

---

## what is a firewall?

- Conceptual
- Technical
- Operational

\$Id: lecture.tex,v 1.4 2003/03/24 22:59:37 mansaxe1 Exp \$

---

## **conceptual**

A firewall is an installation designed to enhance security by regulating traffic between segments of networks.

This builds on the assumption that different networks are more or less secure.

---

## **technical**

A firewall is a number of network elements that together enforce the rules defined at the conceptual level.

These components most of the time perform functions similar to or closely resembling the router, but with forwarding conditions applied. Most routers have a subset of firewalling functions implemented. These often suffice.

---

## **operational**

A security system without guidelines and enforcement is useless.

Any firewall installation must include a policy that dictates which traffic to allow, which to block, why this is done, and what to do when things break.

There must be logging of connections and refused attempts.

The logs must be audited by competent staff.

---

## **concept – reason and design**

Why does the firewall enjoy such universal acceptance?

- The similarity to real-world doors and fences.
- The promise of a cure-all.
- The ability to patch security onto an existing network.
- The notion of *us* versus *them*.

---

## **technical – classes of tools**

Several ways exist to selectively forward IP traffic. The most used are:

- Routing filters
- Packet filters
- Packet filters, with connection state.
- Application level proxies.
- IP-aware bridges with filtering capabilities.

---

## **the routing filter**

Also known as “spoofing filter”

Enforces sane routing by ensuring packets are only allowed from directions to where answers to them would be sent.

Inside source address only allowed from inside; outside address only allowed from outside.

Also good for stopping fake source addresses in denial-of-service attacks.

Found in: Cisco IOS, and most router operating systems, and most firewall shrinkwraps.

---

## **the packet filter**

The packet filter is a device that looks in the IP packet header and decides whether to forward the packet or not based on:

- destination and source address
- source/destination port
- protocol; ie ICMP, UDP, TCP

The filter system checks its rule table and decides to pass or discard per packet.

Discarding can be set to cause ICMP error messages to be sent, or just drop the packet silently.

---

... packet filters...

Found in: Most router OS'es, most Unices (sometimes as third-party) and a pathetic implementation in NT/2K. XP is said to be better.

---

## **stateful filters**

A stateful filter is your basic packet filter but with a sense of memory.

For TCP, the filter machine only accepts SYN packets, ie connection initiators. When a SYN packet arrives, it is checked according to the rule set, and if it matches, it is let through.

---

... stateful filters...

The filter machine stores connection data based on the SYN packet.

When the target responds with SYN/ACK and matching sequence number, this is correlated with the stored SYN.

Packets that match the stream data are let through.

Connection state is torn down on FIN/ACK, or the filter will time out and self-close if no packets are sent in the session.

After closure, again only SYNs are let in.

---

...stateful filters...

For UDP, stateless in nature, a sort of quasi-state can be set up for query-response protocols like DNS. The filter records the port number pairs and sets up a simple match table which makes it possible to track the packet pairs.

Found in: Most router OS'es, most Unices (sometimes as third-party) and in the majority of Personal Firewalls.

---

## **application level proxies**

Application level proxies are protocol-aware software inside the firewall host that intercept traffic on the application level, decide whether it is acceptable and then reshape the calls and replies to suit.

Classic example is the http proxy.

Mail proxying is often done with sendmail, using it as relay.

DNS can be proxied with a caching resolve server.

These proxies need to be written for the protocol or will get very watered-down, so that they only are tcp-generic proxies.

---

... application level proxies...

A properly configured application-level proxy is probably the most safe firewall with uncut cables.

It is also the most intrusive.

Found in: webcaches like Squid or, TIS Firewall Toolkit, most shrinkwrap firewalls. Also, at times, the border between application and proxy is hazy, like Apache, which can do both.

---

## the IP-aware bridge

A different beast that does not route, but instead switches.

Typically built as a 2-port switch in software, bridging segments in the same broadcast domain, but discriminating its forwarding based on L3 info found in frames.

A clever idea, since it is invisible outside the broadcast domain.

OpenBSD pf will do this, as well as SunScreen.

