# Internet Firewalls: Frequently Asked Questions

Paul D. Robertson paul@compuwar.net
Matt Curtin cmcurtin@interhack.net
Marcus J. Ranum mjr@ranum.com

Date: 2004/07/26 15:34:42
Revision: 10.4

This document available in Postscript.and PDF.

## Contents

**1 Administrativia** **4**
1.1 About the FAQ . . . . . 4
1.2 For Whom Is the FAQ Written? . . . . . 4
1.3 Before Sending Mail . . . . . 4
1.4 Where Can I find the Current Version of the FAQ? . . . . . 5
1.5 Where Can I Find Non-English Versions of the FAQ? . . . . . 5
1.6 Contributors . . . . . 5
1.7 Copyright and Usage . . . . . 5

**2 Background and Firewall Basics** **6**
2.1 What is a network firewall? . . . . . 6
2.2 Why would I want a firewall? . . . . . 6
2.3 What can a firewall protect against? . . . . . 7
2.4 What can't a firewall protect against? . . . . . 7
2.5 What about viruses and other malware? . . . . . 8
2.6 Will IPSEC make firewalls obsolete? . . . . . 9
2.7 What are good sources of print information on firewalls? . . . . . 10
2.8 Where can I get more information on firewalls on the Internet? . 11

**3 Design and Implementation Issues** **11**
3.1 What are some of the basic design decisions in a firewall? . . . . 11
3.2 What are the basic types of firewalls? . . . . . 12
3.2.1 Network layer firewalls . . . . . 13
3.2.2 Application layer firewalls . . . . . 13
3.3 What are proxy servers and how do they work? . . . . . 17

3.4 What are some cheap packet screening tools? . . . . . 17
3.5 What are some reasonable filtering rules for a kernel-based packet screen? . . . . . 17
3.5.1 Implementation . . . . . 18
3.5.2 Explanation . . . . . 18
3.6 What are some reasonable filtering rules for a Cisco? . . . . . 19
3.6.1 Implementation . . . . . 19
3.6.2 Explanations . . . . . 21
3.6.3 Shortcomings . . . . . 21
3.7 What are the critical resources in a firewall? . . . . . 22
3.8 What is a DMZ, and why do I want one? . . . . . 23
3.9 How might I increase the security and scalability of my DMZ? . 23
3.10 What is a 'single point of failure', and how do I avoid having one? 24
3.11 How can I block all of the bad stuff? . . . . . 24
3.12 How can I restrict web access so users can't view sites unrelated to work? . . . . . 25

**4 Various Attacks** **26**
4.1 What is source routed traffic and why is it a threat? . . . . . 26
4.2 What are ICMP redirects and redirect bombs? . . . . . 27
4.3 What about denial of service? . . . . . 27
4.4 What are some common attacks, and how can I protect my system against them? . . . . . 28
4.4.1 SMTP Server Hijacking (Unauthorized Relaying) . . . . . 28
4.4.2 Exploiting Bugs in Applications . . . . . 28
4.4.3 Bugs in Operating Systems . . . . . 28

**5 How Do I...** **29**
5.1 Do I really want to allow everything that my users ask for? . . . 29
5.2 How do I make Web/HTTP work through my firewall? . . . . . 29
5.3 How do I make SSL work through the firewall? . . . . . 30
5.4 How do I make DNS work with a firewall? . . . . . 30
5.5 How do I make FTP work through my firewall? . . . . . 31
5.6 How do I make Telnet work through my firewall? . . . . . 31
5.7 How do I make Finger and whois work through my firewall? . . . 32
5.8 How do I make gopher, archie, and other services work through my firewall? . . . . . 32
5.9 What are the issues about X11 through a firewall? . . . . . 32
5.10 How do I make *RealAudio* work through my firewall? . . . . . 33
5.11 How do I make my web server act as a front-end for a database that lives on my private network? . . . . . 33
5.12 But my database has an integrated web server, and I want to use that. Can't I just poke a hole in the firewall and tunnel that port? 34
5.13 How Do I Make IP Multicast Work With My Firewall? . . . . . 34

**6 TCP and UDP Ports** **34**
6.1 What is a port? . . . . . 35
6.2 How do I know which application uses what port? . . . . . 36
6.3 What are LISTENING ports? . . . . . 36
6.4 How do I determine what service the port is for? . . . . . 37
6.5 What ports are safe to pass through a firewall? . . . . . 37
6.6 The behavior of FTP . . . . . 38
6.7 What software uses what FTP mode? . . . . . 39
6.8 Is my firewall trying to connect outside? . . . . . 39
6.9 The anatomy of a TCP connection . . . . . 40

**A Some Commercial Products and Vendors** **41**

**B Glossary of Firewall-Related Terms** **41**

# 1 Administrativia

## 1.1 About the FAQ

This collection of Frequenty Asked Questions (FAQs) and answers has been compiled over a period of years, seeing which questions people ask about firewalls in such fora as Usenet, mailing lists, and Web sites. If you have a question, looking here to see whether it's answered before posting your question is good form. Don't send your questions about firewalls to the FAQ maintainers.

The maintainers welcome input and comments on the contents of this FAQ. Comments related to the FAQ should be addressed to `firewalls-faq@interhack.net`. Before you send us mail, please be sure to see sections 1.2 and 1.3 to make sure this is the right document for you to be reading.

## 1.2 For Whom Is the FAQ Written?

Firewalls have come a long way from the days when this FAQ started. They've gone from being highly customized systems administered by their implementors to a mainstream commodity. Firewalls are no longer solely in the hands of those who design and implement security systems; even security-conscious end-users have them at home.

We wrote this FAQ for computer systems developers and administrators. We have tried to be fairly inclusive, making room for the newcomers, but we still assume some basic technical background. If you find that you don't understand this document, but think that you need to know more about firewalls, it might well be that you actually need to get more background in computer networking first. We provide references that have helped us; perhaps they'll also help you.

