

Redes: Ataques y Defensas

La persona que piensa que su problema puede ser solucionado con la criptografía, no entiende su problema y tampoco entiende la criptografía.

(Butler Lampson o Roger Needham)

Los ataques más comunes a los sistemas en red involucran (en orden):

1. Desbordamiento de buffers (*stack overflows*)
2. Robo (adivinanza o fisgoneo) de passwords
3. Debilidades en los protocolos de red

Ninguno de estos se evita con la criptografía, y no todos se evistan con los *firewalls* (cortafuegos).

Los primeros dos tipos de problema ya se han discutido. Hablamos ahora del tercero.

Problemas con Protocolos de Red

Sistemas como Unix/Linux o Windows vienen con muchos servicios de red preconfigurados y habilitados. En las redes locales, hay varios ataques posibles:

- Manuel puede hacer *sniffing* (fisqueo), poniendo su interfaz LAN en modo *promiscuo* y copiando el password de Alicia cuando ella se conecta.
- Aunque esto es más difícil cuando la red usa conexiones punto-a-punto a través de *switches*, los mismos tienen que ser correctamente configurados para evitarlo. El paquete `dsniff` es muy ilustrativo ...
- Manuel puede esperar que Alicia se haya conectado y “capturar” su sesión usando mensajes ARP para falsificar un mapeo entre el número IP de Alicia y la dirección MAC de él. Después, puede hacer (por ejemplo):

```
rm -rf *
```

- Para que Alicia no se dé cuenta hasta que sea demasiado tarde, Manuel ataca su máquina para desconectarla. Por ejemplo, algunos sistemas basados en DHCP aceptan cualquier cambio de la *máscara de subnet* incluso en el medio de una sesión, y efectivamente desaparecen.
- Alternativamente, se puede atacar a NFS, a NIS, a RPC, etc., todos los cuales tienen vulnerabilidades.

Ataques a TCP/IP

Aparte de los problemas en LANs, las redes TCP/IP son vulnerables porque los protocolos no fueron diseñados pensando en la seguridad.

Resumen del **Three-way Handshake** de TCP:

- Un cliente manda un segmento inicial (con el bit SYN) para solicitar una nueva conexión, y propone un **número de secuencia inicial (ISN)**.
- El servidor responde con un segmento con los bits SYN y ACK, y otro número para la conexión inversa.
- El cliente contesta con un segmento de datos, con el bit de ACK encendido.
- Los ISN deben garantizar que no haya solapamiento con números de secuencia de segmentos viejos.

Esquemáticamente:

$C \rightarrow S$	C, S, SYN, I_C
$S \rightarrow C$	S, C, SYN, I_S, ACK, I_C
$C \rightarrow S$	$C, S, ACK, I_S, datos$

SYN Flooding

Manuel manda muchos segmentos SYN, pero desecha los SYN+ACK de respuesta o nunca los contesta.

El servidor tiene que mantener múltiples conexiones “semi-abiertas”, al menos hasta que se produzca un *timeout* (usualmente en el orden de 15 minutos). Esto es un ejemplo de **denial of service (DOS)** (*denegación de servicio*).

Si Manuel sólo hace esto, será fácil después identificarlo. Para esconderse, Manuel puede emplear *spoofing de IP*: altera el campo de Dirección Fuente en los datagramas IP que envía. En caso del ataque mencionado, no necesita ver las respuestas.

Initial Sequence Number Prediction

Como vimos, en una LAN, Manuel puede escuchar el tráfico entre C y S e insertar sus propios paquetes. En una red WAN, generalmente no le van a llegar las respuestas de S, pero en muchos casos no hace falta leerlos si puede insertar paquetes con números de secuencia correctos:

¿Cómo puede Manuel conocer los números de secuencia si no está escuchando el tráfico?

M→S	C, S, SYN, I_M
S→C	S, C, SYN, I_S , ACK, I_M
M→S	C, S, ACK, I_S , <i>datos</i>

El segundo mensaje no le llega a Manuel, pero si puede predecir sus contenidos, puede generar el tercer mensaje. Esto significa predecir I_S .

En muchas implementaciones, esto es fácil hacer por que cada nuevo I_n es igual a $I_{n-1} + K$, por un constante K (ej. 128,000). Manuel puede realizar pruebas de antemano con conexiones verdaderas a S, para deducir el valor de K que utiliza S.

El tercer paquete del protocolo contiene el primer segmento de datos. Como ya se comentó anteriormente, en muchos casos Manuel puede enviar datos sin necesidad de ver las respuestas.

Una solución efectiva, tanto a este problema como al de los *SYN Floods*, es usar **SYN cookies**.

