivel de Aplicación

No son tan diferentes como se podría pensar, y las últimas tecnologías están desdibujando la distinción, al punto donde ya no es claro si una es "mejor" o "peor" que la otra.

Los firewalls *a nivel de red* generalmente toman sus decisiones basadas en las direcciones de origen, destino y ports en los paquetes IP individuales. Un router simple es el firewall a nivel de red "tradicional", ya que no es capaz de tomar decisiones particularmente sofisticadas acerca de si un paquete está hablándole realmente o de dónde vino en realidad. Los firewall a nivel de red modernos se han vuelto cada vez más sofisticados, manteniendo la información interna acerca del estado de las conexiones que pasan a través de ellos, los contenidos de algunas de las corrientes de datos, etc. Una diferencia importante respecto a los firewalls a nivel de red es que rutean el tráfico directamente a través de ellos, de modo que

para usar uno usualmente se necesita tener asignado un bloque de direcciones IP válidas. Los firewalls a nivel de red tienden a ser bastante rápidos y muy transparentes a los usuarios.

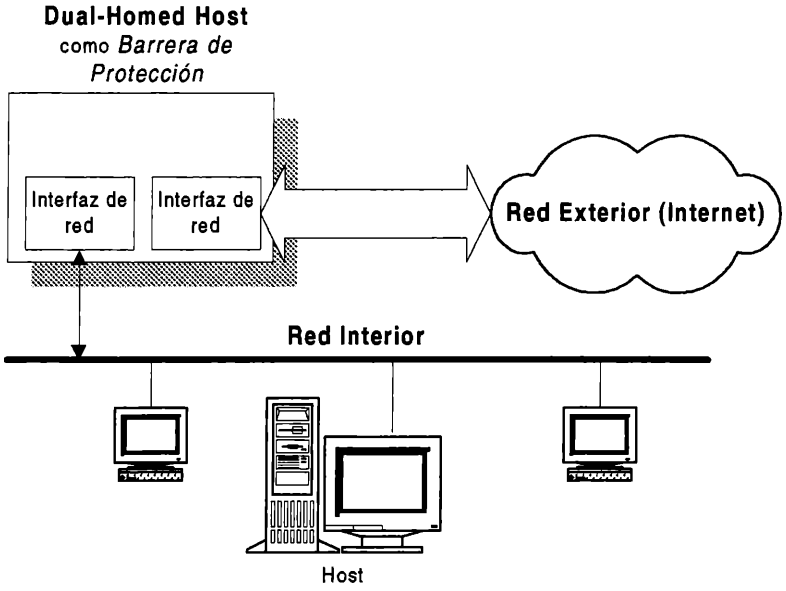
Los firewalls *a nivel de aplicación* generalmente son hosts corriendo servidores proxy - los que no permiten tráfico directo entre las redes - que realizan un logging y una auditoría elaborada del tráfico que pasa a través de los mismos. Como las aplicaciones proxy son componentes de software corriendo sobre el firewall, es un buen lugar para hacer logging y control de acceso. A estos tipos de firewall se los puede usar como traductores de dirección de red, ya que el tráfico va de un "lado" y "fuera" del otro, después de haber pasado a través de una aplicación que enmascara efectivamente el origen de la conexión iniciada. El hecho de tener una aplicación en el camino en algunos casos puede impactar sobre la performance y hacer al firewall menos transparente. Los primeros firewalls a nivel de aplicación, tales como los construidos usando el toolkit de firewall TIS, no son particularmente transparentes a los usuarios finales y puede requerirse algún entrenamiento. En cambio los modernos frecuentemente son completamente transparentes y tienden a proveer informes de auditoría más detallados y a reforzar modelos de seguridad más conservadores que los firewalls a nivel de red.

Cada vez más se busca combinar los dos modelos básicos anteriores y agregar la opción de encriptado, de modo de poder proteger el tráfico que pasa entre ellos al estar sobre la Internet. Los firewalls con encriptado "end-to-end" pueden usarse por las organizaciones con múltiples puntos de conectividad en Internet, para así usar a la Internet como un "backbone privado" sin preocuparse de que las passwords o datos sean "sniffeados".

ARQUITECTURAS DE FIREWALLS

1. Host Dual-Homed o Gateway Dual-Homed

Esta arquitectura está constituida por un host dual-homed, que es una computadora con al menos dos interfaces de red. Tal host actúa como router entre las redes a las que está conectado. Tiene la capacidad de rutear paquetes IP desde una red a la otra, pero este ruteado no se hace directamente. El sistema interno al firewall puede comunicarse con el dual-homed host, y los sistemas fuera del firewall también pueden hacerlo, pero tal comunicación entre sistemas no puede darse directamente entre ellos.

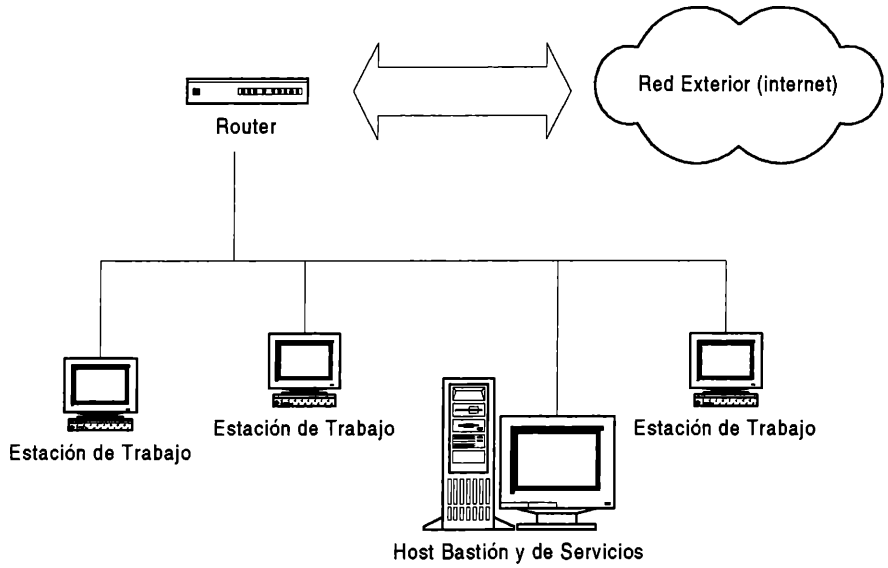


Las aplicaciones pueden compartir datos intercambiando información vía los datos compartidos. El host dual-homed puede proveer varios niveles de control.

2. Screened Host

Esta arquitectura provee servicios desde un host que está conectado a la red interna, usando un router separado. La principal seguridad está dada por el filtrado de paquetes. En la red interna se tiene ubicado al host bastión. El filtrado de paquetes sobre el router apantallado se configura de manera tal que el host bastión es la única maquina de la red interna sobre la que pueden abrir conexiones los host de Internet. A pesar de esto, sólo se permiten cierto tipo de conexiones. Cualquier sistema externo que intente acceder a sistemas o servicios internos tendrá que conectarse con este host. De ahí el alto nivel de seguridad de host que necesita tener el host bastión.

Por otro lado, el filtrado de paquetes también permite que el host bastión abra conexiones permitidas al mundo exterior, según se hayan establecido en la política de seguridad del sitio.



La configuración del filtrado de paquetes en el router apantallado puede ser una de las siguientes:

- Permitir a los host internos que se conecten con hosts de Internet para ciertos servicios (vía filtrado de paquetes)
- Deshabilitar todas las conexiones desde los hosts internos (forzando que usen servicios proxy vía el host bastión)

Estas aproximaciones se pueden mezclar y hacer que coincidan para diferentes servicios; algunos pueden permitirse directamente vía filtrado de paquetes, mientras que otros pueden darse indirectamente vía proxy. Todo depende de la política particular que se quiera reforzar.

Esta arquitectura permite que los paquetes se muevan desde la Internet a la red interna. Al ser más fácil defender a un router que a un host, ya que el primero provee un conjunto muy limitado de servicios, se puede decir que brinda mejor seguridad y mejor uso que la arquitectura dual-homed. Presenta algunas desventajas al compararla con la arquitectura de Screened Subnet: la principal es que si un atacante quiebra al host bastión, no hay nada más en el camino que pueda detener este ataque. El router también presenta un punto de falla. Si el router esta comprometido, la red entera esta disponible para ser atacada.

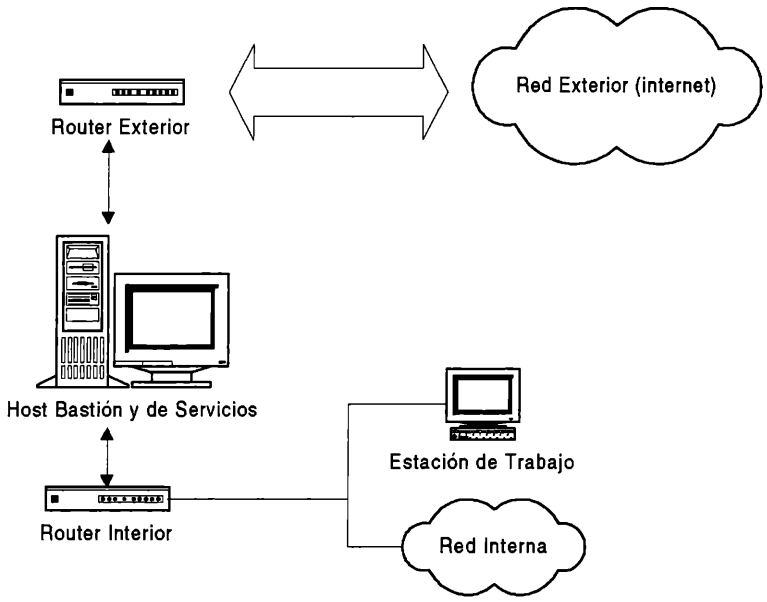
3. Screened Subnet

Es la arquitectura más popular. Agrega un nivel extra de seguridad al de la arquitectura anterior, poniendo una red perimetral (o perímetro) que aísla a la red interna de la Internet.

Los hosts bastiones son las máquinas más vulnerables en la red. Aunque uno se

esfuerce en protegerlas, éstas son las máquinas más buscadas para ser atacadas porque son las únicas vistas desde la red externa. Al aislar al host bastión en un perímetro, se puede reducir el impacto del ataque que se pueda recibir.

Esta arquitectura tiene dos routers apantallados, cada uno conectado al perímetro. Uno esta ubicado entre el perímetro y la red interna y el otro entre el perímetro y la red externa. Para destruir la red interna con este tipo de arquitectura, el atacante debe pasar por el primer router, destruir al host bastión, y lograr pasar por el router interno. Algunos sitios crearon una serie de capas de redes perimetrales entre el mundo externo y la red. Los servicios más peligrosos se pasan de los perímetros más externos a los más internos. La idea es que un ataque a una máquina en el perímetro más externo tendrá una tarea ardua para atacar a las máquinas internas porque deberá atacar a todos los niveles de seguridad de todos los demás perímetros.



Red Perimetral: Es un nivel de seguridad dado por una red adicional entre la red externa y la interna protegida. Si un ataque logra romper el nivel más externo de seguridad (desde la Internet), el perímetro ofrece un nivel adicional de protección entre la red interna y el atacante. La red perimetral está separada de la red interna por un router. El tráfico interno permanece sobre la red interna y no es visible sobre la red perimetral. Los paquetes que corren sobre la red perimetral debería ser solamente o bien desde o hacia el host bastión o bien desde o hacia la Internet.

Host Bastión: Situado dentro del perímetro - puede ser uno o más - es el principal punto de contacto para las conexiones entrantes desde el mundo exterior. Los servicios externos (de clientes internos a servidores en Internet) se manejan de alguna de

estas maneras:

- Configurando filtrado de paquetes sobre ambos routers para permitir que los clientes internos accedan directamente a los servidores externos.
- Configurar los servidores proxy para que corran sobre el Host Bastión de modo de permitir que los clientes internos accedan indirectamente a los servidores externos.

Router Exterior: Situado entre el mundo externo y el perímetro. Realiza un filtrado de paquetes. Las reglas de filtrado de paquetes deben ser esencialmente las mismas en ambos routers. Estas reglas son las que protegen las máquinas del perímetro (el host bastión y el router interno); por lo tanto no es necesario una protección muy fuerte, porque los hosts del perímetro son hosts de seguridad.

Router Interior: Ubicado entre la red interna y el perímetro. Este router no realiza el principal filtrado de paquetes. Permite seleccionar servicios de la red interna. Los servicios que este router permite entre el host bastión y la red interna no necesariamente son los mismos que permite el router externo. La razón para limitar los servicios entre el host bastión y la red interna es que reduce el número de máquinas que pueden ser atacadas desde el host bastión.

Se debe configurar el filtrado de paquetes en ambos routers para permitir que los clientes internos accedan directamente a los servidores externos.

PARTES COMPONENTES DE UN FIREWALL

1. Sistema de Filtrado de Paquetes

El sistema de filtrado de paquetes especifica cómo deberá manejar el router a cada paquete. Cada paquete que pasa entre hosts internos y externos es filtrado de manera selectiva. Su uso permite bloquear cierto tipo de paquetes de acuerdo con la política de seguridad de la red. El tipo de ruteo usado para filtrar paquetes en un Firewall se conoce como ruteado apantallado.

Para entender como se filtra un paquete se necesita conocer la diferencia entre ruteo ordinario y ruteo apantallado.

Un *ruteo ordinario* simplemente mira la dirección destino de cada paquete y elige el mejor camino que conoce hacia ahí. La decisión sobre cómo atender el paquete se basa solamente en el destino del paquete.

Un ruteo apantallado mira el paquete mas detalladamente. Además de determinarse si el paquete puede ser ruteado o no a la dirección destino, se decide si va a ser ruteado o no de

acuerdo a lo que indica la política de seguridad. La comunicación entre las redes interna y externa tiene la responsabilidad de realizar el ruteo; la decisión de rutear o no un paquete es la única protección del sistema. Si la seguridad falla, la red interna estará expuesta. Un router apantallado puede permitir o denegar un servicio, pero no puede proteger operaciones individuales dentro del servicio.

Casi todos los dispositivos actuales para filtrar paquetes (routers de selección o compuertas de filtro de paquetes) operan de la siguiente forma:

El criterio de filtro de paquete debe establecer qué paquetes admitir desde un lugar. A este criterio se lo conoce como "reglas de filtro de paquetes". Cuando un paquete llega al port correspondiente, se analizan los encabezados de los paquetes. La mayoría de los filtros analizan sólo los encabezados IP, TCP o UDP. Las reglas del filtro de paquete se almacenan en un orden específico porque ese va a ser el orden en que se aplican al paquete. Si una regla bloquea la transmisión o recepción del paquete, el paquete no se acepta. Si una regla permite la transmisión o recepción de un paquete, el paquete sigue su camino. Si un paquete no iguala a alguna de las reglas, se rechaza. Es decir, aquello que no esté expresamente permitido, se prohibe. Por eso es importante colocar las reglas en orden correcto. Un error común al configurar las reglas del filtro de paquetes es hacerlo en desorden. Si se las coloca en un orden equivocado, se podrían terminar denegando servicios válidos y permitiendo servicios que se deseaban bloquear. En resumen, se restringen los paquetes entrantes excepto para los ports que corresponden a servicios que la organización pone a disposición del exterior.

El filtrado de paquetes se utiliza para restringir el tráfico de los servicios que se desean denegar y para impedir el tránsito de paquetes dirigidos a otros destinos (siguiendo la política de que *lo que no esta expresamente permitido se prohibe*).

Diseño de filtrado de paquetes

Como dijimos anteriormente, la política de seguridad deberá ser traducida a las reglas de filtrado de paquetes. Estas reglas pueden ser codificadas en una tabla de reglas con el siguiente formato:

Nro. de Regla de Filtro	Acción	Origen	Port Origen	Destino	Port Destino	Opción de flag de Protocolo	Descripción
1	Aceptar	port de Correo	25	*	*	ACK	Permite la conexión con nuestro correo

El asterisco (*) se usa para igualar cualquier valor de esa columna.

En la tabla se muestra un ejemplo que se interpreta como "Permitir la conexión desde cualquier(*) host externo originada en cualquiera(*) de sus ports al port 25 de nuestro host".

En general, los routers de selección pueden filtrar cualquiera de los valores del campo

en los encabezados del protocolo TCP o IP. Para implementar la mayoría de las políticas de seguridad en la red mediante un router de selección se necesitarán especificar sólo los flags TCP, opciones IP y valores de dirección del destinatario y el origen.

Implementación de las reglas de los filtros de paquetes

Cada tipo de dispositivo para filtrado de paquete tiene su propia sintaxis para cómo programar las reglas. El objetivo es mostrar un ejemplo práctico acerca de cómo especificar las reglas de filtrado de paquetes con relación a los routers de selección del distribuidor Cisco.

Qué es una lista de acceso:

Cisco define las listas de acceso como una selección secuencial de condiciones de permiso y negación que se aplica a las direcciones Internet. Estas condiciones sirven para establecer las reglas de filtrado de paquete. Cada paquete se probará contra las condiciones que se encuentran en estas listas, una por una. Cuando el paquete se ajuste a alguna, la regla correspondiente determinará si éste se acepta o rechaza, deteniéndose entonces la prueba de las condiciones. De ahí la importancia del orden de las condiciones. Si no hay condiciones que se correspondan con los paquetes, éstos serán rechazados.

Los routers de Cisco tienen dos tipos de listas de acceso:

- Listas de acceso estándares
- Listas de acceso extendidas

Las listas de acceso *estándares* tienen una sola dirección para las operaciones de correspondencia, con la siguiente sintaxis:

access-list lista (permit / deny) dirección-mascara-de-comodín no-access-list lista

donde :

lista - es un entero con valor en el rango 1 hasta 99 que se utiliza para identificar una o más condiciones de permiso y/o negación. Este valor seria el número de regla.

permit/deny - corresponde a aceptar/rechazar la correspondiente regla de filtrado de paquetes.

dirección y máscara de comodín - los valores son de 32 bits y se escriben por medio de la notación decimal punteada. En la máscara de comodín los bits de dirección correspondientes a 1 se ignoran en la comparación, mientras que los bits de dirección correspondientes a 0 se utilizan en la comparación. Si no se especifica el valor de la máscara de comodín se supone que este valor será 0.0.0.0.

Ejemplo : access-list 1 permit 199.245.180.0 0.0.0.255

El comando "no access-list lista" se utiliza para eliminar toda la lista de acceso. Esto es útil si no se sabe que ya exista una definición de una lista de acceso para ese número y se desea iniciar con un registro en blanco.

Las listas *extendidas* tienen dos direcciones con información opcional del tipo de protocolo para las operaciones de correspondencia; permiten filtrar el tráfico por interfase basándose en las direcciones IP origen y destino, además de la información del protocolo. La sintaxis es la siguiente:

access-list lista (permit/deny) protocolo origen destino masc.-origen masc.-destino [operador operando]

donde

lista - es un valor entero entre 100 a 199 que se utiliza para identificar una o más condiciones de permiso y o negación.

permit/deny - corresponde a aceptar/rechazar una de las reglas de filtrado de paquetes. El resto de la lista extendida no se procesará después de que se produzca una correspondencia.

protocolo - puede contener cualquiera de los siguiente valores: IP, TCP, UDP e ICMP que corresponden a los protocolos que se pueden analizar.

origen y *máscara de origen* - son de 32 bits y se escriben por medio de la notación decimal punteada. Estos valores se utilizan para identificar la dirección IP origen. En la máscara de origen los bits de dirección correspondientes a 1 se ignoran, mientras que los bits en 0 sí se utilizan en la comparación.

destino y *máscara de destino* - estos valores sirven para igualar la dirección IP destinataria; se los puede escribir por medio de la notación decimal punteada. En la mascara de destino los bits de dirección correspondientes a 1 se ignoran, mientras que los bits en 0 sí se utilizan en la comparación.

operador y *operando* - con estos valores se comparan los números de port, puntos de acceso de servicio o nombres de contacto. Estos valores son significativos para los protocolos TCP y UDP. Para éstos valores clave, el operador puede tener cualquiera de los siguientes valores: lt (menor que), eq (igual que), gt (mayor que), neq (diferente de). El operando es el valor decimal del port destinatario del protocolo especificado.

Ejemplo
no access-list 101
access-list 101 deny tcp 132.124.23.55 0.0.0.0 199.245.180.0 0.0.0.255 eq 25

El primer comando elimina cualquier lista de acceso extendida 101 anterior. El segundo comando permite que envíe un paquete TCP desde el anfitrión 132.124.23.55 a la red 199.245.180.0 con port destinatario en 25 (SMTP).

El comando del grupo de acceso sirve para aplicar las definiciones de la lista de acceso en la interfase. La sintaxis es la siguiente:

ip access-group lista

donde :

lista - número entre 1 y 199 que especifica la lista de acceso que se aplica a la interfase.

La palabra *established* indica una conexión establecida en el protocolo TCP. Se produce una correspondencia si en el datagrama TCP están establecidos los bits ACK o RST, lo cual señala que el paquete es propiedad de una conexión existente.

Ejemplo
access-list 102 permit tcp 0.0.0.0 255.255.255.255 181.12.0.0. 0.0.255.255 established

En el ejemplo tenemos una red (181.12.0.0) conectada a Internet, y la política de seguridad requiere que cualquier host en la red interna realice conexión TCP con cualquier host de Internet.

Cómo filtrar las llamadas entrantes y salientes

Para restringir las conexiones entrantes y salientes entre una línea dentro del router Cisco y las direcciones en una lista de acceso, se debe usar el comando de configuración de acceso y clase:

access-class lista (in / out)

donde

lista - es el número de la lista de acceso. Al utilizar el valor *in* se restringe el tráfico entrante y con el valor *out* se restringe el tráfico saliente entre el router Cisco y las direcciones de la lista de acceso respectivamente. Si se desea eliminar las restricciones de acceso para una lista de acceso especificada se debe usar el comando:

no access-class numero de lista de acceso (in / out)

Cómo examinar la colocación de filtros de paquetes y la suplantación de direcciones.

Al diseñar las reglas de los filtros de paquetes se debe especificar si se desea realizar el filtrado en los paquetes entrantes, en los salientes o en ambos del router. Muchos distribuidores lo ejecutan en los paquetes salientes por razones de eficacia. El filtrado se efectúa cuando el router consulta las tablas correspondientes para determinar el destino a seguir. Si el paquete no puede enviarse a una ruta dada o no corresponde a las reglas del filtro, se lo rechaza. Es de suma importancia determinar el lugar preciso donde se debe colocar el router y las interfaces sobre las que se aplican las reglas de los filtrados. Si los routers filtran

los paquetes en el momento en que éstos son enviados afuera, un atacante podría decir que viene de la red interna, cuando esto no es la realidad; en este caso se perderá información porque el router no sabe de qué interfase llegó el paquete, dejando a su red expuesta. Este tipo de ataque se conoce como *suplantación de dirección*. El filtrado de paquetes entrantes puede prevenir la suplantación de dirección. En general, el filtrado debe realizarse tan rápido como sea posible. Hay que tener en cuenta que el router permita que las reglas de filtrado tengan la especificación tanto de los ports origen como de los ports destino en una sola regla. Si se las separa en dos reglas, una para ports origen y otra para ports destino, se puede producir una brecha en la seguridad.

Antes de diseñar las reglas de filtrado de paquete, primero se debe tener en cuenta el comportamiento del servicio de aplicación que se trate de filtrar. Como ejemplo, algunos servicios de aplicación, como FTP, requieren un mecanismo de respuesta donde un host externo pueda necesitar iniciar una conexión con el host interno dentro de un port desconocido al momento de especificar las reglas de filtrado de paquetes. O en otro caso, el protocolo X11 también requiere conexiones de respuesta.

2. Host Bastión

Un host bastión es la presencia pública de la organización a la que pertenece en la Internet. Es el lugar que puede ser recorrido "libremente" por cualquier persona del mundo exterior. Por tal razón se encuentra altamente expuesto a todo tipo de riesgos, y es imprescindible convertirlo en una máquina configurada con máxima seguridad.

Si, por razones de requerimientos del sitio, hay más de un host bastión en el firewall, cada uno respeta y sigue las características y consideraciones técnicas dadas a continuación.

Para **diseñar y construir un Host Bastión** se deben seguir **dos principios básicos** generales:

- Mantenerlo simple.
- Estar preparado para el caso en que se lo comprometa.

La primera consideración es porque el Host Bastión es más fácil de asegurar cuanto más simple es. Por ejemplo, si se quiere habilitar algún servicio dentro del Host, se corre el riesgo de que tenga fallas o errores que pueden ser aprovechados por cualquier atacante que quiera alterarlo o desactivarlo. Por eso se busca que haga lo menos posible. "Debería proveer el conjunto de servicios más chico con los menores privilegios que posiblemente se pueda, mientras esté cumpliendo su papel".

La segunda es porque se debe tener presente que, a pesar de los esfuerzos que se hagan para asegurarlo, siempre está la posibilidad de que alguien o algo pueda 'comprometerlo' ya que, como se dijo más arriba, es la máquina visible desde el mundo

exterior. En caso de que fuera 'quebrado', hay que evitar que esta intrusión se propague a todo el firewall. Como medida de prevención se trata de que las máquinas internas no confíen en él más de lo imprescindible para que funcione. Se deben analizar cada uno de los servicios que provee el Host Bastión a las máquinas internas y establecer los privilegios mínimos necesarios. Una vez que se tomaron estas decisiones, se pueden usar mecanismos para reforzarlas. Por ejemplo, se instalarían mecanismos de control de acceso estándar (passwords, dispositivos de autenticación, etc.) sobre los hosts internos, o se establecería un filtrado de paquetes entre el host bastión y los hosts internos.

Consideraciones a tener en cuenta para la construcción del Host Bastión

1 *Elección de la maquina*: Se debe decidir qué tipo de máquina se va a utilizar teniendo en cuenta el sistema operativo, la velocidad del host bastión, y la configuración de hardware. Conviene emplear un sistema operativo familiar, no uno nuevo y/o desconocido, y para la selección se debe considerar que provea todos los servicios de Internet para las plataformas existentes. En caso de tener que elegir uno desconocido se recomienda utilizar el UNIX porque cuenta con un extenso conjunto de herramientas para construir hosts bastiones; además, es sobre el que se tiene mayor experiencia. En cuanto a la velocidad, no interesa que la CPU sea muy potente porque el tiempo de respuesta para su trabajo depende principalmente de la velocidad de la conexión con el mundo exterior. Otras razones para no preferir una máquina potente es que en el ambiente de los atacantes no da prestigio "quebrar" a una maquina chica y, en caso de estar "comprometida", al ser un poco lenta es menos útil para atacar sistemas internos u otros sitios; además, si fuera rápida, se desperdicia la mayor parte del tiempo esperando una conexión lenta de red y eso es una puerta para que los usuarios del interior utilicen ese sobrante para otras cosas; en esos momentos no se está manteniendo la seguridad del host bastión. Para la configuración del hardware, se busca que sea confiable, por lo que es mejor seleccionar los elementos que están en un rango intermedio.
2 *Elección de la ubicación física*: Es necesario que esté en un lugar físicamente seguro, lejos de personas que puedan atacarlo (p. ej., robarlo, romperlo).
3 *Ubicación del Host Bastión en la red*: Debería estar ubicado sobre una red que no lleve tráfico confidencial, preferiblemente una red especial de su propiedad. Una forma de tratar de resolver el problema es poner al host bastión en una red perimetral en lugar de que esté en una red interna. Usando una red perimetral con un router de filtrado entre ésta y la red interna se tienen ventajas adicionales. Así se limita más su exposición, si es que se compromete al host bastión, reduciendo el número de hosts y servicios que el host bastión puede acceder.
4 *Selección de servicios provistos por el Host Bastión:* Provee cualquier servicio que el sitio necesite para acceder a la Internet, o que quiere ofrecer a la Internet – son los servicios que no creemos seguros para ofrecer directamente vía filtrado de paquetes. En el host bastión no se debería poner servicios que no se van a usar hacia o desde la Internet. Básicamente,

se debería deshabilitar todo lo que no se va a usar, y se debería elegir cuidadosamente lo que sí se va a usar. Se los puede clasificar en:

- *Servicios que son seguros* de proveer vía filtrado de paquetes.
- *Servicios que son inseguros en la forma que se los provee normalmente y se los puede hacer seguros* instalándolos sobre el host bastión.
- *Servicios que son inseguros en la forma que se los provee normalmente y no se los puede hacer seguros;* a estos se los debe deshabilitar o poner sobre un host víctima si son absolutamente necesarios.
- *Servicios que no se usan, o que no se usan en conjunto con la Internet;* deben ser deshabilitados.

5 *No autorización de cuentas de usuarios sobre el Host Bastión*: Mantener las cuentas de los usuarios fuera del Host Bastión es lo que da la mejor seguridad. Hay varias razones:

- Vulnerabilidad de las cuentas en sí.
- Vulnerabilidad de los servicios requeridos para soportar esas cuentas.
- Estabilidad y confiabilidad reducida de la máquina.
- Subversión inadvertida de la seguridad del host bastión por los usuarios.
- Incremento en la dificultad de la detección de ataques.

Construcción del Host Bastión

Para hacerlo se deben seguir estos pasos:

1. Asegurar la máquina.
2. Deshabilitar los servicios no requeridos.
3. Instalar o modificar los servicios que se quieren proveer.
4. Reconfigurar la máquina desde una configuración adecuada para desarrollo dentro de su estado de ejecución final.
5. Ejecutar una auditoría de seguridad para establecer una línea base.
6. Conectar la máquina a la red en donde se la va a usar.

Hay que ser muy cuidadoso respecto a la posibilidad de que se acceda a la máquina desde Internet antes de que se la haya terminado de construir, porque de esa manera cabe la posibilidad de que se transforme en un "mecanismo de ataque" por parte de un atacante en vez de ser de defensa; además - y como consecuencia de lo anterior - no se puede establecer una línea base confiable para las posteriores mediciones de auditoría (podría haber entrado un intruso y entonces no sería detectable).

Formas de asegurar la máquina

Conviene empezar con un sistema operativo estándar, y hacerlo lo más seguro posible. Esto implica:

Instalación mínima del sistema operativo

Se recomienda usar un sistema operativo estándar y original porque de esa manera se tiene la seguridad de conocer exactamente lo que se está usando - se sabe que se está empezando desde una instalación no-modificada.

Conviene dejar lo mínimo imprescindible para no tener que borrarlo más adelante; también se reduce la posibilidad de tener fallas o errores del producto. En última instancia, si queremos alguna facilidad que no está instalada, lo único que hay que hacer es adicionarla.

Corregir las fallas conocidas del sistema

Cuando se trabaja con un sistema operativo conocido, se tienen registradas con mayor facilidad las fallas (o bugs) que presenta, y la solución correspondiente. Por eso se recomienda conseguir una lista detallada de los patches de seguridad y de las advertencias, y analizar cuáles de esas son las adecuadas para tener en cuenta en nuestra instalación.

Usar una lista de chequeo (checklist)

Se recomienda usar una checklist adecuada a la plataforma y al sistema operativo que se está utilizando en la instalación, para no olvidarse de nada referente a la seguridad del host bastión, y no saltear pasos.

Poner a resguardo los system logs

Hay que asegurarse de tener una forma de salvaguardar los system logs del host bastión ya que son importantes por dos razones:

- Son uno de los mejores métodos para determinar si el host bastión está funcionando como debiera.
- Si en algún momento alguien quebrara al host bastión, los logs del sistema son uno de los mecanismos primarios que determinan exactamente qué es lo que pasó. Al examinar los logs y descubrir qué es lo que anduvo mal, se podría llegar a prevenir un suceso similar en el futuro.

El tema es dónde ubicar los systems logs. Se busca que estén en algún lugar conveniente para analizarlos sin dificultad y poder ver qué está haciendo el host bastión. Además, que estén en un lugar seguro como para mantenerlos fuera de cualquier riesgo de alteración en caso de que se los necesite utilizar para reconstruir un incidente. La solución a estos requerimientos es mantener dos copias de los system logs: una por conveniencia y la otra para catástrofes.

- System logs por conveniencia.