We focus predominately on "network" firewalls, but "host" or ""personal" firewalls will be addressed where appropriate.

## 1.3 Before Sending Mail

Note that this collection of frequently-asked questions is a result of interacting with many people of different backgrounds in a wide variety of public fora. *The firewalls-faq address is not a help desk.* If you're trying to use an application that says that it's not working because of a firewall and you think that you need to remove your firewall, please do not send us mail asking how.

If you want to know how to "get rid of your firewall" because you cannot use some application, do not send us mail asking for help. We cannot help you. Really.

Who can help you? Good question. That will depend on what exactly the problem is, but here are several pointers. If none of these works, please don't ask us for any more. We don't know.

- The provider of the software you're using.
- The provider of the hardware "appliance" you're using.

- The provider of the network service you're using. That is, if you're on AOL, ask them. If you're trying to use something on a corporate network, talk to your system administrator.

## 1.4 Where Can I find the Current Version of the FAQ?

The FAQ can be found on the Web at

- `http://www.compuwar.net/pubs/fwfaq/`.
- `http://www.interhack.net/pubs/fwfaq/`.

It's also posted monthly to

- `comp.security.firewalls`,
- `comp.security.unix`,
- `comp.security.misc`,
- `comp.answers`, and
- `news.answers`.

Posted versions are archived in all the usual places. Unfortunately, the version posted to Usenet and archived from that version lack the pretty pictures and useful hyperlinks found in the web version.

## 1.5 Where Can I Find Non-English Versions of the FAQ?

Several translations are available. (If you've done a translation and it's not listed here, please write us so we can update the master document.)

**Norwegian** Translation by Jon Haugsand
`http://helmersol.nr.no/haandbok/doc/brannmur/brannmur-faq.html`

## 1.6 Contributors

Many people have written helpful suggestions and thoughtful commentary. We're grateful to all contributors. We'd like to thank afew by name: Keinanen Vesa, Allen Leibowitz, Brent Chapman, Brian Boyle, D. Clyde Williamson, Richard Reiner, Humberto Ortiz Zuazaga, and Theodore Hope.

## 1.7 Copyright and Usage

# 2 Background and Firewall Basics

Before being able to understand a complete discussion of firewalls, it's important to understand the basic principles that make firewalls work.

## 2.1 What is a network firewall?

A firewall is a system or group of systems that enforces an access control policy between two or more networks. The actual means by which this is accomplished varies widely, but in principle, the firewall can be thought of as a pair of mechanisms: one which exists to block traffic, and the other which exists to permit traffic. Some firewalls place a greater emphasis on blocking traffic, while others emphasize permitting traffic. Probably the most important thing to recognize about a firewall is that it implements an access control policy. If you don't have a good idea of what kind of access you want to allow or to deny, a firewall really won't help you. It's also important to recognize that the firewall's configuration, because it is a mechanism for enforcing policy, imposes its policy on everything behind it. Administrators for firewalls managing the connectivity for a large number of hosts therefore have a heavy responsibility.

## 2.2 Why would I want a firewall?

The Internet, like any other society, is plagued with the kind of jerks who enjoy the electronic equivalent of writing on other people's walls with spraypaint, tearing their mailboxes off, or just sitting in the street blowing their car horns. Some people try to get real work done over the Internet, and others have sensitive or proprietary data they must protect. Usually, a firewall's purpose is to keep the jerks out of your network while still letting you get your job done.

Many traditional-style corporations and data centers have computing security policies and practices that must be followed. In a case where a company's policies dictate how data must be protected, a firewall is very important, since it is the embodiment of the corporate policy. Frequently, the hardest part of hooking to the Internet, if you're a large company, is not justifying the expense or effort, but convincing management that it's safe to do so. A firewall provides not only real security—it often plays an important role as a security blanket for management.

Lastly, a firewall can act as your corporate "ambassador" to the Internet. Many corporations use their firewall systems as a place to store public information about corporate products and services, files to download, bug-fixes, and so forth. Several of these systems have become important parts of the Internet service structure (e.g., `UUnet.uu.net`, `whitehouse.gov`, `gatekeeper.dec.com`) and have reflected well on their organizational sponsors. Note that while this is historically true, most organizations now place public information on a Web server, often protected by a firewall, but not normally on the firewall itself.

## 2.3 What can a firewall protect against?

Some firewalls permit only email traffic through them, thereby protecting the network against any attacks other than attacks against the email service. Other firewalls provide less strict protections, and block services that are known to be problems.

Generally, firewalls are configured to protect against unauthenticated interactive logins from the "outside" world. This, more than anything, helps prevent vandals from logging into machines on your network. More elaborate firewalls block traffic from the outside to the inside, but permit users on the inside to communicate freely with the outside. The firewall can protect you against any type of network-borne attack if you unplug it.

Firewalls are also important since they can provide a single "choke point" where security and audit can be imposed. Unlike in a situation where a computer system is being attacked by someone dialing in with a modem, the firewall can act as an effective "phone tap" and tracing tool. Firewalls provide an important logging and auditing function; often they provide summaries to the administrator about what kinds and amount of traffic passed through it, how many attempts there were to break into it, etc.

Because of this, firewall logs are critically important data. They can be used as evidence in a court of law in most countries. You should safeguard, analyze and protect yoru firewall logs accordingly.

This is an important point: providing this "choke point" can serve the same purpose on your network as a guarded gate can for your site's physical premises. That means anytime you have a change in "zones" or levels of sensitivity, such a checkpoint is appropriate. A company rarely has only an outside gate and no receptionist or security staff to check badges on the way in. If there are layers of security on your site, it's reasonable to expect layers of security on your network.