La idea es que S genera un número aleatorio nuevo (un “cookie”) cada minuto. Si le llega una solicitud de conexión responde con un I_S compuesto por un *hash* criptográfico de los números IP y puertos de fuente y destino más el cookie actual, pero *no mantiene el estado semi-abierto*.

Efectivamente, el I_S contiene información derivada del estado, que no se puede decodificar.

Cuando le llega a S un datagrama correspondiente al tercer paso del protocolo, puede recalcular el hash para verificar que se trata de un cliente legítimo, siempre que el cookie sigue siendo válido. Sólo tiene que recordar los cookies que ha generado durante el último minuto. Si C es legítimo pero lento, su intento de conexión será rechazado y tendrá que probar de nuevo.

Fragmentación de Paquetes

El protocolo IP permite a los enrutadoras fragmentar los datagramas cuyo tamaño sea mayor que el *MTU* de una red. Cada fragmento tiene una copia del encabezado del datagrama original con un campo llamado *offset* que permite el reensamblaje en el anfitrión destino.

En el ataque **Teardrop**, Manuel envía “fragmentos” de un datagrama, pero con errores en el *offset*. Si el receptor no tiene suficiente cuidado, puede sobrescribir parte de su memoria y causar un DOS. En particular, el **Ping Of Death** es un paquete “ICMP ECHO” de tamaño mayor que 64 Kb, que consiste de múltiples fragmentos.

Una variante es poner un *offset* negativo de tal manera que el paquete reensamblado tiene alterado el número de puerto destinatario. Esto es suficiente para engañar a muchos *firewalls* que filtran paquetes entrantes según su número de puerto.

Land : Manuel envía un paquete SYN con los campos de fuente y destino iguales a la dirección de la víctima. Muchas implementaciones de TCP/IP no esperan esto y pueden caerse.

Smurf : Manuel hace “ping” (*ICMP Echo Request*) a la dirección de *broadcast* de una red remota, poniendo la dirección IP de su víctima como fuente. Todas las máquinas en la red local le responden a la víctima, que a veces no puede manejar el tráfico; el método fue inventado para tomar control de los servidores de IRC (*Internet Relay Chat*). En efecto, la red local es un *amplificador*. Para cerrar este hueco, en 1999 se cambió la especificación de ICMP para que los “ping”s a una dirección *broadcast* no se contesten.

El *Distributed Denial of Service* (DDOS) de Febrero 2000 es un ejemplo más sofisticado de **Smurf**, que logró afectar a muchos sitios importantes de e-comercio en el Web.

Algunos de estos ataques pueden ser evitados con un filtro en el enrutador. Por ejemplo, un paquete de afuera de nuestra red no debe tener una dirección de fuente perteneciente a la red misma. Sin embargo, una solución general todavía no se conoce. Una propuesta es que cuando un router entrega un paquete, también genera un mensaje ICMP con detalles del salto anterior, el salto siguiente, etc., y se lo envíe al destinatario final. Esto se hace con baja probabilidad (ej. 1 en 20.000) pero sirve para trazar el origen de inundaciones.

Los sistemas de enrutamiento son vulnerables:

- En vez de interpretar un `Redirect` o `Destination Unreachable` en el contexto de una conexión específica, algunas implementaciones viejas lo aplican a toda comunicación entre un par de anfitriones. Esto puede dejar que Manuel “secuestre” una máquina.
- Manuel manda un datagrama con la opción `loose source routing` de IP, pero aparentemente desde una máquina “de confianza”, que indica una ruta que pase por su propia red. Los estándares requieren que el receptor responda *por la misma ruta*, aunque sea desconocida para él. La mejor defensa es simplemente ignorar los datagramas con dicha opción.
- Es bastante fácil inyectar paquetes **RIP** (*Routing Information Protocol*) falsos. En general, los anfitriones y enrutadoras los van a creer.
- Los protocolos más nuevos, como **OSPF** (*Open Shortest Path First*) tienen un campo de “autenticación”, pero es fácil falsificar (es una contraseña que se puede captar por la red).
- La mejor defensa es que las enrutadoras sean muy “desconfiadas” – deberían estar configuradas para saber cuales rutas pueden aparecer legalmente en una conexión física determinada.

Sondeo

Antes de atacar a una red, Manuel puede dedicar esfuerzos a crear una radiografía de su víctima. Le interesa por ejemplo el número de máquinas que tiene, con sus números IP, sus sistemas de operación, y los servicios que estos ofrecen (en muchos casos sin que sus administradores esten conscientes). Hay varias maneras de recolectar este tipo de información. Cuando se hace sistemáticamente, esto se llama *port scanning*, y puede aplicarse a todos los IP en el rango de la red victima, con todos los puertos “interesantes” en cada uno.

