Firewalls

Sources:

- *Firewalls and Internet Security* by Cheswick, Bellovin, and Rubin
- *TISC Insight*, Volume 4, Issue 1, http://www.tisc2001.com/newsletters/41.html
- Network Security Essentials, 3rd ed., by Stallings
- *Network Firewalls* by K. Ingham and S. Forrest, 2005

A *firewall* is any device, software, or arrangement of equipment that limits network access.

Today, firewalls are found in many devices, e.g., routers, modems, wireless base stations, IP switches.

Software firewalls are available for popular operating systems.

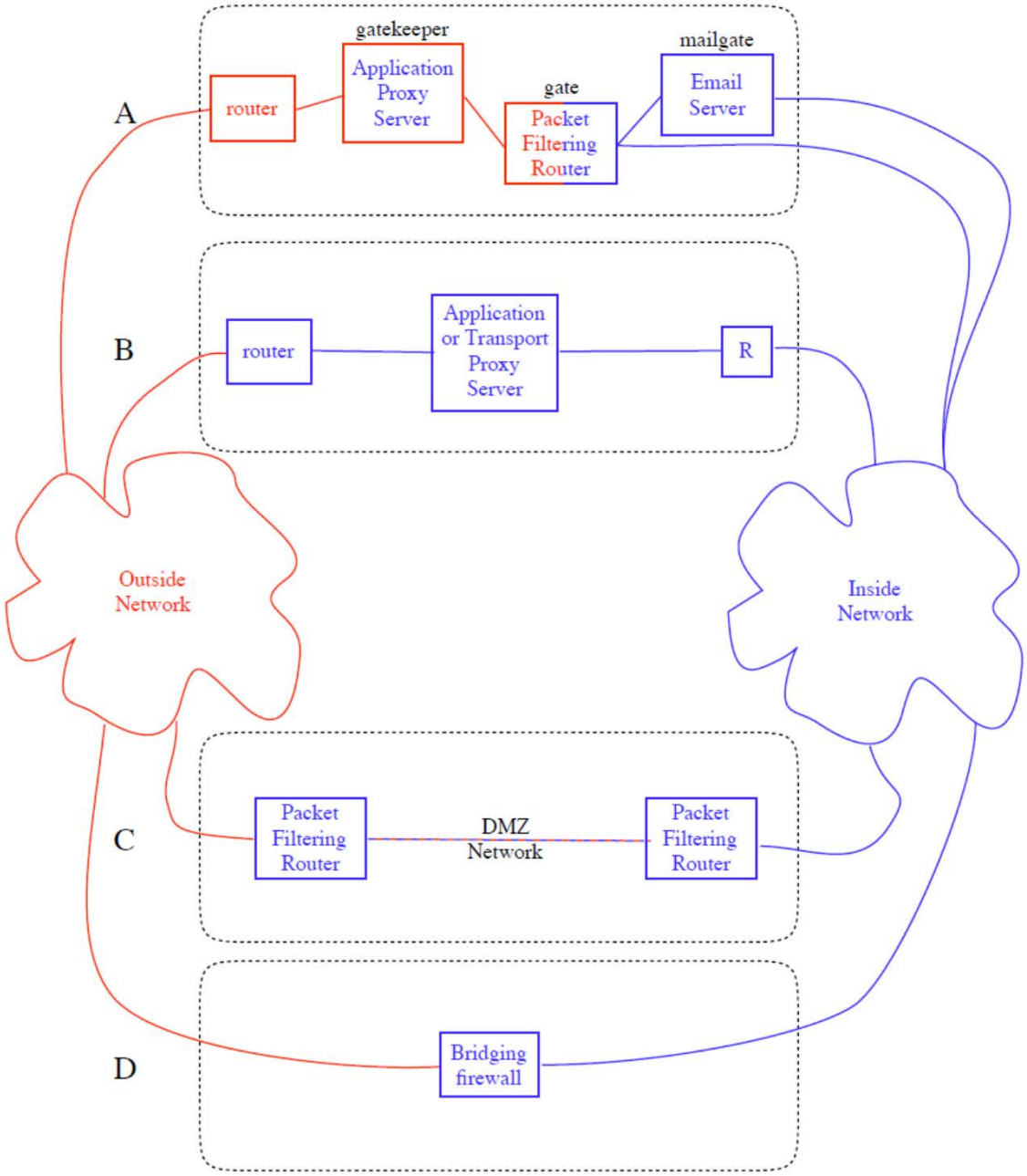
Firewalls can filter network traffic at different levels of the protocol stack.