## 2.4 What can't a firewall protect against?

Firewalls can't protect against attacks that don't go through the firewall. Many corporations that connect to the Internet are very concerned about proprietary data leaking out of the company through that route. Unfortunately for those concerned, a magnetic tape, compact disc, DVD, or USB flash drives can just as effectively be used to export data. Many organizations that are terrified (at a management level) of Internet connections have no coherent policy about how dial-in access via modems should be protected. It's silly to build a six-foot thick steel door when you live in a wooden house, but there are a lot of organizations out there buying expensive firewalls and neglecting the numerous other backdoors into their network. *For a firewall to work, it must be a part of a consistent overall organizational security architecture.* Firewall policies must be realistic and reflect the level of security in the entire network. For example, a site with top secret or classified data doesn't need a firewall at all: they shouldn't be hooking up to the Internet in the first place, or the systems with the really

secret data should be isolated from the rest of the corporate network.

Another thing a firewall can't really protect you against is traitors or idiots inside your network. While an industrial spy might export information through your firewall, he's just as likely to export it through a telephone, FAX machine, or Compact Disc. CDs are a far more likely means for information to leak from your organization than a firewall. Firewalls also cannot protect you against stupidity. Users who reveal sensitive information over the telephone are good targets for social engineering; an attacker may be able to break into your network by completely bypassing your firewall, if he can find a "helpful" employee inside who can be fooled into giving access to a modem pool. Before deciding this isn't a problem in your organization, ask yourself how much trouble a contractor has getting logged into the network or how much difficulty a user who forgot his password has getting it reset. If the people on the help desk believe that every call is internal, you have a problem that can't be fixed by tightening controls on the firewalls.

Firewalls can't protect against tunneling over most application protocols to trojaned or poorly written clients. There are no magic bullets and a firewall is not an excuse to not implement software controls on internal networks or ignore host security on servers. Tunneling "bad" things over HTTP, SMTP, and other protocols is quite simple and trivially demonstrated. Security isn't "fire and forget".

Lastly, firewalls can't protect against bad things being allowed through them. For instance, many Trojan Horses use the Internet Relay Chat (IRC) protocol to allow an attacker to control a compromised internal host from a public IRC server. If you allow any internal system to connect to any external system, then your firewall will provide no protection from this vector of attack.

## 2.5 What about viruses and other malware?

Firewalls can't protect very well against things like viruses or malicious software (malware). There are too many ways of encoding binary files for transfer over networks, and too many different architectures and viruses to try to search for them all. In other words, a firewall cannot replace security-consciousness on the part of your users. In general, a firewall cannot protect against a data-driven attack—attacks in which something is mailed or copied to an internal host where it is then executed. This form of attack has occurred in the past against various versions of *sendmail*, *ghostscript*, scripting mail user agents like *Outlook*, and Web browsers like *Internet Explorer*.

Organizations that are deeply concerned about viruses should implement organization-wide virus control measures. Rather than only trying to screen viruses out at the firewall, make sure that every vulnerable desktop has virus scanning software that is run when the machine is rebooted. Blanketing your network with virus scanning software will protect against viruses that come in via floppy disks, CDs, modems, and the Internet. Trying to block viruses at the firewall will only protect against viruses from the Internet. Virus scanning at the firewall or e-mail gateway will stop a large number of infections.

Nevertheless, an increasing number of firewall vendors are offering "virus detecting" firewalls. They're probably only useful for naive users exchanging Windows-on-Intel executable programs and malicious-macro-capable application documents. There are many firewall-based approaches for dealing with problems like the "ILOVEYOU" worm and related attacks, but these are really oversimplified approaches that try to limit the damage of something that is so stupid it never should have occurred in the first place. Do not count on any protection from attackers with this feature. (Since "ILOVEYOU" went around, we've seen at least a half-dozen similar attacks, including Melissa, Happy99, Code Red, and Badtrans.B, all of which were happily passed through many virus-detecting firewalls and e-mail gateways.)

A strong firewall is never a substitute for sensible software that recognizes the nature of what it's handling—untrusted data from an unauthenticated party—and behaves appropriately. Do not think that because "everyone" is using that mailer or because the vendor is a gargantuan multinational company, you're safe. In fact, it isn't true that "everyone" is using any mailer, and companies that specialize in turning technology invented elsewhere into something that's "easy to use" without any expertise are more likely to produce software that can be fooled. Further consideration of this topic would be worthwhile [3], but is beyond the scope of this document.

## 2.6 Will IPSEC make firewalls obsolete?

Some have argued that this is the case. Before pronouncing such a sweeping prediction, however, it's worthwhile to consider what IPSEC is and what it does. Once we know this, we can consider whether IPSEC will solve the problems that we're trying to solve with firewalls.

IPSEC (IP SECurity) refers to a set of standards developed by the Internet Engineering Task Force (IETF). There are many documents that collectively define what is known as "IPSEC" [6]. IPSEC solves two problems which have plagued the IP protocol suite for years: host-to-host authentication (which will let hosts know that they're talking to the hosts they think they are) and encryption (which will prevent attackers from being able to watch the traffic going between machines).

Note that neither of these problems is what firewalls were created to solve. Although firewalls can help to mitigate some of the risks present on an Internet without authentication or encryption, there are really two classes of problems here: integrity and privacy of the information flowing between hosts and the limits placed on what kinds of connectivity is allowed between different networks. IPSEC addresses the former class and firewalls the latter.

What this means is that one will not eliminate the need for the other, but it does create some interesting possibilities when we look at combining firewalls with IPSEC-enabled hosts. Namely, such things as vendor-independent virtual private networks (VPNs), better packet filtering (by filtering on whether packets have the IPSEC authentication header), and application-layer firewalls will be able to have better means of host verification by actually using the IPSEC

authentication header instead of "just trusting" the IP address presented.

## 2.7 What are good sources of print information on firewalls?

There are several books that touch on firewalls. The best known are:

- Building Internet Firewalls, 2d ed.