En general, se comienza con un *ping* para ver si la máquina está escuchando. Sin embargo, si la red está protegida por un firewall, éste a veces rechaza los *pings* desde afuera, para evitar sondeos.

Para eludir esta protección, Manuel envia paquetes ICMP tipo *Echo Reply*, que el firewall no va a filtrar. Si la dirección de destino es inválida, la enrutadora de la red victima (o el mismo firewall) responde con otro paquete *ICMP* de error. Si no pasa nada, es que el anfitrión sí existe ...

Una vez armada la lista de posibles victimas, Manuel tiene varias opciones para probar sus puertos, por ejemplo:

Sondeo simple : Se solicita una conexión al puerto. Para que la máquina no lo registre en sus bitacoras, Manuel no responde al *SYN+ACK*.

Sondeo de FIN : Envía un paquete *FIN* al puerto. Si no hay servicios allí, el anfitrión generalmente responde con un *RST*. En cambio si hay servicios, el *FIN* será descartado silenciosamente. Funciona mejor cuando la red en sí es confiable.

Del lado de la victima, los *Sistemas de Detección de Intrusos* observan el patrón de paquetes que llegan y suenan la alarma si ven algo sospechoso. Para eludirlos, Manuel puede variar sus patrones de ataques, usando muchos IPs de origen distintos, probando máquinas y puertos en orden aleatorio, con largos intervalos de tiempo aleatorios entre ellos. Esto es sumamente difícil detectar, sobre todo si la red tiene mucho tráfico legítimo.

Defensas

Control de Configuración

La seguridad de una red depende de un buen control de la configuración de los equipos que la componen:

- Que estén usando una versión actualizada del sistema de operación, con todos los *patches* al día.
- Que no haya huecos evidentes (ej. archivo de passwords modificable).
- Que los passwords “de fábrica” se hayan cambiado.
- Que haya disciplina organizacional para mantener todo esto en todos los equipos.

El problema es que se necesita invertir recursos y tiempo para lograr esto, y puede ser difícil explicarlo a la alta gerencia y a los usuarios.

Existen algunas herramientas útiles que ayuden al administrador, por ejemplo:

- Programas para hacer *scanning* de vulnerabilidades, ej. **SATAN** o **Nessus**.
- Utilitarios para mantener consistencia entre múltiples sistemas, ej. **rdist** o **CVS**.
- Detectores de modificaciones, ej. **Tripwire**

Firewalls

Cuando es problemático controlar todos los equipos en forma individual, una opción popular es encerrarlos dentro de una “cerca” y vigilar el tráfico entrando y saliendo, mediante un **cortafuegos**.

Un firewall es más que simplemente un enrutador, anfitrión o combinación de estos que controla el acceso a la red interna. Es un *enfoque* de seguridad que existe para implementar una *política* claramente definida. Puede:

- Prohibir que ciertos servicios (ej. NFS) sean accesibles desde “afuera”.
- Evitar ataques basados en manipulación de rutas, ICMP etc.
- Controlar acceso a ciertos anfitriones internos por parte de ciertos anfitriones externos.
- Concentrar decisiones de seguridad en un solo sitio.
- Funcionar, al menos parcialmente, en forma transparente para los usuarios internos.
- Esconder información sobre la estructura de la red interna – máquinas, subredes, dominios etc.
- Mantener registros de intentos de acceso, y de ataques.

Desventajas

- Bloquea acceso a ciertos servicios que los usuarios pueden necesitar, ej. X11, NFS, FTP etc. A veces se puede paliar la situación usando “tunneling”.
- Las *puertas traseras* (**back doors**): ej. si un usuario conecta un modem a la máquina en su oficina. Difícil evitar.
- No protege contra **data-driven attacks**, ej. virus. Esto requiere controles en cada anfitrión (y una política que determina cuales son dichos controles). Los virus se pueden comunicar por FTP, HTTP, email etc., pero requieren acciones intencionales del usuario para ser activados.
- Un firewall puede afectar al flujo eficiente (*throughput*) de datos entre la red interna y el mundo externo.
- Se concentra toda la seguridad en un sólo sitio.
- Si el problema es fuga de información por parte de “traidores”, un firewall no da mucha protección: es fácil mandar información en un mensaje de correo, o llevarla en un diskette.
- Del mismo modo, los ataques que vienen desde “adentro” no son controlables por un firewall. Algunas redes corporativas tienen firewalls internos para este escenario.
- No hay protección contra nuevos tipos de ataque que no fueron previstos en el diseño.