Machines such as email and web servers are often placed in a DMZ ("demilitarized zone").

They are not allowed to make connections to machines inside the firewall.

Machines on the inside *are* allowed to make connections to DMZ machines.

If a server in the DMZ is compromised, it cannot directly attack machines on the inside.



[Ingham & Forest]

Three main categories of firewalls:

- Packet filter
- Circuit gateways
- Application gateways

Commonly, more than one of these is used.

A *dynamic packet filter* is a combination of a packet filter and a circuit-level gateway.

Packet Filters

Packet filters are incorporated into *routers*.

They work by *dropping packets* based on:

- Source and destination address
- Options in network header
- Transport protocol and flags and options in transport header
- Source and destination ports
- Interface that received packet
- Whether packet is inbound or outbound

Packet filters are *stateless*: decisions are based solely on the contents of the current packet.

Filtering may be done at the incoming interface, outgoing interface, or both.

The system administrator creates a list of acceptable machines and services and a *stoplist* of unacceptable ones.

Packet filters work well for blocking spoofed packets.

Ingress filtering is used to prevent IP packets with forged source addresses from entering a network, e.g.,

- Addresses from inside the destination network
- Loopback addresses

Egress filtering has two purposes:

1. To prevent packets containing invalid source addresses from leaving a network
2. To prevent communication to unauthorized or questionable TCP and UDP ports from valid source addresses

It works by:

- Denying all directed broadcast packets from being forwarded
- Allowing only those IP addresses assigned by the network administrator to trusted hosts to pass through the firewall

To do packet filtering correctly, you need intimate knowledge of TCP and UDP *port utilization* on a number of operating systems.

To configure a packet filter, you must:

1. Create a security policy.

2. Specify allowable types of packets.

3. Describe these types using the syntax supported by your firewall.

Most security policies require finer control than packet filters alone can provide.

Example: You might want to allow any host to connect to machine *A*, but only to send or receive mail.

Example [Cheswick *et al*]: Suppose that your security policy states:

- Inbound mail (SMTP, port 25) is permitted, but only to your gateway.
- Mail from a particular site SPIGOT is to be blocked, because they are host spammers.
- All other packets are to be rejected.

A filter that implements such a ruleset might look like:

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	Don't trust them
allow	OUR-GW	25	*	*	Our SMTP port
block	*	*	*	*	Default

To allow incoming messages to our mail server and outgoing messages to other mail servers, expanded notation is needed:

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25	*	Our packets to their SMTP port
allow	*	25	*	*	ACK	Their replies
allow	*	*	OUR-GW	25	*	Their packets to our SMTP port
block	*	*	*	*	*	Default

Question: Why isn't it safe to allow packets from port 25 on any source?

Packet filtering can't identify which *user* is causing which network traffic.

It is *difficult* to write correct filters.

Packet Filtering with State

Originally, packet filters ignored the state of a TCP connection

A machine outside the firewall can send packets that seem to be part of an established connection.

Attacks against bugs in the TCP/IP protocol stack can pass through stateless firewalls.

Stateful firewalls record and check the state of a TCP connection.

A packet filter can record both network level and transport level data.

E.g., a router can monitor the initial TCP packets with the SYN flag set and allow the return packets only until the FIN packet is sent and acknowledged.

Routing Filters

It is also important to filter routing information.

Routers must control what *routes they advertise* over various interfaces.

This can prevent outside attackers from reaching potentially vulnerable machines within a network.

Such machines can still be reached through:

- *Compromised internal hosts* reachable from the Internet
- IP *source routing*

IP source routing should be *disabled*, if possible.

Packet Filtering Performance

Packet filters provide a cheap and useful level of security.

Their filtering abilities come with the router software.

There is some performance penalty for packet filtering.

This depends on the number of filtering rules that are applied.

Application-Level Filtering

A packet filtering doesn't know much about the traffic it is limiting.

It looks only at packet headers.

Application-level filters (*gateways*, *proxies*) deal with the details of the service they are checking.

They can enforce *application level protocols*.

Hence, they are usually more complex than packet filters.

Special purpose code can be used for each desired application.

Example: An application-level filter for email will understand RFC 2822 headers and MIME-formatted attachments.

It may also be able to identify virus-infected software.