  **Authors** Elizabeth D. Zwicky, Simon Cooper, and D. Brent Chapman
  **Publisher** O'Reilly
  **Edition** 2000
  **ISBN** 1-56592-871-7

- Firewalls and Internet Security: Repelling the Wily Hacker

  **Authors** Bill Cheswick, Steve Bellovin, Avi Rubin
  **Publisher** Addison Wesley
  **Edition** 2003
  **ISBN** 020163466X

- Practical Internet & Unix Security

  **Authors** Simson Garfinkel and Gene Spafford
  **Publisher** O'Reilly
  **Edition** 1996
  **ISBN** 1-56592-148-8
  **Note** Discusses primarily host security.

Related references are:

- Internetworking with TCP/IP Vols I, II, and III

  **Authors** Douglas Comer and David Stevens
  **Publisher** Prentice-Hall
  **Edition** 1991
  **ISBN** 0-13-468505-9 (I), 0-13-472242-6 (II), 0-13-474222-2 (III)
  **Comment** A detailed discussion on the architecture and implementation of the Internet and its protocols. Volume I (on principles, protocols and architecture) is readable by everyone. Volume 2 (on design, implementation and internals) is more technical. Volume 3 covers client-server computing.

- Unix System Security—A Guide for Users and System Administrators

  **Author** David Curry
  **Publisher** Addison Wesley
  **Edition** 1992
  **ISBN** 0-201-56327-4

### 2.8 Where can I get more information on firewalls on the Internet?

**Site Security Handbook** `http://www.rfc-editor.org/rfc/rfc2196.txt` The Site Security Handbook is an information IETF document that describes the basic issues that must be addressed for building good site security. Firewalls are one part of a larger security strategy, as the Site Security Handbook shows.

**Firewalls Mailing List** `http://www.isc.org/index.pl?/ops/lists/firewalls/` The internet firewalls mailing list is a forum for firewall administrators and implementors.

**Firewall-Wizards Mailing List** `http://honor.icsalabs.com/mailman/listinfo/firewall-wizards` The Firewall Wizards Mailing List is a moderated firewall and security related list that is more like a journal than a public soapbox.

**Firewall HOWTO** `http://www.linuxdoc.org/HOWTO/Firewall-HOWTO.html` Describes exactly what is needed to build a firewall, particularly using Linux.

**Firewall Toolkit (FWTK) and Firewall Papers** `ftp://ftp.tis.com/pub/firewalls/`

**Marcus Ranum's firewall related publications** `http://www.ranum.com/pubs/`

**Texas A&M University security tools** `http://www.net.tamu.edu/ftp/security/TAMU/`

**COAST Project Internet Firewalls page** `http://www.cerias.purdue.edu/coast/firewalls/`

## 3 Design and Implementation Issues

### 3.1 What are some of the basic design decisions in a firewall?

There are a number of basic design issues that should be addressed by the lucky person who has been tasked with the responsibility of designing, specifying, and implementing or overseeing the installation of a firewall.

The first and most important decision reflects the policy of how your company or organization wants to operate the system: is the firewall in place explicitly to deny all services except those critical to the mission of connecting to the Net, or is the firewall in place to provide a metered and audited method of "queuing" access in a non-threatening manner? There are degrees of paranoia between these positions; the final stance of your firewall might be more the result of a political than an engineering decision.

The second is: what level of monitoring, redundancy, and control do you want? Having established the acceptable risk level (i.e., how paranoid you are) by resolving the first issue, you can form a checklist of what should be monitored, permitted, and denied. In other words, you start by figuring out your overall

objectives, and then combine a needs analysis with a risk assessment, and sort the almost always conflicting requirements out into a laundry list that specifies what you plan to implement.

The third issue is financial. We can't address this one here in anything but vague terms, but it's important to try to quantify any proposed solutions in terms of how much it will cost either to buy or to implement. For example, a complete firewall product may cost between $100,000 at the high end, and free at the low end. The free option, of doing some fancy configuring on a Cisco or similar router will cost nothing but staff time and a few cups of coffee. Implementing a high end firewall from scratch might cost several man-months, which may equate to $30,000 worth of staff salary and benefits. The systems management overhead is also a consideration. Building a home-brew is fine, but it's important to build it so that it doesn't require constant (and expensive) attention. It's important, in other words, to evaluate firewalls not only in terms of what they cost now, but continuing costs such as support.

On the technical side, there are a couple of decisions to make, based on the fact that for all practical purposes what we are talking about is a static traffic routing service placed between the network service provider's router and your internal network. The traffic routing service may be implemented at an IP level via something like screening rules in a router, or at an application level via proxy gateways and services.

The decision to make is whether to place an exposed stripped-down machine on the outside network to run proxy services for telnet, FTP, news, etc., or whether to set up a screening router as a filter, permitting communication with one or more internal machines. There are benefits and drawbacks to both approaches, with the proxy machine providing a greater level of audit and, potentially, security in return for increased cost in configuration and a decrease in the level of service that may be provided (since a proxy needs to be developed for each desired service). The old trade-off between ease-of-use and security comes back to haunt us with a vengeance.

## 3.2 What are the basic types of firewalls?

Conceptually, there are three types of firewalls:

1. Network layer
2. Application layer
3. Hybrids

They are not as different as you might think, and latest technologies are blurring the distinction to the point where it's no longer clear if either one is "better" or "worse." As always, you need to be careful to pick the type that meets your needs.

Which is which depends on what mechanisms the firewall uses to pass traffic from one security zone to another. The International Standards Organization

(ISO) Open Systems Interconnect (OSI) model for networking defines seven layers, where each layer provides services that "higher-level" layers depend on. In order from the bottom, these layers are physical, data link, network, transport, session, presentation, application.

The important thing to recognize is that the lower-level the forwarding mechanism, the less examination the firewall can perform. Generally speaking, lower-level firewalls are faster, but are easier to fool into doing the wrong thing.