Estrategias de Seguridad

Privilegios

Hay dos clases de política que se pueden adoptar, tanto para acceso a servicios (quienes son los usuarios o sitios remotos que pueden acceder a cuales servicios), como en el diseño del firewall:

- *Todo es permitido al menos que sea expresamente prohibido*
- *Todo es prohibido al menos que sea expresamente permitido*

El segundo obedece el *Principio de Privilegio Mínimo*; por lo tanto es más seguro pero puede molestar más a los usuarios. Hay numerosos ejemplos de fallas de seguridad cuando este principio se ha ignorado.

Defensa en Profundidad

Es importante no confiar en un sólo mecanismo de seguridad:

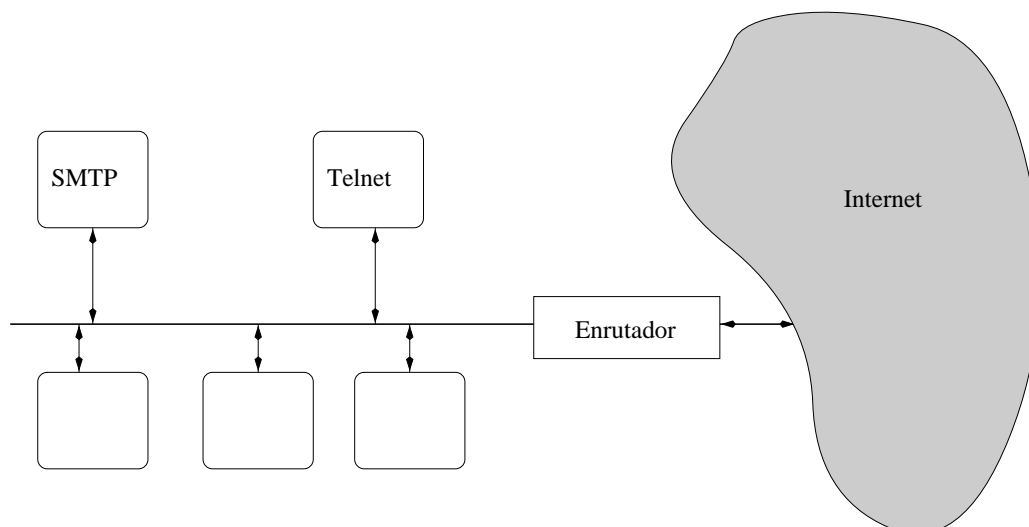
- Un firewall debe ser complementado por seguridad en los anfitriones.
- Un buen firewall consiste de varios componentes que se apoyan mutuamente, ejemplos: un filtro de paquetes que rechaza paquetes que supuestamente ya fueron rechazados por un filtro anterior; si una máquina no debe recibir correo, poner filtros en el firewall pero también quita los programas de correo de la máquina.

Filtros a Nivel Paquete

Un **enrutador** (*router*) tiene como objetivo interconectar redes distintas al nivel de paquete, ej. datagramas IP. La mayoría de los enrutadores modernos tienen facilidades para la implementación de *reglas de filtraje*, basadas en:

- Dirección IP de fuente y destino
- Puerto local TCP o UDP de fuente y destino
- Tipo de protocolo (TCP, UDP, ICMP, ...)
- Interfaz física en que el paquete llegó, o en que saldrá

El enrutador no entiende nada de protocolos de aplicación. Sin embargo, puede aprovechar el hecho de que algunos puertos tienen una función convencional en el Internet:



El enrutador permite especificar acciones a tomar en base a estos criterios, ej.:

Tipo	IP Fuente	IP Destino	P. Fuente	P. Destino	Acción	Obs
tcp	*	123,4,5,6	> 1023	23	Sí	<i>Telnet</i>
tcp	*	123,4,5,7	> 1023	25	Sí	<i>SMTP</i>
tcp	*	123,4,5,8	> 1023	25	Sí	<i>SMTP</i>
tcp	129,6,48,254	123,4,5,9	> 1023	119	Sí	<i>NNTP</i>
udp	*	123,4.*.*	> 1023	123	Sí	<i>NTP</i>
*	*	*	*	*	No	<i>Default</i>

Aunque el enrutador debe tratar a cada paquete por separado, y no puede mantener información de “estado” respecto a conexiones, sabe que un segmento TCP para abrir una conexión no tiene su bit *ACK* encendido, mientras todo otro segmento sí lo tiene. Entonces fácilmente puede prohibir intentos de conexión a ciertos puertos (se supone que los anfitriones rechazarán a segmentos que no corresponden a conexiones abiertas). Esto no funciona con UDP, por supuesto.

Hay dos servicios comunes (FTP y X11) que requieren que un anfitrión externa establezca conexiones con uno interno, en ambos casos a través de puertos arbitrarios. En el caso de FTP, a veces es posible invertir la dirección de la conexión (no el flujo de datos) con el comando *PASV*, pero no todos los servidores lo entienden. X11 típicamente usa puertos en el rango 6000 – 6100, así que es razonable bloquear conexiones a estos desde afuera.