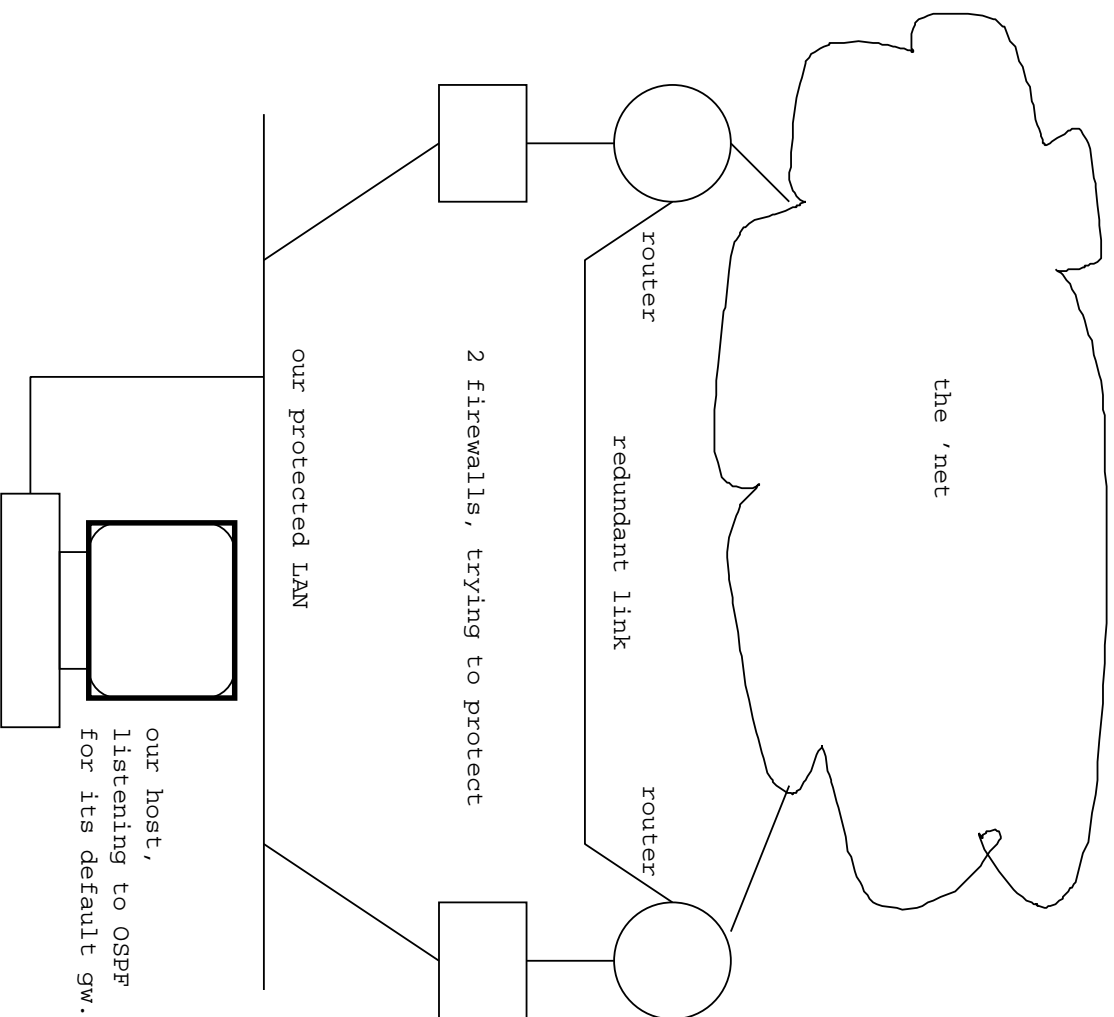
---

## death of redundancy

Most firewalls implemented as separate boxes are detrimental to dynamic routing. Exceptions would be:

- static packet filters lacking state.
- firewalling elements not in the redundant path at all.
- firewalls knowing about the routing information updates. (do such beasts exist?)
- spoofing filters may or may not be affected, depending on network design.

... redundancy deceased...



---

# **operations and philosophy**

---

## **advantages of the firewall concept**

As stated, several compelling reasons exist for using a firewall:

The notion of inside / outside suits workplace-centric computing well.