These days, most firewalls fall into the "hybrid" category, which do network filtering as well as some amount of application inspection. The amount changes depending on the vendor, product, protocol and version, so some level of digging and/or testing is often necessary.

### 3.2.1 Network layer firewalls

These generally make their decisions based on the source, destination addresses and ports (see Appendix 6 for a more detailed discussion of ports) in individual IP packets. A simple router is the "traditional" network layer firewall, since it is not able to make particularly sophisticated decisions about what a packet is actually talking to or where it actually came from. Modern network layer firewalls have become increasingly sophisticated, and now maintain internal information about the state of connections passing through them, the contents of some of the data streams, and so on. One thing that's an important distinction about many network layer firewalls is that they route traffic directly though them, so to use one you either need to have a validly assigned IP address block or to use a "private internet" address block [5]. Network layer firewalls tend to be very fast and tend to be very transparent to users.

In Figure 1, a network layer firewall called a "screened host firewall" is represented. In a screened host firewall, access to and from a single host is controlled by means of a router operating at a network layer. The single host is a bastion host; a highly-defended and secured strong-point that (hopefully) can resist attack.

*Example Network layer firewall*: In Figure 2, a network layer firewall called a "screened subnet firewall" is represented. In a screened subnet firewall, access to and from a whole network is controlled by means of a router operating at a network layer. It is similar to a screened host, except that it is, effectively, a network of screened hosts.

### 3.2.2 Application layer firewalls

These generally are hosts running proxy servers, which permit no traffic directly between networks, and which perform elaborate logging and auditing of traffic passing through them. Since the proxy applications are software components running on the firewall, it is a good place to do lots of logging and access control. Application layer firewalls can be used as network address translators, since traffic goes in one "side" and out the other, after having passed through an application that effectively masks the origin of the initiating connection.