Problemas con Filtros de Paquete

- La especificación de las reglas es compleja (ej. permitir Telnet de algunos sitios pero no en general), y no se pueden probar antes de poner en servicio.
- Los enrutadores no mantienen muchos registros de sus acciones. Tampoco pueden discriminar entre distintas clases de usuario.
- Es muy difícil manejar datagramas fragmentados (no es asunto para un enrutador). Existe software de filtraje para anfitriones tipo Unix que puede hacerlo (screen), pero implica usar un anfitrión como enrutador.
- Algunos no toman en cuenta el puerto de origen del paquete. Esto limita su utilidad.
- Servicios basados en UDP son muy difíciles de filtrar, ej. RPC usa el portmapper para asignar puertos dinámicamente.
- Algunos no discriminan según la interfaz física por la cual llega el paquete (ej. para detectar intentos de falsificar direcciones IP – una dirección de fuente que es de la red interna en un paquete que viene de afuera).
- Algunos no obedecen las reglas en el orden en el cual son especificadas, sino que las reordenan “por eficiencia”, cambiando la semántica.
- El lenguaje de especificación es propietario y distinto para cada marca de enrutador.

Pasarelas

Para tener más inteligencia que un filtro, es necesario que el firewall entienda algunos de los protocolos de aplicación.

Un **dual-homed gateway** es un anfitrión que se ubica entre la red interna y externa, en lugar del enrutador. No hace *forwarding* de paquetes IP, por lo tanto todo acceso desde afuera tiene que conectarse a algún puerto en la pasarela, cada uno atendido por un **proxy** que puede implementar una política de seguridad particular para su protocolo.

base. In this situation, there is still some hope, since the vandal may leave traces on the firewall, and may be detected. If the firewall is completely destroyed the private network can undergo attack from any external system and reconstructing the course of an attack becomes nearly impossible.

In general, firewalls can be viewed in terms of reducing the zone of risk to a single point of failure. In a sense, this seems like a bad idea, since it amounts to putting all of one's eggs in a single basket, but practical experience implies that at any given time, for a network of non-trivial size, there are at least a few hosts that are vulnerable to break-in by even an unskilled attacker. Many corporations have formal host security policies that are designed to address these weaknesses, but it is sheer foolishness to assume that publishing policies will suffice. A firewall enhances host security by funneling attackers through a narrow gap where there's a chance of catching or detecting them first. The well-constructed medieval castle had multiple walls and interlocking defense points for exactly the same reason.

Firewalls and Their Components

There may be a hundred combat postures, but there is only one purpose: to win. — Heiho Kaden Sho

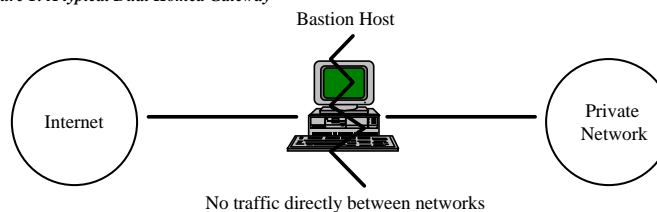
In discussing firewalls there is often confusion of terminology since firewalls all differ slightly in implementation if not in purpose. Various discussions on USENET indicate that the term "firewall" is used to describe just about any inter-network security scheme. For the sake of simplifying discussion, some terminology is proposed, to provide a common ground:

Screening Router — A screening router is a basic component of most firewalls. A screening router can be a commercial router or a host-based router with some kind of packet filtering capability. Typical screening routers have the ability to block traffic between networks or specific hosts, on an IP port level. Some firewalls consist of nothing more than a screening router between a private network and the Internet.

Bastion host — Bastions are the highly fortified parts of a medieval castle; points that overlook critical areas of defense, usually having stronger walls, room for extra troops, and the occasional useful tub of boiling hot oil for discouraging attackers. A bastion host is a system identified by the firewall administrator as a critical strong point in the network's security. Generally, bastion hosts will have some degree of extra attention paid to their security, may undergo regular audits, and may have modified software.

Dual Homed Gateway — Some firewalls are implemented without a screening router, by placing a system on both the private network and the Internet, and disabling TCP/IP forwarding. Hosts on the private network can communicate with the gateway, as can hosts on the Internet, but direct traffic between the networks is blocked. A dual homed gateway is, by definition, a bastion host.