Esta copia es la que se usa sobre una base regular para monitorear el flujo de actividad de la máquina. Sobre estos logs es donde se corren diaria y semanalmente los informes de análisis automatizados.

Al pensar en donde guardarlos, se puede optar entre dejarlos en el host bastión con la ventaja de no establecer un logging para ir a otro sistema ni tener que configurar el

filtrado de paquetes para permitirlo, dándonos una mayor simplicidad, o sino dejarlos en un host interno obteniendo así facilidad de acceso, porque en el host bastión no se tienen herramientas para examinarlos. Además, evitamos conectarnos al host bastión, lo cual es una medida requerida de seguridad.

- System logs para catástrofes.

Esta copia es la que se usará en caso de una catástrofe. La otra copia puede no estar disponible en el momento necesario o podemos ya no estar seguros de su integridad. Hay varias maneras de crearlos:

a) Forma más simple: Conectar una impresora en línea a un port serial del host bastión y mandar una copia de todo lo que pasa a esa salida.

b) Forma más efectiva: Conectar una PC dedicada a un port serial del host bastión, como dispositivo de logging de vuelco seguro con la condición de que no esté conectada en red. La PC se configura de manera tal que cuando bootea inicia un programa en modo "registro", y cada tanto los archivos de log se rotan y se recortan para que el sistema nunca funcione sin espacio. La ventaja respecto al anterior está dada por la facilidad de realizar un análisis automatizado de los logs, ya que están registrados en un disco rígido. El único riesgo estaría dado por la posibilidad de que alguna persona "no confiable" pueda acercarse a la máquina.

- Instalación de los system logs.

En un ambiente UNIX, el logging se maneja a través de syslog. Este servicio registra mensajes desde clientes locales y remotos, junto con los correspondientes códigos de prioridad (dan la importancia) y facilidad (indica de qué subsistema viene). Dado un mensaje, se puede optar qué hacer basándonos en los códigos mencionados más arriba. Por ejemplo, ignorarlo, dirigirlo hacia otro sistema, mostrarlo en la pantalla de ciertos o todos los usuarios, registrarlo en uno o más archivos, o cualquier combinación de estas opciones.

Cuando configuramos el registro de mensajes dirigido a archivo, podemos elegir entre varias opciones que nos darán distintas formas de monitorear lo que está pasando.

Hay dispositivos de red - tales como routers - que pueden configurarse para hacer un log de mensajes vía syslog. Esto simplifica el proceso de recolección de los mensajes.

El servicio que ofrece *syslog* es útil y recomendable, a pesar de ciertas debilidades que presenta y se mencionan a continuación:

- El logging remoto vía syslog no es totalmente confiable, porque se basa en un protocolo UDP - con la inseguridad respecto al origen y recepción de los paquetes que esto implica.
- Tampoco se tiene la certeza de que el sistema receptor está disponible o accesible.

Por la última razón es que se recomienda trabajar con una máquina de vuelco seguro (dropsafe) conectada localmente al host bastión.

Deshabilitar los servicios innecesarios

Al saber que es imprescindible contar con un host bastión lo más compacto posible se debe tener en cuenta que una de las puertas de ingreso para los atacantes son los servicios que se puedan encontrar en él, porque es común que todo servicio pueda presentar fallas o errores de configuración, llevándonos así a problemas de seguridad.

El objetivo es eliminar todo aquello que no es necesario: ni siquiera conviene dejarlo anulado, hay que borrarlo.

¿Cómo se manejan los servicios?

Sobre las máquinas UNIX, la mayoría de los servicios se maneja de una de estas dos maneras:

- Controlando cuándo empiezan y quién los usa.
- Por archivos de configuración específicos del servicio.

Hay dos formas en que se inician los servicios en los sistemas UNIX:

- En el momento de booteo desde los archivos /etc/rc de la máquina.
- Bajo demanda por el daemon inetd (que a su vez se inicia en el momento de booteo).

¿Cómo deshabilitar los servicios?

Como primera medida hay que tomar precauciones tales como tener una manera de bootear la máquina para cuando se quiere deshabilitar un servicio crítico. Además, tener una copia de cada archivo que se quiere modificar, para poder reconstruir en caso de errores.

Es preferible comentar las líneas en lugar de borrarlas, porque queda una indicación de lo que se quiso hacer; junto con esto, conviene anotar las razones por las que se las deshabilitó, para posteriores pruebas.

Después de haber rebooteado y testeado la máquina, y verificar que todo funciona como buscábamos, se recomienda eliminar a los ejecutables de los servicios anulados, para que nadie pueda utilizarlos, ni siquiera por error. Otra opción es encriptarlos con algún producto seguro.

¿Qué servicios se deberían dejar habilitados?

Ciertos servicios son esenciales para el funcionamiento de la máquina, y hay que dejarlos habilitados. Además, se necesitarán procesos de servidor para los que se decidieron proveer en el host bastión, p.ej., FTP, SMTP, Telnet real o proxy, y servidores DNS.

¿Qué servicios se deberían deshabilitar?

Hay tres reglas sencillas que se aplican para determinar los servicios a deshabilitar:

- Si no es necesario, se desconecta.
- Si no se sabe lo que hace, se desconecta.

- Si desconectarlo trae problemas, ahora se sabe lo que hace, por lo tanto se puede volver atrás o replantearse como trabajar sin ese servicio.

Lista de servidores que pueden anularse en un host bastión:

- NFS y servicios relacionados
- Otros servicios RPC (Remote Procedure Call)
- Servicios de booting
- Servicios BSD de comandos 'r'
- Routed
- fingerd
- ftpd
- Otros servicios

Instalación y modificación de servicios

Nos podemos encontrar con servicios que no son provistos por el sistema operativo que estamos usando; y con algunos otros que sí provee pero son inadecuados para usar en un entorno de seguridad.

A los servicios que dejamos habilitados se los tiene que proteger al máximo posible. Existen distintos paquetes que realizan esta tarea: el TCP Wrapper, el programa *netacl* del paquete TIS FWTK para mejorar la seguridad, o el SOCKS. También se tendría que proveer logging.

Reconfigurar para producción

Cuando estamos seguros de que todo funciona como esperábamos, podemos reconfigurar al host bastión para instalarlo en producción. Se necesitan hacer varias cosas:

- Reconfigurar y reconstruir el kernel.
- Remover todos los programas innecesarios.
- Montar todos los filesystems que sea posible en modo read-only.

Al dejar reducido al host bastión a su menor expresión, se nota lo difícil que se hace la actualización o instalación de nuevos productos, lo cual era en parte nuestro objetivo.

Reconfigurar y reconstruir el kernel

En esta etapa, el primer paso es reconstruir el kernel para eliminar las capacidades que no son necesarias. Con esto, además de reducir el tamaño del kernel, dejando más memoria libre para otros propósitos, los atacantes no tienen la posibilidad de explotar estas capacidades.

Al deshabilitar las capacidades del kernel, hay que tener mucho cuidado por las

numerosas interrelaciones que existen entre ellas. De ahí la dificultad de determinar exactamente para qué se usa cada una.

Cuando se tenga el kernel "mínimo" trabajando, conviene borrar o encriptar el kernel con "las funciones completas" y reemplazarlo con una copia backup del kernel mínimo que se está usando. Así se evita que un cracker trate de introducirse en nuestra máquina y use la copia de backup para rebootearla y entonces restaurar todas las funciones que habíamos deshabilitado. Por idénticas razones, se busca borrar los archivos y programas necesarios para construir un nuevo kernel.

Remover los programas no esenciales

Normalmente se deberían remover los programas que no son esenciales para la operación diaria. El host bastión sólo está proveyendo servicios Internet, no necesita ser un entorno agradable para trabajar. Al eliminar programas, se elimina el riesgo de que los atacantes utilicen las fallas que pudieran contener. Uno de los programas más buscados son los *setuid/setgid*; también los sistemas Windows, por los problemas de seguridad que presentan, y los compiladores, porque cualquier atacante los puede usar para construir sus propias herramientas.

Pero, en vez de simplemente borrar lo que no queremos dejar al alcance de un atacante, conviene reemplazarlo por programas que activen una alarma al querer ejecutarlos. Hay quien prefiere que detengan el sistema (halt) después de la alarma.

Montar los filesystems en modo Read-Only

Para evitar que cualquiera sea capaz de cambiar la configuración del host bastión - una vez que quedó terminada - se trata de montar en modo Read-Only a la mayoría de los filesystems, dentro de lo posible.

Todavía es mejor si se puede usar hardware protegido contra escritura; un atacante puede llegar a remontar discos con permiso de escritura sin tener acceso físico a la máquina, pero no puede hacer nada si el hardware está protegido contra escritura.

Siempre hay que dejar cierta parte del filesystem para escritura de cosas como, por ejemplo, un espacio de scratch, los system logs, y el spool del mail. Se podría utilizar un disco RAM, pero tenemos que estar seguros de que nuestro sistema operativo lo soporta, de que tenemos suficiente RAM, y de que podemos correr el riesgo de perder lo que esté en el disco RAM si por alguna razón se rebootea la máquina.

Ejecutar una Auditoría de Seguridad

Hay dos razones para ejecutar la auditoría de seguridad:

1. Brinda una manera de asegurarnos que no nos olvidamos de nada durante la configuración total del sistema.
2. Establece una "línea base", o base para comparación, contra la cual comparar futuras auditorías.

De esta manera vamos a ser capaces de detectar cualquier violación a la máquina.

Paquetes de Auditoría

Muchos paquetes de auditoría tienen dos propósitos básicos:

- Chequear buscando los agujeros de seguridad conocidos. Estos son los agujeros que no fueron cubiertos por los administradores del sistema, explotados por los atacantes en los break-ins del sistema, o los documentados en papers y libros de seguridad.
- Establecer una base de datos para checksums de todos los archivos en un sistema; haciendo esto se le permite a un administrador de sistema que pueda reconocer futuros cambios en los archivos - particularmente los cambios no autorizados.

La efectividad de los paquetes que chequean los agujeros de seguridad conocidos es muy dependiente del sistema operativo y de la versión sobre la que trabajamos.

Uso de los Paquetes de Auditoría

Después de realizar varias repeticiones de ejecución del paquete de auditoría, y haber creado la línea base inicial, se debe guardar una copia de las herramientas usadas y estos resultados iniciales de auditoría en algún lugar seguro. Desde ningún punto de vista se deben dejar en el host bastión. Si en el peor de los casos alguien lo viola, ya no nos sirve de referencia la auditoría base ni los resultados de futuras auditorías, porque un intruso podría haber adulterado la original - quitándonos la posibilidad de usarla posteriormente para detectar cambios ilegales en el sistema. También podrían cambiar el software de auditoría, y de esa manera generar siempre un reporte que reproduzca la línea base. Tampoco es conveniente dejar los resultados periódicos al alcance de cualquiera porque sería fácil ver qué es lo que se audita y qué no.

Según sean las capacidades y necesidades de la organización, se recomienda ejecutar la auditoría diariamente o semanalmente. Cuanto más frecuentemente se haga, mejor, porque se le dificulta a los atacantes la posibilidad de quebrarlo, detectar el sistema automatizado de auditoría, y subvertirlo antes de que corra la siguiente auditoría.

Checksums para auditoría

Un checksum es el número calculado a partir de los contenidos del archivo, que va a cambiar si el archivo cambia. El cálculo del checksum consume tiempo, pero no tanto como leer dos veces el mismo archivo haciendo una comparación bit a bit. Además, el almacenamiento de los checksums lleva menos espacio que el de todo el archivo. Pero no son una representación completa de los archivos, y se pueden obtener números iguales para distintos archivos. Hay diferentes algoritmos de checksum. Son muy útiles en auditoría porque permiten detectar en una forma simple los cambios que un intruso pudo realizar sobre algún archivo.

Conexión de la máquina

Ahora ya está todo listo como para finalmente conectarla a la red de destino y empezar a usarlo en producción.

Operación del Host Bastión

Aprender cuál es el Perfil de Uso Normal

Para poder detectar anormalidades durante el funcionamiento del host bastión que nos indiquen que puede estar comprometido, primero se necesita saber qué es "normal". Se logra haciéndonos algunas preguntas, como:

- ¿Cuántos trabajos tienden a ejecutarse a la vez?
- ¿Cuánto tiempo de CPU consumen estos trabajos relativos entre sí?
- ¿Cuál es la carga típica a diferentes horas a través del día?

Considerar la escritura de un software para Monitoreo Automático

Generalmente los logs terminan siendo utilizados solamente después de un break-in cuando, de hecho, tendrían que usarse para detectar, y tal vez detener, las intrusiones. Debido a que cada sistema operativo y sitio es diferente, cada host bastión está configurado distinto, y cada sitio tiene diferentes ideas acerca de cuál debería ser la respuesta del sistema de monitoreo. Por eso el monitoreo tiende a ser muy específico del sitio y del host en los detalles.

Protección de la Máquina y los Backups

Observar cuidadosamente los Rebooteos

No siempre es obvio detectar que alguien quebró la seguridad; a veces se deben sacar conclusiones a partir de comportamientos extraños de la máquina, rebooteos no previstos, o retardos en el sistema.

Una vez en funcionamiento, ya configurado y en producción, el host bastión debería ser un sistema muy estable. Por eso, al producirse algún crash o rebooteo conviene investigar si hubiera sido el resultado de un posible ataque.

Una medida más de seguridad es no permitirle el rebooteo automático De esta manera evitamos que un atacante fuerce un rebooteo.

Hacer Backups seguros

Los backups del host bastión son tramposos debido a los asuntos de seguridad. La duda es siempre en quién confiar.

Normalmente, los backups se hacen sobre unidades de cinta conectadas directamente al host bastión, no en discos que permanecen conectados a la máquina. Cualquiera sea el medio a utilizar, como primera medida debe ser removible, para que no quede al alcance de atacantes o de usuarios bien intencionados pero torpes que puedan comprometerlos.

Los backups del host bastión no se hacen sólo para prevenirnos contra catástrofes de sistemas normales como roturas de disco. También son una herramienta que se puede utilizar posteriormente para investigar un break-in o algún otro incidente de seguridad. Dan una forma de comparar lo que está corrientemente en el disco del host bastión con

lo que estaba antes del incidente.

Según la frecuencia con que se hagan los backups, hay que analizar el tema de los logging. Si no hacemos un backup diario, el logging se *debe* hacer sobre algún sistema que no sea el host bastión. Los logs son de gran valor para reconstruir lo que sucedió después de un incidente. Si estuvieran en el host bastión, no serían un material confiable para trabajar, porque podrían haber sido alterados o destruidos.

Los backups del host bastión se tienen que guardar con el mismo cuidado con que se guarda la máquina. Ahí se tiene guardada la información de configuración para el host. Un atacante que tenga acceso a los backups está teniendo el análisis de la seguridad del host bastión en sus manos sin tener que tocarlo.

3. Sistemas Proxy

El Proxy es un sistema intermediario entre los hosts internos de una red y los hosts de Internet de forma tal que recibe los requerimientos de unos y se los pasa a los otros, previa verificación de accesos y privilegios. Es decir, todas las comunicaciones entre el usuario interno y la Internet pasan por el servidor proxy en lugar de permitir que los usuarios se comuniquen directamente con otros servidores de la Internet. Los proxy normalmente corren sobre una maquina del firewall; podrían estar en un host "dual-homed" o en hosts "bastión". Solamente son efectivos si se utilizan junto a métodos de restricción de tráfico IP entre clientes y servidores reales. De este modo, un cliente no podrá atravesar el servidor proxy para comunicarse con un servidor real utilizando este protocolo.

Un cliente proxy es parte de una aplicación que habla con el servidor real en la red externa en nombre del cliente interno.

En la actualidad los clientes de Web más conocidos, como el Internet Explorer y el Netscape Navigator, poseen soporte de proxy.

Un servidor proxy provee ciertas mejoras de seguridad. Permite que los sitios concentren servicios a través de un host específico para permitir monitoreo, ocultamiento de la estructura interna, etc. Esta concentración de servicios crea un destino atractivo para un intruso potencial. El tipo de protección requerido para un servidor proxy depende grandemente del protocolo proxy en uso y de los servicios que se están pasando por un proxy. Un buen punto de inicio es la regla general de limitar el acceso a sólo aquellos hosts que necesitan los servicios, y limitando el acceso por esos hosts a solamente esos servicios.

Funcionamiento

El programa cliente del usuario se comunica con el servidor Proxy enviando un pedido de conexión con un servidor real. El servidor Proxy evalúa este pedido y decide si se permitirá

la conexión. Si la permite, envía al servidor real la solicitud recibida desde el cliente. De este modo, un servidor Proxy se ve como "Servidor" cuando acepta pedidos de clientes y como "Cliente" cuando envía solicitudes a un servidor real. Una vez establecida la comunicación entre un cliente y un servidor real, el servidor Proxy actúa como un retransmisor, pasando comandos y respuestas de un lado a otro. Un punto importante a tener en cuenta en este tipo de conexión es que es totalmente transparente. Un usuario nunca se entera de que existe un "intermediario" en la conexión que ha establecido. La comunicación entre el programa cliente y el servidor Proxy puede realizarse de dos formas distintas:

- Software Cliente estándar: El Software Cliente debe saber cómo opera el servidor Proxy, cómo contactarlo, cómo pasar la información al servidor real, etc. Se trata de un programa cliente modificado que establece el contacto de manera automática para que cumpla con ciertos requerimientos.
- Procedimientos de Usuario estándar: El usuario utiliza un Cliente estándar para conectarse con un servidor Proxy y usa diferentes procedimientos (comandos del servidor Proxy) para pasar información acerca del servidor real al cual quiere conectarse. El servidor Proxy realiza la conexión con el servidor real.

Tipos de servidores Proxy

Existen dos tipos básicos de servidores Proxy:

- *Servidor Proxy de Aplicación*: Es un servidor que conoce sobre una aplicación en particular y provee servicios proxy para la misma. Entiende e interpreta comandos de un protocolo en particular. Cuando se trabaja con este tipo de servidor, hace falta contar con uno para cada tipo de servicio. Recibe también el nombre de servidor Dedicado. Para los clientes Web, las modificaciones para soportar un proxy a nivel de aplicación son mínimas. No es necesario compilar versiones especiales de clientes Web con librerías Firewall, los clientes estándares pueden ser configurados para ser clientes de un proxy. Esto es importante cuando no se tiene el código fuente para modificar, especialmente para los clientes Web comerciales. Los usuarios no necesitan modificaciones especiales para obtener información FTP, Gopher y Wais a través del Firewall. Un único cliente Web con un servidor proxy maneja todos estos casos. Además, el proxy estandariza la aparición de los listados FTP y Gopher, en lugar de que cada cliente tenga su propio manejo especial. Usando HTTP entre un cliente y un proxy no se pierde la funcionalidad del protocolo FTP, Gopher. Hacer proxy permite registros(logging) de alto nivel de las transacciones de los clientes, incluyendo direcciones IP del cliente, fecha y hora, URL, conteo de byte y código de éxito. Cualquier campo normal y de meta-información que de una transacción HTTP son candidatos para el registro. Además es posible filtrar las transacciones de los clientes a nivel de protocolo de aplicación (como por ejemplo los access control list - acl - de Squid); el proxy puede controlar el acceso a servicios para métodos individuales, host y dominio, etc.

- *Servidor Proxy de Circuito*: Crea un circuito virtual entre el cliente y el servidor real sin interpretar el protocolo de la aplicación; también son llamados Proxys Genéricos. El servidor Proxy de Circuito provee soporte para un gran conjunto de protocolos. Como ejemplo se tiene el Socks que actúa en la capa de aplicación. El software necesario para trabajar con este producto contiene:

⇒ SOCKS Server, que corre sobre plataformas UNIX o basadas en UNIX.

⇒ SOCKS Client Library, que se encuentra entre la capa de aplicación y la de transporte del cliente.

⇒ Librerías para soportar aplicaciones Macintosh y Windows.

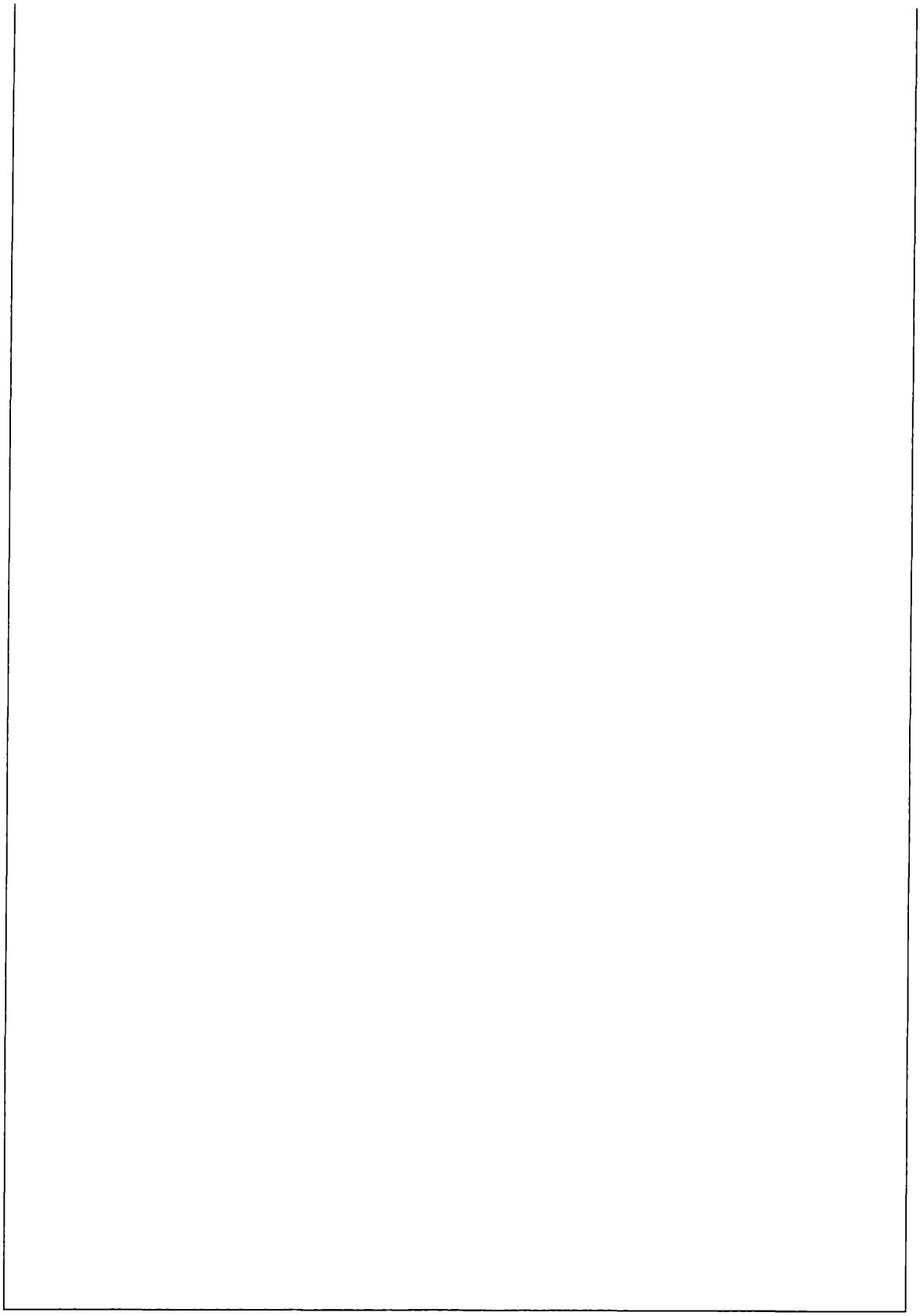
Socks requiere para el cliente el Custom Client Software, pero es posible reutilizar el software cliente de manera que se pueda llamar a un servidor de este tipo y utilizar sus servicios. Para lograrlo, se debe recompilar el programa junto con la Socks Client Library, reemplazando todas las llamadas a la red estándar con las llamadas de la versión de Socks. Es un sistema proxy equipado con seguridad, auditoria, tolerancia a fallas y notificación de alarmas. Establece un canal de datos seguro entre dos host dentro de un entorno cliente/servidor. La generación de logs durante una conexión no es lo suficientemente detallada como para facilitar su inspección. Por estas razones, el Socks es un producto capaz de proporcionar servicio a múltiples protocolos.

Ventajas de utilizar servidores Proxy:

- Permite a los usuarios acceder a los servicios de Internet ocultando totalmente la red interna.
- Permite un buen servicio de log a nivel de cada aplicación. Debido a que todo el tráfico pasa a través del servidor Proxy, se puede registrar gran cantidad de información con fines de auditoría y seguridad.

Desventajas de utilizar servidores Proxy:

- A menudo se requiere la modificación del software cliente.
- Hay software que está disponible sólo para ciertas plataformas: Por ejemplo,: IGATEWAY es un paquete Proxy para FTP y TELNET de SUN que corre sólo sobre SUN. La disponibilidad de estos tipos de paquetes en general no es inmediata.
- En el caso de servidores Proxy de aplicación se requiere un servidor Proxy para cada servicio.
- Algunos servicios no son viables para trabajar con servidores Proxy (por ejemplo, talk).
- El uso de servidores Proxy introduce algún retardo en las comunicaciones.
- Los servidores Proxy de circuitos no brindan control específico sobre las aplicaciones.



Parte II: ANÁLISIS DE SERVICIOS PARA PROVEER A TRAVÉS DE UN FIREWALL

• PARTE II - ANALISIS DE SERVICIOS PARA PROVEER A TRAVES DE UN FIREWALL

En una organización hay muchos servicios que se quisieran proveer a sus usuarios, donde algunos pueden ser externos. Por muchas razones de seguridad se intenta aislar a los servicios en hosts dedicados. En otros casos se hace así por razones de performance.

Como ya mencionamos, un firewall puede construirse sobre diferentes dispositivos físicos (routers y host bastión) usando de forma conveniente las variantes que ofrecen tanto el filtrado de paquetes como los servicios proxy. Por lo tanto, para poder construir un firewall:

◊ Se debe conocer como funciona cada uno de los servicios.
◊ Conveniencias, ventajas y desventajas de la instalación particular de cada uno.
◊ En caso de realizarla, qué mecanismo utilizar para que el servicio sea seguro.

En muchos casos, los servicios que un sitio puede proveer tienen diferentes niveles de acceso y modelos de confianza. Siempre se recomienda que los servicios que sean esenciales para la seguridad u operación ininterrumpida de un sitio, estén ubicados fuera (de la red interna) sobre una máquina dedicada con un acceso muy limitado (respondiendo al modelo "denegar todo"), antes que sobre una máquina que provea uno o más servicios - algo que tradicionalmente ha sido menos seguro, o requiere mayor accesibilidad por usuarios que accidentalmente pueden alterar la seguridad. También es importante distinguir entre hosts que operan dentro de diferentes modelos de confianza (p.ej., todos los hosts dentro de un firewall y cualquier host sobre una red expuesta).

De ser posible, cada servicio debería estar corriendo sobre una máquina diferente cuya única ocupación sea proveer un servicio específico. Esto ayuda a aislar intrusos y limitar el daño potencial.

Hay muchos tipos de servicios y cada uno tiene sus propios requerimientos de seguridad. Estos requerimientos variarán con la intención de uso del servicio. Por ejemplo, un servicio que sólo debería usarse dentro de un sitio (p.ej., NFS) puede requerir mecanismos de protección diferentes a los de un servicio provisto para uso externo. A veces puede resultar suficiente proteger al servidor interno del acceso del exterior. Sin embargo, un servidor WWW que suministra una página para ser vista por los usuarios en la Internet, requiere protección built-in. Esto es, el servicio/protocolo/servidor deben proveer cualquier seguridad que sea requerida para prevenir acceso y modificación no autorizada de la base de datos de la Web.

Los servicios internos (es decir, los que se usarán sólo por los usuarios dentro un sitio) y los externos (los servicios hechos deliberadamente disponibles para los usuarios fuera del sitio), en general tendrán requerimientos de protección que difieren entre sí. Por lo tanto, es inteligente aislar los servicios internos en un conjunto de host servidores y los servicios externos en otro conjunto de hosts. No es conveniente que los servidores internos y externos

estén ubicados sobre el mismo host. De hecho, muchos sitios llegan a tener hasta un conjunto de subredes (o aún diferentes redes) que son accesibles desde el mundo exterior y otro conjunto que es accesible sólo desde el sitio. Generalmente hay un firewall que conecta estos conjuntos. Hay que tener mucho cuidado para asegurar que dicho firewall está operando adecuadamente.

Una forma de servicio externo que merece alguna consideración especial es el acceso anónimo o *guest*. Esto puede ser o bien FTP anónimo o bien login *guest* (no autenticado). Es extremadamente importante asegurar que los servidores FTP anónimos y las identificaciones de usuario *guest* estén cuidadosamente aislados de cualquier host o file system de los que se pudieran aprovechar los usuarios externos. Otra área sobre la que se debe prestar especial atención se refiere al acceso anónimo de escritura. Un sitio puede ser legalmente responsable del contenido de la información disponible públicamente, tal que es recomendado el monitoreo cuidadoso de la información depositada por los usuarios anónimos.

CARACTERÍSTICAS DE LOS SERVICIOS ANALIZADOS

1. Correo Electrónico

Desde el punto de vista del usuario, el correo electrónico es uno de los servicios más fundamentales de Internet. Pero, desafortunadamente, también es uno de los más vulnerables. Los servidores de mail son objetivos de lo más tentadores, ya que aceptan datos arbitrarios desde hosts externos arbitrarios. El proceso que va desde que se escribe un mail hasta que llega a destino, se lleva a cabo por distintos programas. Un usuario puede escribir un mail sobre un agente de usuario que podría ser E*udora*, *pine*, etc. Cuando el mail se mandó, el agente de transporte de mail realiza el intercambio entre el host origen y el host destino. El software de transporte utilizado depende del tipo de enlace (por ejemplo, si utilizo TCP/IP normalmente se hará con SMTP). En un sistema de correo se tienen las siguientes partes:

- Un *servidor* que acepta correo desde hosts externos o lo envía a hosts externos.
- Un *agente de distribución* que pone el correo en el buzón correcto sobre el host local.
- Un *agente de usuario* que deja que el receptor lea el correo y componga el correo saliente.

Cada una de estas partes es vulnerable por una razón diferente:

- El servidor acepta directamente comandos (relacionados a la distribución del correo) desde hosts externos; por esta razón, si el servidor no es seguro, puede terminar dándole inmediatamente a un atacante todo el acceso que él tiene. Un firewall puede proteger contra *ataques de canal de comandos* restringiendo el número de máquinas sobre las que los atacantes pueden abrir canales de comando y proveyendo un servicio asegurado sobre esas maquinas.

- El agente de distribución necesita permisos especiales porque necesita ser capaz de escribir en cada buzón de usuario. Aunque el agente de distribución no necesita hablar al mundo externo, si puede ser subvertido de alguna manera, el intruso obtiene un acceso muy amplio.
- El agente de usuario corre como un usuario, y no habla al mundo externo, lo cual limita su potencia y accesibilidad; sin embargo, generalmente puede correr otros programas arbitrarios en respuesta a los datos recibidos.

Un firewall no puede hacer mucho contra los *ataques dirigidos por los datos*; al dato hay que dejarlo pasar, o no sería realmente capaz de recibir mail. En algunos casos es posible descartar/filtrar caracteres especiales que serían "peligrosos" en las direcciones de mail si alguien pudiera reconocerlos. La mejor elección, a pesar de todo, sería ejecutar agentes de usuario y de distribución actualizados y educar a los usuarios. Hacer esto protegerá contra la mayoría de los *ataques dirigidos-por-datos*. En cualquier caso, como este tipo de ataque tiende a ser complicado y difícil para recuperar la información obtenida, es relativamente raro.

Fallas de línea de comando

Las fallas de línea de comando están fuera del alcance de un firewall, ya que sólo pueden ser desplegados por alguien que es capaz de ejecutar comandos sobre su sistema. Uno de los propósitos de un firewall es impedir que los atacantes tengan esta capacidad.

A continuación se describen los dos protocolos que se usan más comúnmente para el correo electrónico: SMTP y POP.