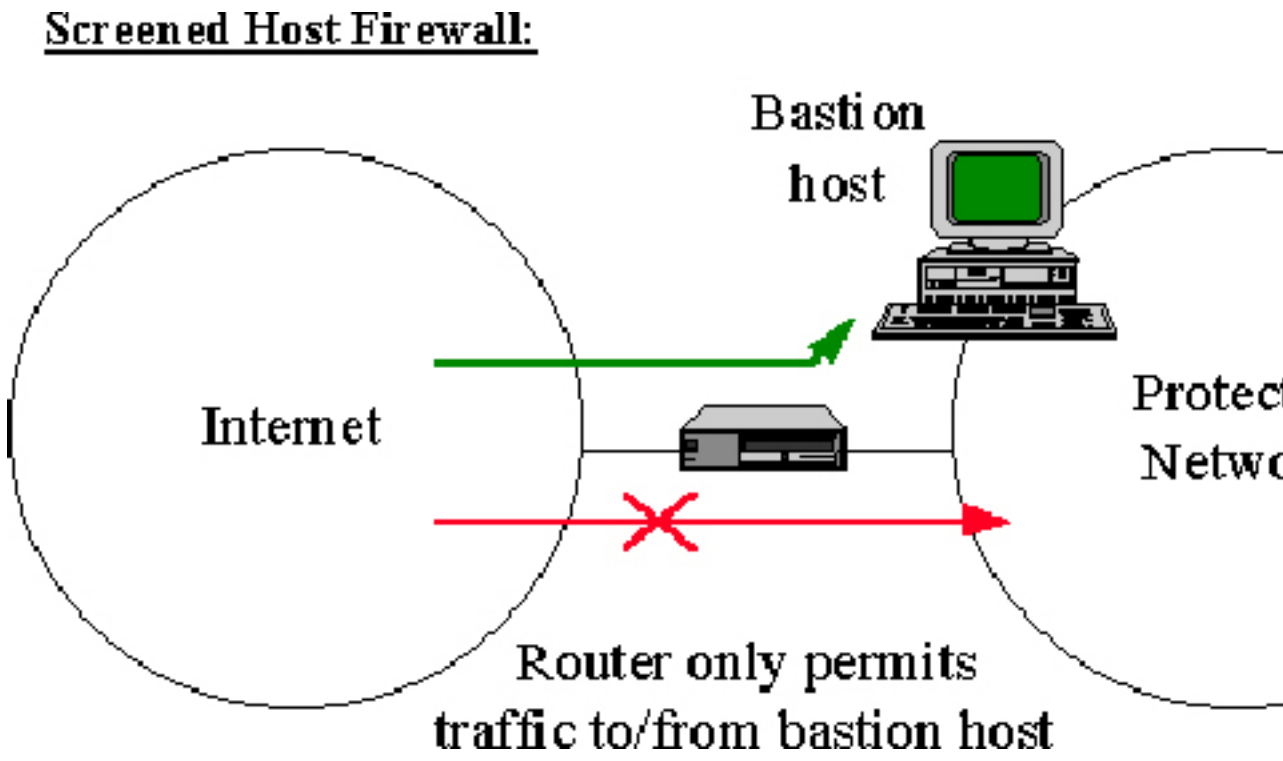


Figure 1: Screened Host Firewall

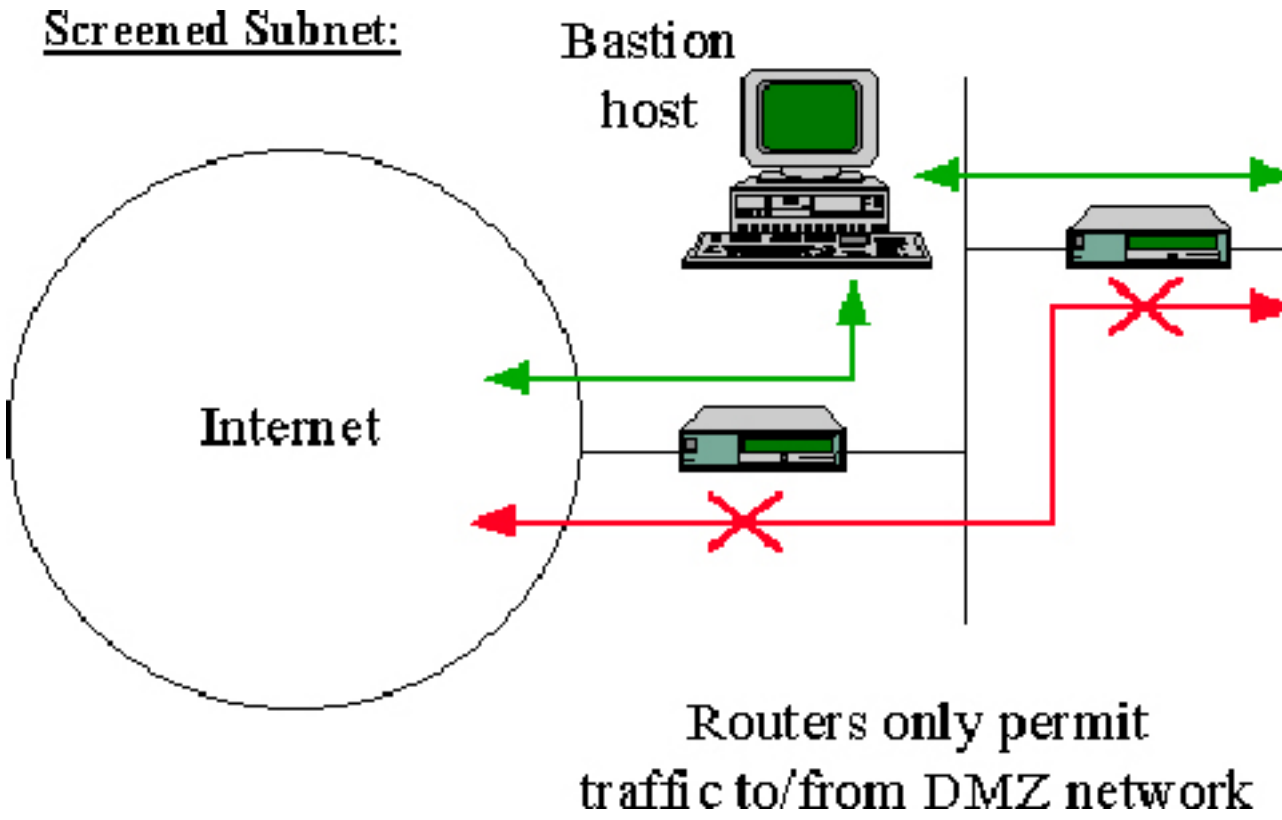


Figure 2: Screened Subnet Firewall

Having an application in the way in some cases may impact performance and may make the firewall less transparent. Early application layer firewalls such as those built using the TIS firewall toolkit, are not particularly transparent to end users and may require some training. Modern application layer firewalls are often fully transparent. Application layer firewalls tend to provide more detailed audit reports and tend to enforce more conservative security models than network layer firewalls.

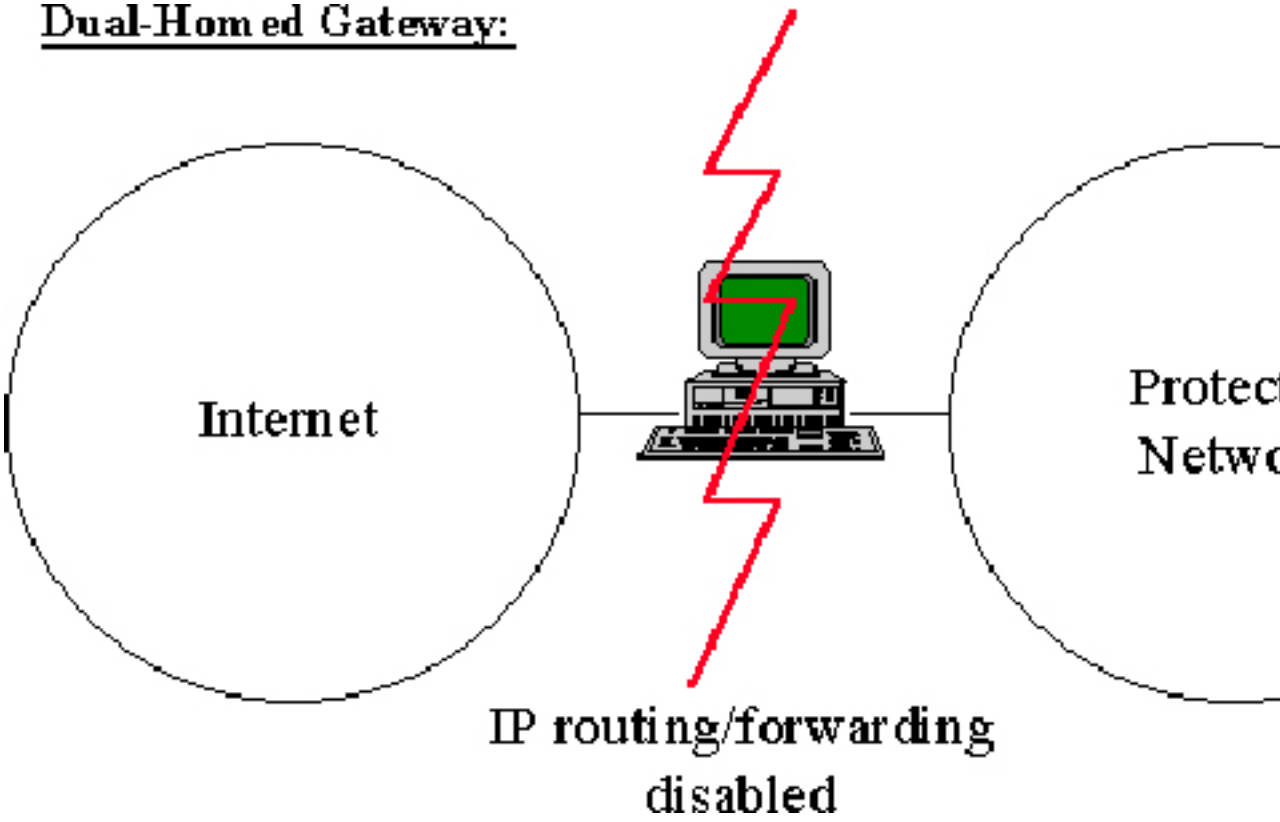


Figure 3: Dual Homed Gateway

*Example Application layer firewall*: In Figure 3, an application layer firewall called a "dual homed gateway" is represented. A dual homed gateway is a highly secured host that runs proxy software. It has two network interfaces, one on each network, and blocks all traffic passing through it.