Figure 1: A typical Dual Homed Gateway



3

Dado que la pasarela es el punto de control, es también un blanco de ataques. Es altamente recomendable que:

- La pasarela no corra procesos innecesarios
- Los usuarios no puedan hacer login en ello

- Se evalúe cuidadosamente la conveniencia de protocolos como NFS o NIS que no se van a “exportar” al mundo externo

Cualquier anfitrión muy reforzado se llama un **bastión**.

Ejemplo: un *proxy* para Telnet (entrante) corre en una pasarela. Cuando llega un segmento de “abrir conexión”:

- Chequea la dirección IP del origen del segmento
- Decide si lo va a aceptar, según su política
- Autentifica al usuario, quizás con S/KEY o similar, y verifica con cual anfitrión interno quiere conectarse
- Si aprueba, crea una sesión TCP al anfitrión indicado
- Pasa todo el tráfico TCP en ambas direcciones
- Toma un registro de sus acciones

Algo similar se puede hacer para FTP, SMTP etc. Incluso la pasarela puede filtrar el protocolo como tal, ej. para prohibir que un cliente FTP externo haga PUT. Ventajas de esta estructura:

- Es posible esconder los nombres y estructura de la red interna, con excepción de los anfitriones que ofrecen servicios.
- La autenticación y registro pueden ser tan robustos como se quiere
- Se centraliza el control en un sólo sitio
- Las reglas de filtro para el enrutador son menos complejos – puede rechazar todo tráfico que no va a la pasarela

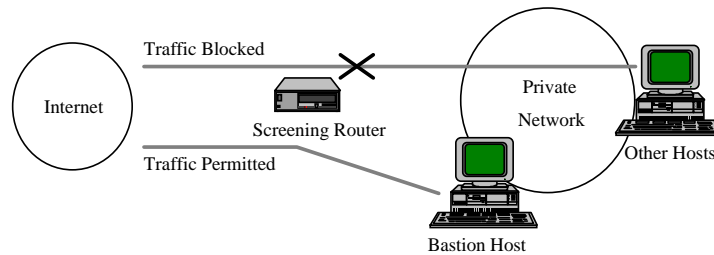
Una desventaja es que se modifica el comportamiento de los clientes, y/o de los usuarios externos.

Otros Diseños

Un principio básico de la seguridad de datos es que es mejor tener dos niveles de protección que uno. Una configuración bastante popular para un firewall es el **screened host**, que combina un filtro de paquetes con una pasarela:

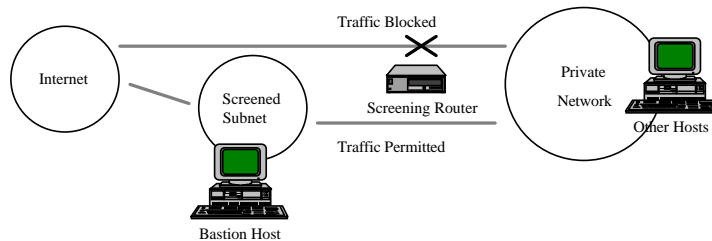
Screened Host Gateway — Possibly the most common firewall configuration is a screened host gateway. This is implemented using a screening router and a bastion host. Usually, the bastion host is on the private network, and the screening router is configured such that the bastion host is the only system on the private network that is reachable from the Internet. Often the screening router is configured to block traffic to the bastion host on specific ports, permitting only a small number of services to communicate with it.

Figure 2: A typical Screened Host Gateway



Screened Subnet — In some firewall configurations, an isolated subnet is created, situated between the Internet and the private network. Typically, this network is isolated using screening routers, which may implement varying levels of filtering. Generally, a screened subnet is configured such that both the Internet and the private network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked. Some configurations of screened subnets will have a bastion host on the screened network, either to support interactive terminal sessions or application level gateways.

Figure 3: A typical Screened Subnet



Application Level Gateway (or "proxy gateway") — Much of the software on the Internet works in a store-and-forward mode; mailers and USENET news collect input, examine it, and forward it. Application level gateways are service-specific forwarders or reflectors, which usually operate in user mode rather than at a protocol level. Generally, these forwarding services, when