Simple Mail Transfer Protocol (SMTP)

El intercambio de correo electrónico entre servidores de mail se maneja en la Internet con SMTP. Un servidor SMTP de un host acepta correo y examina la dirección destino para decidir si derivar el mail localmente o enviarlo a otra máquina. Si decide derivarlo localmente, recodifica los encabezados de mail y la dirección de distribución a la forma adecuada para el programa de distribución local, y luego manda el mail a ese programa. Si decide enviar el mail a otra máquina, modifica los headers, contacta a esa máquina (usualmente vía SMTP, pero a veces vía UUCP u otro protocolo), y envía el correo.

Para los sistemas UNIX, el más comúnmente usado es el Sendmail. Hay otros mailers (smail 3, MMDF, Z-Mail) que no se usan tanto como Sendmail, y por lo tanto tienen menos gente que esté buscando las fallas de seguridad que pudieran tener. Este hecho, más que cualquier superioridad inherente, es probablemente la razón por la aparente mayor seguridad de estos mailers alternativos; no es que no tengan problemas, lo que pasa es que la gente no los conoce.

Mejorando la seguridad SMTP con smap y smapd

Un paso importante que puede dar un firewall para mejorar la seguridad es prevenir que los atacantes hablen SMTP directamente a Sendmail y, en lugar de esto, usen un servidor sustituto. Por suerte, esto es viable. SMTP quiere decir "Simple Mail Transport Protocol", y realmente es simple. Sólo hay más o menos seis comandos en el protocolo que un servidor SMTP tiene que implementar de modo de aceptar el correo que llega.

El resultado de usar este servidor sustituto es que un atacante nunca tiene una conexión SMTP directa a Sendmail o a cualquier otro servidor SMTP complejo. Tal sistema no protege contra agujeros de seguridad dirigidos-por-datos, pero tales agujeros serían muy difíciles de controlar por cualquier firewall. Afortunadamente, este tipo de agujero de seguridad parece ser muy raro en Sendmail.

Post-Office Protocol (POP)

SMTP se usa para intercambiar mail entre servidores. En cambio, POP es un protocolo cliente/servidor para manejar los buzones electrónicos de los usuarios. Con POP, un buzón de usuario (el verdadero archivo donde se guarda el correo entrante del usuario para su acceso posterior) se conserva sobre un servidor, en vez de hacerlo sobre la PC del usuario. Probablemente, el servidor está disponible para aceptar el correo entrante de una forma más consistente que la PC del usuario (particularmente si se trata de una PC portátil y que sólo a veces está conectada a la red). Cuando el usuario quiere su e-mail, accede a su buzón usando un programa cliente (tal como Eudora o Z-Mail) sobre su propia máquina mediante el protocolo POP.

Hay dos temas principales de seguridad involucrados con el uso de POP a través de la Internet:

1. Los servidores y clientes POP estándar envían la password de usuario POP a través de la Internet en forma explícita, de modo que cualquiera que haga snooping sobre la conexión puede capturarla y reusarla más tarde. En muchos casos, la password POP es la misma que la del login del usuario; por consiguiente, cualquiera que la obtenga va a conseguir todos los privilegios del usuario - no sólo el correo electrónico. Hay variantes más seguras de POP que usan Kerberos (generalmente llamadas KPOP) y passwords no reusables para autenticación (llamadas generalmente APOP), pero estas variantes más seguras no están soportadas o disponibles ampliamente. Es más, se pueden presentar problemas para encontrar una combinación de clientes y servidores que soporten estas variantes y que trabajen bien en nuestro sitio.
2. Sin observar los temas de autenticación, también hay que considerar la sensibilidad del email que los usuarios estarán accediendo sobre la Internet vía POP porque será visible por cualquiera que esté haciendo snooping sobre sus sesiones POP. Muchos sitios deciden que, a pesar de los temas de autenticación, el e-mail de los usuarios internos generalmente es demasiado sensitivo al riesgo de ser "observado" por alguien

monitoreando sus sesiones POP. Estos sitios deciden proveer métodos alternativos de acceso, tales como dial-up, que no son tan susceptibles para hacerles snooping. Si se les da a los usuarios la capacidad de llegar a la red en el interior del firewall (por ejemplo, con modems y PPP o SLIP), se les puede dar acceso POP mientras están viajando, sin permitirlo a través de la Internet.

2. Transferencia De Archivos

El FTP es la norma estándar para la transferencia de archivos en Internet. Hay algunos protocolos especializados, usados por aplicaciones, donde FTP no es el adecuado. TFTP se usa por dispositivos dedicados para transferir archivos de configuración. FSP es un protocolo basado en UDP que se usa cuando las conexiones basadas en TCP no funcionan o no están permitidas. UUCP se utiliza para transferencia batch, especialmente a través de líneas telefónicas.

File Transfer Protocol (FTP)

FTP se usa para transferir archivos de una maquina a otra, sin restricciones en el tipo de archivo a transferir. Hay dos tipos de acceso FTP: *FTP de usuario* y *FTP anónimo*. El FTP de usuario requiere una cuenta en el servidor y permite que el usuario acceda a cualquier archivo sobre el que tiene acceso si fue logoneado. El FTP anónimo es de acceso libre y se usa para acceder a archivos en todo el mundo.

Si un sitio proporciona un servidor FTP anónimo, cualquiera en Internet puede iniciar una conexión FTP a ese sitio, decirle al servidor que su nombre de login es *anonymous*, y acceder a cualquier archivo que esté disponible.

Brindar un servicio de FTP anónimo

Si se está proporcionando un servicio FTP anónimo, el desafío es asegurar que el servidor FTP anónimo ponga a disposición sólo la información que desea que sea visible desde el exterior y que no dé acceso a la gente del exterior a otra información supuestamente privada en la máquina.

Al configurar el FTP anónimo se tiene que tener la precaución de limitar qué información está disponible en la máquina que brinda el servicio FTP anónimo.

Muchos servidores FTP realizan un *cbroot* en el área del FTP anónimo antes de que el servidor FTP comience a procesar los comandos de un usuario anónimo. Para soportar tanto al FTP anónimo como al FTP de usuario, el servidor FTP necesita acceder a todos los archivos. Esto significa que *cbroot*, al que normalmente se lo considera como extremadamente seguro, no garantice a un servidor FTP debido a que el servidor no está siempre ejecutando el entorno *cbroot*. Para solucionar este problema, puede

cambiarse la configuración *inetd* de modo tal que, en vez de iniciar el servidor FTP directamente, inicie este *cbroot* y luego al servidor FTP. Esto hace que los usuarios no anónimos también estén limitados.

Ser cuidadoso con los directorios que pueden escribirse en el área de FTP anónimo

Los sitios generalmente brindan áreas de FTP anónimo, tanto para que la gente externa pueda dejar archivos en ellos como también para transferir archivos hacia ellos. Estos directorios que pueden escribirse serán encontrados y usados por Internet como espacio de almacenamiento y de distribución para material ilícito. Esto origina los siguientes problemas:

- Consume sus recursos, espacio de disco y amplitud de banda de la red especialmente en la conexión de Internet e interfiere con el uso legítimo de estos recursos.
- Potencialmente expone a su sitio a una acción ilegal por ayudar a la piratería de software.
- Incluso si no se realizan acciones ilegales, un incidente de este tipo podría generar una publicidad negativa para su sitio u organización.

¿Cómo puede evitarse el mal uso de las áreas de FTP anónimo? La primera pregunta que debe hacerse es: ¿Necesito de verdad un área donde tiene que escribirse para FTP anónimo?

Generalmente los archivos se envían a través de correo electrónico. Si realmente se necesita esta área, a continuación se describen varias maneras de evitar la vulnerabilidad:

- Hacer que los directorios de entrada sean solamente de escritura.
- Inhabilitar la creación de directorios y ciertos archivos.
- Carga por pre-arreglo: se crean directorios con nombre secretos que sólo los puede acceder el que conoce su nombre.
- Supresión de archivos.

Trivial File Transfer Protocol (TFTP)

TFTP es un protocolo trivial de transferencia de archivos; es más simple que FTP y está diseñado para implementarse en ROM para sistemas de booteo sin-disco como las terminales X, y en routers. Con TFTP no hay autenticación; un cliente TFTP simplemente se conecta al servidor y pregunta por un archivo, sin decir para quién es. Si el archivo es uno al que el servidor puede acceder, el servidor le da el archivo al cliente. Por esta razón, se necesita ser muy cuidadoso respecto a lo que el servidor TFTP puede acceder (si se tiene uno), y a qué clientes pueden acceder al servidor.

Generalmente, no hay razón para permitir TFTP a través del firewall, aún si se lo usa internamente. La gente no transfiere archivos con TFTP.

File Service Protocol (FSP)

FSP es un protocolo de transferencia de archivo diseñado para evadir las restricciones FTP. Muy pocos sitios proveen servicio soportado por FSP; la mayoría de los servidores FSP están instalados por los usuarios o los atacantes, sin el consentimiento o conocimiento de los administradores de la máquina. De hecho, la actividad FSP es frecuentemente un indicio de que los atacantes han comprometido un sitio; generalmente usan FSP para mover sus archivos de sitio en sitio. Sin embargo, la actividad FSP puede ser difícil de detectar, porque no hay números de port estándar para uso de los clientes y servidores FSP.

UNIX-to-UNIX Copy Protocol (UUCP)

UUCP fue diseñado para transferir archivos entre máquinas UNIX, incluyendo correo electrónico y news; es el sistema de red original de UNIX. UUCP primariamente se usó como un protocolo dial-up (como el único protocolo sobre una conexión moderna), pero algunos sitios, particularmente aquellos con servicio Internet de dial-up intermitente, usan UUCP sobre TCP para transferir correo y news desde su proveedor de servicio. UUCP permite que el proveedor de servicio recolecte y guarde todo el mail y news del sitio; cuando el sitio levanta su conexión Internet y se contacta al proveedor de servicio usando UUCP, el proveedor de servicio transfiere inmediatamente todo el correo y news acumulado.

Una de las características agradables de UUCP es que, una vez que se estableció una conexión entre dos sitios, esa conexión se usa para transferir todos los datos pendientes entre los dos sitios, sin observar cuál sitio inició la conexión. Esto significa que se podrían permitir sólo las conexiones UUCP de salida, y hacer un poll al proveedor de servicio sobre una base regular (por ejemplo, una vez por hora) para tanto enviar los mensajes de salida como recolectar los mensajes entrantes.

Muchos sitios ya no usan UUCP. Si no se tiene una necesidad especial, simplemente se lo puede bloquear a través del firewall.

Ya que UUCP usa passwords reusables, alguien que haya hecho snooping sobre una de las sesiones UUCP podría conectarse después al proveedor de servicio en lugar del verdadero usuario, o hacerse pasar por el proveedor de servicio al usuario. Las passwords generalmente se guardan no-encriptadas en los archivos de configuración de la máquina llamante, de modo que esto se puede hacer aún sin hacer primero snooping sobre la conexión.

3. Terminal Access (Telnet)

Telnet permite que un usuario remoto acceda al shell de comandos sobre otra computadora. Está soportado por muchas plataformas en la Internet, aún en sistemas MS-DOS y Microsoft Windows (el cual provee acceso a un shell de DOS vía un servidor Telnet). La

principal excepción es el Sistema Operativo Macintosh, el cual no tiene un shell orientado a comando de línea para darle acceso a los usuarios.

Aunque el uso más común de Telnet es el acceso a terminal remota, muchos clientes Telnet soportan la especificación de números de port arbitrario para acceder a servicios TCP basados-en-texto en otros ports. Esto es útil si se tiene un servicio para el cual no se quiere distribuir un cliente dedicado; por ejemplo, se usa para dar acceso a MUDs (Multi-User Domains) y MOOs (Multi-User Domains, Object Oriented), que son entornos multiusuario para juegos, entornos de trabajo colaborativo, o áreas chat. Los clientes Telnet también se usan bastante frecuentemente para protocolos de debugging que normalmente se acceden por clientes dedicados. Por ejemplo, la gente chequeará servidores SMTP o verificará los nombres de usuario usando *telnet hostname 25* para conectarse directamente al servidor SMTP sobre el port 25 y ahí tipeará comandos SMTP. Es importante entender que, aunque se puede estar usando un programa llamado *Telnet* para estos propósitos, todo lo que se está haciendo es abrir una conexión TCP simple al número de port especificado. El programa *telnet* no usa el protocolo Telnet (el cual provee cosas del tipo negociación de opción entre cliente y servidor, modos línea-por-vez y carácter-por-vez, etc.) a menos que se esté hablando al servidor sobre el port Telnet estándar (port 23). Acá se discute sólo el uso de los clientes Telnet que acceden a servidores Telnet.

Los Telnet entrante y saliente tienen implicaciones de seguridad muy diferentes. Muchos sitios quieren permitir que sus usuarios accedan al servicio Telnet saliente, de modo que sus usuarios puedan obtener los shell de comandos y servicios de información provistos vía Telnet sobre sistemas remotos en la Internet. Por otro lado, muchos sitios no quieren permitir (o quieren permitir pero bajo un control muy estricto) el acceso entrante de Telnet en sus sitios.

Obviando si el acceso es entrante o saliente, Telnet es un protocolo de texto plano (como muchos otros). Cualquier información que sus usuarios accedan o provean sobre una sesión Telnet va a ser visible por cualquiera que haga snooping sobre la conexión Telnet. Hay versiones encriptadas de Telnet, pero por ahora ninguna es muy usada.

Los usuarios deberían usar en los hosts externos passwords distintas a las que usan en los hosts internos, porque podrían ser *sniffed.*

4. Ejecucion De Comando Remoto

Existe un conjunto de protocolos primarios para permitir que los usuarios ejecuten comandos sobre sistemas remotos. Esta sección describe los comandos 'r' de BSD.

Comandos 'r' de BSD

Los comandos 'r' BSD (*rsh, rlogin, rcp, rdump, rrestore* y *rdist*) están diseñados para

proveer un acceso remoto conveniente (acceso sin password) a servicios tales como ejecución de comando remoto (rsh), login remoto (rlogin), y copiado remoto de archivo (rcp y rdist).

Estos comandos son extremadamente útiles; son seguros para usar sólo en un entorno en el que todas las máquinas sean más o menos confiables para seguir las reglas. Mientras que puede ser adecuado usar estos servicios dentro de una LAN, casi nunca es apropiado emplearlos a través de la Internet. Puede usarse un proxy para permitir alguno en forma segura, particularmente hacia el exterior.

La dificultad que presentan es que usan autenticación basada-en-dirección. El servidor mira la dirección origen del pedido y decide si confía en el host remoto o no para decirle quién es el usuario (esto está controlado por el archivo /etc/hosts.equiv sobre los sistemas UNIX).

Un atacante que convence a uno de estos servidores de que está viniendo desde una máquina "confiable", esencialmente puede obtener acceso completo e irrestricto al sistema. Puede convencer al servidor fingiendo ser una máquina confiable y usando su dirección IP, confundiendo al DNS de tal modo que DNS crea que la dirección IP del atacante mapea al nombre de una máquina confiable, o por cualquier otra cantidad de métodos.

Si falla el chequeo de host confiable dado más arriba (es decir, si el usuario no viene de un host confiable), muchos de estos servicios simplemente deniegan el pedido del cliente y lo desconectan. El servidor rlogin, sin embargo, le pedirá una password al usuario si es que falló el chequeo de host confiable. La password ingresada se envía como texto plano por la red, igual que en Telnet, por eso hay que preocuparse por los atacantes que capturan passwords haciendo *sniffing.*

5. Network News Transfer Protocol (NNTP)

NNTP es el servicio que generalmente se usa para transferir noticias Usenet a través de la Internet (como dijimos antes, UUCP sobre TCP a veces se usa con este propósito). Un servidor de news es el lugar donde las noticias Usenet fluyen entrando y saliendo de la organización y al cual acceden los usuarios (vía clientes de news) para leer y dar noticias. Hay muchos nuevos servidores disponibles, incluyendo B-News, C-News, e INN, y generalmente hablan NNTP entre ellos de modo que pueden transferir noticias entre los sitios. Además, muchos nuevos clientes usan NNTP para acceder a servidores de news para usuarios leyendo noticias.

Formas Peligrosas de Instalar NNTP en un entorno de Firewall

Puede parecer obvio que el host bastión debería ser el servidor de noticias Usenet del sitio. Sin embargo, hay algunos problemas sutiles que pueden provocarse por tal configuración.

1. Es conveniente dedicar una máquina para el servicio de noticias; news tiene tendencia a absorber todo el espacio disponible en disco y el tiempo de procesamiento, y no

coexiste bien con servicios críticos. Si se usa un host bastión para las noticias, probablemente se necesitará tener múltiples hosts bastiones, lo cual en sí no es algo malo, pero aumenta la sobrecarga de mantenimiento. Puede no ser costoso tener múltiples hosts bastiones como para proveer el servicio de news sobre uno de ellos.

2. Si el servidor de news es el host bastión, no se puede tener ningún grupo de noticias privado o propietario para discusiones internas. No se querrá que esos newsgroups se vean expuestos si se cae al host bastión, o si alguno del exterior consigue conectarse al servidor NNTP y lo convence de que le muestre los newsgroups.

3. Si el host bastión es el servidor de news, tendrá que elegirse una entre cuatro aproximaciones para dejar que los usuarios lean las noticias; cada una de estas aproximaciones, dadas a continuación, tienen sus propios problemas.

- ***Dejar que los usuarios se conecten al host bastión para que lean las noticias.***

  Esta es la aproximación más directa. En el host bastión los usuarios ejecutan un newsreaders. Esto implica dar cuentas sobre el host bastión, lo cual puede comprometer seriamente la seguridad del mismo. Pero esto puede no ser muy popular entre los usuarios, ya que les requerirá usar cualquier newsreader de UNIX que esté disponible sobre el host bastión.

- ***Usar sólo clientes NNTP para leer las noticias.***

  Es un poco más segura que la anterior. Hace que los usuarios usen sólo clientes NNTP para leer las noticias en el host bastión. El principal problema con esta aproximación es que requiere que los usuarios usen newsreaders con la capacidad NNTP. Mientras muchos, tal vez aún la mayoría, tienen la capacidad NNTP, hay algunos que todavía son de uso común (como ciertas versiones de *nn*) que no tienen esa capacidad. Dependiendo del servidor NNTP que se esté usando, puede ser imposible obtener características de un newsreader popular como el paso de artículo a través de NNTP. Pueden tenerse dificultades buscando un servidor que tenga todas las características que se quiere para transferir noticias y dar servicio de newsreaders.

- ***Exportar noticias a los clientes vía NFS.***

  La tercera aproximación a tomar para hacer que el host bastión sea su servidor de news es exportar las noticias a los clientes vía NFS. Esta aproximación resuelve uno de los problemas que hemos identificado: permite usar newsreaders sin la capacidad NNTP.

- ***Confiar las noticias a un servidor de noticias interno a través del host bastión***

  Aquí simplemente se confían las noticias a través del host bastión, y se hace a algún sistema interno el verdadero servidor de news. La forma más obvia de hacerlo es instalando un servidor de news sobre el host bastión que simplemente derive noticias hacia y desde el servidor de news interno. El problema con esta aproximación es que requiere gran cantidad de mantenimiento. Los sistemas de news acoplados estrechamente son muy susceptibles a problemas de cascading. Un problema sobre uno provoca que se detenga el flujo de noticias, deteniendo cosas sobre el otro sistema

y provocando problemas ahí; cuando se arregla el primer problema, el otro sistema fluye en el primero, y todo el proceso empieza de nuevo. Sin embargo, si se tienen administradores de news experimentados y una gran cantidad de espacio en disco, puede que a esta aproximación se la encuentre aceptable, y se la pueda instalar para permitir newsgroups privados.

Formas Convenientes de Instalar NNTP en un Entorno de Firewall

Una de las diferencias claves entre NNTP y otros protocolos, tal como SMTP, está en cómo son usados. Se podrían obtener conexiones SMTP desde todo el mundo, similarmente a como la gente le manda correo electrónico al sitio desde cualquier parte. Por el otro lado, solamente obtendrá conexiones legítimas de NNTP desde su sitio de alimentación de noticias.

La forma más conveniente y confiable de tratar con news sobre un firewall usualmente es convenir que las noticias fluyan directamente entre su(s) sitio(s) de alimentación y su servidor interno de news. Esto puede conseguirse o bien a través de filtrado de paquetes, o a través de un sistema proxy sobre el host bastión, como se describe en las siguientes secciones.

Si se instala NNTP usando o bien filtrado de paquetes o bien un servidor proxy, se reducirá grandemente la vulnerabilidad a los ataques vía NNTP. Con NNTP configurado como describimos acá, las dos condiciones siguientes tendrían que ser verdaderas para ser atacado vía NNTP:

- El servidor NNTP debería tener que tener alguna falla o error de configuración que podría ser explotado.
- El ataque tendría que venir (o parecer que viniera de) uno de los sitios de alimentación, ya que los sitios de alimentación son los únicos sitios que están habilitados para abrir conexiones NNTP al servidor.

NNTP es un protocolo relativamente simple. No hay ataques conocidos que usen NNTP en sí. Hubo un ataque dirigido-a-datos conocido sobre sistemas de news de UNIX, que confiaba en la buena voluntad de algún software de servidor de news para crear newsgroups automáticamente. Era posible especificar nombres de newsgroup con ";" en ellos, tal que cuando el servidor de news iba a crear el grupo, también ejecutaba comandos contenidos después del ";" en el nombre de grupo deformado. Como los mensajes de control fueron deformados, no fueron reconocidos y se propagaron por todos los servidores, pero muchos sitios de cualquier modo fueron inmunes porque no crearon newsgroups automáticamente. Esta falla fue corregida en versiones posteriores de varios paquetes de software de servidor de news. Como conclusión se resaltaba la importancia de permanecer actualizado.

6. World Wide Web (WWW) Y HTTP

La existencia de la World Wide Web es el principal factor del crecimiento explosivo de la Internet. Desde la introducción del paquete NCSA Mosaic (la primer interfase gráfica de usuario a la WWW que ganó amplia aceptación), el tráfico WWW sobre la Internet creció a una tasa explosiva, bastante más rápido que cualquier otro tipo de tráfico (p.ej., e-mail SMTP, archivos de transferencia FTP, sesiones de terminales remotas Telnet, etc.). Ciertamente, se querrá dejar que los usuarios usen un browser para acceder a sitios Internet, y seguramente querrá ejecutar un sitio propio, si hace algo que podría beneficiarse con publicidad.

Muchos browsers WWW son capaces de usar protocolos distintos, HTTP el cual es el protocolo básico de la Web. Por ejemplo, esos browsers usualmente también son clientes Gopher o FTP. Muchos también son clientes NNTP, SMTP y Archie. Usan una notación única, consistente, llamada Ubicación de Recurso Uniforme (URL - Uniform Resource Locator) para especificar conexiones de varios tipos.

Temas de Seguridad en HTTP

Los programas externos ejecutados por los servidores HTTP generalmente son shell scripts escritos por usuarios que tienen información sobre la que quieren proveer acceso, pero que saben poco o nada de shell scripts seguros (lo cual no es trivial ni para un experto). Lo mejor que se puede hacer es tratar de proveer un entorno seguro en donde puedan ejecutarse los scripts. No debería haber nada en el entorno que pudiera preocuparlo si se llega a revelar al mundo. Nada debería confiarse a la máquina sobre la que se está corriendo el servidor. Si se instala el entorno de esta manera, aún si de alguna manera los atacantes consiguen quebrar este ambiente restringido y obtienen un acceso total a la máquina, no podrán ir muy lejos para quebrar la parte realmente interesante de su red interna.

Los problemas de seguridad de los clientes HTTP son bastante más complejos que los de los servidores HTTP. La base de estos problemas cliente es que los clientes HTTP (como Mosaic y Netscape Navigator) generalmente están diseñados para ser extensibles y ejecutar programas externos particulares para tratar con tipos de datos particulares. Esta extensibilidad puede ser abusada por un atacante. Los servidores HTTP pueden proveer datos en cualquier formato: archivos de texto plano, archivos HTML, documentos PostScript, archivos "still" video (GIF y JPEG), archivos de movie (MPEG), archivos de audio, etc. Estos servidores usan MIME, para formatear los datos y especificar su tipo. Los clientes HTTP generalmente no intentan entender y procesar todos estos diferentes formatos de dato. Entienden unos pocos (tales como HTML, texto plano y GIF), y confían en programas externos para tratar con el resto. Estos programas externos harán lo que sea apropiado para el formato. Por ejemplo, los browsers UNIX de la Web confrontados con un archivo PostScript comúnmente invocarán al programa GhostScript, y los browsers UNIX de Web frente a un archivo JPEG comúnmente invocarán al programa *xv*. El usuario controla (generalmente vía un archivo de configuración) qué tipo de datos conoce el cliente HTTP, qué programa invocar para cada tipo de dato, y qué argumento

pasar a esos programas. Si el usuario no proveyó su propio archivo de configuración, el cliente HTTP generalmente usa un default interno o uno de todo el sistema.

Todos estos programas externos presentan dos temas de seguridad:

- ¿Cuáles son las capacidades inherentes de los programas externos de las que podría aprovecharse un atacante?
- ¿Qué nuevos programas (o nuevos argumentos para los programas existentes) podrían agregarse a su configuración por un atacante?

¿Qué se puede hacer?

Debido a que los clientes Mac y PC parecen menos susceptibles a algunos de los problemas del lado cliente, algunos sitios toman la decisión de permitir acceso WWW sólo desde las Macs o PCs. Otros van aún más lejos y limitan el acceso a máquinas particulares (generalmente ubicadas en lugares fácilmente accesibles como bibliotecas o cafeterías) que fueron configuradas cuidadosamente como para que no tengan información sensitiva La idea es: si ocurre algo malo, afectará solamente a esta máquina que se reconstruye fácilmente. La máquina no puede usarse para acceder a datos de la compañía en otras máquinas.

Hay otra complicación de los clientes WWW en entornos en los que el filtrado de paquetes es parte de la solución firewall: no todos los servidores HTTP corren en el port 80. Para solucionar esto, habría que considerar usar servidores proxy para acceso HTTP. Si se hace esto, los clientes internos hablan sobre los ports estándar a través del sistema de filtrado de paquetes al servidor proxy, y el servidor proxy habla sobre ports arbitrarios (porque está fuera del sistema de filtrado de paquetes) al servidor real.

HTTP Seguro

Pueden escucharse discusiones de HTTP Seguro y preguntarse cómo se relaciona con los firewalls y con la configuración de los servicios. El HTTP Seguro no está diseñado para resolver los tipos de problemas que estuvimos discutiendo en esta sección. Está diseñado para tratar con temas de privacidad encriptando la información que pasa vía HTTP. Un mecanismo del tipo HTTP Seguro es necesario para poder hacer negocios usando HTTP de tal modo que cosas como números de tarjeta de crédito puedan pasarse a través de la Internet sin miedo de que sean capturados por sniffers de paquetes.

Se pueden proveer autenticación y a la vez encriptado con el HTTP Seguro. Si nos conectamos a sitios conocidos, que corren HTTP Seguro, y que autentican, podríamos asegurar que no se está hablando con un sitio hostil. Sin embargo, aún cuando el HTTP Seguro está liberado y en amplio uso, no resulta ser una aproximación popular y práctica.

No se podría decir que HTTP sea inseguro; el hecho es que HTTP está transfiriendo programas en otros lenguajes. Por esta razón se está trabajando sobre protocolos del tipo HTTP que sean seguros.

7. Otros Servicios De Información

Además de los servicios de la World Wide Web descritos en la sección previa, también hay otros servicios de información populares, incluyendo Gopher, WAIS y Archie.

Gopher

Gopher es una herramienta que maneja menúes basada en texto que sirve para mirar archivos y directorios por la Internet. Cuando un usuario selecciona un ítem de menú Gopher, recupera el archivo especificado y lo muestra apropiadamente. Esto significa que si un archivo está comprimido, automáticamente Gopher lo descomprime; si es una imagen GIF, automáticamente corre un visor GIF.

Los asuntos de seguridad son esencialmente los mismos - desde el punto de vista cliente y servidor - que los descritos en la sección precedente para clientes y servidores HTTP.

Muchos de los browsers WWW comunes tales como Mosaic y Netscape Navigator son clientes Gopher como también clientes HTTP. Estos browsers generalmente soportan el proxying de Gopher vía SOCKS o vía servidores proxy HTTP tales como el servidor HTTP CERN. Aún si no puede encontrar un cliente Gopher dedicado que haga proxying probablemente puede usar en su lugar uno de estos browsers WWW (y un servidor proxy apropiado). Sus usuarios pueden preferir esta aproximación a usar un cliente Gopher separado, ya que significa que tendrán una sola aplicación - un cliente para HTTP, Gopher, y varios otros protocolos - para aprender y configurar, antes que una aplicación separada para cada protocolo.

Wide Area Information Servers (Wais)

WAIS indexa grandes bases de datos de texto tales que puedan ser consultadas eficientemente por palabras claves simples o expresiones booleanas más complicadas.

Los servidores WAIS presentan los mismos problemas de seguridad que los otros servicios. ¿ Un atacante puede usar este servidor para acceder a algo que no debería?

Se atiende este problema igual que con los otros servidores: Asegurar y restringir el entorno en donde se ejecuta el servidor. De esta manera, se asegura que aún si el atacante sale del servidor, no hay nada más de interés que pueda encontrarse en la máquina.

Los clientes WAIS generalmente son programas aislados. Ayudan a encontrar los servidores WAIS, envían consultas a esos servidores, muestran los resultados, envían nuevas consultas basadas en los resultados previos, etc. La información WAIS está generalmente basada en texto, por lo que los clientes WAIS no tienen los problemas de los clientes HTTP y Gopher cuidando la seguridad de los datos que recuperan.

Archie

Archie es un servicio Internet que deja que los usuarios busquen cadenas o expresiones regulares a través de índices de servidores FTP anónimos que coincidan con nombres de archivo y directorio en los archivos FTP. Archie continuamente hace poll de servidores públicos de FTP anónimo para mantenerse actualizado con lo que está disponible en los sitios.

Hay varias formas de dejar que los usuarios accedan a los servidores Archie, incluyendo Telnet, e-mail, y gateways WWW, y el protocolo Archie dedicado. Los sitios corriendo un servidor Archie prefieren generalmente que la gente los acceda vía el protocolo Archie dedicado ya que es más eficiente para ellos y les permite atender a un mayor número de usuarios.

Acceso Archie vía Telnet: Si se permite Telnet saliente, los servidores Archie pueden accederse abriendo una sesión Telnet a la máquina y logoneándose como "archie" (sin password). Esto le dará acceso a un programa de consultas Archie orientado al comando de línea. Ahora puede o bien capturar un log de la sesión Telnet para salvar sus resultados o decirle al servidor que envíe por e-mail los resultados obtenidos.

Acceso Archie vía e-mail: Generalmente, los servidores Archie pueden también accederse vía e-mail. Si se envía un mensaje e-mail a "archie" a uno de los sitios Archie, éste tratará el mail como una consulta a procesarse y devolverá los resultados vía e-mail.

Acceso Archie vía filtrado de paquetes y el protocolo Archie dedicado: Los servidores Archie pueden accederse directamente con clientes Archie dedicados. El problema es que Archie está basado en UDP, y muchos sitios bloquean todo el tráfico UDP salvo un conjunto restringido de paquetes UDP que pueden pasar por el firewall para evitar problemas de seguridad con servicios basados en RPC como NFS y NIS/YP.

Lo que hace posible proveer acceso Archie directo vía filtrado de paquetes es que hay muy pocos servidores Archie. Muchos sitios se consideran suficientemente seguros como para abrir un boquete en el filtrado de paquete para Archie. Hay varias razones para esto:

- La lista de servidores es chica y bastante estable.
- Para ser atacado a través de este boquete, el ataque tendría que venir desde uno de estos sitios. Dado que todos estos son sitios conocidos y muy fuertemente usados, cualquier break-in va a ser notado rápidamente, particularmente si quiebra al servidor Archie para usar su port para atacar otro sitio. Hasta la adulteración de paquetes va a llevar a situaciones de fuera de servicio que se detectarán rápidamente.
- Aún si alguien quebró uno de los servidores Archie, ¿cuáles son las posibilidades de que el atacante podría seguir sobre nuestro sitio en particular?

