CSCI 7000-001 — Firewalls and Packet Filtering

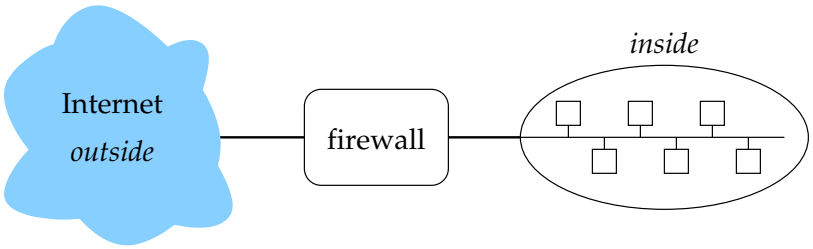
November 1, 2001

> Firewalls are the wrong approach. They don't solve the general problem, and they make it very difficult or impossible to do many things. On the other hand, if I were in charge of a corporate network, I'd never consider hooking up to the Internet without one. And if I were looking for a likely financially successful security product to invest in, I'd pick firewalls.
>
> —Charlie Kaufmann

1 What is a Firewall?

A *firewall* is a computer that:

- is "topologically between" a (*inside*) network and the (*outside*) rest of the Internet
- protects the *inside* network from attacks coming from the *outside*.



2 Firewall design principles and assumptions

- all traffic from the inside to the outside, and vice-versa, must go through the firewall
- the firewall filters (i.e., drops) network traffic according to some policy. The policy is expressed by a configurable set of *rules*.
- the firewall itself must be immune to penetration and denial-of-service attacks.

3 Why firewalls are needed/good?

- firewalls are not needed. In theory
  - inside systems can implement very good security
  - inside systems can be made just as secure as a firewall with good application management, system administration, and audit
  - firewall security is actually very primitive. System-specific security can solve a lot more problems than the ones addressed by a firewall.
- firewalls are very convenient. In practice
  - applications contain, and will always contain, an enormous amount of bugs, most of which translate into vulnerabilities
  - it is a pain in the neck to design and implement secure applications. It does make sense to ignore security in many cases. A firewall allows you to ignore security to a large extent.
  - application management, system administration, and audit are a cost
  - a firewall is a single choke point, therefore it allows easier (cheaper) administration and audit
  - a firewall performs a very simple function (it applies simple filtering rules), therefore it can be a simple piece of software, therefore it will contain less bugs than any other system inside
  - a firewall is a good place to put other network-, but not necessarily security-related services (DNS, web cache, etc.)

4 Fundamental limitations of firewalls

It is important to understand the fundamental limitations of firewalls

- often there are ways to bypass the firewall. That is, you can get to the inside network without going through the firewall. Examples:
  - dial-in connections
  - wireless networks reaching outside secure physical perimeter
- malicious or naive insiders
- viruses. A firewall can not possibly scan data at every level, therefore it is always possible for an attacker to send malicious code through some allowed data channel (e-mail, web, etc.)

5 Types of firewalls

packet filter a packet filter applies a set of rules to each incoming and outgoing IP packet. Filter rules are typically based on values in the IP header field and/or values in higher-level headers (e.g., TCP or UDP). The only thing a packet filter can do is drop packets

application-level gateway a.k.a. *proxy*

circuit-level gateway this is what we would call a *tunnel server*. An example of a circuit-level gateway is the SOCKS package. Note that SOCKS connections are not encrypted

bastion host a platform for application-level or circuit-level gateways.

6 Combination of firewalls

Different types of firewalls or multiple firewalls can work together:

- multiple firewalls can be used for the *inside network*, to create separate sub-domains with different levels of security protections. This is obviously a coarse-grained subdivision mechanism because it relies on topological separation.
- in most cases, application-level gateways are combined with packet filters

7 Defining policies with packet filters

Packet filters allow the enforcement of policies based on filtering rules. The most typical form of rule uses

- source and destination (IP) address
- type of protocol
- the source and destination port (for TCP or UDP)
- IP or TCP flags

8 Packet filters: examples

action	*source*		*destination*		*protocol*	*flags*
	IP address	*port*	*IP address*	*port*		
block	*	*	*inside*	>1023	TCP	SYN