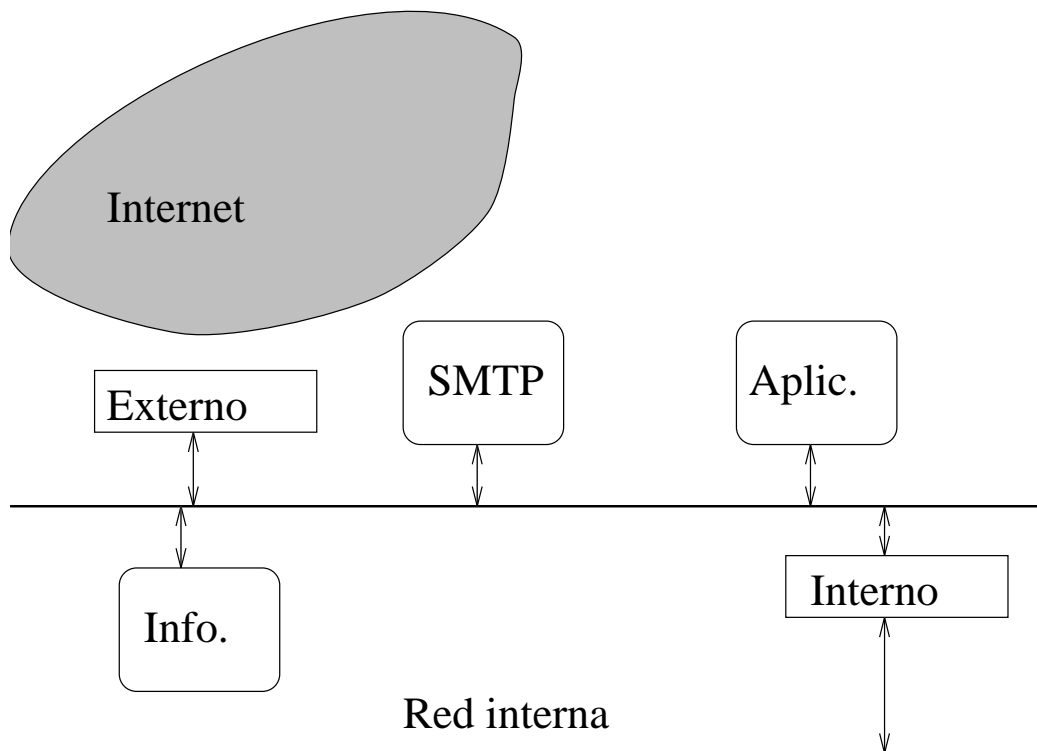
Las reglas para el filtro son, por ejemplo:

- Rechazar paquetes para servicios que no queremos ofrecer
- Rechazar paquetes con la opción **source-routing** de IP
- Rechazar paquetes de "iniciar conexión" destinados a

cualquier anfitrión interno que no sea la pasarela

La pasarela corre los *proxies*, y actúa como servidor de correo (este último no necesita ser implementado como *proxy*). Los *proxies* también sirven para permitir a anfitriones internos llamar a servicios externos (ej. WWW, FTP, etc.)

Para redes grandes con mucho tráfico tras la “frontera”, la configuración más elaborada se llama un **screened subnet**. Adicionalmente, reduce la vulnerabilidad porque reparte las funciones entre varios sitios:



La red “intermedia” se llama el **perímetro**, o el **DMZ** (*demilitarized zone*). El enrutador externo pasa solamente:

- Tráfico de aplicaciones entre la pasarela y el Internet
- Correo desde el servidor de correo y el Internet
- Tráfico de FTP, WWW etc. entre el Internet y el servidor de información

El enrutador interno comunica solamente entre la red interna y los sistemas en el DMZ:

- Tráfico de aplicaciones: la pasarela
- Correo: el servidor de correo
- FTP, WWW etc.: el servidor de información

Ventajas de este diseño:

- Ningún sistema interno es alcanzable directamente desde el Internet, y vice versa (igual que el *dual-homed gateway*), pero se elimina la necesidad de que la pasarela sea doblemente conectada. Esto aumenta la eficiencia.
- Los dos enrutadores proveen cierta redundancia, en el sentido que el adversario tendría que penetrar a ambos.
- Se puede configurar DNS de tal manera que sólo los sistemas en el DMZ se conocen desde afuera.
- La pasarela puede implementar cualquier política de autenticación.
- En casos excepcionales, se puede “bajar la guardia” para permitir conexiones directas entre sistemas internos y determinados sistemas externos “de confianza”. Esto tiene sus riesgos.
- Alternativamente, dichos sistemas internos se pueden colocar directamente en el DMZ. Esta es una ventaja significativa.

Caballos de Troya, Virus, y Gusanos

Genéricamente se le llama **malware** al software que intencionalmente hace algo dañino para el usuario. Aunque hay cierto solapamiento de terminología, el uso típico es:

Troyano (o *Caballo de Troya*) tiene una función visible que es benigno, por ejemplo un juego, pero “trás bamablins” hace algo distinto, para aprovecharse de los privilegios del usuario.

Gusano Un programa que se replica, usualmente a través de la red. Por ejemplo, puede abrir una conexión SMTP a otra máquina y disfrazarse como mensaje de correo electrónico.

Virus Un gusano que se esconde dentro de otro programa, para replicarse.