También se puede decidir que se es capaz de dar una oportunidad y abrir suficientemente el filtrado de paquetes para permitir que los usuarios accedan a los principales servidores Archie en el mundo, o al menos los servidores por defecto que se configuraron en el cliente que se está distribuyendo.

8. Servicios De Búsqueda De Información

Los servicios *finger* y *whois* miran información referente a sitios y usuarios sobre la Internet. Son similares a los servicios reales de búsqueda de información acerca de gente.

Finger

El servicio *finger* mira información relativa a los usuarios. Esta información puede incluir el nombre real de la persona, nombre de usuario, e información de cuando se logoneó más recientemente y desde dónde. También se puede usar para mostrar la lista de los usuarios conectados actualmente a un host. Se lo diseñó para permitir que la gente se encuentre, pero da más información de la que uno probablemente quisiera que estuviera disponible.

Se recomienda que se limiten los pedidos *finger* entrantes a un host bastión, y que ejecute un servidor *finger* reemplazante sobre ese host.

Los pedidos *finger* salientes son también medianamente problemáticos. Porque son posibles los ataques *dirigidos por datos.*

Whois

whois es otro protocolo para mirar información, similar al *finger*. Comúnmente se lo usa para obtener información pública acerca de hosts, redes, dominios, y de la gente que los maneja desde varios Centros de Información de Red (NICs, tales como *rs.internic.net*). Los sitios en general no proveen su propio servidor *whois*; simplemente acceden a los servidores *whois* en los NICs.

Los datos disponibles vía whois no necesariamente son de mucho interés para los usuarios normales.

Por el otro lado, no se conocen problemas de seguridad con los clientes *whois*, y cualquiera que ocurrió tendría que ser *dirigido por datos.*

9. Servicios De Conferencia En Tiempo Real

Hay varios servicios disponibles en la Internet que permiten que la gente interactúe en tiempo real sobre la Internet, incluyendo *talk*, IRC, y servicios varios provistos sobre MBONE.

Talk

Talk es un sistema de conferencia entre dos usuarios basado en texto en tiempo real;

permite que dos personas establezcan entre sí una sesión 'chat'. Cada una de sus pantallas se divide en dos secciones; lo que una persona tipea aparece en la otra sección.

Talk usa UDP para negociar las conexiones entre los dos sitios y luego usa TCP para mover en realidad los datos de vuelta y hacia los participantes. UDP se usa entre el cliente llamante y el servidor que responde, y otra vez entre el cliente que responde y el servidor llamante; luego TCP se usa entre los dos clientes.

Debido a la incompatibilidad de los protocolos, *talk* falla frecuentemente aún entre sitios que no hacen filtrado de paquetes, o entre máquinas de diferentes tipos dentro del mismo sitio. Los clientes y servidores *talk* generalmente se proveen sólo sobre máquinas UNIX.

Internet Relay Chat (IRC)

IRC es un sistema de conferencia multiusuario basado en texto en tiempo real. Los usuarios corren programas clientes IRC para conectarse a servidores IRC. Los servidores IRC se distribuyen en un "spanning tree" (árbol ligado), y hablan unos con otros para pasar mensajes a todos los clientes.

Muchos de los problemas de seguridad con IRC están en relación con quién lo usa y cómo, no al protocolo en sí. Más, algunos de los usuarios frecuentes de IRC tienen el hábito desagradable de persuadir a los nuevos usuarios para que inocentemente ejecuten comandos que no hacen cosas claras y dejan basura en los sistemas. Aunque existen problemas sobre IRC estos pueden ser manejables.

Se debería ser capaz de correr seguramente un servidor IRC en un entorno restringido sobre un host bastión, pero sería algo más grotesco ejecutar al servidor sin tener clientes locales que lo pudieran acceder, y un servidor que pudiera acceder a la Internet probablemente no sería seguro para que le hablen los clientes. Se puede querer correr uno dentro del firewall para conferencias IRC privadas.

Varios clientes IRC soportan algo llamado Direct Client Connections (DCC — Conexiones de Cliente Directas). DCC permite que dos clientes IRC negocien y establezcan una conexión TCP directa entre ellos, salteando a todos los servidores excepto para la negociación inicial.

El Backbone Multicast (MBONE)

Muchas tecnologías de red proveen un mecanismo para que una estación de envío dirija un mensaje a una estación de recepción en particular que se conoce como *unicasting*, y también proveen otro método para que una estación de envío dirija un mensaje a todas las posibles estaciones receptoras llamado *broadcasting*. Algunas tecnologías de red también proporcionan algo intermedio para que una estación de envío dirija un mensaje a un conjunto particular de estaciones receptoras sin mandarlo a todas las estaciones: esto se llama *multicasting*.

El multicasting es particularmente útil cuando se está tratando con aplicaciones de un alto ancho de banda como conferencias de audio y video. Con tales aplicaciones se puede tener a un grupo de estaciones recibiendo la misma corriente de paquetes, y la corriente puede consumir una significante fracción del ancho de banda de red disponible. Si una corriente dada consume un 10% del ancho de banda disponible de la red (que no es poco común), no se querrá hacer unicast con cada host interesado porque cada uno de estos consumiría otro 10% del ancho de banda, limitándonos a 10 host participantes, suponiendo siempre que no se hace otra cosa con la red. Tampoco se buscará hacer broadcast a todos los hosts a menos que todos (o casi todos) estuvieran realmente interesados en esa corriente, ya que ubica una carga significativa sobre cada host para procesar un paquete broadcast y después decidir ignorarlo.

El multicasting IP provee una forma de enviar paquetes a grupos de hosts IP. Con multicasting IP, a los grupos de hosts que desean recibir un tipo particular de paquete (p.ej. una corriente de video en particular) se les asigna una dirección multicast IP que todos comparten. Una dirección multicast IP se ve como una dirección IP normal en el rango 224.0.0.0 hasta 239.255.255.255. Las direcciones de multicast IP solamente se usan como direcciones destino; nunca son válidas como direcciones origen (los paquetes multicast usan la dirección IP regular del host enviante como su dirección origen).

Los grupos multicast son algo así como los canales de televisión de cable. Hay una amplia variedad de canales (grupos multicast) disponibles, tales como HBO, CNN, ESPN, y MTV, pero muchos hogares (hosts) se suscriben sólo a unos pocos de los canales. Algunos grupos multicast son permanentes; esto es, ciertas direcciones son reservadas para ciertos usos, tales como encuentros de la Internet Engineering Task Force (IETF), mantenimiento de videos seleccionados de la NASA, etc. Otros grupos multicast son transitorios: instalados para un propósito o evento particular y se cierran cuando ya no son necesarios, para ser reusados por otra cosa más adelante.

Hoy en día, el multicasting se usa en la Internet básicamente para servicios de conferencia en tiempo real, incluyendo servicios de video, audio, y pizarra electrónica. Además está empezando a usarse para otros servicios, tales como la transmisión eficiente de noticias de Usenet.

Los productos comerciales, tales como hosts y routers, están empezando a soportar multicasting. Algunas tecnologías de red tales como Ethernet soportan multicasting directamente. Se puede enviar un paquete Ethernet de información a una dirección Ethernet mágica sobre un segmento Ethernet dado, y todos los hosts sobre ese segmento que están escuchando esa dirección Ethernet recibirán ese paquete. Otras tecnologías de red tales como líneas contratadas punto a punto, no soportan multicast, de modo que el efecto tiene que ser el de convertir un paquete multicast en una serie de paquetes unicast duplicados, cada uno direccionado a uno de los participantes multicast.

Obviamente, se quiere limitar el número de paquetes unicast duplicados sobre estas líneas contratadas punto a punto, de modo que una solución común para enlazar dos redes con capacidad de multicast (tales como Ethernet) sobre una red sólo unicast (tal como la línea

T1) es crear un *túnel* sobre la red unicast, con routers multicast (llamados *mrouters*) en cada extremo del túnel. Estos mrouters toman paquetes IP multicast, los encapsulan en paquetes IP unicast, y los envían (vía IP unicast regular) a través del túnel al *mrouter* del otro extremo, el cual lo desencapsula para volverlo paquetes IP multicast. Al crear una red de *mrouters* y túneles, se puede crear una red multicast virtual sobre el tope de un backbone unicast.

El MBONE es el Backbone Multicast ad hoc sobre la Internet, y es la red anterior de *mrouters* y túneles. Sus participantes son los sitios que están interesados en usar multicasting IP por diferentes servicios en la Internet.

El multicasting IP levanta varios asuntos de firewalls. Si un sitio usa tunneling para tomar parte en el MBONE, ¿cómo se ven los paquetes para los túneles? ¿Qué podría enviarse a través de los túneles? Si un sitio no usa túneles, pero usa multicasting directamente, ¿cómo lo va a tratar el sistema de filtrado de paquetes del sitio? ¿Los atacantes pueden acceder a servicios no-multicasting (tales como SMTP, NFS, NIS/YP, etc.) vía multicast, estuviera o no el túnel?

Comúnmente el tunneling de multicast IP está hecho con encapsulación de IP-sobre-IP. Esto es, un paquete multicast IP se encapsula en un paquete IP común, de la misma forma que se hace con un paquete TCP o UDP. En vez del paquete usual IP que contiene por ejemplo un paquete UDP, se tiene un paquete IP unicast que contiene un paquete IP multicast que, en cambio, contiene un paquete UDP. Por eso, si los paquetes multicast tunneled necesitan cruzar un sistema de filtrado de paquetes, el sistema necesita reconocer paquetes IP-sobre-IP, de la misma forma que reconoce paquetes TCP, UDP, o ICMP. Si el sistema de filtrado de paquetes no reconoce IP-sobre-IP por nombre, pero en cambio permitirá que se especifiquen protocolos por número, se necesita saber que IP-sobre-IP es el protocolo número 4 (por comparación, ICMP es 1, TCP es 6, y UDP es 17).

Los túneles multicast IP usualmente se hacen con paquetes IP ruteados de origen, pero esta práctica provocó varios problemas y ya no es recomendada.

Para prevenir que un túnel multicast se use como una puerta trasera de entrada o salida de una red, el código corriente públicamente disponible del *mrouter* solamente aceptará paquetes multicast a través del túnel; no aceptará paquetes unicast empujados a través del túnel en un intento de saltear al firewall. Si se está usando un router multicast comercial, antes que el código públicamente disponible fuera de la Internet, debería verificarse que se comportará en una forma similar.

Si se tienen routers y una topología de red que soporta multicast directamente, sin túneles, todavía habrá que preocuparse de cómo cualquier sistema de filtrado de paquetes que se use va a resolver esto. No debería ser demasiado difícil, ya que desde un punto de vista de filtrado de paquetes, los paquetes multicast se ven como paquetes comunes con algunas direcciones destino inusuales (en el rango 224.0.0.0 a 239.255.255.255). Tratarlos como se haría con cualquier cosa: bloquearlos a todos por defecto y permitir los que se entienden y se quieren soportar. Recuerde que cada una de estas direcciones multicast van a aplicarse a múltiples máquinas internas, y que si se están aceptando paquetes multicast desde el exterior,

entonces todas las máquinas internas que están aceptando esos paquetes tendrán que ser protegidas contra ataques del mundo exterior - como si se estuviera aceptando cualquier otro paquete directamente desde el mundo exterior.

Si el túnel está restringido sólo a paquetes multicast, o si se está usando multicast directamente sin tunneling, todavía está la pregunta de cómo los hosts responderán a los paquetes multicast direccionados a ports regulares, tales como ports NIS/YP o NFS. Desafortunadamente, el comportamiento varía de sistema operativo a sistema operativo, y aún de versión a versión dentro del mismo sistema operativo.

10. Sistema De Nombres De Dominio (DNS)

El DNS es un sistema de base de datos distribuida que traduce nombres de host a direcciones IP y direcciones IP a nombres de host. También es el mecanismo estándar de Internet para almacenar y acceder a otros tipos de información sobre los host, y proporciona información acerca de un host en particular al resto del mundo. Por ejemplo, si un host no puede recibir mail directamente pero otra maquina puede hacerlo en su lugar y pasárselo, esta información es comunicada con un registro MX en DNS.

Los clientes DNS incluyen cualquier programa que necesite hacer algunas de las siguientes cosas:

- Traducir un nombre de host a una dirección IP.
- Traducir una dirección IP a un nombre de host.
- Obtener otra información publicada acerca de un host (tal como su registro MX).

Fundamentalmente, cualquier programa que use nombres de host puede ser un cliente DNS. Esto comprende esencialmente a todo programa que tenga algo que ver con el trabajo en red, incluyendo tanto programas clientes y servidores para Telnet, SMTP, FTP y cualquier otro servicio de red. DNS es por lo tanto un servicio fundamental de red sobre el cual se basan otros servicios de red.

Pueden usarse otros protocolos para dar este tipo de información. Por ejemplo, NIS/YP se usa para dar información de host dentro de una red. Sin embargo, DNS es el servicio utilizado para este propósito a través de la Internet, y los clientes que necesitan acceder a hosts de Internet tendrán que usar DNS en forma directa o indirecta. En redes que utilizan NIS/YP u otros métodos en forma interna, el servidor para el otro protocolo generalmente actúa como un proxy DNS para el cliente. Muchos clientes también se pueden configurar para usar múltiples servicios, de modo tal que si falla la búsqueda de un host, lo volverá a intentar con otro método. Así, podría comenzar buscando en NIS/YP, el cual mostrará solamente host locales; si esto falla intenta con DNS, o podría comenzar buscando con DNS y si esto falla entonces probar con un archivo de su propio disco (de modo tal que se puedan incluir nombres personales favoritos, por ejemplo).

El DNS esta diseñado para despachar preguntas y respuestas entre clientes y

servidores, por lo que los servidores pueden actuar en nombre de los clientes o de otros servidores. Esta capacidad es muy importante en el momento de construir un firewall que maneja servicios DNS en forma segura.

Un DNS funciona de la siguiente manera: cuando un cliente necesita alguna información en especial, por ejemplo la dirección IP de un host, le pregunta a su servidor DNS local para obtener esa información. El servidor DNS local primero examina su propio cache para ver si ya conoce la respuesta a la pregunta del cliente. Si no la conoce, el servidor DNS local pregunta a otros servidores DNS para resolver la pregunta de su cliente. Cuando el servidor local obtiene la respuesta, o decide que no puede obtenerla por alguna razón, éste cachea la información obtenida y le responde al cliente.

Estas preguntas y respuestas son transparentes para el cliente. Para el cliente, éste se ha comunicado solamente con el servidor local; no sabe si el servidor local tuvo que contactarse con otros servidores para obtener la respuesta.

Datos DNS

El DNS es una base de datos estructurada en forma de árbol, con servidores para varios subárboles distribuidos a través de la Internet. Hay varios tipos de registro definidos en el árbol, tales como:

Tipo de registro	*Uso*
A	Traduce nombre de host a dirección IP
PTR	Traduce dirección IP a Nombre de host
CNAME	Traduce alias de host a nombre de host (nombre canónico)
HINFO	Da información del hardware y software acerca de un host
NS	Delega una zona del árbol DNS a otro servidor
SOA	Indica el inicio de autoridad para una zona del árbol DNS
TXT	Registros de texto no estructurados.

Hay dos árboles de datos DNS separados: uno para obtener información por nombre de host (tal como dirección IP, registros CNAME, registros HINFO, o registros TXT que corresponden a nombres de host dados) y otro para obtener información por dirección IP (el nombre de host para una dirección dada).

Problemas de Seguridad en DNS

Hay algunos problemas de seguridad que se describen a continuación:

⇒ Respuestas falsas a las preguntas DNS

El primer problema de seguridad con el DNS es que muchos clientes y servidores DNS pueden ser engañados por un atacante al crear información falsa. Muchos clientes y servidores no verifican si todas las respuestas que obtienen se relacionan con las preguntas que en realidad hicieron, o si las respuestas que obtienen están viniendo del servidor al que se las mandaron. Los servidores, en particular, pueden almacenar estas respuestas extras sin pensar en ellas, y más tarde responder preguntas con estos datos falsos almacenados. Esta falta de chequeo puede permitir a un atacante darle

datos falsos a los clientes y servidores. Por ejemplo, podría usar esta capacidad para cargar el cache del servidor con información que dice que su dirección IP mapea hacia el hostname de un host en el cual confía para un acceso sin password vía rlogin (Esta es una de la varias razones por la cual no debería permitir que los comandos BSD "r" crucen su firewall).

NOTA: Las nuevas versiones de DNS para UNIX (BIND 4.9 y posteriores) chequean estas preguntas falsas y son menos susceptibles a estos problemas. Las primeras versiones y clientes y servidores DNS para otras plataformas pueden todavía tener este problema.

⇒ Datos que se corresponden mal o no concuerdan entre los árboles de las direcciones IP y de los nombres de host en DNS

El ataque descrito más arriba señala el problema de la falta de correspondencia de datos entre los árboles del hostname y de las direcciones IP en el DNS. En un caso como el anterior, si se busca el nombre de host correspondiente a la dirección del atacante (esto se llama *búsqueda inversa*) se obtiene el nombre de un host en el cual se confía. Si luego se busca la dirección IP de este nombre de host (lo que se llama *doble búsqueda inversa*), debería verse que la dirección IP no concuerda con la que el atacante estaba usando. Esto debería ser una alerta de que algo sospechoso está sucediendo.

Cualquier programa que toma decisiones de autenticación o autorización basándose en información de nombres de host que obtiene del DNS debería ser muy cuidadoso para validar los datos con este método de búsqueda inversa e inversa doble. En algunos sistemas operativos esta verificación se hace en forma automática a través de la función de gethostbyaddr() (por ejemplo en SunOS 4.x y superior). En la mayoría de los sistemas operativos se tiene que hacer esta verificación por uno mismo. Habría que estar seguro de conocer cómo trabaja el SO y de que los daemons que están tomando tales decisiones en el sistema hagan las validaciones apropiadas. Es más, conviene no hacer autenticación o autorización basada solamente en el nombre del host o en la dirección IP porque no hay forma de estar seguro de que un paquete provenga de la dirección IP de la que dice venir, a menos que haya alguna clase de autenticación criptográfica dentro del paquete que solamente puede ser generada por el origen verdadero.

Algunas implementaciones de búsqueda inversa doble fallan en hosts con direcciones múltiples; por ejemplo en un host dual-homed usado para proxy. Si ambas direcciones son registradas con el mismo nombre, una búsqueda DNS por nombre retornará a las dos, pero muchos programas solamente leerán la primera. Si sucede que la conexión proviene de la segunda dirección, la búsqueda inversa doble fallará incluso si el host está correctamente registrado. Aunque debería evitar usar implementaciones de búsqueda inversa doble que tengan estas fallas, también se trata de asegurar que en los host multi-homed visibles desde el exterior las búsquedas por dirección devuelvan un nombre diferente para cada dirección, y que estos nombres solamente tengan una

dirección de retorno cuando se realizan búsquedas.

⇒ Revelación de demasiada información a los atacantes.

Otro problema que puede encontrarse al soportar DNS con un firewall es que se puede hacer visible cierta información que no se desea revelar. Algunas organizaciones ven a los nombres de host internos (así como otra información acerca de los host internos) como información confidencial. Por eso se desean proteger estos nombres porque pueden revelar nombres de proyectos o el tipo de host, lo cual puede hacer más fácil el ataque.

Hasta la información de nombre de host más simple puede ser útil para un atacante que desea burlarse. El uso de la información de este modo es un ejemplo de lo que normalmente se llama un *ataque de ingeniería social*. El atacante primero examina los datos del DNS para determinar el nombre de uno o varios host clave. Esos hosts serán incluidos en una lista como servidores DNS para el dominio, o como gateways MX para otros hosts. Luego el atacante llama o visita su sitio, presentándose como técnico y dice que necesita trabajar en estos hosts, para lo cual solicitará las palabras claves. Los ataques de ingeniería social requieren mucho descaro del atacante pero suceden muy a menudo.

Además de los nombres de host internos, con frecuencia se coloca otra información dentro del DNS que resulta útil a nivel local pero que convendría que no estuviera al alcance de un atacante. Los registros DNS de recursos, TXT y HINFO, son particularmente reveladores:

- HINFO (registro de información del host): nombra el hardware y la versión de sistema operativo que una máquina está corriendo. Esta información es muy útil para administradores de sistemas y redes pero también le dice a un atacante exactamente qué lista de fallas probar cuando atacan la máquina.
- TXT (registro de información textual): son esencialmente registros de texto cortos, sin formato, usados por diferentes servicios para brindar información diversa.

Los atacantes obtendrán mucha información del DNS de su sitio al contactarse con su servidor DNS y pedir una transferencia de zona como si fueran un servidor secundario de su sitio. Se puede evitar esto con filtrado de paquetes (bloqueando las preguntas DNS basadas en TCP, lo cual desafortunadamente bloqueará más que sólo una transferencia de zona) o a través de la directiva *xfernets* de la implementación corriente de BIND.

Configuración de DNS para ocultar información

Una opción que se adopta al querer reducir los problemas de DNS a nivel de seguridad es configurarlo para revelar únicamente los datos que se quieren dejar disponibles para las consultas de los usuarios. Aprovechando la capacidad de envío de preguntas, se puede dar a los hosts internos una visión irrestricta de los datos de DNS tanto internos como externos mientras que a los host externos se los puede restringir a una visión muy limitada de los datos

DNS internos.

Para lograr este ocultamiento, se pueden seguir los pasos dados a continuación:

⇒ Configuración de un servidor DNS "falso" en un host bastión para ser usado por el mundo exterior

El primer paso para ocultar información DNS hacia el mundo exterior es configurar un servidor DNS falso en un host bastión. Éste será el servidor del dominio que es nombrado por los registros del Servidor de Nombre mantenido por su dominio padre. Si se tienen múltiples servidores con estas características para que el mundo les hable, se hace que el servidor falso sea el primario del conjunto de servidores autorizados y el resto, secundarios.

El servidor falso debe saber todo acerca de la información que su dominio quiere que sea revelada al mundo exterior. Esta información incluye básicamente nombres de host y direcciones IP de los siguientes hosts:

- Las maquinas en su red perimetral.
- Cualquier maquina del mundo exterior que necesite contactar directamente.

Además, necesitará publicar los registros MX para cualquier nombre de dominio o host que se use como parte de las direcciones de e-mail, por ejemplo, para que la gente pueda responder a los mensajes.

También se necesitará publicar información falsa para cualquier máquina que pueda contactar directamente al mundo exterior. Muchos servidores sobre Internet, por ejemplo la mayoría de los servidores FTP anónimos, insisten en conocer el nombre de host de cualquier máquina que se contacte con ellos. En los registros DNS de recurso, los registros A y PTR manejan las búsquedas por nombre y dirección.

Las máquinas que tienen dirección IP y necesitan nombre de host realizan búsquedas inversas. Con una búsqueda inversa, el servidor comienza con la dirección IP remota de la conexión de entrada y busca el nombre de host del cual viene la conexión. Toma la dirección IP, la permuta, y busca el registro PTR para ese nombre, y debería retornar el nombre de host para el host que tiene esa dirección, que el servidor utiliza para sus logs o para otras cosas.

¿Como manejarse con estas búsquedas inversas? Si todo lo que estos servidores quisiesen hacer fuera logonearse, se podría crear simplemente un registro PTR genérico. Este indica que todo un rango de direcciones pertenece a un host desconocido en un dominio particular.

Hay un problema al hacer solamente estas búsquedas. Al validar los datos devueltos por DNS, cada vez son más los servidores que están haciendo una búsqueda inversa doble, y no se conversará con el que realiza la consulta inicial a menos que la búsqueda inversa doble tenga éxito. Esta búsqueda es necesaria para la gente que brinda un servicio donde se necesita algún grado de autenticación del host solicitante

En una búsqueda inversa doble, un cliente DNS:

- Realiza una búsqueda inversa para traducir una dirección IP a un nombre de host.

- Hace una búsqueda común sobre ese nombre de host para determinar su dirección IP nominal.
- Compara esta dirección nominal IP con la dirección IP original.

El servidor falso necesita brindar información falsa consistente para todos los hosts en el dominio cuyas direcciones IP van a ser vistas en el mundo exterior. Para cada dirección IP que se tenga, el servidor falso debe publicar un registro PTR con un nombre de host falso y el registro A correspondiente que mapee el nombre de host falso de regreso a la dirección IP. Cuando se conecta con algún sistema remoto que intenta hacer una búsqueda inversa de su dirección IP para determinar el nombre de host, este sistema obtendrá el nombre de host falso. Si luego el sistema intenta hacer una búsqueda inversa doble para traducir el nombre del host a una dirección IP, obtendrá una dirección IP que concuerda con la dirección IP original y satisface el chequeo de consistencia.

Si esta usando *proxy* para conectar un host interno al mundo exterior, *no* se necesitará configurar información falsa en su host interno; simplemente se necesita poner información en el host que corre el servidor proxy. El mundo externo verá solamente la dirección del servidor proxy.

⇒ Configuración de un servidor DNS real en un sistema interno para que lo usen hosts internos.

Las máquinas internas necesitan usar la información real de DNS respecto de los hosts, no la información falsa presentada para el mundo externo. Esto se hace a través de un servidor estándar configurado sobre algún sistema interno. Las máquinas internas puede desear encontrar información sobre las máquinas externas.

Una forma de lograrlo es brindando acceso a información DNS externa, al configurar al servidor DNS interno para que interrogue a DNS remotos en forma directa para resolver preguntas de clientes internos acerca de hosts externos. Tal configuración requeriría abrir el filtro de paquetes para permitirle al servidor DNS interno hablar con los servidores DNS remotos que podrían estar en cualquier host de Internet. Esto es un problema porque el DNS está basado en UDP, y necesita bloquear todo el UDP para bloquear el acceso externo a los servicios vulnerables basados en RPC, como NFS y NIS/YP.

Afortunadamente, el servidor DNS más común (el programa *named* en UNIX) brinda una solución a este problema: la directiva *forwarders* en el archivo de configuración del servidor /etc/named.boot.

El uso de mecanismo *forwarders* en realidad no tiene nada que ver con el ocultamiento de la información en el servidor DNS interno; sí tiene que ver con hacer el filtro de paquete lo más estricto posible, es decir aplicar el **principio de menor privilegio**, haciendo esto de modo tal que el servidor DNS interno solamente necesite estar habilitado para hablar con el servidor DNS del host bastión y no con los servidores DNS en toda la Internet.

⇒ Consultas de los clientes DNS internos al servidor interno

El próximo paso es configurar el cliente DNS interno para que haga todas las preguntas del servidor interno. En sistemas UNIX esto se hace a través de los archivos */etc/resolv/conf*.

Hay dos casos:

- Cuando el servidor interno recibe una consulta referente a un sistema interno, o sobre un sistema externo que está en su cache, responde directa e inmediatamente porque ya conoce la respuesta a tales preguntas.
- Cuando el servidor interno recibe una consulta referente a un sistema externo que no está en su cache, el servidor interno envía esta consulta al servidor del host bastión. Una vez que éste obtiene la respuesta desde el servidor DNS apropiado de Internet, se la devuelve al servidor interno. Entonces, el servidor interno responde al cliente original y la guarda en el cache.

⇒ Los clientes DNS del bastión también consultan al servidor interno

La clave para toda esta configuración de ocultamiento de la información es que los clientes DNS en el host bastión deben pedir información al servidor interno, no al servidor en el host bastión. De esta manera, los clientes DNS en el host bastión pueden usar los nombres de hosts reales y lo mismo para hosts internos, pero los clientes externos no pueden acceder a los datos internos.

⇒ Qué necesita permitir el sistema de filtrado de paquetes

De modo de que este esquema de forwarding de DNS funcione, cualquier sistema de filtro de paquetes entre el host bastión y el sistema interno tiene que permitir lo siguiente:

- Preguntas DNS desde el servidor interno al host bastión: paquetes UDP desde el port 53 en el servidor interno al port 53 en host bastión y paquetes TCP desde ports mayores a 1023 en el servidor interno al port 53 en el host bastión.
- Respuestas a esas preguntas desde el host bastión al servidor interno: paquetes UDP desde el port 53 en el host bastión al port 53 en el servidor interno y paquetes TCP - con el seteo del bit ACK - desde el port 53 en el host bastión a ports mayores a 1023 en el servidor interno.
- Preguntas DNS desde clientes DNS del host bastión a servicios internos: paquetes UDP y TCP desde el ports mayores a 1023 en el host bastión al port 53 en el servidor interno.
- Respuestas desde el servidor interno a los clientes DNS del host bastión: paquetes UDP y TCP - con el seteo del bit ACK - desde el port 53 en el servidor interno a port mayores de 1023 en el host bastión.

Configuración de DNS sin ocultar información

Si no se considera ocultar los datos de su servidor interno al mundo, se deberían seguir teniendo el servidor DNS del host bastión y el servidor DNS interno; sin embargo, uno de éstos puede ser un servidor secundario del otro. Generalmente resulta más fácil hacer al servidor

DNS del bastión uno secundario del servidor DNS interno, y mantener los datos DNS en el servidor interno.

11. Syslog

syslog se usa para manejar mensajes de log en una forma centralizada. Se inició como una forma de registrar mensajes centralizadamente para un conjunto de máquinas UNIX, pero muchos dispositivos de red (routers, hubs, etc.) ahora usan *syslog* para brindar estado e información de uso. Tales dispositivos generalmente no tienen una forma de registrar localmente esta información, ya que no poseen algún medio de almacenamiento para escribirla.

Los atacantes frecuentemente intentarán saturar al servidor *syslog* del sitio de modo de cubrir sus pistas, tal que el servidor llegue a quedarse sin espacio en disco y pare de registrar nuevos mensajes, o tal que la evidencia de sus actividades se pierda en el montón.

12. Servicios De Manejo De Red

A continuación se describen protocolos que se usan por usuarios y por programas para manejar y mantener redes: SNMP, RIP, e ICMP - y, donde es apropiado, herramientas tales como *ping* y *traceroute*.

Simple Network Management Protocol (SNMP)

SNMP es un mecanismo estandarizado de manejo y monitoreo remoto para dispositivos de red como hubs, routers y bridges, así como también para servidores y estaciones de trabajo. La teoría es que cualquier estación capaz de manejar SNMP debería ser capaz de monitorear y controlar cualquier dispositivo de red.

Normalmente, las estaciones de manejo SNMP actúan como clientes contactando servidores SNMP en los distintos dispositivos de red para requerir información o para efectuar comandos. Algunas veces, los dispositivos de red actúan como clientes SNMP para contactar servidores SNMP especiales (conocidos como servidores *trap*) sobre estaciones de manejo para dar información crítica que no puede esperar hasta la próxima vez que la estación de manejo reciba el dispositivo. Los servidores trap SNMP están separados de los servidores comunes SNMP al punto que una máquina dada pueda ejecutar ambos - esto es, tanto un servidor SNMP (y por lo tanto ser manejable vía SNMP) como un servidor trap SNMP (puede ser una estación de manejo y recibir traps desde otros dispositivos).

En general, no se quiere que alguien desde el exterior pueda manejar nuestra red vía SNMP. Por eso, no se debería permitir que SNMP cruce el firewall, y se debería configurar cuidadosamente (o deshabilitar) el SNMP sobre los sistemas que están fuera del firewall para

que los atacantes no puedan usarlo para cambiar esa configuración.

SNMP sí soporta alguna seguridad rudimentaria; cuando se solicita la información, el solicitante necesita especificar la "comunidad" en la que está. Diferentes comunidades pueden mostrar diferente información, y puede requerirse una password reusable para ciertas comunidades. Esto es una seguridad bastante primitiva; cualquiera que esté haciendo *sniffing* de paquetes puede descubrir fácilmente un nombre y password de comunidad.

No todos los dispositivos SNMP soportan aún esto. Afortunadamente, las características más inseguras de SNMP (por ejemplo, permitir que los clientes SNMP seteen parámetros) también son las más complejas de implementar, y por eso las menos frecuentes.