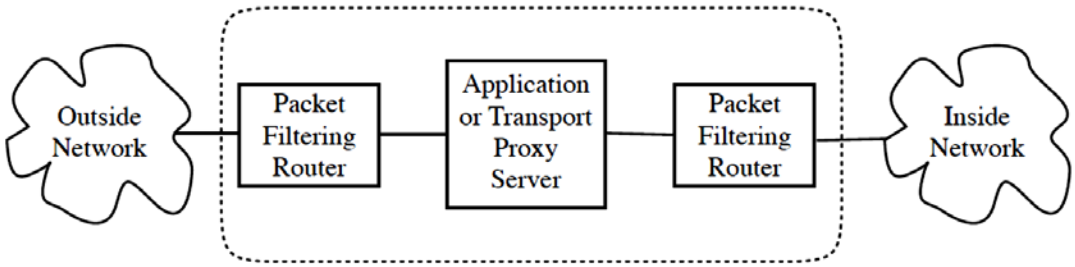


Figure 4: A network using a proxy server. Some commercial products combine all of the machines shown in the dashed lines into one to reduce the cost.

[Ingham and Forrest]

Application gateways make it easy to *log* and *control all* incoming and outgoing traffic.

Mail can be checked for indications that *proprietary or restricted data* is passing the gateway.

Web queries can be checked for conformance with *company policies*.

Dangerous mail attachments can be stripped off.

Electronic mail usually passes through an application gateway.

The principal disadvantage of application-level gateways is the need for a specialized program or user interface for each service provided.

Some commercial firewalls include a *large suite* of application-level gateways.

In practice, this means that only the most important services will be provided.

With a gateway or proxy, each packet requires *two trips* through the *complete network protocol stack*, which hurts performance.

The client program may have to be *changed* to accommodate an application gateway.

It must know the gateway's *authentication method* and communicate the *destination* to the gateway.

Transparent gateways/proxies address this.

It acts as if it were the remote machine.

Circuit-Level Gateways

Circuit-level gateways work at the TCP level.

TCP connections are *relayed* through a computer that copies bytes between two connections.

It may also *log or cache* the contents.

When a client wishes to connect to an external server, it connects to the relay host.

They relay host connects to the server.

The name and IP address of the client is usually not available to the server.

The relay host handles pathological IP streams and emits only normal, well-behaved packets.

Circuit-level gateways can *bridge* two networks that don't share any IP connectivity or DNS processing.

They are generally used to create specific connections between isolated networks.

In some cases, a circuit connection is made *automatically*, as part of the gateway architecture.

A particular TCP service might be relayed from an external host to an internal machine.

In other cases, the connection service needs to be told the desired destination.

In this case a simple protocol is used between the caller and the gateway.

Example: the SOCKS protocol

- SOCKS call replaces normal *socket call*
- All outbound traffic uses the proxy
- A system using SOCKS and TCP is *transparent* to the user

In general these relay services *do not examine* the bytes as they flow through.

They may log the number of bytes and the destination.

Uncooperative *inside users* can *subvert* the intent of the gateway by advertising unauthorized services.

Controls are also needed to ensure that external users cannot “launder” connections through the relay host.

Dynamic Packet Filters

Dynamic packet filters are the most common type of firewall technology.

They are intended to permit virtually all client software to operate while giving network administrators full control over traffic.

Transit packets are examined as by an ordinary packet filter.

The identity of outgoing packets is also noted, and incoming packets for the same connection are allowed to pass through.

Dynamic packet filters have additional features for filtering the packet contents on certain connections, such as FTP and RPC.

There are two primary ways to implement dynamic packet filters:

- Make *dynamic changes* to a conventional packet filter's ruleset.

- Implement the dynamic aspects of the filter using *circuit-like semantics*, by terminating the connection on the firewall itself.

The second option is safer than the first.

With it, the firewall connects to the *ultimate destination*.

With this option, a dynamic packet filter can be used together with an application proxy.

Distributed Firewalls

The newest form of firewall is the *distributed firewall*.

With these, each individual host enforces a shared security policy.

The policy itself is set by a central *management node* or *policy server*.

With this scheme, there is *no central point of failure*.

Machines that aren't inside a topologically isolated space can also be protected (e.g., mobile computers).

With this scheme, it is hard to allow in certain services selectively.

What Firewalls Cannot Do

Firewalls are useless against attacks from the *inside*.

This includes attacks launched via malicious code executed on an internal machine.

Firewalls act at some layer of the protocol stack.

They are “blind” to what happens at higher layers.

“Transitive trust” through a firewall is also a problem.

If A trusts B through its firewall and B trusts C, then A trusts C, whether it wants to or not.

Finally, firewalls may have bugs.