This rule blocks all incoming TCP connections to non-privileged ports.

action	*source*		*destination*		*protocol*	*flags*
	IP address	*port*	*IP address*	*port*		
allow	*	*	mailer.*inside*	25	TCP	SYN
block	*	*	*inside*	25	TCP	SYN

This rule blocks all incoming TCP connections to port 25 for all hosts, except for the mailer host.

action	*source*		*destination*		*protocol*	*flags*
	IP address	*port*	*IP address*	*port*		
allow	*	*	mailer.*inside*	25	TCP	
block	*	*	*inside*	25	TCP	

This rule blocks all incoming TCP traffic to port 25 for all hosts, except for the mailer host.

action	*source*		*destination*		*protocol*	*flags*
	IP address	*port*	*IP address*	*port*		
block	*inside*	*	*outside*	80	TCP	SYN

This rule blocks all outgoing TCP connections to port 80.

action	*source*		*destination*		*protocol*	*flags*
	IP address	*port*	*IP address*	*port*		
block	*	*	*	*	UDP	

This rule blocks all UDP traffic.

9 Methodology

It is good to follow good software engineering practices when setting up a firewall or a set of firewalls

- specification: policy definition
- design
- testing
- configuration management, and maintenance

10 Examples using IP Filter

10.1 basic concepts

```
block in all
block out all
```

sets up defaults: block all traffic

```
# pass packets from host firewall to any destination
pass in from firewall to any
```

The action is either *block* or *pass*. The direction is either *in*, packets entering this (firewall) host, or *out* for packets that this host is sending out.

10.2 rule processing and precedence

Rules are stored in a configuration file, and are processed top to bottom. The *last* matched rule applies. The idea is to express rules starting from the most generic to the most specific.
You can short-circuit the evaluation with the `quick` keyword, so for example:

```
block in quick all
block out all
pass in from firewall to any
```

would cause the evaluation of the rule set to terminate as soon as the first rule is matched.

10.3 specifying interfaces

You can apply rules to specific interfaces:

```
block in on eth0 from localhost to any
```

drops all inbound packets from localhost coming from eth0.

10.4 specifying addresses and networks

host and network addresses and specified in the intuitive way:

```
block in on eth0 from mynet/26 to any
```

blocks packets coming from a subnet defined by the address of *mynet* with a 26 bit mask.

```
block in on eth0 from mynet/255.255.255.192 to any
```

same effect, but this time we explicitly give a mask

```
block in on eth0 from myhost to any
```

the default mask is */32* or *255.255.255.255*.
Of course, you can specify destination addresses in the same way

```
block out quick on eth0 from any to 192.168.0.0/16
block out quick on ppp0 from any to 172.16.0.0/12
block out quick on ppp0 from any to host.foo.bar
```

10.5 specifying protocols

#
```
block in on eth0 proto icmp all
```

block all incoming ICMP packets

```
pass in on eth0 proto 4 all
```

allow all IP packets carrying protocol 4 (encapsulated IP)

10.6 options and ports

```
block in on eth0 proto tcp/udp from nastyhost to any
```

blocks UDP and TCP coming from nastyhost.

```
pass in quick on eth0 proto icmp from any to any icmp-type 0
block in quick on eth0 proto icmp from any to any
```

allows ICMP packets with ICMP type 0 (`ECHO_REPLY`)

```
block out quick on eth1 proto tcp from any to any port = 23
```

the above rule blocks all outgoing TCP traffic to port 23 (Telnet).

10.7 logging packets

```
block in log quick on ppp0 from 20.20.20.0/24 to any
```

if matched, the above rule drops the packet and logs it.
You can also log passed traffic:

```
pass in log quick on eth0 from host.net.bla to any
```

11 Evaluation of Packet Filters

- does it filter inbound or outbound traffic, or both?
- what subset of attributes can you check? Protocol, source address, destination address, source port, destination port, etc.
- can you filter on established connections?
- how are "other" protocol handled? (protocols other than TCP and UDP?) For example, can you filter routing protocols? In which directions?

- can you filter on arbitrary bit fields?
- evaluate the filter language. For example, can you control the order of application of filtering rules.
- what kind of auditing capabilities do you have? Can you log rejected packets?