Routing Information Protocol (RIP)

RIP es el protocolo de ruteo más viejo en la Internet. De hecho, es el predecesor de IP. Todavía es el protocolo de ruteo más usado sobre las redes IP locales. Los routers (incluyendo máquinas de propósito general con múltiples interfaces, que pueden actuar como routers) usan RIP para difundir periódicamente a cuáles redes saben cómo llegar, y lo lejos que esas redes están. Con estos datos un router o host puede determinar qué redes son alcanzables y elegir el mejor camino (el más corto) para cada una. Los servidores RIP generalmente sólo difunden esta información cada 30 segundos más o menos para que alguien interesado la escuche, pero un cliente RIP puede solicitar una actualización especial del servidor RIP, con lo cual provoca que el servidor responda directamente al cliente con la información pedida.

Ping

El programa *ping* chequea la conectividad de la red. La aplicación ping genera un paquete ICMP de "echo request". El sistema destino responde con un paquete ICMP "echo response". Típicamente ICMP está implementado en el kernel, de modo que es el kernel el que genera el paquete "echo response"; en casi ningún sistema hay un servidor ICMP separado. (Sobre algunas máquinas, el "echo response" en realidad se genera en la interfase de red en sí, no en el sistema operativo) *ping* no es el único programa que usa "echo ICMP"; otros incluyen *spray* y casi cualquier herramienta dedicada al manejo de red.

ping es una herramienta útil para disparo de problemas de red, y es razonablemente segura. Probablemente se querrá permitir *ping* al exterior desde al menos las máquinas que usa su equipo de operaciones de red y al interior desde al menos las máquinas del centro de operaciones de red de su proveedor de servicio de red.

Como consecuencia de donde está implementado, es casi imposible deshabilitar respuestas a *ping* sobre los hosts individuales; la única forma de controlarlo es con filtrado de paquetes.

Hay dos peligros en permitir "echo ICMP":

- Puede usarse para un ataque de denegación de servicio; esto es, saturar la red. Aunque

con cualquier protocolo que se acepte se puede tener este problema, el echo ICMP es particularmente tentador porque hay programas comúnmente disponibles diseñados para el testeo de la red (incluyendo algunas versiones de *ping*) que dejan saturar las redes con opciones simples de comando de línea.

- Cualquiera que envía pedidos echo ICMP y recibe respuestas echo ICMP desde su red puede descubrir cuántas máquinas se tienen, y en qué direcciones de red están; esto incrementa la eficiencia de cualquier ataque futuro.

Traceroute

traceroute es una aplicación que muestra la ruta que los paquetes toman hacia un destino IP particular. Ya que ningún sistema conoce el camino completo al destino (apenas el siguiente paso hacia el destino). *traceroute* trabaja construyendo cuidadosamente paquetes UDP especiales. La dirección destino de los paquetes es el host remoto; el port destino es un port UDP no-usado sobre el host remoto. Trabaja con el campo "time to live" (TTL) de los paquetes, se lo setea muy bajo (empezando en 1) de modo que los paquetes se rechazarán por los routers intermedios si quedan en loop dentro de la red. Mirando de donde vinieron los rechazos (mensajes ICMP "time to live exceeded"), *traceroute* puede determinar cuáles son los routers intermedios.

TTL normalmente no es de interés desde el punto de vista de un firewall. Cuando se crea un paquete, su campo TTL se setea en algún valor (típicamente 16, 30 o 255). Cada router que maneja el paquete durante su viaje decrementa en 1 el valor de TTL. Si TTL llega a cero, se asume que el paquete está en alguna clase de loop; se lo encapsula con un mensaje ICMP "time to live exceeded", y se lo devuelve a la dirección origen.

Por eso, el primer router que maneja el primer paquete construido especialmente por *traceroute* decrementará el campo TTL (lo deja en 0), y devuelve un mensaje ICMP "time to live exceeded", diciéndole a *traceroute* la dirección del primer router (el origen del mensaje ICMP).

Luego *traceroute* construye otro paquete UDP, esta vez con TTL en 2, y lo envía. Este paquete obtiene el segundo router antes de que TTL llegue a 0, y TTL sabe que el router que devuelve el mensaje ICMP "time to live exceeded" para el paquete es el segundo router en el camino hacia el destino. Con el mismo criterio *traceroute* va construyendo paquetes para determinar el camino al destino. *traceroute* sabe que terminó cuando recibe un mensaje "service unavailable", en lugar de un mensaje ICMP "time to live exceeded" desde algún router intermedio.

Si *traceroute* no puede llegar al host destino (u obtener algo de él), eventualmente terminará por tiempo.

Otros Paquetes ICMP

Hay ciertos tipos de mensaje ICMP que se usan para el manejo de red y que no tienen

programas asociados con ellos. Estos son generados e interpretados automáticamente por varios programas y dispositivos de red.

Qué hacer con los mensajes ICMP depende del mensaje y de la dirección en que está yendo. Los otros tipos de mensaje ICMP que probablemente se quiere permitir, entrantes y salientes, son "source quench" (usado por un receptor para decir al origen que "vaya más despacio" porque está enviando datos muy rápido) y "parameter problem" (que es una clase de código "catch-all" para retornar cuando hay un problema con los headers de paquetes que no puede informarse de otra manera).

Muchos otros tipos de mensajes ICMP tienen el potencial de cambiar la información local sobre los hosts (por ejemplo, 'redirect' provoca cambios a las tablas de ruteo del host), de modo que probablemente no se querrán permitir tales mensajes de entrada a través de los filtros de paquetes.

En general, solamente se querrá permitir ICMP saliente cuando tenga la posibilidad de hacer algo bueno. Tanto "source quench" como "parameter problem" se usan para conseguir que el host origen sea más agradable con nosotros, y están permitiendo salida. Cualquiera de los tipos ICMP que indican que la conexión no se puede hacer ("destination unavailable", "network unavailable", "service unavailable", "destination administratively unavailable", o "network administratively unavailable", por ejemplo) ayudarán a que un atacante pruebe la red sin darnos mucho beneficio, y deben bloquearse estas salidas.

13. Network Time Protocol (NTP)

El NTP permite setear los relojes en los sistemas muy precisamente, hasta con 100ms y a veces aún 10ms. Para cierto tipo de aplicaciones y protocolos es extremadamente importante conocer la hora exacta.

Es más fácil relacionar información desde múltiples máquinas (archivos de log, por ejemplo, cuando se analiza un intento de break-in) cuando los relojes sobre esas máquinas están todos sincronizados. Es útil saber quién fue atacado exactamente, y en qué orden, si se va a analizar qué hizo después - y qué haría a continuación.

Los servidores NTP se comunican con otros servidores NTP siguiendo una jerarquía para distribuir información de reloj. Cuanto más cerca está el sistema a un reloj de referencia (un reloj atómico, radio reloj, o algún otro reloj definitivo), más alto está en la jerarquía. Los servidores se comunican entre sí frecuentemente, para estimar y registrar el retardo de red entre ellos tal que este retardo pueda ser compensado. Los clientes NTP simplemente le preguntan a los servidores la hora corriente sin preocuparse acerca de la compensación por los retardos de las comunicaciones.

¿Realmente se necesita configurar NTP para que trabaje con un firewall? Esta es la primera decisión. En cualquiera de los casos dados a continuación no se necesita ejecutar NTP a través del firewall; simplemente lo puede ejecutar internamente:

- Si tiene una fuente horaria ajustada dentro de la red interna - por ejemplo un reloj de radio recibiendo señales horarias desde los relojes atómicos del National Bureau of Standards sobre una de sus estaciones de radio (o el equivalente desde las organizaciones de estándares no de EE.UU.), o un reloj satelital recibiendo datos desde los satélites Global Positioning System (GPS).
- Si se está más preocupado por tener hora que sea consistente dentro de la red que entre la red y el mundo exterior.

Si se quiere correr NTP a través del firewall, la mejor forma es instalar un servidor NTP sobre un host bastión que habla a múltiples servidores NTP externos y otro servidor NTP sobre algún host interno que hable al host bastión. (Se busca que el host bastión hable a múltiples servidores NTP externos ya que incrementa la precisión y lo hace más difícil de engañar). Luego se configuran clientes NTP internos y otros servidores NTP internos para hablar al servidor interno que habla con el servidor del bastión. Se necesita configurar cualquier sistema de filtrado de paquetes entre el servidor interno y el host bastión para permitir lo siguiente:

- Preguntas desde el servidor NTP interno al servidor NTP del host bastión: paquetes UDP desde el port 123 en el servidor interno al port 123 en el host bastión.
- Respuestas del servidor NTP del host bastión al servidor NTP interno: paquetes UDP desde el port 123 del host bastión al port 123 del host interno.

14. Network File System (NFS)

El protocolo NFS está diseñado para permitir que los sistemas accedan archivos a través de la red sobre un sistema remoto, tan convenientemente como si los archivos estuvieran en discos conectados directamente. Las máquinas pueden ser servidores NFS (exportando sus discos para que se accedan desde otras máquinas), clientes NFS (accediendo discos exportados por servidores NFS), o ambos. NFS está fuertemente usado dentro de LANs, pero es inseguro para permitirlo a través de un firewall por varias razones.

El problema primario con NFS es la frágil autenticación de los pedidos. El acceso a un cierto filesystem NFS-exportado es todo o nada; dada una máquina cliente, es confiable para acceder al filesystem o no. Si el servidor confía en una máquina cliente dada, entonces cree cualquier cosa que el cliente le dice acerca de quién está tratando de acceder a los archivos. Éste luego usa esa información para autorización de acuerdo a los mecanismos estándar de protección de archivo de UNIX (es decir, usuario, grupo y otros permisos). Hay unas pocas protecciones extras que algunos servidores NFS pueden aplicar opcionalmente.

La confianza del servidor en el cliente se establece cuando el cliente monta el filesystem desde el servidor. El cliente contacta al servidor y le dice al servidor a qué filesystem quiere acceder. El servidor chequea si el cliente está habilitado para acceder a ese filesystem o no (basado en la dirección IP). Si lo está, el servidor le da al cliente un "file handle", que va a

ser usado por el cliente para todos los posteriores accesos al filesystem.

Una vez que el cliente montó el filesystem (y recibió un manejador de archivo desde el servidor) el cliente envía un pedido al servidor cada vez que quiere actuar en un archivo sobre ese filesystem. El pedido describe la acción que el cliente quiere tomar e incluye al manejador de archivo obtenido desde el servidor, de modo que el servidor asume que el cliente está autorizado a solicitar esa acción.

Los principales problemas de seguridad de NFS son:

- La vulnerabilidad a la adulteración de direcciones pues el servidor NFS confía en la dirección IP para autenticar hosts clientes.
- El servidor NFS confía en el cliente para autenticar al usuario, haciéndolo vulnerable a cualquier usuario que ha comprometido una máquina cliente.
- El servidor NFS no rechequea la autenticación del cliente en cada pedido. El servidor supone que si el cliente usa un manejador de archivo válido, el cliente está autorizado para acceder a ese filesystem. Un atacante con un file handle capturado o adulterado puede acceder al filesystem tan fácilmente como lo puede hacer un cliente legítimo.

El manejador de archivo para un filesystem se establece cuando el filesystem se crea sobre el servidor o cuando se monta por primera vez sobre un nuevo punto de instalación del servidor o desde un nuevo par disco/controladora. Desafortunadamente, el método para generar manejadores de archivo es claramente predecible, y un atacante puede probablemente adivinar el file handle si puede adivinar cuándo fue creado el filesystem. Muchos servidores NFS permitirán cualquier número de malas adivinanzas sin preocuparse. Si de alguna manera el atacante puede adivinar al menos el día que se creó el filesystem - lo cual es especialmente fácil si el ataque es sobre una máquina instalada recientemente - probablemente puede adivinar el file handle sin demasiada dificultad, probando el rango de manejadores de archivo que habría sido asignado a filesystems creados ese día.

Los servidores NFS no tienen mecanismo para cancelar file handles; una vez que un manejador de archivo fue realizado, es bueno hasta que el filesystem del servidor es montado o reinicializado en algún otro lado (aún si es solamente temporario; cuando se lo remonta otra vez en su ubicación original, se tendrá un nuevo file handle). Un cliente que tuvo acceso una vez puede mantenerse usando el file handle que obtuvo aún si los controles de acceso (el archivo */etc/exports* sobre la mayoría de los servidores NFS UNIX) se cambian para denegar el acceso a ese cliente.

Bajo NFS, *root* puede tratarse diferente a los usuarios normales. Algunos servidores NFS UNIX siempre tratan al *root* de la misma manera que a los usuarios normales, es decir, el usuario *root* del cliente obtiene el mismo acceso que tendría el usuario *root* del servidor. Algunos de ellos siempre traducen el usuario *root* del cliente a un UID conocido como 'nobody' que nunca se usa como un usuario regular.

Traduciendo *root* a 'nobody' la mejora que producen en la seguridad es menor. Cualquiera que sea capaz de ser *root* sobre el cliente es capaz de ver y hacer cualquier cosa que el usuario puede hacer. La traducción oculta sólo los archivos del servidor con acceso

restringido por el mismo *root*. Aún probablemente se querrá usar la traducción dondequiera que se pueda por la protección mínima que da, pero no se debería sentir que hace segura la exportación de filesystems a clientes posiblemente hostiles.

Está disponible una mejor protección para el servidor exportando el filesystem en modo read-only. Si el filesystem se exporta puramente en read-only (los hosts no están autorizados a escribirlo) puede ser razonablemente cierto que los datos no puedan ser modificados vía NFS. Si se permite que cualquier host lo escriba, va a ser vulnerable a la adulteración.

Los clientes NFS también pueden estar en peligro desde los servidores NFS. Por ejemplo, un filesystem montado-NFS puede contener programas *setuid*; los usuarios en el cliente serían capaces de usar esos programas para volverse root. Las entradas de dispositivos sobre una partición montada NFS se consideran para aplicarse a los dispositivos de los clientes, no a los dispositivos del servidor. Alguien con una cuenta sobre un cliente NFS y un permiso root sobre un servidor NFS puede usar esto para tener, si fuera inconveniente, un acceso read-write ilimitado a todos los datos en el cliente.

Algunos de los clientes NFS proveen opciones para *mount* que pueden usarse para deshabilitar dispositivos y *setuid/setgid* sobre los filesystems montados. Si *mount* no está disponible para otros usuarios más que para root, o si siempre usa estas opciones para usuarios distintos a root, esto protegerá al cliente del servidor. Si estas opciones no están disponibles, aún si solamente root puede montar filesystems, debería considerarse que el montar un filesystem es equivalente a autorizar un acceso root a la máquina servidor.

Si se tiene un requerimiento inmutable para ejecutar NFS a través del firewall a algún otro sitio, debería investigar o bien una conexión privada a ese sitio o encriptado a nivel de red. Tenga cuidado que NFS generalmente no funciona bien sobre redes corriendo en algo menor que la velocidad Ethernet. Se podrían estudiar alternativas tales como FTP para esta situación particular o filesystems de red alternativos tales como el Andrew File System (AFS).

15. Network Information Service/Yellow Pages (NIS/YP)

NIS/YP está diseñado para proveer acceso distribuido a información administrativa centralizada (tal como tablas de host, archivos de password, alias de e-mail a lo largo del sitio, etc.) compartida por las máquinas en un sitio.

El problema principal con NIS/YP es que su seguridad no es lo suficientemente buena como para proteger adecuadamente algunos de los datos que contiene. En particular, los servidores NIS/YP de un sitio generalmente contienen el archivo de passwords compartido (equivalente al archivo */etc/passwd* en un sistema aislado) del sitio, lleno de passwords encriptadas. Todo lo que se necesita para conseguir datos desde un servidor NIS/YP es el nombre de dominio de NIS/YP con el que los datos están asociados. Un atacante que puede hablar al servidor NIS/YP del sitio, y que puede adivinar qué eligió el sitio como nombre de dominio del servidor NIS/YP (frecuentemente el mismo o un derivado de su nombre de dominio

regular de Internet), puede solicitar cualquier información que tiene el servidor. Si el atacante consigue el archivo de password compartido, estas passwords pueden crackearse a la comodidad del atacante.

NOTA : Las transferencias NIS/YP incluyen las passwords encriptadas aún si las máquinas están configuradas para usar passwords ocultas y las passwords encriptadas no son legibles en el servidor NIS/YP.

Unos pocos servidores NIS/YP (notablemente los de Sun) soportan un archivo de configuración llamado *securenets*. Esto permite usar una autenticación de dirección IP para controlar a qué hosts le va a pasar datos el servidor NIS/YP. Esta es una mejora de gran magnitud en la seguridad NIS/YP. Cambia los ataques NIS/YP de juegos de adivinanzas (adivinar el nombre de dominio y el servidor NIS/YP y se obtiene un premio) a requerir que se hagan las mismas adivinanzas y luego determinar a qué direcciones le va a contestar el servidor NIS/YP y le va a falsificar paquetes. Desafortunadamente, un orden de magnitud probablemente no es una mejora suficiente para datos tan cruciales como las passwords encriptadas. Mientras que *securenets* (si se lo tiene disponible) lo protegerá de atacantes casuales que quieren entrar en cualquier sitio que puedan, no lo protegerá de un atacante que conoce el sitio y quiere atacarlo en particular.

El truco es, entonces, impedir que un atacante hable con el servidor NIS/YP.

16. X11 Window System

El sistema Window X11 posee una serie de problemas para brindarlo a través de un firewall. Vale aclarar que la mayoría de los sistemas Window provistos por los distribuidores UNIX están o bien basados en o son muy similares a X11. Desde el punto de vista del firewall, muchas de las consideraciones son las mismas para las distintas versiones.

El primer problema con X11 es que la relación cliente/servidor está por detrás de muchos otros protocolos. El "servidor" X11 es la unidad display/mouse/teclado, y los "clientes" son los programas de aplicación manejando ventanas o interactuando con el mouse o el teclado en ese servidor. Por eso, el servidor típicamente está dentro del firewall (sentado en el escritorio del usuario), y los clientes están fuera (ejecutando en cualquier computadora remota a la que accedió el usuario).

Los servidores X11 tienen ciertas capacidades que los hacen muy tentadores para los atacantes. Hay gran cantidad de ataques que se pueden llevar adelante teniendo acceso a un servidor X11, incluyendo:

- Conseguir vuelcos de pantalla - obteniendo una copia de cualquier información que está siendo mostrada sobre la pantalla en cualquier momento.
- Lectura de lo tecleado - por ejemplo, leyendo una password de usuario a medida que la va escribiendo en el teclado.
- Insertar teclas como si fueran tipeadas por el usuario; esto permite potencialmente que el

atacante haga todo tipo de acciones "desagradables", especialmente en una ventana donde el usuario habitualmente está ejecutando un shell de root.

X11 tiene ciertos mecanismos de seguridad, pero hasta la fecha han probado o bien ser muy débiles o bien ser engorrosos para ser realmente útiles contra determinado ataque. Uno, conocido comúnmente como el mecanismo 'magic cookie', se basa en un 'secreto' compartido entre el servidor y los clientes legítimos. Los clientes están habilitados para acceder al servidor sólo si pueden probar que conocen el 'secreto'. El problema es que, aunque nunca se pase directamente este 'secreto' entre el servidor y el cliente X11, muchas de las formas en que los usuarios lo dejan disponible ante el cliente y el servidor (por ejemplo, vía un archivo accesible por NFS en su directorio que es accesible por el cliente y por el servidor) hacen que quede comprometido de modo tal que cualquiera haciendo snooping sobre la red lo puede conocer. Mientras que el mecanismo 'magic cookie' es teóricamente seguro, en la práctica no se puede medir la seguridad que da porque no hay una forma fácil de usarlo correctamente.

Otro de los mecanismos de seguridad de X11, llamado el mecanismo xhost, permite que el usuario le diga al servidor desde qué direcciones IP remotas debería aceptar las conexiones. Se supone que los usuarios sólo autorizan hosts específicos donde tratan de ejecutar clientes X11. El problema es que los usuarios se olvidan de preautorizar a los hosts antes de inicializar a los clientes, y los clientes son rechazados. Después que esto ocurre unas pocas veces, muchos usuarios deshabilitan los controles todos juntos.

Pocos sitios necesitan correr X11 a través de firewalls desde la Internet. Hay sitios WWW ocasionales que proveen clientes X como una forma de proveer displays en tiempo real, pero hay pocos y bastante alejados entre sí. Si se necesitara permitir X11 desde la Internet, debería considerarse usar uno de los proxys X11, por ejemplo, el x-gw en el TIS Firewall Toolkit.

17. Protocolos De Impresión (LPR y LP)

El sistema de impresión *lpr* de BSD es muy similar al de los comandos *'r'* discutido anteriormente. A diferencia de los comandos *'r'*, *lpt* autoriza hosts, no usuarios individuales, y aceptará trabajos desde hosts en */etc/printers.equiv* como en */etc/hosts.equiv*.

Dadas las deficiencias de *lp* y *lpr*, muchos distribuidores de UNIX implementan sus propias soluciones de impresión remota. Otras plataformas pueden soportar *lp*, *lpr*, un protocolo separado, o alguna combinación. Algunas impresoras son dispositivos de red en su propio derecho, a veces hablando *lp* o *lpr* directamente, y a veces (particularmente las impresoras más viejas) hablando un protocolo desarrollado por el fabricante de la impresora.

Toda la amplia variedad de otros protocolos de impresión de red comparte una característica común; no son más seguros que *lpr* y soportan posibles ataques de denegación de servicio sobre impresoras.

Algunos ataques son solamente fastidiosos, tales como imprimir páginas con basura (o,

mejor todavía, algo ofensivo para la víctima o sus compañeros de trabajo) hasta que la impresora se queda sin papel. Desafortunadamente, el más desagradable de todos está habilitado por una característica de seguridad en PostScript. PostScript fue diseñado para proteger ciertos comandos peligrosos solicitando una password antes de que fueran ejecutados. Esta password se guarda en un chip EEPROM de la impresora, y está seteado de fábrica en "0" sobre cada marca de fábrica de impresora PostScript. Siempre es lo mismo porque *debe* ser siempre el mismo. Algunos de los comandos de los diseñadores PostScript considerados peligrosos son usados rutinariamente por drivers PostScript, y si se cambia la password, esos drivers ya no funcionarán.

La inteligencia y vulnerabilidad de los dispositivos PostScript hace importante proteger a las impresoras del acceso desde Internet. Asegúrese de que ha bloqueado cualquier protocolo de impresión remota que usan sus máquinas e impresoras. Se necesitará chequear cada tipo de impresora y cada tipo de máquina separadamente.

CARACTERÍSTICAS DE FILTRADO Y SERVICIOS PROXY

Volviendo a la lista de protocolos y servicios, se necesita dar el comportamiento de cada uno ante el filtrado de paquetes o el servicio proxy para ver como configurarlos ante las decisiones de seguridad que se hayan tomado.

1. Correo Electrónico

Simple Mail Transfer Protocol (SMTP)

Características de filtrado de paquetes de SMTP

SMTP es un servicio basado en TCP. Los receptores SMTP usan el port 25. Los iniciadores usan un port aleatorio superior a 1023.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	25	(a)	mail entrante, origen a receptor
OUT	Int.	Ext.	25	>1023	Si	mail entrante, receptor a origen
OUT	Int.	Ext.	>1023	25	(a)	mail saliente, origen a receptor
IN	Ext.	Int.	25	>1023	Si	mail saliente, receptor a origen

(a) ACK no se setea en el primer paquete; sí en los restantes

Normalmente se configuran los filtrados de paquetes para permitir SMTP entrante y saliente sólo entre hosts externos y el host bastión, y entre el host bastión y los servidores de

mail internos.

Características proxy de SMTP

En sí es adecuado para hacer proxying porque SMTP es un protocolo store-and-forward. Cualquier servidor SMTP puede ser un proxy. Por eso, muchos sitios dirigen las conexiones SMTP a un host bastión que ejecuta un servidor SMTP seguro que hace las veces de proxy. El utilizar proxying protege de las conexiones no deseadas, pero no del mal uso de las conexiones; es decir, no se quiere dejar que los hosts externos dialoguen con un servidor estándar SMTP inseguro, aún cuando se haga a través de un proxy.

Post-Office Protocol (POP)

Características de Filtrado de Paquetes de POP

POP es un servicio basado en TCP. Los servidores POP de la versión corriente del protocolo POP (que se conoce como POP3 y es por el momento la versión más común en uso) usan el port 110. Los servidores para el POP2 - protocolo más viejo - usan el port 109. Los clientes POP usan ports por encima de 1023.

Sentido	Direcc. Origen	Direcc. Dest.	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	110 / 109	(a)	conexión POP entrante, cliente a servidor
OUT	Int.	Ext.	110 / 109	>1023	Si	conexión POP entrante, servidor a cliente
OUT	Int.	Ext.	>1023	110 / 109	(a)	conexión POP saliente, cliente a servidor
IN	Ext.	Int.	110 / 109	>1023	Si	conexión POP saliente, servidor a cliente

(a) ACK no se setea en el primer paquete; si en los restantes

Una conexión POP de salida permitiría que los usuarios recuperen los mails desde otros sitios. Esto no es más peligroso que permitir un Telnet de salida, y probablemente se quiera proporcionar tal conexión POP si hay demanda. Por el otro lado, POP sobre la Internet es lo suficientemente raro como para que alguien tuviera interés en un POP saliente en el sitio, y su uso no tiene sentido.

Las conexiones POP entrantes son aquellas que posibilitan que gente en otros sitios lean el mail enviado a ellos en nuestro sitio. Si se quiere permitir un POP entrante, se deberían limitar las conexiones POP a un servidor corriendo sobre un único host. Esto acotará el número de cuentas vulnerables y la cantidad de información que está pasando a través de la Internet. Aunque no hay vulnerabilidades conocidas en los servidores POP, si se encontraran algunas, sólo deberían corregirse en un único host, sin preocuparse por todos los hosts internos. Ya que POP requiere cuenta de usuario, no se recomienda ejecutarlo sobre el host bastión normal. Aunque se pueda prevenir a los usuarios de hacer login sobre las cuentas POP, representan una disputa de mantenimiento.

Características proxy de POP

POP es algo directo para hacer proxy porque usa una única conexión. Los clientes

POP precompilados habilitados para proxy (por ejemplo, los que trabajan con SOCKS) no están ampliamente disponibles. Esto se debe a que POP es raro a través de la Internet. Los clientes POP de UNIX se encuentran disponibles en código fuente y tienen una adaptación trivial a los sistemas de proxy para cliente modificado.

No hay una forma fácil de hacer proxy de cliente modificado para las conexiones entre clientes internos y servidores externos, o clientes externos y servidores internos, a menos que todos los clientes se conecten al mismo servidor.

2. Sistema De Nombres De Dominio (DNS)

Características de Filtrado de paquetes de DNS

Hay dos tipos de actividades de red DNS: búsqueda y transferencia de zona.

- La *búsqueda* es cuando un cliente DNS o un servidor DNS que actúa en nombre de un cliente le pide información a un servidor DNS.
- La *transferencia de zona* ocurre cuando un servidor DNS (servidor secundario) le pide a otro servidor DNS (servidor primario) cualquier dato que el servidor primario conozca sobre una parte dada del árbol de nombres DNS. La transferencia de zonas sucede solamente entre servidores que se supone están brindando la misma información; un servidor no tratará de hacer transferencia de zona desde otro servidor en forma aleatoria en circunstancias normales. La gente ocasionalmente hace transferencia de zona para recolectar información; esto está bien cuando se está calculando cuál es el nombre de host más popular en Internet pero esta mal cuando están tratando de descubrir qué hosts pueden atacar a su sitio.

Por razones de rendimiento las búsquedas de DNS se ejecutan usando UDP. Si se pierden algunos de los datos en la transmisión por UDP, la búsqueda se volverá a hacer usando TCP. Hay muchas excepciones, por ejemplo las máquinas que corran el sistema operativo AIX de IBM que siempre usan TCP incluso para la pregunta inicial.

Un servidor DNS usa el port 53 para todas sus actividades UDP y como su port servidor para TCP. Usa un port aleatorio por encima de 1023 para los pedidos de TCP. Un cliente DNS usa un port aleatorio por encima de 1023 tanto para UDP como para TCP. Así se puede diferenciar entre lo siguiente:

- Una pregunta del cliente al servidor, donde el port de origen está por encima de 1023 y el destino es el port 53.
- Una respuesta del servidor al cliente, donde el port origen es el 53 y el port destino está por encima del 1023.
- Una pregunta o respuesta de servidor a servidor, al menos con UDP, donde los ports origen y destino están en el 53; con TCP, el servidor que hace la pregunta usará un port por encima del 1023.

Las transferencias de zona DNS son ejecutadas usando TCP. La conexión se inicia desde un port aleatorio por encima de 1023 en el servidor secundario (el que solicita los datos)

a un port 53 en el servidor primario que envía los datos requeridos por el secundario. Un servidor secundario también debe hacer una pregunta DNS regular de un servidor primario para decidir cuándo hacer la transferencia de zona.

Sentido	Direcc. Origen	Direcc. Destino	Protocolo	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	UDP	>1023	53		entrada de una pregunta cliente al servidor
OUT	Int.	Ext.	UDP	53	>1023		respuesta a una pregunta servidor al cliente
IN	Ext.	Int	TCP	>1023	53		pregunta de entrada cliente al servidor
OUT	Int.	Ext.	TCP	53	>1023	1	respuesta a una pregunta cliente al servidor
OUT	Int.	Ext.	UDP	>1023	53		salida de una pregunta cliente al servidor
IN	Ext.	Int.	UDP	53	>1023		respuesta a una pregunta de salida, servidor al cliente
OUT	Int.	Ext.	TCP	>1023	53		salida de una pregunta cliente al servidor
IN	Ext.	Int.	TCP	53	>1023	1	respuesta a una pregunta de salida, servidor al cliente
IN	Ext.	Int.	UDP	53	53		pregunta y respuesta entre dos servidores
OUT	Int.	Ext.	UDP	53	53		pregunta y respuesta entre dos servidores
IN	Ext.	Int.	TCP	>1023	53		pregunta desde un servidor externo al servidor interno, también requiere la transferencia de zona desde el servidor externo secundario.
OUT	Int.	Ext.	TCP	53	>1023	1	respuesta desde un servidor interno al servidor externo, también una respuesta a una transferencia de zona de un servidor secundario externo.
OUT	Int.	Ext.	TCP	>1023	53		pregunta desde un servidor interno a uno externo.
IN	Ext.	Int.	TCP	53	>1023	1	pregunta desde un servidor externo a un servidor interno.

Características proxy de DNS

El DNS está estructurado de manera tal que los servidores siempre actúen como proxys para los clientes. También es posible usar una característica de DNS llamada *forwarding* (despacho) de modo que un servidor DNS sea efectivamente un proxy para otro servidor.

En muchas implementaciones sería posible modificar las librerías DNS para que usen un proxy de cliente modificado.

3. Transferencia De Archivos

File Transfer Protocol (FTP)

Características de filtrado de paquetes para FTP

FTP usa dos conexiones TCP separadas: una para transportar los comandos y resultados entre el cliente y el servidor (comúnmente llamada *canal de comandos*) y la otra para transportar cualquier archivo y lista de directorios transferidos (*canal de datos*). En el servidor final, el canal de comando usa el port 21 y el canal de datos normalmente usa el port 20. El cliente usa ports por encima del 1023 para ambos canales.

Al comenzar una sesión FTP, un cliente primero se asigna dos ports TCP, cada uno por encima de 1024. El primero lo usa para abrir la conexión del canal de comandos con el servidor y luego realiza el comando PORT de FTP para decirle al servidor el número del segundo port, que lo va a usar como canal de datos. El servidor luego abre la conexión con el canal de datos. Esta conexión del canal de datos va hacia atrás desde la mayoría de los protocolos, la cual abre la conexión desde el cliente al servidor. Esta apertura hacia atrás complica las cosas para los sitios que están intentando hacer un filtrado de paquetes al inicio de la conexión para asegurar que todas la conexiones TCP sean iniciadas desde adentro, debido a que los servidores FTP externos intentarán iniciar las conexiones de datos hacia los clientes internos en respuesta de las conexiones de comandos abiertas desde esos clientes internos. Además, estas conexiones comenzarán a ir a los ports conocidos para estar en un rango no seguro.