Most firewalls now lie someplace between network layer firewalls and application layer firewalls. As expected, network layer firewalls have become increasingly "aware" of the information going through them, and application layer firewalls have become increasingly "low level" and transparent. The end result is that now there are fast packet-screening systems that log and audit data as

they pass through the system. Increasingly, firewalls (network and application layer) incorporate encryption so that they may protect traffic passing between them over the Internet. Firewalls with end-to-end encryption can be used by organizations with multiple points of Internet connectivity to use the Internet as a "private backbone" without worrying about their data or passwords being sniffed. (IPSEC, described in Section 2.6, is playing an increasingly significant role in the construction of such virtual private networks.)

## 3.3 What are proxy servers and how do they work?

A proxy server (sometimes referred to as an application gateway or forwarder) is an application that mediates traffic between a protected network and the Internet. Proxies are often used instead of router-based traffic controls, to prevent traffic from passing directly between networks. Many proxies contain extra logging or support for user authentication. Since proxies must "understand" the application protocol being used, they can also implement protocol specific security (e.g., an FTP proxy might be configurable to permit incoming FTP and block outgoing FTP).

Proxy servers are application specific. In order to support a new protocol via a proxy, a proxy must be developed for it. One popular set of proxy servers is the TIS Internet Firewall Toolkit ("FWTK") which includes proxies for Telnet, rlogin, FTP, the X Window System, HTTP/Web, and NNTP/Usenet news. SOCKS is a generic proxy system that can be compiled into a client-side application to make it work through a firewall. Its advantage is that it's easy to use, but it doesn't support the addition of authentication hooks or protocol specific logging. For more information on SOCKS, see `http://www.socks.nec.com/`.

## 3.4 What are some cheap packet screening tools?

The Texas A&M University security tools include software for implementing screening routers. Karlbridge is a PC-based screening router kit available from `ftp://ftp.net.ohio-state.edu/pub/kbridge/`.

There are numerous kernel-level packet screens, including *ipf*, *ipfw*, *ipchains*, *pf*, and *ipfwadm*. Typically, these are included in various free Unix implementations, such as FreeBSD, OpenBSD, NetBSD, and Linux. You might also find these tools available in your commercial Unix implementation.

If you're willing to get your hands a little dirty, it's completely possible to build a secure and fully functional firewall for the price of hardware and some of your time.

## 3.5 What are some reasonable filtering rules for a kernel-based packet screen?

This example is written specifically for *ipfwadm* on Linux, but the principles (and even much of the syntax) applies for other kernel interfaces for packet screening on "open source" Unix systems.

There are four basic categories covered by the *ipfwadm* rules:

**-A** Packet Accounting

**-I** Input firewall

**-O** Output firewall

**-F** Forwarding firewall

*ipfwadm* also has masquerading (`-M`) capabilities. For more information on switches and options, see the *ipfwadm man* page.

### 3.5.1 Implementation

Here, our organization is using a private (RFC 1918) Class C network 192.168.1.0. Our ISP has assigned us the address 201.123.102.32 for our gateway's external interface and 201.123.102.33 for our external mail server. Organizational policy says:

- Allow all outgoing TCP connections
- Allow incoming SMTP and DNS to external mail server
- Block all other traffic

The following block of commands can be placed in a system boot file (perhaps `rc.local` on Unix systems).

```
ipfwadm -F -f
ipfwadm -F -p deny
ipfwadm -F -i m -b -P tcp -S 0.0.0.0/0 1024:65535 -D 201.123.102.33 25
ipfwadm -F -i m -b -P tcp -S 0.0.0.0/0 1024:65535 -D 201.123.102.33 53
ipfwadm -F -i m -b -P udp -S 0.0.0.0/0 1024:65535 -D 201.123.102.33 53
ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0 -W eth0

/sbin/route add -host 201.123.102.33 gw 192.168.1.2
```

### 3.5.2 Explanation

- Line one flushes (`-f`) all forwarding (`-F`) rules.
- Line two sets the default policy (`-p`) to `deny`.
- Lines three through five are input rules (`-i`) in the following format:

  *ipfwadm* **-F** (forward) **-i** (input) **m** (masq.) **-b** (bi-directional) **-P** protocol)[protocol]**-S** (source)[subnet/mask] [originating ports]**-D** (destination)[subnet/mask][port]

- Line six appends (`-a`) a rule that permits all internal IP addresses out to all external addresses on all protocols, all ports.
- Line eight adds a route so that traffic going to 201.123.102.33 will be directed to the internal address 192.168.1.2.

## 3.6 What are some reasonable filtering rules for a Cisco?

The example in Figure 4 shows one possible configuration for using the Cisco as filtering router. It is a sample that shows the implementation of as specific policy. Your policy will undoubtedly vary.