A very effective containment method for insecure applications and hosts.

Probably as good for keeping people in as for keeping people out.

---

## collateral damage

The general problem with the firewall is that computers become unreachable. This is often regarded as Good but might be detrimental to people trying to get their work done.

NAT is often bundled with firewalls:

Security-wise, NAT is as bad or good as no NAT.

Application-level proxies can become very confused, and are significantly harder to configure in a NAT situation. Traceability might suffer since addresses are no longer unique.

---

... collateral damage...

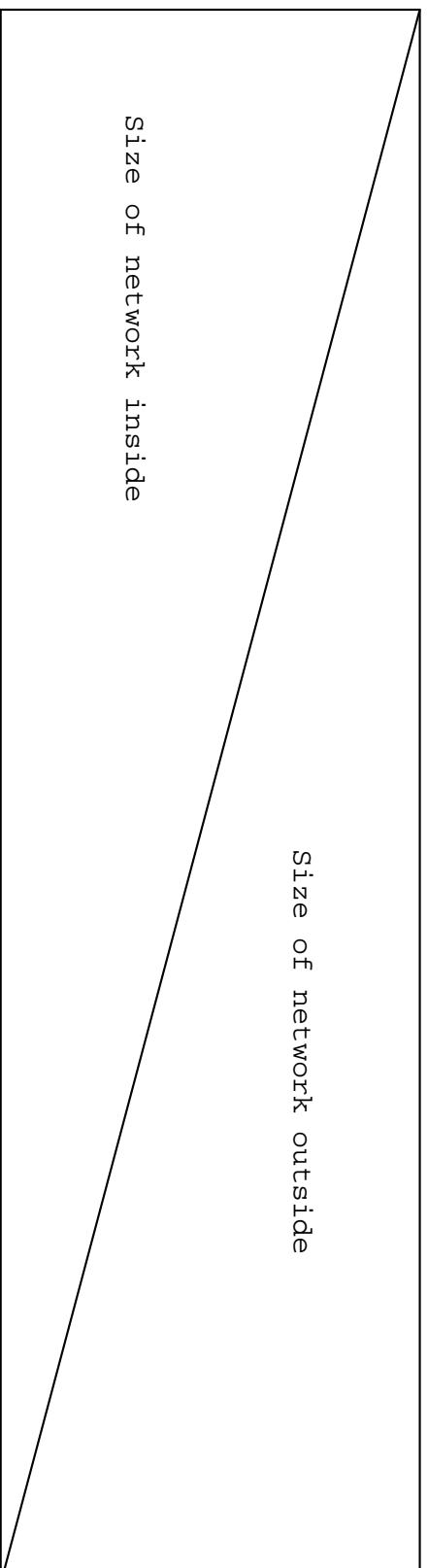
Leakage of addresses in application data will invalidate any security by obscurity.

It will become close to impossible to traverse the firewall with some applications like SIP telephony or peer-to-peer software without fragile proxy solutions.

---

# efficiency of a firewall

Efficiency, average



```
$Id: efficiency.obj,v 1.1 2003/03/24 15:28:07 mansaxel Exp $
```

---

... efficiency...

Since the efficiency of a firewall can be argued to be directly inversely related to the number of hosts it protects, there are strong reasons for keeping containment areas very small.

The extreme case is the personal firewall, or even the partitioning of multiuser systems into virtual hosts inside one mother operating system.

---

... efficiency...

Keeping partitions small also has added benefit:

The problems are close to the user.

The load on the firewall is small; no scaling issues with computers.

In the personal case, no NAT can/need be employed, thus there are no problems with in-band IP addresses like in SIP.

---

... efficiency...

The problems are management-centric:

More places where to configure.

More things that break.

More staff required.

A host setup on the net, without firewall kit, will not automatically be protected.

---

## **IPv6**

Most commercial firewall software lacks v6 support.

Some open-source systems are v6-aware, like ipf and pf, and the packet filter systems associated with the Linux kernel.

---

**the end**

questions?

opposition?

feedback?

**mansaxel@sUNET . se**