La mayoría de los servidores FTP (especialmente aquellos usados por la mayoría de los sitios FTP anónimos en Internet) y muchos clientes FTP soportan un modo alternativo que permite al cliente abrir tanto los canales de comandos como los de datos hacia el servidor. Este modo se llama modo *pasivo* o modo PASV después que el comando FTP el cliente envía hacia el servidor para iniciar este modo. Si tanto el cliente y el servidor FTP que se están tratando de conectar soportan el modo pasivo, lo pueden usar en lugar del modo regular. De esta forma se pueden evitar problemas de filtrado al inicio de la conexión porque todas las conexiones serán abiertas desde adentro por el cliente.

Para usar el modo *pasivo* un cliente FTP se asigna dos ports TCP, utilizando el primer port para contactar al servidor FTP. Sin embargo, en lugar de emitir el comando 'port' para decirle al servidor el segundo port del cliente, envía el comando PASV. Esto hace que el servidor asigne un segundo port propio para el canal de datos y le diga al cliente el número de ese port. (Por razones de arquitectura los servidores usan ports aleatorios por encima de 1023 para esto y no el port 20 como en el modo normal. No se podrían tener dos servidores en la misma maquina escuchando simultáneamente las conexiones de datos en modo PASV que entran en el port 20). El cliente luego abre la conexión de datos desde su port al port de datos que el servidor le acaba de comunicar.

No todos los clientes FTP soportan el modo pasivo. Si no lo soporta, se mencionará como una característica en la documentación. Algunos clientes brindan los dos modos y alguna manera de especificar cuál modo usar. A veces el modo pasivo falla y las transferencias se cuelgan.

Si el cliente o servidor FTP no soportan modo pasivo y se desea permitir FTP vía filtrado de paquetes se tendrá que poner una excepción para este caso especial en las reglas de filtrado de paquetes y ahí permitir al servidor abrir el canal de datos hacia atrás para el

cliente. De todos modos será vulnerable a los atacantes que lancen una conexión desde el port 20 del lado del atacante hacia un port por encima del 1023 en su extremo. Por lo tanto se debería restringir esta excepción de caso especial tanto como sea posible, por ejemplo asociándolo a la dirección de un cliente o de un servidor particular que no soporte modo pasivo.

Algunas implementaciones dinámicas de filtrado de paquetes controlan los comandos enviados a través de canales de comando y observan el comando PORT que el cliente envía hacia el servidor. Este comando le dice al servidor qué port está escuchando el cliente, para que el servidor abra el canal de datos. Estas implementaciones también colocan en el lugar una excepción temporaria de tiempo limitado en las reglas de filtrado de paquetes para permitir al servidor abrir el canal de datos hacia atrás para el cliente.

Sentido	Direcc. Origen	Direcc. Dest.	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	21		requiere una entrada FTP
OUT	Int.	Ext.	21	>1023	1	responde a un requerimiento de entrada
OUT	Int.	Ext.	20	>1023		creación del canal de datos para un requerimiento FTP normal, de salida
IN	Ext.	Int.	>1023	20	1	respuesta del canal de datos al requerimiento de entrada FTP, modo normal
IN	Ext.	Int.	>1023	>1023		creación del canal de datos para un requerimiento de FTP de entrada, modo pasivo.
OUT	Int.	Ext.	>1023	>1023	1	respuesta del canal de datos al requerimiento de entrada FTP, modo pasivo.
OUT	Int.	Ext.	>1023	21		salida de un requerimiento FTP
IN	Ext.	Int.	21	>1023	1	respuesta al requerimiento de salida
IN	Ext.	Int.	20	>1023		creación del canal de datos para un requerimiento de salida modo normal.
OUT	Int.	Ext.	>1023	20	1	respuesta al canal de datos para un requerimiento de FTP de salida de modo normal
OUT	Int.	Ext.	>1023	>1023		creación del canal de datos para un requerimiento de salida modo pasivo
IN	Ext.	Int.	>1023	>1023	1	respuesta al canal de datos para un requerimiento de un FTP de salida modo pasivo

(*) el protocolo siempre es TCP.

Características proxy de FTP

Debido a los problemas con el modo pasivo, el proxy es una solución adecuada para el FTP de salida. Un cliente proxy normal permite hablar en forma confiable con los servidores externos sin tener que permitir conexiones TCP entrantes para el canal de datos a otro host excepto al host bastión que está haciendo el proxy. De ahí que se pueda optar por hacer proxy de FTP aún permitiendo que otros protocolos pasen a través del firewall vía filtrado de paquetes. Tanto el cliente como los procedimientos modificados para proxy están disponibles para FTP.

El paquete SOCKS incluye un cliente FTP para UNIX que fue modificado para usar SOCKS. El FTP involucra múltiples conexiones TCP simultáneas, por lo que modificar otros clientes FTP por cuenta propia implica mucho esfuerzo.

Muchos lugares utilizan combinaciones de filtrado de paquetes y proxy como soluciones para FTP. A veces se usa proxy para las conexiones en modo normal y filtrado de paquetes para las conexiones en modo pasivo. También se usa una solución combinada para agregar seguridad usando un servidor proxy FTP que usa modo pasivo para hacer las conexiones externas, sin tener en cuenta el modo que usa para hablar a los hosts internos; esto convierte todas las conexiones que cruzan el firewall a modo pasivo y posibilita que se ajusten los filtros de paquetes que protegen al host que está haciendo el proxying. Además, evita que se usen servidores que no soportan el modo pasivo.

Trivial File Transfer Protocol (TFTP)

Características de Filtrado de Paquetes de TFTP

TFTP es un protocolo basado en UDP. Los servidores usan el port 69. Los clientes usan ports por encima de 1023.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	69	(a)	pedido TFTP entrante
OUT	Int.	Ext.	69	>1023	(a)	respuesta a requerimiento de entrada
OUT	Int.	Ext.	>1023	69	(a)	requerimiento TFTP de salida
IN	Ext.	Int.	69	>1023	(a)	respuesta a requerimiento de salida

(a) los paquetes son todos UDP; no tienen bit ACK.

Características proxy de TFTP

El TFTP no se lleva bien con el proxy. Generalmente, los clientes TFTP se implementan en hardware, sin usuarios involucrados. Si se quiere, un proxy podría soportar fácilmente al TFTP, proveyendo la misma seguridad mínima que se puede brindar a través del filtrado de paquetes.

File Service Protocol (FSP)

Características de Filtrado de Paquetes de FSP

FSP es intencionalmente difícil de suprimir con filtrado de paquetes. Si se filtran todos los servicios UDP excepto los necesarios, y se restringen esos a host específicos, se lo bloqueará excepto sobre los hosts donde se están permitiendo otros servicios UDP. Si se quiere permitir FSP, es muy simple; se habilitan los paquetes UDP desde y hacia un host, a cualquier número de port, y en consecuencia se está autorizando que sea un cliente y/o servidor FSP.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción

IN	Ext.	Int.	>1023		(a)	pedido FSP entrante
OUT	Int.	Ext.		>1023	(a)	respuesta a requerimiento de entrada
OUT	Int.	Ext.	>1023		(a)	requerimiento FSP de salida
IN	Ext.	Int.		>1023	(a)	respuesta a requerimiento de salida

(a) los paquetes son todos UDP; no tienen bit ACK.

Características proxy de FSP

No hay paquete proxy estándar que soporte FSP. Podría escribirse un proxy FSP dedicado, porque usar un proxy genérico para una aplicación UDP que no usa un port estándar es tan difícil como querer suprimirlo con filtrado de paquetes.

UNIX-to-UNIX Copy Protocol (UUCP)

Características de Filtrado de Paquetes de UUCP

UUCP sobre TCP es obviamente un servicio basado en TCP. Los servidores usan el port 540. Los clientes usan ports por encima de 1023.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	540	(a)	conexión UUCP entrante, cliente a servidor
OUT	Int.	Ext.	540	>1023	Si	conexión UUCP entrante, servidor a cliente
OUT	Int.	Ext.	>1023	540	(a)	conexión UUCP saliente, cliente a servidor
IN	Ext.	Int.	540	>1023	Si	conexión UUCP saliente, servidor a cliente

(a) ACK no se setea en el primer paquete pero si en los restantes

Características proxy de UUCP

Al ser un protocolo store-and-forward, no se usa con proxys de propósito general. Se lo configura para hacer su propio proxying. Es muy adecuado para hacer proxy de cliente y procedimiento modificados, ya que es un protocolo de conexión única directa que se usa con un número chico de clientes.

4. Terminal Access (Telnet)

Características de Filtrado de Paquetes de Telnet

Telnet es un servicio basado en TCP. Los servidores Telnet normalmente usan el port 23 (aunque se los puede setear para que usen cualquier otro). Los clientes Telnet usan ports por encima de 1023.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	23	(a)	sesión entrante, cliente a servidor
OUT	Int.	Ext.	23	>1023	Si	sesión entrante, servidor a cliente
OUT	Int.	Ext.	>1023	23	(a)	sesión saliente, cliente a servidor
IN	Ext.	Int.	23	>1023	Si	sesión saliente, servidor a cliente

(a) ACK no se setea en el primer paquete pero sí en los restantes

Características proxy de Telnet

Telnet está bien soportado por los proxys. SOCKS provee un cliente Telnet para UNIX modificado; modificar los clientes para otras plataformas es bastante trivial. Los proxys SOCKS deberían permitir la conexión a otros ports que no sean el estándar de Telnet, si tales conexiones son correctas de acuerdo al archivo de configuración del servidor SOCKS.

5. Ejecucion De Comando Remoto

Características de Filtrado de Paquetes de comandos 'r' BSD

Los comandos 'r' son servicios basados en TCP. Para el servidor, usan los ports (conocidos) 513 (*rlogin*) o 514 (*rsh, rcp, rdump, rrestore*, y *rdist*; estos son diferentes clientes para el mismo servidor). Son algo inusuales respecto a que usan ports aleatorios por *debajo* de 1023 para el lado cliente.

Esta característica para el lado cliente es un atentado para el esquema de seguridad que permite acceso sin-password a los servicios ya que los pedidos vienen desde un host y un usuario confiables, como se asumió más arriba. La idea es que, si el pedido viene desde un port menor a 1023 del lado cliente, entonces el pedido debe estar OK con root desde la máquina cliente; si así no fuera, el cliente nunca hubiera podido obtener el port menor a 1023 para usarlo.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	<1023	513	(a)	*Rlogin* entrante, cliente a servidor
OUT	Int.	Ext.	513	<1023	Si	*Rlogin* entrante, servidor a cliente
OUT	Int.	Ext.	<1023	513	(a)	*Rlogin* saliente, cliente a servidor
IN	Ext.	Int.	513	<1023	Si	*Rlogin* saliente, servidor a cliente
IN	Ext.	Int.	<1023	514	(a)	*rsh/rcp/rdump/rrestore/rdist* entrante, cliente a servidor
OUT	Int.	Ext.	514	<1023	Si	*rsh/rcp/rdump/rrestore/rdist* entrante, servidor a cliente
IN	Ext.	Int.	<1023	<1023	Si	*rsh* entrante, canal de error, cliente a servidor
OUT	Int.	Ext.	<1023	<1023	(a)	*rsh* entrante, canal de error, servidor a cliente
OUT	Int.	Ext.	<1023	514	(a)	*rsh/rcp/rdump/rrestore/rdist* saliente, cliente a servidor
IN	Ext.	Int.	514	<1023	Si	*rsh/rcp/rdump/rrestore/rdist* saliente, servidor a cliente
OUT	Int.	Ext.	<1023	<1023	Si	*rsh* saliente, canal de error, cliente a servidor
IN	Ext.	Int.	<1023	<1023	(a)	*rsh* saliente, canal de error, servidor a cliente

(a) ACK no se setea en el primer paquete pero si en los restantes

Características proxy de comandos 'r'

El único comando 'r' que se usa en la Internet es el *rlogin*. El TIS FWTK provee un servidor proxy para *rlogin* que usa procedimientos de usuario modificados para dar *rlogin* saliente. Para los otros comandos 'r' no hay proxys ampliamente disponibles.

6. Network News Transfer Protocol (NNTP)

Características de Filtrado de Paquetes de NNTP

NNTP es un servicio basado en TCP. Los servidores usan el port 119. Los clientes NNTP (incluyendo los servidores transfiriendo news a otros servidores) usan ports por encima de 1023.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	119	(a)	news entrantes
OUT	Int.	Ext.	119	>1023	Si	respuestas a news entrantes
OUT	Int.	Ext.	>1023	119	(a)	news salientes
IN	Ext.	Int.	119	>1023	Si	respuestas a news salientes
(b)	Int	News Server	>1023	119	(a)	cliente Newsreader leyendo noticias
(b)	News Server	Int	119	>1023	Si	servidor enviando artículos a cliente lector

(a) ACK no se setea en el primer paquete pero si en los restantes
(b) ambos extremos son internos en la mayoría de los casos

Características proxy de NNTP

NNTP es un protocolo store-and-forward, capaz de hacer su propio proxying. También es fácil de hacer el proxy como un protocolo directo de conexión única. Resulta sencillo modificar a los clientes para que usen un proxy de cliente modificado genérico como el SOCKS.

Uso de Filtrado de Paquetes con NNTP

Si se instaló un sistema de filtrado de paquetes entre el servidor interno de news y el sitio de alimentación de news, se puede convenir pasar NNTP entre ellos. Conviene que el sistema de filtrado permita lo siguiente:

- Conexiones NNTP entrantes, tal que se puedan recibir news: esto es, paquetes TCP desde ports mayores a 1023 sobre el sistema remoto al port 119 del servidor de news, y paquetes TCP con el bit ACK seteado desde el port 119 de nuestro servidor de news a ports mayores a 1023 sobre el sistema remoto (reglas A y B).
- Conexiones NNTP salientes, tal que se puedan enviar news: esto es, paquetes TCP desde ports mayores a 1023 en nuestro servidor de news al port 119 en el sistema remoto, y paquetes TCP con el bit ACK seteado desde port 119 sobre el sistema remoto a ports superiores a 1023 de nuestro servidor de news (reglas C y D).

Regla	Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Acción
A	IN	proveedor de servicio	nuestro servidor	>1023	119	(a)	permitir
B	OUT	nuestro servidor	proveedor de servicio	119	>1023	Si	permitir
C	OUT	nuestro servidor	proveedor de servicio	>1023	119	(a)	permitir
D	IN	proveedor de servicio	nuestro servidor	119	>1023	Si	permitir

(a) el primer paquete no lo tiene seteado; sí los restantes

7. World Wide Web (WWW) Y HTTP

Características de Filtrado de Paquetes de HTTP

HTTP es un servicio basado en TCP. Los clientes usan ports aleatorios por encima de 1023. La mayoría de los servidores usan el port 80.

La capacidad de proveer estos servicios sobre ports no estándar, a pesar de tener cierta utilidad, complica considerablemente las cosas desde el punto de vista de filtrado de paquetes. Si los usuarios quieren acceder a un servidor que se ejecuta en un port no estándar, se tienen que tomar varias determinaciones:

- Se le puede decir a los usuarios que no pueden hacerlo, pasando a ser una situación aceptable o no según el entorno.
- Se puede agregar una excepción especial para ese servicio en la configuración del filtrado de paquetes. Esto es malo para los usuarios porque significa que primero tienen que reconocer el problema y entonces esperar hasta que se lo pueda arreglar, y es malo para el administrador porque constantemente se tendrán que estar agregando excepciones a la lista de filtrado.
- Se puede tratar de convencer al propietario del servidor para que lo mueva al port estándar. Mientras se insiste en el uso de los ports estándar tanto como sea posible por ser una buena solución en el largo plazo, no se pueden obtener resultados inmediatos.
- Se puede usar algún tipo de versión proxy del cliente. Esto requiere instalación en la red propia, y puede restringir la elección de los clientes. Pero dos de los clientes más populares, Mosaic y Netscape, soportan proxying.
- Si se puede filtrar sobre el bit ACK, pueden permitirse las conexiones salientes sin tener en cuenta el port destino. Esto abre una amplia gama de servicios, incluyendo FTP en modo pasivo. También hay un notable incremento de la vulnerabilidad.

Los ports no estándar que usan los servidores en general son fácilmente reconocibles: 81, 800, 8000 y 8080.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	80	(a)	sesión entrante, cliente a servidor
OUT	Int.	Ext.	80	>1023	Si	sesión entrante, servidor a cliente
OUT	Int.	Ext.	>1023	80	(a)	sesión saliente, cliente a servidor
IN	Ext.	Int.	80	>1023	Si	sesión saliente, servidor a cliente

(a) ACK no se setea en el primer paquete pero sí en los restantes

Características proxy de HTTP

Varios clientes HTTP (tales como Mosaic y Netscape Navigator) soportan diferentes esquemas de proxy en forma transparente. Algunos clientes soportan SOCKS; otros soportan proxy transparente de usuario; y otros, ambos.

8. Otros Servicios De Información

Gopher

Características de Filtrado de Paquetes de Gopher

Gopher es un servicio basado en TCP. Los clientes Gopher usan ports superiores a 1023. La mayoría de los servidores Gopher usan el port 70.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	70	(a)	sesión entrante, cliente a servidor
OUT	Int.	Ext.	70	>1023	Si	sesión entrante, servidor a cliente
OUT	Int.	Ext.	>1023	70	(a)	sesión saliente, cliente a servidor
IN	Ext.	Int.	70	>1023	Si	sesión saliente, servidor a cliente

(a) ACK no se setea en el primer paquete pero sí en los restantes

Características proxy de Gopher

El servidor proxy de TIS FWTK sirve tanto para Gopher como para HTTP. SOCKS no incluye un cliente modificado. En general no es difícil de modificar el cliente Gopher para usarlo con SOCKS.

Wide Area Information Servers (WAIS)

Características de Filtrado de Paquetes de WAIS

WAIS es un servicio basado en TCP. Los clientes usan ports aleatorios por encima de 1023. Los servidores usan el port 210, pero a veces no.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	210	(a)	sesión entrante, cliente a servidor
OUT	Int.	Ext.	210	>1023	Si	sesión entrante, servidor a cliente
OUT	Int.	Ext.	>1023	210	(a)	sesión saliente, cliente a servidor
IN	Ext.	Int.	210	>1023	Si	sesión saliente, servidor a cliente

(a) ACK no se setea en el primer paquete pero sí en los restantes

Características proxy de WAIS

Si para acceder a WAIS se usa un browser que maneja proxy, como Netscape Navigator, automáticamente se tiene el soporte del cliente. Como un protocolo de conexión directa única con toda la información especificada por el usuario, WAIS se dirige al proxying de tanto el cliente como el procedimiento modificado. En los clientes WAIS aislados comúnmente está disponible el soporte SOCKS.

Archie

Característica de Filtrado de Paquetes de Archie

Archie es un servicio basado en UDP. Los clientes Archie dedicados usan ports por encima de 1023; los servidores usan el port 1525.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
OUT	Int.	Ext.	>1023	1525	(a)	consulta saliente, cliente a servidor
IN	Ext.	Int.	1525	>1023	(a)	respuesta entrante, servidor a cliente

(a) los paquetes UDP no tienen el bit ACK

Muchos sitios con sistema de filtrado de paquetes típicamente descartan todo el tráfico UDP, y abren específicamente agujeros restringidos entre sus hosts internos y su host bastión (*no* todo el mundo exterior) para los servicios claves basados en UDP.

Características proxy de Archie

El Archie es un servicio que no lo soporta SOCKS porque está basado en UDP, pero se le puede hacer proxy usando el UDP Packet Relayer.

9. Servicios De Búsqueda De Información

Finger

Características de Filtrado de Paquetes de finger

finger es un servicio basado en TCP. Los servidores usan el port 79. Los clientes usan ports por encima de 1023.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	79	(a)	consulta entrante, cliente a servidor
OUT	Int.	Ext.	79	>1023	Si	respuesta saliente, servidor a cliente
OUT	Int.	Ext.	>1023	79	(a)	consulta saliente, cliente a servidor
IN	Ext.	Int.	79	>1023	Si	respuesta entrante, servidor a cliente

(a) ACK no se setea en el primer paquete pero sí en los restantes

***Características proxy de* 'finger'**

SOCKS provee un cliente *finger* modificado para UNIX, y los clientes *finger* sobre otras plataformas deberían modificarse fácilmente para usar SOCKS.

Whois

***Característica de Filtrado de Paquetes de* 'whois'**

whois está basado en TCP. Los servidores usan el port 43. Los clientes usan ports aleatorios por encima de 1023.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
OUT	Int.	Ext.	>1023	43	(a)	consulta saliente, cliente a servidor
IN	Ext.	Int.	43	>1023	Si	respuesta entrante, servidor a cliente

(a) ACK no se setea en el primer paquete pero sí en los restantes

***Características proxy de* 'whois'**

SOCKS no provee un cliente *whois* modificado, y TIS FWTK no provee un servidor proxy *whois*. Como *whois* es un protocolo directo de una conexión con todos los datos especificados por el usuario, sería trivial modificar los clientes *whois* para SOCKS, y relativamente simple escribirle un servidor proxy de procedimiento modificado.

10. Servicios De Conferencia En Tiempo Real

Talk

***Características de Filtrado de Paquetes de* 'talk'**

A fin de habilitar *talk* a través del firewall, habría que permitir conexiones TCP en donde ambos extremos están usando ports arbitrarios por encima de 1023; esto no es seguro debido a las vulnerabilidades de servidores como los X11 que usan ports por encima de 1023.

Sentido	Direcc. Origen	Direcc. Destino		Port Origen	Port Destino	ACK	Descripcion
IN	Ext.	Int.	UDP	>1023	517 / 518	(a)	cliente externo contactando a servidor interno
OUT	Int.	Ext.	UDP	517 / 518	>1023	(a)	servidor interno respondiendo a cliente externo
OUT	Int.	Ext.	UDP	>1023	517 / 518	(a)	cliente interno contactando a servidor externo
IN	Ext.	Int.	UDP	517 / 518	>1023	(a)	servidor externo respondiendo a cliente interno
OUT	Int.	Ext.	TCP	>1023	>1023	(b)	cliente interno comunicándose con cliente externo
IN	Ext.	Int	TCP	>1023	>1023	(b)	cliente externo comunicándose con cliente interno

(a) los paquetes UDP no tienen bit ACK
(b) ACK no se setea en el primer paquete pero sí en los restantes

***Características proxy de* 'talk'**

No hay proxys disponibles para *talk*. Como *talk* involucra simultáneamente a clientes internos y externos, casi debería ser un servidor proxy de procedimiento modificado. No lo podría manejar un servidor genérico porque involucra a TCP y UDP. Escribir un proxy *talk* es difícil, y el proceso es extremadamente frágil.

Internet Relay Chat (IRC)

Característica de Filtrado de Paquetes de IRC

IRC es un servicio basado en TCP. Generalmente los servidores escuchan las conexiones entrantes (desde ambos clientes y otros servidores) sobre el port 6667. Los clientes (y servidores contactando a otros servidores) usan ports por encima de 1023.

Los clientes usan ports por encima de 1023 para hablar a otros clientes usando DCC. Para empezar, el cliente llamante pasa una invitación al cliente llamado a través de los canales del servidor IRC normal. La invitación incluye un número de port TCP donde el cliente llamante está esperando una conexión entrante. El cliente llamado, si elige aceptar la invitación, abre una conexión TCP a ese port.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	6667	(a)	cliente o servidor externo contactando servidor interno
OUT	Int.	Ext.	6667	>1023	Si	servidor interno contestando a cliente o servidor externo
OUT	Int.	Ext.	>1023	6667	(a)	cliente o servidor interno contactando servidor externo
IN	Ext.	Int.	6667	>1023	Si	servidor externo contestando a cliente o servidor interno
IN	Ext.	Int.	>1023	>1023	(a)	conexión DCC pedida por cliente interno; el cliente externo responde la invitación desde un cliente interno
OUT	Int.	Ext.	>1023	>1023	Si	conexión DCC pedida por cliente interno
OUT	Int.	Ext.	>1023	>1023	(a)	conexión DCC pedida por cliente externo; el cliente interno responde la invitación desde un cliente externo
IN	Ext.	Int.	>1023	>1023	Si	conexión DCC pedida por cliente externo

(a) ACK no se setea en el primer paquete pero sí en los restantes

***Características proxy de* IRC**

Debido a la arquitectura de árbol, cualquier servidor IRC se utiliza como servidor proxy. Para configurar a IRC como proxy, se pone un servidor IRC sobre el host proxy y se apuntan hacia éste a los clientes.

11. Syslog

***Características de Filtrado de Paquetes de* 'syslog'**

syslog es un servicio basado en UDP. Los servidores *syslog* (que registran mensajes loggeados por otros sistemas) trabajan sobre el port 514 de UDP. Los clientes *syslog* generalmente (pero no siempre) usan ports por encima de 1023 para hablar con los servidores. Los servidores *syslog* nunca envían mensajes de vuelta a los clientes; en el caso de querer configurar al servidor *syslog* para que pase mensajes hacia otros servidores *syslog*, el servidor origen generalmente usa el port 514 como el port cliente.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	514	(a)	cliente externo contactando servidor syslog interno

OUT	Int.	Ext.	>1023	514	(a)	cliente interno contactando servidor syslog externo
IN	Ext.	Int.	514	514	(a)	servidor syslog externo pasando mensaje a servidor interno
OUT	Int.	Ext.	514	514	(a)	servidor syslog interno pasando mensaje a servidor externo

(a) los paquetes UDP no tienen bit ACK

Características proxy de 'syslog'

Se dice que *syslog* es un protocolo que se hace proxy a sí mismo porque los servidores, en general, se pueden configurar para sólo pasar los mensajes que reciben hacia otros servidores *syslog.*

12. Servicios De Manejo De Red

Simple Network Management Protocol (SNMP)

Características de Filtrado de Paquetes de SNMP

SNMP es un servicio basado en UDP. Los servidores SNMP (en dispositivos de red) escuchan el port 161 de UDP y TCP. Los servidores trap SNMP (en estaciones de manejo) escuchan el port 162 de UDP y TCP. Los clientes SNMP generalmente usan ports por encima de 1023 para hablar con los servidores trap y los comunes.

Sentido	Direcc. Origen	Direcc. Destino		Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	UDP	>1023	161	(a)	Estación de manejo externa (cliente) contactando dispositivo de red SNMP interno (servidor)
OUT	Int.	Ext.	UDP	161	>1023	(a)	Dispositivo de red SNMP interno (servidor) respondiendo a una estación de manejo externa (cliente)
IN	Ext.	Int.	TCP	>1023	161	(b)	Estación de manejo externa (cliente) contactando a dispositivo interno de red (servidor)
OUT	Int.	Ext.	TCP	161	>1023	Si	Dispositivo de red interno SNMP (servidor) respondiendo a estación de manejo externa (cliente)
OUT	Int.	Ext.	UDP	>1023	161	(a)	Estación de manejo interno (cliente) contactando a un dispositivo de red externo SNMP (servidor)
IN	Ext.	Int.	UDP	161	>1023	(a)	Dispositivo de red SNMP externo (servidor) respondiendo a una estación de manejo interna (cliente)
OUT	Int.	Ext.	TCP	>1023	161	(b)	Estación de manejo interna (cliente) contactando a un dispositivo de red SNMP externo (servidor)
IN	Ext.	Int.	TCP	161	>1023	Si	Dispositivo de red SNMP externo (servidor) respondiendo a una estación de manejo interna (cliente)
IN	Ext.	Int.	UDP	>1023	162	(a)	Dispositivo de red externo (cliente) contactando a estación de manejo SNMP

							interna (servidor trap)
OUT	Int.	Ext.	UDP	162	>1023	(a)	Estación de manejo SNMP interna (servidor trap) respondiendo a dispositivo de red externo (cliente)
IN	Ext.	Int.	TCP	>1023	162	(b)	Dispositivo de red externo (cliente) contactando a estación de manejo SNMP interna (servidor trap)
OUT	Int.	Ext.	TCP	162	>1023	Si	Estación de manejo SNMP interna (servidor trap) respondiendo a un dispositivo de red externo (cliente)
OUT	Int.	Ext.	UDP	>1023	162	(a)	Dispositivo de red interno (cliente) contactando a una estación SNMP de manejo externa (servidor trap)
IN	Ext.	Int.	UDP	162	>1023	(a)	Estación de manejo SNMP externa (servidor trap) respondiendo a dispositivo de red interno (cliente)
OUT	Int.	Ext.	TCP	>1023	162	(b)	Dispositivo de red interno (cliente) contactando una estación SNMP de manejo externo (servidor trap).
IN	Ext.	Int.	TCP	162	>1023	Si	Estación de manejo SNMP externa (servidor trap) respondiendo a dispositivo de red interno (cliente)

(a) los paquetes UDP no tienen bit ACK
(b) ACK no se setea en el primer paquete; sí en los restantes

Características proxy de SNMP

Como SNMP es un protocolo que no se usa mucho a través de la Internet, no hay proxys muy difundidos para SNMP. Aunque es un protocolo directo de conexión única, es frecuente que se lo implemente en dispositivos para los que el código fuente y los compiladores no estén al alcance de la mano. Esto hace que sea casi imposible de usar con un sistema proxy de cliente modificado como el SOCKS. Pero no debería ser difícil de hacerlo funcionar con sistemas proxy de procedimiento modificado.

Routing Information Protocol (RIP)

Características de Filtrado de Paquetes de RIP

RIP es un servicio basado en UDP. Los servidores RIP escuchan en el port 520 las transmisiones de otros servidores y pedidos de los clientes. Los servidores RIP generalmente envían sus transmisiones desde el port 520. Los clientes RIP generalmente usan ports superiores a 1023.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	520	(a)	pedido de cliente externo a servidor interno
OUT	Int.	Ext.	520	>1023	(a)	respuesta de servidor interno a cliente externo
OUT	Int.	Ext.	>1023	520	(a)	pedido de cliente interno a servidor externo
IN	Int.	Ext.	520	>1023	(a)	respuesta de servidor externo a cliente interno
IN	Ext.	Broadcast	520	520	(a)	broadcasting de servidor externo a servidores internos
OUT	Int.	Broadcast	520	520	(a)	broadcasting de servidor interno a servidores externos

(a) los paquetes UDP no tienen bit ACK

Características proxy de RIP

No tiene sentido hacer un proxy de RIP a otro host cualquiera en la red porque RIP permite que un host desarrolle tablas de ruteo que son específicas del lugar donde está el host en la red.

Ping

***Características de Filtrado de Paquetes de* 'ping'**

Muchos sistemas de filtrado de paquetes dejan filtrar paquetes ICMP en casi la misma forma que los paquetes TCP y UDP, especificando el código del tipo de mensaje ICMP en vez del número de port origen o destino de TCP o UDP. Si el sistema de filtrado de paquetes tiene esta capacidad, su documentación debería incluir una lista de los códigos numéricos o palabras clave que entiende el sistema de filtrado de paquetes.

Para permitir que el programa *ping* opere en el exterior (es decir, haciendo *ping* sobre hosts remotos), tendrá que permitir paquetes de pedido "echo ICMP" salientes y paquetes de respuesta "echo ICMP" entrantes. Para permitir *ping* entrante (es decir, un host remoto haciendo *ping* sobre un host local), tendrá que permitir paquetes de pedido "echo ICMP" entrantes y paquetes de respuesta "echo ICMP" salientes.

Sentido	Direcc. Origen	Direcc. Destino	Protocolo	Tipo de Mensaje	Descripción
IN	Ext.	Int.	ICMP	8	*ping* entrante
OUT	Int.	Ext.	ICMP	0	respuesta al *ping* entrante
OUT	Int.	Ext.	ICMP	8	*ping* saliente
IN	Ext.	Int.	ICMP	0	respuesta al *ping* saliente

(a) los mensajes ICMP no tienen números de port origen o destino; en cambio, tienen un único campo de tipo de mensaje. Los paquetes ICMP tampoco tienen bit ACK.

***Características proxy de* 'ping'**

Al no estar basado en TCP ni en UDP, el *ping* no va a trabajar con los servidores proxy genéricos de amplia difusión en la Internet para proxying de cliente modificado. Al no transmitir datos provistos por el usuario al host destino, no se hace posible el proxy de procedimiento modificado. En un entorno de proxy puro, el *ping* tendrá que ser provisto permitiendo que los usuarios se conecten al host proxy y lo ejecuten desde ahí.