El artículo de Ken Thompson “*Reflections on Trusting Trust*” demuestra las limitaciones prácticas de detectar a los troyanos.

La Teoría de No-Decidibilidad de Turing demuestra que no es posible, ni siquiera en teoría, tener un “detector de virus” perfecto.

El caso más famoso de un ataque DOS (Denegación de Servicio) fue el **Gusano de Morris** en Noviembre de 1988.

Atacó a más de 6000 equipos en el Internet, todos basados en Unix BSD o sus variantes, usando vulnerabilidades en varios programas, incluyendo `sendmail` y `fingerd`, y la existencia de archivos `.rhost` (que indicaban las demás máquinas en que ésta máquina “confiaba”). Cargaba adentro un pequeño diccionario de passwords comunes, y cambiaba su propio nombre a `sh` para no ser visible en los listados de procesos.

El autor siempre ha dicho que no quería montar un DOS, sino experimentar con la idea del gusano, pero se le escapó (tenía un detector de copias que no funcionó correctamente).

El ataque fue suprimido en menos de dos días. Los que tardaron más tiempo en solucionarlo eran los administradores que desconectaron sus equipos de la red para no ser infectados

....

Un gusano o virus tiene dos componentes:

El mecanismo de replicación

Originalmente se usaban los diskettes. El gusano se escondía dentro de un archivo ejecutable y cambiaba las primeras instrucciones del código para que el CPU saltara a él antes de ejecutar el programa (los `.com` en MS-DOS siempre empezaban en la dirección `0x100`). Entre sus otras acciones, el código buscaba otros ejecutables y los infectaba con copias de sí mismo. Alternativamente, si el programa a infectar se llamaba `programa.exe`, se podía crear `programa.com` y MS-DOS lo encontraría primero. Otros virus infectan al sector de arranque o la table de particiones.

Hoy en día es mas común usar el correo electrónico. El gusano busca la libreta de direcciones del usuario (en Windows ésta se encuentra en un lugar fijo) y fabrica mensajes de correo a cada uno, con un `Subject:` y texto diseñados para provocar el interés de la víctima, más un `attachment` que contiene el gusano. Se adopta la identidad del usuario previamente infectado como remitente del mensaje. Además, se suele esconder la naturaleza del `attachment` mediante adaptación de su nombre. Por ejemplo, `cancion.wav` puede ser un ejecutable cuyo verdadero nombre es `cancion.wav.exe`.

Para que un mensaje pueda infectar a la máquina remota, se requiere que el `attachment` sea ejecutado. Afortunadamente hay algunos clientes de correo que hacen esto automáticamente ...

El *payload* (cargamento activo)

Es la parte que hace daño, por ejemplo después de ser activado por un *trigger*, puede:

- Cambiar los datos del usuario, replicarse en la red, causar el modem a llamar un número 900, robar claves criptográficos.
- Instalar una “puerta trasera” para que el autor pueda entrar fácilmente más adelante. Algunos de los virus más dañinos hacen cambios pequeños y paulatinos en un archivo, que no se van a notar en mucho tiempo, y así hacen inútiles los respaldos.
- Robar los passwords del usuario, por ejemplo los que le dan acceso a su cuenta bancaria.

Los programas anti-virus tratan de detectar los archivos infectados usando búsqueda de patrones conocidos. El diccionario de patrones se actualiza frecuentemente.

Los *virus polimórficos* tratan de evadir detección haciendo cambios aleatorios (pero no dañinos) en su propio código cada vez que se replican. Por ejemplo consisten de unas pocas instrucciones, una clave aleatoria, y un bloque encriptada con la clave. La clave cambia con cada replicación.

En este momento muchos de los virus importantes son programados como macros de Microsoft Word, y el *vector* de infección es por attachments de correo electrónico, generalmente usando Microsoft Outlook o Outlook Express.

Sin embargo, cualquier formato de archivo que permite *contenido ejecutable* es potencialmente un vector para virus: un ejemplo obvio es Java, que fue diseñado para permitir el envío de pequeños programas (applets) a través de la red. Otro ejemplo, menos obvio, es Postscript, que es en realidad un lenguaje de programación completo.

La única forma de limitar el efecto de los virus es ejecutar el contenido recibido en un ambiente muy controlado, o **sandbox**, que sólo permite ciertos tipos de interacción con el medio ambiente. Por ejemplo, en el caso de Java el sandbox es parte del JVM (*Java Virtual Machine*) e impone ciertas reglas: no se puede abrir conexiones de red sino al anfitrión de origen; no se permite crear archivos locales, etc. Evitar un DOS es más difícil: un applet puede consumir recursos de CPU prácticamente sin límite.

El concepto de sandbox es nada más que un mecanismo de control de acceso, como ya hemos visto.