Traceroute

***Características de Filtrado de Paquetes de* traceroute**

Para permitir *traceroute* saliente a través del filtrado de paquetes (es decir, alguien ejecutando *traceroute* desde el interior hacia un destino externo), tienen que permitirse los paquetes UDP construidos 'salientes', y los paquetes ICMP relevantes 'entrantes' (particularmente "time to live exceeded" y "service unavailable"). Para permitir *traceroute*

entrante, tendrá que permitir los paquetes UDP construidos 'entrantes', y los mensajes ICMP relevantes 'salientes'.

Si se quiere limitar esta capacidad a las máquinas usadas por el centro de operaciones de red de su proveedor de servicio de red, además de ejercer un control más estrecho sobre los paquetes UDP que atraviesan el firewall, también se protegen servicios basados en RPC como NFS y NIS/YP y se frena a los atacantes que quieren usar *traceroute* para descubrir qué direcciones del sitio están realmente asignadas a los hosts.

En algunas versiones de *traceroute* puede decirse (vía comando de línea o una opción en tiempo de compilación) qué rango de ports UDP usar para el destino. Por eso se debe establecer de antemano qué ports serán los habilitados a *traceroute* a través del firewall.

Sentido	Dirección Origen	Dirección Destino	Port Origen	Port Destino	ACK		Descripción
OUT	Int.	Ext.	UDP	(b)	(b)	(a)	exploración de traceroute saliente
IN	Ext.	Int.	ICMP	(a)	(a)	11	"TTL exceeded" entrante
IN	Ext.	Int.	ICMP	(a)	(a)	3	"service unavailable" entrante
IN	Ext.	Int.	UDP	(b)	(b)	(a)	exploración de traceroute entrante
OUT	Int.	Ext.	ICMP	(a)	(a)	11	"TTL exceeded" saliente
OUT	Int.	Ext.	ICMP	(a)	(a)	3	"service unavailable" saliente

(a) los paquetes UDP tienen ports origen y destino; los paquetes ICMP sólo tienen campo de tipo de mensaje. Ninguno de ambos tiene bit ACK.
(b) los ports origen/destino del paquete UDP probe traceroute varían por implementación, invocación, y/o argumentos de comando de línea.

***Características proxy de* traceroute**

No es posible hacer proxy de procedimiento modificado sobre *traceroute*. Sí, como *ping*, podría ser fácilmente soportado por un servidor proxy de cliente modificado reconocido en ICMP, pero tal servidor no está disponible ampliamente.

Otros Paquetes ICMP

Características de Filtrado de Paquetes de ICMP

Muchos sistemas de filtrado de paquetes dejarán que se filtren los paquetes ICMP basados en el campo tipo de mensaje de la misma manera en que dejan filtrar paquetes UDP y TCP basados en los campos número de port origen y destino. Acá hay algunos tipos de mensaje ICMP comunes, y cómo se deberían manejar (si se debiera dejarlos pasar por el firewall, o bloquearlos).

Tipo de mensaje (a)	Descripción
0	Echo reply (respuesta a ping)
3	Destino no alcanzable. Puede indicar un host no alcanzable, red no alcanzable, port no alcanzable, u otro.
4	Source quench (alguien diciendo "slow down; you're talking too fast"); probablemente debería permitirse
5	Redirigir (alguien diciendo al destino para cambiar una ruta); se supone que es ignorado por los sistemas a menos que venga desde un router conectado directamente, pero de cualquier modo

	probablemente debería bloquearse. En particular, asegúrese que los routers que son parte de su firewall lo ignoran.
8	Echo request (generado por ping); probablemente debería permitirse.
11	Time exceeded (parece que el paquete esta en loop); probablemente debería deshabilitarse.
12	Problema de parámetros (problema con el header de un paquete); probablemente debería permitirse.

(a) los mensajes ICMP no tienen números de port origen o destino; en su lugar tienen un único tipo de mensaje ICMP. Los paquetes ICMP tampoco tienen bit ACK.

13. Network Time Protocol (NTP)

Características de Filtrado de Paquetes de NTP

NTP es un servicio basado en UDP. Los servidores NTP usan el port conocido 123 para hablar entre sí y a los clientes NTP. Los clientes NTP usan ports aleatorios superiores a 1023. Como con DNS, puede decirse la diferencia entre:

- Una consulta NTP cliente-a-servidor -- port origen encima de 1023, port destino 123.
- Una respuesta servidor-a-cliente -- port origen 123, port destino encima de 1023.
- Una consulta o respuesta servidor-a-servidor -- ports origen y destino 123.

A diferencia de DNS, NTP nunca usa TCP, y NTP no tiene operación de transferencia de zona análoga a DNS.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	123	(a)	Pregunta entrante, cliente a servidor
OUT	Int.	Ext.	123	>1023	(a)	Respuesta a pregunta UDP entrante, servidor a cliente
OUT	Int.	Ext.	>1023	123	(a)	Pregunta saliente, cliente a servidor
IN	Ext.	Int.	123	>1023	(a)	Respuesta a pregunta UDP saliente, servidor a cliente
IN	Ext.	Int.	123	123	(a)	Pregunta o respuesta entre dos servidores
OUT	Int.	Ext.	123	123	(a)	Pregunta o respuesta entre dos servidores

(a) los paquetes UDP no tienen bit ACK

Características proxy de NTP

Como una aplicación basada en UDP, no puede hacérsele proxy con SOCKS, pero puede usarse con el UDP Packet Relayer. Al emplear una jerarquía de servidores, el NTP puede configurarse para ejecutar sobre un host bastión sin usar un proxy explícito.

14. Network File System (NFS)

Características de Filtrado de Paquetes de NFS

NFS es un servicio basado en RPC. Es muy difícil manejar servicios basados en RPC con un sistema de filtrado de paquetes, ya que normalmente los servidores no usan números de port predecibles. Mientras que los números de port que usan son demasiado impredecibles para un sistema de filtrado de paquetes, no son tan impredecibles como para que un atacante

los pueda encontrar. (Si no hay otra cosa, el atacante podría tratar de enviar pedidos NFS a todos los ports para ver cuál responde como se esperaría que respondiera un servidor NFS).

Generalmente NFS se provee sobre UDP; hay servidores y clientes NFS basados en TCP disponibles, pero son raros y poco usados. NFS es algo inusual para un servicio basado en RPC en el que normalmente se usa un número de port predecible (port 2049).

Características proxy de NFS

Sobre los protocolos basados en RPC es tan complicado aplicar un filtrado de paquetes como un servicio proxy. El NFS es problemático para implementar con proxy porque trabaja con grandes cantidades de datos que están intercambiándose constantemente, y en donde el retardo es muy notable para el usuario final. Un servidor que haga proxy de NFS necesita atender múltiples conexiones transfiriendo paquetes grandes a alta velocidad.

15. Network Information Service/Yellow Pages (NIS/YP)

Características de Filtrado de Paquetes de NIS/YP

NIS/YP es un servicio basado en RPC, generalmente provisto sobre UDP (ídem NFS), en donde los servidores no usan un número de port predecible. Esta característica lo hace complicado para armar las reglas de filtrado, pero adivinar un número de port no resulta algo tan difícil de averiguar para un atacante.

Características proxy de NIS/YP

Al ser un protocolo basado en RPC, sigue consideraciones similares a las de NFS. Un proxy aislado no es adecuado para tratar con las vulnerabilidades del NIS/YP.

16. X11 Window System

Características de Filtrado de Paquetes de X11

X11 está basado en TCP. Se usa el port 6000 para el primer servidor X11 sobre una máquina. Esta elección de ports presenta otro problema para los sistemas de filtrado de paquetes: los ports X11 están en el medio del rango de ports "por encima de 1023" que muchas aplicaciones usan para ports aleatorios del lado cliente. Por eso, cualquier esquema de filtrado de paquetes que permite paquetes de entrada en ports superiores a 1023 (de modo de permitir paquetes desde servidores remotos a clientes locales) necesita ser muy cuidadoso para no permitir conexiones de entrada a servidores X11. Se puede hacer esto o bien bloqueando totalmente el acceso al rango de ports usado por estos servidores (lo que puede ser una proposición falsa debido a la posibilidad de múltiples servidores por máquina, por la

discusión más abajo) o usando un filtrado de "inicio de conexión" (mirando al bit ACK) para deshabilitar las conexiones TCP entrantes a cualquier port.

Algunas máquinas corren múltiples servidores X11. El primer servidor está en el port 6000, el segundo en el 6001, etc.

A veces en realidad se tienen múltiples disposiciones display/keyboard/mouse, pero más frecuentemente los múltiples servidores son servidores *virtuales* para alguna terminal X remota. X11 es un protocolo muy difuso, con un alto uso del ancho de banda; no se ejecuta bien sobre enlaces dial-up. Una de las soluciones que se ha adoptado (por ejemplo, el paquete *XRemote* de NCD) es ejecutar un servidor X11 virtual sobre una máquina bien-conectada (por ejemplo, enlazada por Ethernet a las máquinas sobre las que están corriendo los programas clientes), y luego hablar algún otro protocolo más frugal sobre el enlace lento entre este servidor virtual y la terminal X real.

Por eso, para bloquear el acceso a todos estos servidores, suponiendo que no se puede hacer el filtrado de 'inicio de conexión', se necesita bloquear el acceso a los ports 6000 hasta 6000+*n*, donde *n* es algún número indeterminado. No se quiere hacer *n* demasiado chico, ya que podrían exponerse a ataques a algunos servidores X11 virtuales. Por el otro lado, no se lo quiere hacer demasiado grande porque se están bloqueando ports en el rango de los ports aleatorios que podrían usarse por los clientes de otras aplicaciones. Tampoco se quiere tener sin trabajar a clientes de otros protocolos (p.ej., Telnet o FTP) simplemente porque sucede que se eligió aleatoriamente un port que está bloqueado para prevenir un acceso X11.

Hay algo a favor: la forma en que muchos sistemas operativos reservan tales ports aleatorios. Generalmente, cuando una aplicación cliente le pide al sistema operativo que reserve un port aleatorio para su uso, el kernel reserva el próximo port disponible después del último reservado (envolviendo al comienzo del espacio de número de port cuando sea necesario). Si sucede que un cliente se posesiona de un port bloqueado debido a X11, el cliente fallará. Si el usuario trata de ejecutar el cliente unas pocas veces más, en cada oportunidad el cliente obtendrá un nuevo port, y eventualmente tendrá éxito cuando el port reservado se mueva por encima del rango bloqueado.

Una aproximación común (suponiendo que no se puede hacer filtrado sobre el inicio de conexión para bloquear conexiones externas a servidores internos) es bloquear, digamos, cuatro ports en todos los hosts, y más ports en los que se supone o se sabe que la gente correrá muchos servidores X11 virtuales, p.ej., los hosts a los que la gente hace dial-in desde las terminales X de sus casas. Una aproximación más directa es usar proxying para dirigir conexiones a un host bastión que no está corriendo un sistema window. Puede hacer conexiones salientes sobre cualquier port sin preocuparse acerca de dar con el rango bloqueado, ya que no necesita un rango bloqueado.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	>1023	6000+n	(a)	conexión X11 entrante al n-ésimo servidor, cliente a servidor
OUT	Int.	Ext.	6000+n	>1023	Si	conexión X11 entrante al n-ésimo servidor, servidor

						a cliente
OUT	Int.	Ext.	>1023	6000+n	(a)	conexión X11 saliente al n-ésimo servidor, cliente a servidor
IN	Ext.	Int.	6000+n	>1023	Si	conexión X11 saliente al n-ésimo servidor, servidor a cliente

(a) ACK no está seteado en el primer paquete; sí en los restantes.

17. Protocolos De Impresión (LPR y LP)

Características de Filtrado de Paquetes de lpr

lpr está basado en TCP. Los servidores usan el port 515. Los clientes usan ports aleatorios por debajo de 1023, como los comandos '*r*' de BSD.

Sentido	Direcc. Origen	Direcc. Destino	Port Origen	Port Destino	ACK	Descripción
IN	Ext.	Int.	<1023	515	(a)	*lpr* entrante, cliente a servidor
OUT	Int.	Ext.	515	<1023	Si	*lpr* entrante, servidor a cliente
OUT	Int.	Ext.	<1023	515	(a)	*lpr* saliente, cliente a servidor
IN	Ext.	Int.	515	<1023	Si	*lpr* saliente, servidor a cliente

(a) ACK no está seteado en el primer paquete; sí en los restantes.

Características proxy de lpr

El *lpr* es un protocolo store-and-forward, capaz de poder configurarse para que haga su propio proxy. Simplemente se corre una configuración estándar sobre el servidor proxy y se lo configura para que maneje las impresoras que uno quiera o para que derive la impresión hacia otro servidor. Esto no provee mejoras en la seguridad sobre el *lpr* directo, pero dejará que se pueda cruzar un host no ruteado.

Características proxy y de Filtrado de paquetes de lp

lp en sí no provee soporte de impresión remota. Maneja la impresión a través de la red usando o bien rsh, que es cubierto por encima con otros comandos '*r*' de BSD, o *lpr*. Para determinar que se está usando la configuración de la impresora, hay que configurar una impresora remota y leer su archivo de interfase.

18. Analizando Otros Protocolos

¿Cómo se hace para analizar protocolos que no se han discutido acá? La primer cuestión a preguntarse es: ¿Realmente se necesita ejecutar el protocolo a través del firewall, o hay alguna otra forma satisfactoria de proveer o acceder al servicio deseado usando un protocolo ya soportado por el firewall?

Si realmente se necesita proveer un protocolo a través del firewall, y no se discutió anteriormente, ¿cómo determinar que ports usa, etc.? Mientras que a veces es posible

determinar esta información a partir de la documentación del programa o del protocolo, la forma más fácil de descubrirlo usualmente es preguntar a alguien tal como miembros de la lista de mailing de Firewalls.

Si se tiene que determinar la respuesta por uno mismo, la forma más fácil de hacerlo es usualmente empíricamente. Acá está lo que se debería hacer:

1. Establecer un sistema de testeo que esté corriendo lo menos posible antes que la aplicación que se quiere testear.
2. A continuación, establecer otro sistema que monitoree los paquetes hacia y desde el sistema (usando *etherfind* o *tcpdump* o algún otro paquete que nos deje mirar el tráfico sobre la red local).
3. Ejecutar la aplicación sobre el sistema de testeo y ver qué registra el sistema de monitoreo.

Se puede necesitar repetir este proceso por cada implementación cliente y cada implementación servidor que se trate de usar. Ocasionalmente hay diferencias impredecibles entre implementaciones.

Parte III: DEFINICIÓN DE UN PROTOTIPO

• PARTE III - DEFINICIÓN DE UN PROTOTIPO

ARQUITECTURA DEL PROTOTIPO

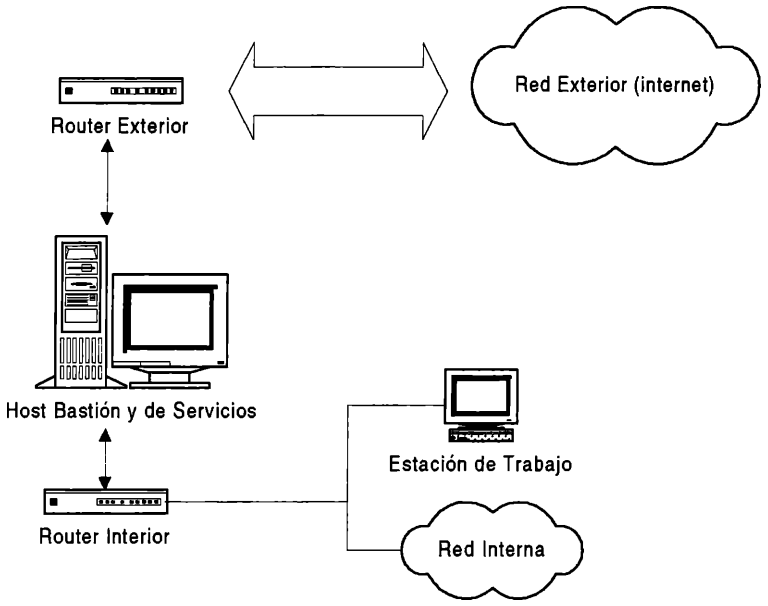
En base a lo estudiado anteriormente se definió un prototipo de firewall con arquitectura del tipo Screened-Subnet con dos routers (uno interno y otro externo) y un host bastión como único integrante de la red perimetral que se forma. Dentro de este esquema, el trabajo se focalizó en analizar dos políticas para implementación de los servicios utilizando reglas de filtrado de paquetes en ambos routers, configuradas de idéntica manera por ser así el mejor método para preservar la seguridad ante los potenciales ataques, y empleando servicios proxy para tratar a los servicios con la herramienta dentro del firewall que mejor se adecua (o complementa) en cada caso. Para proporcionar el servicio proxy se utilizó el protocolo Socksv.5 por ser un protocolo estándar de distribución libre establecido por la IETF en la RFC 1928 al proveer soporte para TCP (propio de la versión 4) y agregarse el correspondiente a UDP (facilidad incorporada en la versión 5).

Para llevar adelante la implementación de un firewall que ejemplifique una de las tantas posibilidades existentes se optó por un prototipo (mínimo) que consta de:

A nivel de Hardware

- ✓ Un router exterior que hace de router de selección, dejando pasar sólo los paquetes permitidos. Cada paquete que pase por él será controlado para que se comporte de acuerdo a las reglas de seguridad establecidas, porque es el encargado de proteger al host bastión de los ataques provenientes del exterior.
- ✓ Un host (PC) con dos plaquetas de red en donde cada plaqueta es una interfase que comunica con cada router, funcionando como host bastión y ubicada sobre la red perimetral. Corre bajo un LINUX 2.0.34. No está definida puramente como host bastión debido a que ahí también se están ejecutando servidores de mail y DNS, por lo que cumple dos funciones: host de servicios y host bastión. Tiene habilitado el "IP forwarding". En realidad éste es un caso extremo de red perimetral porque está formada por este único host. Según se indicó en la descripción de la arquitectura "Screened Host", es el único punto de contacto para establecer la comunicación entre el mundo exterior y el interior.
- ✓ Un router interior que principalmente permite seleccionar los servicios de la red interna aplicando filtrado de paquetes para minimizar el número de máquinas que pueden ser atacadas desde el host bastión. La función que cumple es la de proteger a la red interna si se logró quebrar el host bastión.
- ✓ Una red interna - representada al menos por una PC (host) con Windows 95.

✓ Uso de direcciones IP válidas que se asignaron siguiendo las recomendaciones dadas en la RFC 1597 (Ver Apéndice - página 115).



A nivel de Software

✓ Reglas de filtrado de paquetes implementables en el router interior para proveer servicios en forma directa.

✓ Proxy a nivel de aplicación para proveer indirectamente los servicios que no se pueden controlar total o parcialmente con filtrado de paquetes. Se eligió el SOCKSv5 por ser un protocolo estándar de distribución libre que fue establecido por la IETF en la RFC 1928. El Socks fue diseñado para comunicar aplicaciones cliente/servidor basadas en TCP. La versión 5 fue ampliada para permitir trabajar además con UDP.

✓ También se utilizó el AVENTAIL AutoSocks para probar el Telnet (ante algunos inconvenientes de implementación/incompatibilidad de versiones con ese servicio en particular bajo el Socks).

✓ En el host bastión se tienen dos servicios: el SendMail (servidor de correo) y el Named (servidor de DNS).

Productos utilizados:

- Linux 2.0-34: preinstalado.
- Sendmail 8-9.1: preinstalado.

- Named : preinstalado.
- Socks5-v1.0r6 : producto de distribución libre.
- Sockscap32 : para socksificar aplicaciones en Windows 95 (32 bits).
- AutoSOCKS from Aventail: para probar Telnet porque con el Sockscap no funcionó. Era una licencia de evaluación por 30 días.

POLÍTICAS DE SEGURIDAD

En el planteo del prototipo se asumió que los usuarios internos son confiables (es decir, no intentarán subvertir el firewall) y que no hay una necesidad particular de monitorear o registrar sus actividades Internet.

Una vez definido el prototipo, se establece la política a seguir basada en los criterios determinados como convenientes para la Organización.

La política de seguridad que se adopta es: *lo que no se permite de manera expresa, esta prohibido.*

Entre los protocolos y servicios considerados inseguros para permitir a través de un firewall, basándonos en los descriptos en la sección anterior, se encuentran los mencionados a continuación con la recomendación de bloquearlos específicamente (aunque se trate de una política abierta) ya sea mediante filtrado de paquetes o mediante servicio proxy (se opta por el más conveniente en cada caso):

- UDP: Por ser un protocolo sin conexión no se puede identificar el emisor y la solicitud de la respuesta. También los números de ports se pueden suplantar con facilidad. Son muy difíciles de filtrar, por lo tanto se bloquearán todos los servicios a excepción de DNS.
- TFTP: Es un servicio basado en UDP. Permite que los usuarios reciban y envíen archivos en una forma punto-a-punto. TFTP no requiere autenticación, por lo que no soporta seguridad. No se lo debería considerar ni para uso interno; en caso de necesitarlo se lo debe configurar en una forma restringida sobre su propio host, de modo tal que el servidor sólo tenga acceso a un conjunto predeterminado de archivos. TFTP no debería instalarse sobre hosts que soportan acceso externo de FTP o Web. Probablemente el uso más común de TFTP es para bajar los archivos de configuración en un router.
- FSP: Como puede ser usado para mover archivos de un sitio a otro sin ningún tipo de restricción, es muy peligroso porque entonces se podría transferir cualquier tipo de información.
- Comandos 'r' BSD, rexec, rex: son muy inseguros porque generalmente proveen acceso sin pasword.
- SNMP: No se lo permite desde el exterior de la red porque estaría dándose la capacidad de

conocer la estructura de la red interna a cualquier usuario capaz de manejar SNMP.

- RIP: si se permite este protocolo, se podría proporcionar información falsa en forma deliberada sobre el router dentro de la red interna. También se debe inhabilitar el enrutamiento fuente (ruteado de origen) en el router de selección para completar el bloqueo del protocolo; esto se hace en el Sistema Operativo de los router Cisco con el comando *'no ip source-router'*.
- *talk*: al ser un servicio que trabaja con UDP y TCP en distintos momentos, se hace casi imposible permitirlo en forma segura mediante filtrado de paquetes o haciendo proxy. Si fuera imprescindible habilitarlo, se necesitaría poner una máquina víctima sobre una red perimetral, que es no-confiable y que no tiene datos confidenciales, y permitir que los usuarios se conecten y ejecuten *talk* desde ahí.
- RPC: Es un protocolo que corre sobre TCP O UDP según sea su implementación, cuyo mayor problema es que al asignar el número de port en forma dinámica y por encima de los valores de ports bien conocidos, no permite realizar filtrado de paquetes y desarrollar un proxy para controlarlo.
- NFS: Es un servicio basado en RPC. Presenta problemas de fragilidad de autenticación y facilidad de adulteración (ver página 94).
- NIS/YP: Es un servicio basado en RPC. Presenta problemas de seguridad similares a los de NFS.
- lp, lpr: Ambos son protocolos deficientes en cuanto a la seguridad. Uno de los problemas puede presentarse con las impresoras PostScript modernas (ver página 96).

Además de los protocolos y servicios anteriores, hay otros dados a continuación que, aunque no están específicamente prohibidos en una definición de firewall, en el caso particular de nuestro prototipo no son instalables porque no se cuenta con el hardware necesario o no son necesarios por el tipo de software que se maneja:

- UUCP: Es un protocolo que trabaja haciendo transferencia de archivos sólo sobre UNIX. Como en nuestro prototipo no se utiliza UNIX en los hosts de la red interna, no es necesario habilitarlo.
- NNTP: Para un buen funcionamiento se recomienda la instalación del servidor NNTP en un host bastión *exclusivo* para este servicio, contando con que el firewall tiene más de un host bastión instalado. Esta recomendación es porque maneja mucha información, consumiéndose mucho espacio del disco y tiempo de proceso.
- POP: No es necesario permitir este protocolo porque estamos trabajando con Sendmail (vía SMTP). En el caso de habilitarlo, se tiene que considerar el riesgo de que pueden revelarse las passwords reusables, a menos que o bien no se está preocupado acerca de la sensibilidad del mail en sí o bien se tiene un canal encriptado por el cual hacer la transferencia. Si se tienen usuarios que quieren transferir mails desde otros sitios vía POP, permitirlo vía filtrado de paquetes, tal vez restringido a conexiones desde sitios específicos o a hosts específicos en nuestro lado. El proxying podría ser una solución efectiva para

algunos problemas POP, pero se necesitaría hacer como mínimo unas modificaciones menores al código.

- NTP: No es de interés en nuestro prototipo la ejecución de NTP a la Internet. De querer hacerlo, conviene usar un servidor NTP sobre un host bastión como un proxy para un servidor interno.
- IRC: Aunque es posible hacer proxy de IRC, o sólo permitir IRC a través de filtros, en general no es una buena idea su instalación a través de un firewall a causa de la debilidad de los clientes. Además, en el caso particular de nuestro prototipo, no recomendamos su uso porque el método más seguro sería poniendo una máquina víctima no confiable y sin datos confidenciales sobre una red perimetral para dejar que los usuarios se conecten a ésta para ejecutar IRC.
- *syslog*: no se instala el servidor *syslog* porque en nuestra arquitectura deberíamos configurarlo sobre el host bastión. Tal situación no es recomendada porque es muy fácil realizar un ataque de saturación de datos enviando mensajes de syslog, y de esta manera se comprometería al host bastión. De querer instalarlo, la configuración óptima es no permitir syslog entrante desde el mundo externo usando filtrado de paquetes.
- X11: En nuestro prototipo no es necesario porque no se manejan host internos con UNIX. En el caso de tener que permitirlo, habría que bloquear a los clientes en la Internet que se quieran conectar a servidores X11 en la red interna. En este caso convendría utilizar un servidor proxy X11 (como el del TIS FWTK) corriendo sobre un host bastión.

Contando con el prototipo definido, se puede proporcionar una amplia gama de alternativas de seguridad generando variantes que van desde la definición de una política llamada **abierta**, en donde se provee la mayor cantidad de servicios medianamente seguros, hasta la que ofrece una política **cerrada**, en donde se restringen los servicios a los mínimos indispensables.

Prototipo siguiendo una "Política Abierta".

En la definición de la política abierta sobre nuestro prototipo, se permiten los protocolos y servicios dados a continuación, descartando los anteriormente mencionados.

- DNS: Para esta política abierta se configuró un único servidor DNS en el host bastión: el *named*, por ser el servidor de nombres de Internet. Para su correcto funcionamiento, en los archivos de configuración del servidor se activó la directiva *forwarding*. No se puede hacer ocultamiento de información porque el host de servicios (único) es el servidor primario interno y externo de DNS. No se realizó la configuración recomendada para mayor seguridad, que sería instalar un servidor secundario sobre el host bastión y uno primario sobre un host interno debido a la arquitectura con que contamos en el prototipo. El filtrado de paquetes permite la comunicación entre el host bastión y el mundo exterior.
- SMTP: en este protocolo se usan las características normales de store-and-forward para

enviar todos los correos de entrada y salida a través del host bastión. Se usa filtrado de paquetes para restringir las conexiones SMTP de los host externos a sólo el host bastión y desde el host bastión a un servidor interno específico. Además se permite que cualquier sistema interno envíe el correo de salida al host bastión.

- FTP: Se brinda el servicio vía servidor proxy. Para el FTP común se prohibe el modo entrante usando filtrado de paquetes. En el FTP anónimo se habilita sólo el de salida. Si se contara con clientes que soporten el modo pasivo, se lo podría proveer de forma segura con filtrado de paquetes. Necesitaríamos instalar un proxy FTP sobre el host de servicios si quisiéramos soportar clientes que no usen el modo pasivo. Podríamos soportar ambos modos, pasivo y no pasivo, pero hacer esto no es algo conveniente en una estructura como la nuestra. En un sitio pequeño lo mejor es hacer uno o el otro, no tratar de mantener los dos. En nuestro prototipo soportamos el FTP común. Se deniega el uso del FTP común y del anónimo de entrada porque son muy riesgosos.
- Telnet: Se provee solamente el de salida con filtrado de paquetes. No se proveerá un servicio de Telnet entrante debido a que se puede comprometer al host bastión.
- HTTP: Se aconseja usar un host bastión dedicado, pero eso no es posible en nuestra configuración de prototipo. Esta recomendación se debe a los cuidados que hay que tener al configurar el servidor previendo las posibles formas en que alguna persona pudiera emplear para cargar un programa al sistema en algún lugar (vía mail o FTP, por ejemplo) y luego ejecutarlo vía el servidor HTTP. Otra opción a nivel servidor es hacer un proxy para HTTP, ya que además de los beneficios de seguridad ofrece los beneficios de optimizar el uso de ancho de banda de la red. Con respecto a los clientes, se configuran cuidadosamente y se advierte a los usuarios para que no los reconfiguren basados en advertencias externas.
- Gopher: Se siguen básicamente las mismas recomendaciones que para HTTP. De querer instalar un cliente, conviene usar un browser Web tal como Netscape Navigator antes que un cliente dedicado. Lo habitual es que los usuarios demanden acceso WWW antes que acceso Gopher, de modo que solamente se tendría que analizar y asegurar una sola aplicación.
- WAIS: No se instaló un servidor Wais. De querer hacerlo, se lo debe ejecutar sobre el host bastión y sobre el port estándar de WAIS. Hay servidores que no usan el port estándar sino ports aleatorios. Si no importa permitir que los usuarios accedan a todos los ports TCP, puede usarse filtrado de paquetes para examinar el bit ACK y poder permitir conexiones salientes a esos ports (pero no conexiones entrantes desde esos ports). Si ese no es el interés, conviene restringir a los usuarios a los servidores sobre el port estándar (210 de TCP) o usar proxying. Si es posible, usar un browser Web tal como Netscape Navigator para el cliente WAIS antes que un cliente dedicado. Es el mismo criterio que para Gopher.
- Archie: No se instala un servidor Archie (es un servicio UDP) porque conviene accederlo, en el caso de querer hacerlo, vía los gateways Web. Se lo define sólo de salida mediante filtrado de paquetes.

- *finger*: No se instaló un servidor *finger* porque la recomendación indicaba hacerlo sobre el host bastión (en el caso de los pedidos entrantes, en donde se ejecutaría un servicio *finger* de reemplazo). Sí se permiten los pedidos *finger* de salida haciendo filtrado de paquetes, pero conviene considerar la ejecución de un cliente *finger* de reemplazo.
- whois: En nuestro prototipo no se instala un servidor whois, por lo que no es necesario permitir las consultas whois entrantes. Las consultas salientes whois se permiten - siguiendo las recomendaciones - al menos desde las máquinas que se consideran convenientes o seguras para usar.
- ICMP: Para posibilitar el trabajo con este protocolo a través del firewall se usa filtrado de paquetes. No deben aceptarse los redireccionamientos ICMP que se difunden desde una red externa hacia la red interna porque puede causar estragos en la tabla de enrutamiento de la red interna. Se permiten los "echo requests" ICMP salientes y se limitan los "echo request" entrantes a los que vienen de máquinas con una legítima necesidad de probar la red (tal como el centro de operaciones de red del proveedor de red). Se permiten las "echo responses" ICMP en ambos sentidos. Sólo se permiten los tipos de mensajes ICMP seguros.
- *ping*: Se bloquea el que viene desde el exterior usando filtrado de paquetes porque se puede divulgar el mapa lógico de la red interna. Se permite sólo el saliente.
- *traceroute*: Se limitan los pedidos traceroute entrantes a los que vienen de máquinas con la legítima necesidad de probar la red, y a su vez se limita el rango de port usado. Se permite el traceroute de salida.

Prototipo siguiendo una "Política Cerrada".

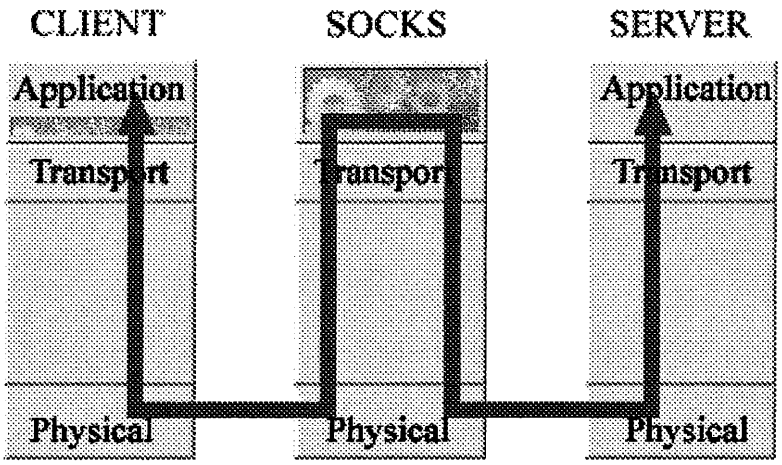
En la definición de la política cerrada sobre nuestro prototipo sólo se permiten los protocolos y servicios básicos dados a continuación, configurados siguiendo las consideraciones dadas:

- DNS, Telnet y FTP: siguen el mismo criterio que en la política abierta.
- HTTP: puede ser provisto vía filtrado de paquetes o vía un servidor proxy. Es mucho más razonable proveer HTTP vía un servidor proxy. Se permite un browser Web tal como Netscape Navigator que está soportado vía Socks.
- SMTP: Las entradas de Mail deberían ser direccionadas al host y las salidas deberían salir vía el servidor de servicios. No es aconsejable dirigir el mail a las maquinas internas. Así se tienen entradas desde un único punto. Es más fácil rutear los mail de salida desde aquí que enviar directamente los mail que teníamos. Permitir filtrar los paquetes del mail de entrada y salida solamente direccionando al host de servicios.

IMPLEMENTACIÓN DEL FIREWALL VÍA SOCKS

Para llevar adelante la implementación de las políticas descritas más arriba, se eligió el proxy SOCKSv5 por ser un estándar de la Internet, dado por la IETF en la RFC 1928, dado que es un protocolo diseñado para proveer un ambiente de trabajo para aplicaciones cliente/servidor en los dominios de TCP y UDP, para usar de manera conveniente y segura los servicios de un firewall de red. El protocolo es conceptualmente una "capa intermedia" entre la capa de aplicación y la capa de transporte, y como tal no provee servicios gateway de capa de red, tales como envío de mensajes ICMP. Como dato adicional se tuvo en cuenta que, aunque no está particularmente especificado que el Socksv5 es Y2K, al manejar las fechas/horas usando las estructuras del Sistema Operativo se asume que no habrá problemas de fechas para el año 2000 si el sistema operativo lo soporta.

Incluye dos componentes primarios: las librerías Socks para el cliente y para el servidor. La implementación del servidor se realiza a nivel de capa de aplicación y la librería cliente está entre las capas de transporte y de aplicación del cliente.



Generalmente se usa al Socks como un firewall a nivel de circuito a causa de su simplicidad. Debido a que el servidor Socks aparece como el cliente al servidor de la aplicación, los clientes están ocultos. El servidor Socks provee un único punto de acceso para el tráfico externo; autentica y autoriza los pedidos, establece una conexión proxy, y redirige los datos. En sus orígenes se pretendía usarlo como firewall a nivel de red.

Los firewalls construidos en base a Socksv5 tienen las siguientes *ventajas*:

- Simplicidad y flexibilidad.
- Acceso de red transparente a través de múltiples firewalls.
- Fácil desarrollo de métodos de autenticación y encriptación.
- Desarrollo rápido de nuevas aplicaciones de red.

- Extensión simple de las políticas de seguridad de red.
- Screening y filtrado de tráfico de red flexible.

Nota : Para Socksv5 los clientes pueden pasar nombres de host no resueltos para que los servidores Socksv5 los resuelvan. Socks va a trabajar si el cliente o los servidores Socks5 pueden resolver un host.

Socks realiza cuatro *funciones básicas*:

1 Atender pedidos de conexión.
2 Establecer el circuito proxy entre el cliente y el servidor.
3 Transmitir los datos de la aplicación.
4 Realizar autenticación.

Actualmente se encuentran disponibles las versiones: SOCKSv4 y SOCKSv5. La ventaja de la v5 es que incluye soporte proxy de UDP, extiende el ambiente de trabajo para incluir esquemas de fuerte autenticación generalizados (variando desde la forma básica de nombre_de_usuario/password hasta el trabajo con Kerberos), y extiende el esquema de direcciones para abarcar direcciones de nombre de dominio y de IPv6.

El paquete Socks5 incluye los siguientes clientes de aplicación socksificados: telnet, ftp, archie, ping, traceroute, finger, y whois. También incluye un socksificador de cliente dinámico, el runsocks, para socksificar dinámicamente aplicaciones como Netscape. Se proveen tres formas flexibles de configurar las aplicaciones cliente socksificadas:

- Configuración por defecto.
- Configuración de ambiente.
- Configuración de libsocks5.conf.

La aplicación cliente realiza un requerimiento al SOCKS para poder comunicarse con el servidor. Este requerimiento incluye la dirección del servidor, el tipo de conexión y la identidad del usuario. Cuando un cliente quiere establecer una conexión a un objeto que es alcanzable sólo vía firewall, debe abrir una conexión TCP o UDP - según corresponda - al port de socks apropiado sobre el sistema servidor de socks. Si el pedido de conexión es exitoso, el cliente entra a una negociación por el método de autenticación que va a usar, autentica con el método elegido, y luego envía un pedido de retransmisión. El servidor socks evalúa el pedido y, o bien establece la conexión apropiada o bien la deniega. Una vez establecido el circuito entre el cliente y el servidor, Socks transmite los datos de la aplicación entre ambos. Como todos los datos pasan por el Socks, brinda la posibilidad de poder filtrarlos, monitorearlos, auditarlos, etc. Por este motivo, no se lo considera simplemente como un firewall sino que también está en condiciones de realizar tareas administrativas en la red.

Nota: Si el método de autenticación negociado incluye encapsulación con el propósito de

chequeo de integridad y/o confidencialidad, los pedidos DEBEN ser encapsulados en la encapsulación dependiente del método.

Hay cuatro tipos de entradas en el archivo de configuración de socks5:

- Líneas **auth** - identifican los tipos de autenticación que usa el demonio socks5. El servidor requiere las líneas **auth** para realizar autenticación.
- Líneas **route** - identifican al servidor las interfaces de red que debería usar para las conexiones de salida.
- Líneas proxy - identifican al servidor cómo debería establecer una conexión proxy: directamente o a través de otro servidor proxy. Estas entradas DEBEN incluirse para hacer conexiones servidor-a-servidor (VPN).
- Líneas **permit/deny** - identifican al servidor los pedidos de conexión que debería autorizar. El orden que se le dé a estas líneas es crítico porque el servidor las chequea en orden secuencial. Cuando se detecta una coincidencia, se detiene el chequeo. Al no haber coincidencia, se deniega la autorización. Por lo tanto, debe haber al menos una entrada permit en el archivo socks5.conf.

Configuración y funcionamiento del Socksv5

El servidor Socks convencionalmente está ubicado en el port 1080 de TCP. Al ejecutarse el servidor, lee dos archivos: el *socks5.conf* (configuración del servidor) y el *libsocks5.conf* (configuración del cliente). Estos archivo generalmente se encuentran en el subdirectorio */etc/* o en */usr/local/etc/.* También se puede cambiar su ubicación seteando la variable de entorno "--with-srvconffile=filename" que indique dónde ir a buscarlos.

En el archivo *socks5.conf* se da la información necesaria para que el servidor pueda determinar:

- La interface que usa para alcanzar una dirección.
- Cuando el servidor debería conectarse directamente a una dirección.
- Cuando el servidor debería usar otro servidor proxy.
- Los requerimientos necesarios para hacer una conexión proxy.

Este archivo de configuración contiene seis secciones:

- *ban host*: identifica los host desde los cuales no se aceptan conexiones.
- *autenticación*: identifica el tipo de autenticación que se puede usar (desde nombre_de_usuario/password hasta el trabajo con Kerberos). Esto se define con las entradas auth. Si no se incluyen en la configuración, se asume que no hay autenticación.
- *ruteo:* en el caso de hosts multi-homed se utiliza para definir cuales son las interfaces externas e internas. Así se previene que las maquinas externas se hagan pasar por maquinas internas requiriendo que cada máquina utilice la interface que les corresponde

para establecer el vínculo. Para los hosts single-homed no se necesitan entradas de ruteo. (Nota: si se tienen conexiones PPP, se considera que estamos frente a un sistema multi-homed).

- *variables y flags*: controla el tipo y cantidad de logging y mensajes de información. Se definen con el comando "set".
- *proxies:* describe las direcciones que los clientes puede alcanzar solamente a través de otros servidores Socks e identifican como el daemon contacta al host. Se contacta al host en forma directa cuando el archivo de configuración no contiene una entrada para ese host.
- *control de acceso*: En esta sección se determina cuando el servidor permite o deniega un pedido para establecer una conexión. Se deniega una pedido si la línea de control de acceso no coincide con lo solicitado, aún después de haberse autenticado al host. Hay dos tipos de líneas: "permit" y "deny".

El orden en que se encuentran las secciones y las líneas dentro de estas secciones son muy importantes para la obtención del resultado final.

En el archivo *libsocks5.conf* se da la información usada por la librería cliente para determinar:

⇒ si la conexión es directa o a través de un servidor Socks.
⇒ qué tipo de servidor Socks usar.
⇒ el port sobre el servidor a través del cual se hace la conexión.

Hay una única sintaxis para las líneas. En primer lugar se identifica al tipo de servidor proxy, indicando si se trata de un servidor socks4, socks5, o si es una conexión directa. Como segundo parámetro se especifica un comando que puede ser: 'c' (connect), 'b' (bind), 'u' (UDP), 'p' (ping), 't' (traceroute), o '-' (cualquier comando). En tercer y cuarto lugar se dan el host destino y el port destino. Como opcionales se pueden dar listas de usuarios y/o lista de servidores para conectarse siguiendo un orden de preferencia.

Volviendo a nuestro prototipo, se hicieron diferentes pruebas sobre los archivos de configuración, quedando algunas líneas siempre fijas y variando otras para probar distintos efectos. A continuación damos los archivos de configuración con las líneas numeradas para que sea más fácil hacer la referencia aclaratoria.

Parámetros de configuración del sistema operativo (Linux) necesarios para la configuración del SOCKS.

(Named usa el 'FORWARDERS only' - se lo instaló en 163.10.10.1)

Loopback 127.0.0.1

eth0 inet addr 163.10.10.4

eth1 inet addr 10.10.0.1

SOCKS5_NOREVERSEMAP ==> SOCKS5_REVERSEMAP

SOCKS5_NOSERVICENAME ==> SOCKS5_SERVICENAME

Se cambió la configuración de estas dos variables para trabajar con DNS.

Datos de prueba del "socks5.conf"

1. auth - - -
2. route 10.10.0. - 10.10.0.1
3. route - - 163.10.10.4
4. set SOCKS5_BINDINTFC 10.10.0.1:1080
5. set SOCKS5_NOIDENT
6. permit - - 10.10.0. - - -
7. deny - - - - - 53

Significado de cada una de las líneas de configuración:

1. Se trabaja sin autenticación.
2. Se dirige todo el tráfico de la red interna (10.10.0.) a la interfase definida como interna (estamos en un host multi-homed), la 10.10.0.1.
3. Se dirige todo el tráfico de la red externa a la interfase definida como externa, la 163.10.10.4.
4. Se especifican el número de host y de port sobre el que se está ejecutando al servidor de socks5. De no ser así, se asume el host 0.0.0.0.
5. SOCKS5_NOIDENT suprime pedidos *ident* (en el caso de que se tenga la librería ident y esté habilitada la opción '--with-ident') para que el socks5 no trate de intercambiar el protocolo ident con los clientes de aplicación socksificados para cada conexión, reduciendo los timeouts.
6. Se autoriza la entrada de todos los protocolos y/o servicios que se originaron desde la red interna.
7. Se prohibe la salida de todos los protocolos que usan el port 53 (DNS).

Opcionales (no se definieron simultáneamente entre sí):

a) setenv SOCKS5_SERVER 163.10.5.147
b) setenv NONETMASKCHECK
c) setenv SOCKS5_PWFILE /etc/socks5.passwd
d) # (se uso noproxy, socks5, deny)

Explicación :

a) Se indica en una variable de entorno de Linux que el servidor de Socks está en la dirección 163.10.5.147.
b) Se indica en una variable de entorno de Linux que no se hace chequeo de máscara de red. Por defecto, el demonio chequea la máscara de red y se conecta directamente a los hosts sobre la misma subred antes de consultar al archivo de configuración.
c) Se indica en una variable de entorno de Linux la ubicación del archivo de passwords de Socks5.
d) Se usaron diferentes combinaciones de 'noproxy', 'socks5', 'deny'.

Datos de prueba del "libsocks5.conf"

1. socks5 - - - - 10.10.0.1
2. socks5 b 163.10.20.4 - - 10.10.0.1
3. socks5 p 163.10.20.4 - - 10.10.0.1
4. socks5 t - - - 10.10.0.1
5. socks5 - 163.10.20.4 telnet - 10.10.0.1
6. socks5 - 163.10.5.66 telnet - 10.10.0.1
7. socks5 - - telnet - 10.10.0.1
8. noproxy - 10.10.0. - -

Significado de cada una de las líneas de configuración:

1. Indica la dirección IP en donde está el servidor socks5 para los clientes internos.
2. Autoriza el 'bind' para un servidor socks5 desde la interfase 163.10.20.4 a la 10.10.0.1
3. Idem anterior para el 'ping'.
4. Autoriza el 'traceroute' para un servidor socks5 desde cualquier host hacia el 10.10.0.1
5., 6., 7. Autoriza el 'Telnet' desde distintos hosts a la interfase definida como interna en el host de servicios (10.10.0.1).
8. Permite la conexión directa de cualquier aplicación desde cualquier host interno a la red externa.

Opcionales (no se definieron simultáneamente):

a) set SOCKS5_DEBUG 3
b) set SOCKS5_USER (userid)

Explicación :

a) Activa el debugging, estableciendo uno de los niveles posibles. En este caso el '3' indica emisión de un listado de información.
b) Identifica el nombre-de-usuario para la autenticación de Username/Password.

En la etapa de prueba de protocolos y servicios se encontraron inconvenientes durante la utilización del Telnet. Después de hacer diferentes cambios en la configuración del cliente y del servidor de Socksv5 y del Linux, se optó por probar con otro cliente proxy, el AutoSOCKS de Aventail (versión comercial de Socks) con el cual se verificó que los problemas no eran de configuración sino del cliente que se había instalado.

• CONCLUSIONES

Llegado el momento de las conclusiones, queremos resaltar dos aspectos que para nosotros son de vital importancia: la experiencia desde el punto de vista profesional y desde el punto de vista humano.

Aspecto profesional

Luego de haber experimentado distintas técnicas de estudio y de implementación de distintos trabajos durante el transcurso de la carrera, podemos decir, que la investigación llevada adelante alrededor del tema elegido, resultó brindarnos una nueva experiencia de metodología de trabajo. Además, el hecho de enfrentar a una investigación, sin interesar - pero sin desmerecer - cual es el tema en particular, nos marca una de las alternativas laborales como para completar el nivel de formación necesario para cerrar un ciclo de estudio perteneciente a la Universidad.

El tema que elegimos para realizar el trabajo no fue desarrollado con profundidad en los contenidos de las materias que nosotros cursamos, pero en el transcurso de la investigación nos permitió afianzar conocimientos básicos necesarios para el desarrollo que habían sido adquiridos durante la carrera. Por lo que nos sentimos capacitadas para enfrentar cualquier trabajo cuyo tema escape a los contenidos de las materias debido a que tenemos la preparación necesaria y suficiente como para hacerlo.

Tenemos que señalar que el tema elegido Firewall es de total actualidad, que día a día cobra un mayor auge y popularidad como consecuencia de la inserción de la Internet en todos los ámbitos laborales y/o comerciales de la sociedad de hoy. Lo cual nos beneficia debido a que nos enfrentamos a una de las problemáticas del mercado laboral permitiendo actualizamos respecto a la existencia de los distintos productos que se ofrecen a nivel comercial y/o de libre distribución que responde al mismo.

Aspecto humano

La experiencia de haber trabajado en el Linti fue muy importante debido a que en todo momento recibimos una total colaboración de parte de la gente brindándonos su tiempo, conocimientos, experiencia, así como todo el equipamiento necesario (ambiente de redes, un router y trabajar sobre Internet) y la bibliografía que fuimos necesitando para llevar adelante nuestro trabajo.

• APENDICE - CONCEPTOS BÁSICOS

• *Protocolos, Ports y Sockets*

Una vez que los datos se rutean a través de la red y se distribuyen a un host específico, deben ser dirigidos al usuario o proceso correcto. A medida que los datos se mueven hacia arriba y hacia abajo de las capas de TCP/IP, se necesita un mecanismo para distribuir los datos a los protocolos correctos en cada capa. El sistema debe ser capaz de combinar datos desde varias aplicaciones a unos pocos protocolos de transporte, y desde los protocolos de transporte a IP. La combinación de muchas fuentes de datos en una única corriente de datos se llama multiplexado. Los datos que llegan de la red deben ser demultiplexados - divididos para distribuir a múltiples procesos. Para realizar esto, IP usa números de protocolo para identificar protocolos de transporte, y los protocolos de transporte usan números de port para identificar aplicaciones.

Números de Protocolo

Un sistema sólo necesita incluir los números de los protocolos que utiliza en realidad.

Números de Ports

Los números de port no son únicos entre protocolos de capa de transporte; los números son sólo únicos dentro de un protocolo de transporte específico. En otras palabras, TCP y UDP pueden, y lo hacen, asignar ambos los mismos números de port. La combinación de protocolo y número de port es la que especifica en forma única el proceso específico al que deben mandarse los datos.

Sockets

Los ports bien-conocidos son números de ports estandarizados que permiten que las computadoras remotas sepan a que port conectarse para un servicio particular de red. Esto simplifica el proceso de conexión porque tanto el enviador como el receptor saben de antemano que los datos dirigidos para un proceso específico usarán un port específico. Por ejemplo, todos los sistemas que ofrecen Telnet, lo ofrecen sobre el port 23.

Hay un segundo tipo de número de port llamado port alocado dinámicamente. Como el nombre lo implica, los ports alocados dinámicamente no están preasignados. Se asignan a los procesos cuando es necesario. El sistema asegura que no asigna el mismo número de port a dos procesos, y que los números asignados están por encima del rango de números de port estándar.

Para identificar únicamente a cada conexión, el port origen se asigna a un número de port alocado dinámicamente, y el número de port bien-conocido se usa para el port destino.

La combinación de una dirección IP y un número de port se llama socket. Un socket identifica únicamente a un único proceso de red dentro de toda la Internet. A veces los términos "socket" y "número de port" se usan indistintamente. De hecho, los servicios conocidos se referencian frecuentemente como "sockets bien-conocidos". En el contexto de esta discusión,

un "socket" es la combinación de una dirección IP y un número de port. Un par de sockets, un socket para el host receptor y otro para el enviador, definen la conexión para protocolos orientados a la conexión tales como TCP.

La combinación de los dos sockets identifica únicamente esta conexión; ninguna otra conexión en la Internet tiene este par de sockets.

- ***RFC 1597 - Asignación de direcciones IP para Redes Privadas (Concepto general)***

Describe métodos para preservar el espacio de direcciones IP al no reservar direcciones IP globalmente únicas para los hosts privados; a la vez se permite una completa conectividad entre todos los hosts dentro de una empresa así como entre todos los hosts públicos de diferentes empresas.

Como consecuencia de la proliferación de la tecnología TCP/IP, incluso fuera de la Internet, un gran número de empresas no conectadas usan esta tecnología y sus capacidades de direccionamiento para comunicaciones intra-empresariales.

Los hosts dentro de las empresas que usan IP pueden particionarse en tres categorías:

⇒ los que no requieren acceso a hosts en otras empresas o sobre la Internet;

⇒ los que necesitan acceso a un número limitado de servicios externos (p.ej. E-mail, FTP, netnews, login remoto) que pueden manejarse por gateways de capa de aplicación;

⇒ los que necesitan acceso de capa de red fuera de la empresa (proporcionado vía conectividad IP).

Por razones de seguridad, muchas empresas usan gateways de capa de aplicación (p.ej. firewalls) para conectar sus redes internas a la Internet. La red interna usualmente no tiene acceso directo a la Internet, por eso solo uno o más host de firewall son visibles desde la Internet. En este caso la red interna puede usar números IP no únicos. (Nota: las interfaces de los routers sobre una red interna usualmente no necesitan estar accesibles directamente desde el exterior de la empresa).

La Internet Assigned Numbers Authority (IANA) reservó los siguientes tres bloques del espacio de direcciones IP para redes privadas:

10.0.0.0	-	10.255.255.255
172.16.0.0	-	172.31.255.255
192.168.0.0	-	192.168.255.255

Al primer bloque se lo conoce como el "bloque de 24 bits"; al segundo como el "bloque de 20 bits", y al tercero como el "bloque de 16 bits". Las direcciones dentro de este espacio de direcciones privado serán únicas sólo dentro de la empresa.

Como antes, cualquier empresa que necesita espacio de direcciones globalmente único se requiere que lo obtenga desde un registro de Internet. A una empresa que solicita direcciones IP para su conectividad externa nunca se le asignarán direcciones de los bloques

definidos más arriba.

Se llaman "privados" a los hosts que no necesitan tener conectividad de capa de red fuera de la empresa. Son los que usan el espacio de direcciones privadas. Los hosts privados pueden comunicarse con todos los otros hosts dentro de la empresa, tanto públicos como privados. Sin embargo, no pueden tener conectividad IP hacia algún host externo. Pero todavía tienen acceso a los servicios externos vía relevos de capa de aplicación.

Mover un host de privado a público o viceversa involucra un cambio de direcciones IP.

Debido a que las direcciones privadas no tienen un significado global, la información de ruteo acerca de las redes privadas no deberá propagarse sobre los vínculos interempresariales, y los paquetes con direcciones origen o destino privadas no deberían reenviarse sobre esos enlaces. En el caso de los routers en las redes que no usan el espacio privado de direcciones, especialmente los de los proveedores de servicio de Internet, se espera que estén configurados para rechazar la información de ruteo referente a las redes privadas. Si tal router recibe tal información el rechazo no deberá tratarse como un error de protocolo de ruteo.

La ventaja obvia de usar espacio de direcciones privado para la Internet sin limitación es conservar el espacio de direcciones globalmente único sin usar donde la unicidad global no se requiera.

Las empresas en sí ganan mucha flexibilidad en el diseño de red teniendo más espacio de direcciones a su disposición que el que podrían obtener del pool globalmente único. Esto posibilita esquemas de direccionamiento convenientes operacional y administrativamente así como caminos de crecimiento más fáciles.

El uso del espacio de direcciones privadas provee una elección segura para las empresas, evitando las colisiones una vez que es necesaria la conectividad al exterior.

Es conveniente observar que, aunque no sea necesario renumerar los hosts, con el Classless Inter-Domain Routing (CIDR) una empresa que está conectada a la Internet puede ser inducida a renumerar sus hosts públicos, al cambiar de Proveedor de Servicio de Red. Las herramientas para facilitar la renumeración (p. ej. DHCP) lo harían ciertamente menos importante. (También se observa que la clara división de hosts públicos y privados y la necesidad resultante de renumerar hace más difícil la conectividad externa no controlada, de modo que de alguna manera la necesidad de renumerar podría verse como una ventaja.

Una estrategia recomendada es diseñar primero la parte privada de la red y usar el espacio privado de direcciones para todos los enlaces internos. Luego planear subredes públicas en los lugares necesarios y diseñar la conectividad externa.

Este diseño no es fijo permanentemente. Si cierto grupo de hosts requiere cambiar de estado más adelante, se lo puede realizar renumerando sólo los hosts involucrados e instalando otra red física si se necesita.

Usar múltiples (sub)redes IP sobre el mismo medio físico tiene muchos peligros latentes.

Pasar a un host de ser público a privado involucrará un cambio de dirección y en la mayoría de los casos, conectividad física.

Se recomienda principalmente que los routers que conectan a las empresas con las redes externas se configuren con los filtros de paquetes y de ruteado apropiados en ambos extremos del enlace a fin de prevenir fuga de información de paquete o de ruteado. Una empresa también debería filtrar cualquier red privada de la información de ruteado entrante para protegerse de las situaciones de ruteado ambiguas que pueden ocurrir si se rutea al espacio de direcciones interno apuntando hacia el exterior.

Si dos sitios de la misma empresa necesitan conectarse usando un proveedor de servicio externo, pueden considerar usar un tunel IP para prevenir escapes de paquetes de la red privada.

Una posible aproximación para evitar las fugas de RRs de DNS es ejecutar dos nameservers, un servidor externo autoritativo para todas las direcciones IP globalmente únicas de la empresa y un servidor de nombres interno autoritativo para todas las direcciones IP de la empresa, tanto públicas como privadas. Para asegurar la consistencia entre ambos servidores se los debería configurar sobre los mismos datos, de los cuales el nameserver externo sólo recibe una versión filtrada.

Los analizadores sobre todos los hosts internos, públicos y privados, consultan sólo al servidor interno. El servidor externo resuelve las consultas de los analizadores externos a la empresa y está enlazado al DNS global. El servidor interno dirige todas las consultas para información fuera de la empresa al servidor de nombres externo, por eso todos los hosts internos pueden acceder al DNS global. Esto asegura que la información referente a los hosts privados no llega a los analizadores y servidores de nombres externos a la empresa.

A pesar de que usar el espacio de direcciones privado mejora la seguridad, no es un sustituto para medidas de seguridad dedicadas.

Con el esquema descripto, muchas grandes empresas necesitarán solamente un bloque relativamente pequeño de direcciones IP del espacio globalmente único. La Internet se beneficia a través de la conservación del espacio global - que extenderá su tiempo de vida. Las empresas se benefician a partir del incremento de flexibilidad dado por un espacio de direcciones privado relativamente extenso.

- ***La Tabla de Ruteado***

Los gateways rutean datos entre redes pero todos los dispositivos de red, tanto hosts como gateways, deben tomar decisiones de ruteado. Para muchos hosts, las decisiones de ruteado son simples:

* Si el host destino está en la red local, el dato se dirige (distribuye) al host destino.
* Si el host destino está en una red remota, el dato se envía a un gateway local.

Debido a que el ruteado está orientado a la red, IP hace decisiones de ruteado basadas en la porción de red de la dirección. El módulo IP determina la parte de red de la dirección IP de destino chequeando los bits de orden superior de la dirección para determinar la clase de la

dirección. La clase de la dirección determina la porción de la dirección que IP usa para identificar la red. Si la red destino es la red local, se le aplica la máscara de subred local a la dirección destino.

Después de determinar la red destino, el módulo IP busca la red en la tabla de ruteo local (también llamada la forwarding table). Los paquetes se rutean hacia su destino según se dirige por la tabla de ruteado. La tabla de ruteo puede construirse por el administrador del sistema o por protocolos de ruteado, pero el resultado final es el mismo; las decisiones de ruteado son simples miradas a la tabla.

Sobre un sistema UNIX, se pueden ver los contenidos de la tabla de ruteo con el comando 'netstat –nr'. La opción '-r' muestra la tabla de ruteo, y la '-n' muestra la tabla en formato numérico.

Ejemplo :

peanut% netstat -nr

Tabla de Ruteo:

Destination	Gateway	Flags	Refcnt	Use	Interface
127.0.0.1	127.0.0.1	UH	1	298	lo0
default	128.66.12.1	UG	2	50360	le0
128.66.12.0	128.66.12.2	U	40	111379	le0
128.66.2.0	128.66.12.3	UG	4	1179	le0
128.66.1.0	128.66.12.3	UG	10	1113	le0
128.66.3.0	128.66.12.3	UG	2	1379	le0
128.66.4.0	128.66.12.3	UG	4	1119	le0

La primer entrada de la tabla es la ruta de retorno (loopback) para el host local. Esta es la dirección de retorno (loopback) mencionada anteriormente como un número reservado de la red. Ya que cada sistema usa la ruta loopback para enviarse datagramas a sí mismo, esta entrada está en cada tabla de ruteo de host. El flag H se setea porque es una ruta a un host específico (127.0.0.1) no una ruta a toda una red (127.0.0.0).

Otra entrada única en una tabla de ruteado es la que tiene la palabra 'default' en el campo destino. Esta entrada es para la ruta por defecto, y el gateway especificado en esta entrada es el gateway por defecto. El gateway por defecto se usa donde no hay una ruta específica en la tabla para una dirección de red destino. Por ejemplo, esta tabla de ruteado no tiene una entrada para la red 192.178.16.0. Si IP recibe algún datagrama direccionado para esta red, lo enviará vía el gateway por defecto 128.66.12.1.

En base a la visión de la tabla de ruteado se puede decir que este host (peanut) está conectado directamente a la red 128.66.12.0. La entrada en la tabla de ruteo para esa red no especifica un gateway externo; es decir, la entrada en la tabla de ruteo para 128.66.12.0 no

tiene seteado el flag G. Por lo tanto, peanut debe estar conectado directamente a esa red.

Todos los gateways que aparecen en tablas de ruteo están sobre redes conectadas directamente al sistema local. En el ejemplo anterior esto significa que, sin tener en cuenta la dirección destino, el gateway direcciona todo lo que empiece con 128.66.12. Esta es la única red a la que está conectada directamente peanut, y por eso es la única red a la que peanut puede distribuir directamente los datos. Los gateways que peanut usa para alcanzar al resto de la Internet deben estar en la subred de peanut.

- ***Sitio***

Se conoce con este nombre a cualquier parte de la organización que posee computadoras y recursos relacionados con la red. Dichos recursos incluyen:

◊ Estaciones de trabajo
◊ Computadoras anfitrión y servidores
◊ Dispositivos de Interconexión: routers, bridges, etc.
◊ Servidores de terminal
◊ Software para red y aplicaciones
◊ Cables de red
◊ Información en archivos y bases de datos
◊ Otros dispositivos que tengan acceso a la Internet

Por eso un "sitio" puede ser un usuario final de servicios Internet o un proveedor de servicio tal como una red de nivel medio.

- ***Comando forwarders***

El comando forwarders de UNIX le dice al servidor que, si no conoce por si mismo la información (o bien desde su propia zona de información o desde su cache), debería reenviar la consulta a un servidor especifico y dejar que este otro servidor obtenga la respuesta, antes que tratar de contactar servidores por toda la Internet en un intento de determinar la respuesta por sí mismo. En el archivo de configuración /etc/named.boot, se establece la línea forwarders para apuntar al servidor falso sobre el host bastión; el archivo tambien necesita contener una linea esclava para decirle que solo use los servidores sobre la linea forwards, aun si los forwarders son lentos en responder.

BIBLIOGRAFÍA

- Internet Security - "Building Internet FIREWALLS"
D. Brent Chapman and Elizabeth D. Zwicky / Edit. O'Reilly & Associates, Inc. - 1995

- "Redes Globales de Información con Internet y TCP/IP". Principios básicos, Protocolos y Arquitecturas
Douglas E. Comer / Edit. 3ra. edición

- "Internet y Seguridad en Redes"
Karanjit Siyan, Ph.D. - Chris Hare / Edit. Prentice Hall - 1995

- RFC 1597 - "Address Allocation for Private Internets"
IETF Network Working Group - Marzo 1994

- RFC 1928 - "SOCKS Protocol Version 5"
IETF Network Working Group - Marzo 1996

- RFC 1929 - "Username/Password Autentication for SOCKS V5"
IETF Network Working Group - Marzo 1996

- RFC 1961 - "GSS-API Autentication Method for SOCKS V5"
IETF Network Working Group - Junio 1996

- Internet Firewall FAQs
Marcus J. Ranum
<ftp://ftp.greatcircle.com/pub/firewalls/FAQ>

- Securing a Network
<http://www.iss.net/vd/checklist.html>

- *UniverCD - Documentación de la Serie CISCO*
CD - Cisco

Tabla de Contenidos

INDICE