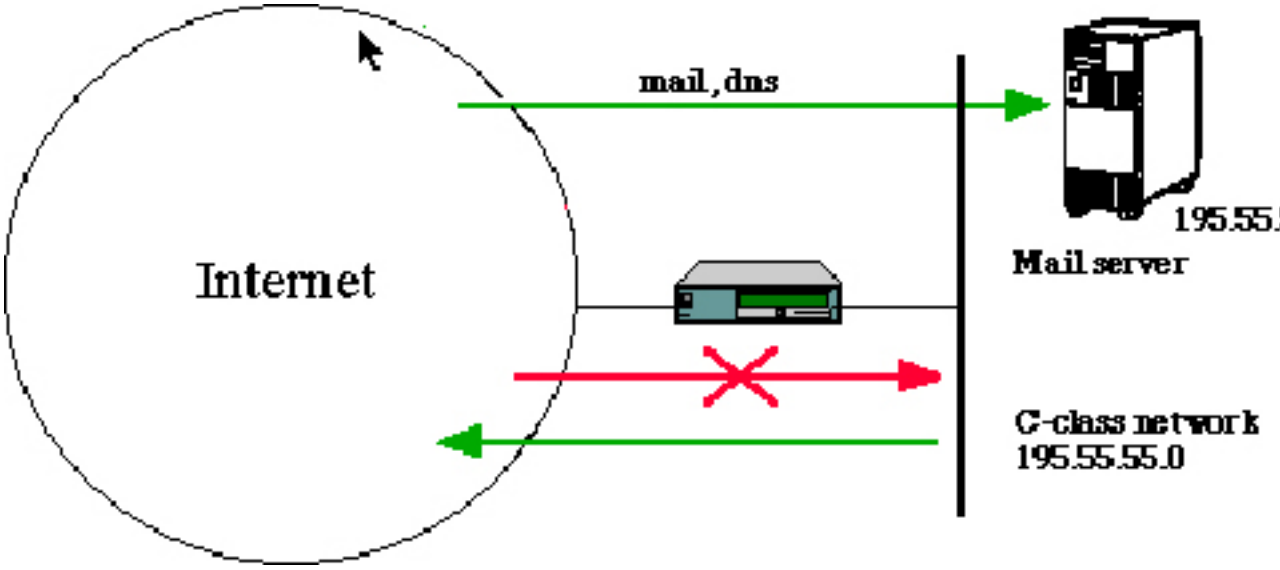


Figure 4: Packet Filtering Router

In this example, a company has Class C network address 195.55.55.0. Company network is connected to Internet via IP Service Provider. Company policy is to allow everybody access to Internet services, so all outgoing connections are accepted. All incoming connections go through "mailhost". Mail and DNS are only incoming services.

### 3.6.1 Implementation

- Allow all outgoing TCP-connections
- Allow incoming SMTP and DNS to mailhost
- Allow incoming FTP data connections to high TCP port (>1024)

- Try to protect services that live on high port numbers

Only incoming packets from Internet are checked in this configuration. Rules are tested in order and stop when the first match is found. There is an implicit deny rule at the end of an access list that denies everything. This IP access list assumes that you are running Cisco IOS v. 10.3 or later.

```
no ip source-route
!
interface ethernet 0
ip address 195.55.55.1
no ip directed-broadcast
!
interface serial 0
no ip directed-broadcast
ip access-group 101 in
!
access-list 101 deny ip 127.0.0.0 0.255.255.255 any
access-list 101 deny ip 10.0.0.0 0.255.255.255 any
access-list 101 deny ip 172.16.0.0 0.15.255.255 any
access-list 101 deny ip 192.168.0.0 0.0.255.255 any
access-list 101 deny ip any 0.0.0.255 255.255.255.0
access-list 101 deny ip any 0.0.0.0 255.255.255.0
!
access-list 101 deny ip 195.55.55.0 0.0.0.255
access-list 101 permit tcp any any established
!
access-list 101 permit tcp any host 195.55.55.10 eq smtp
access-list 101 permit tcp any host 195.55.55.10 eq dns
access-list 101 permit udp any host 192.55.55.10 eq dns
!
access-list 101 deny tcp any any range 6000 6003
access-list 101 deny tcp any any range 2000 2003
access-list 101 deny tcp any any eq 2049
access-list 101 deny udp any any eq 2049
!
access-list 101 permit tcp any 20 any gt 1024
!
access-list 101 permit icmp any any
!
snmp-server community FOOBAR RO 2
line vty 0 4
access-class 2 in
access-list 2 permit 195.55.55.0 0.0.0.255
```

### 3.6.2 Explanations

- Drop all source-routed packets. Source routing can be used for address spoofing.
- Drop directed broadcasts, which are used in smurf attacks.
- If an incoming packet claims to be from a local net, loopback network, or private network, drop it.
- All packets which are part of already established TCP-connections can pass through without further checking.
- All connections to low port numbers are blocked except SMTP and DNS.
- Block all services that listen for TCP connections on high port numbers. X11 (port 6000+), OpenWindows (port 2000+) are a few candidates. NFS (port 2049) runs usually over UDP, but it can be run over TCP, so you should block it.
- Incoming connections from port 20 into high port numbers are supposed to be FTP data connections.
- Access-list 2 limits access to router itself (telnet & SNMP)
- All UDP traffic is blocked to protect RPC services

### 3.6.3 Shortcomings

- You cannot enforce strong access policies with router access lists. Users can easily install backdoors to their systems to get over "no incoming telnet" or "no X11" rules. Also crackers install telnet backdoors on systems where they break in.
- You can never be sure what services you have listening for connections on high port numbers. (You can't be sure of what services you have listening for connections on low port numbers, either, especially in highly decentralized environments where people can put their own machines on the network or where they can get administrative access to their own machines.)
- Checking the source port on incoming FTP data connections is a weak security method. It also breaks access to some FTP sites. It makes use of the service more difficult for users without preventing bad guys from scanning your systems.

Use at least Cisco version 9.21 so you can filter incoming packets and check for address spoofing. It's still better to use 10.3, where you get some extra features (like filtering on source port) and some improvements on filter syntax.

You have still a few ways to make your setup stronger. Block all incoming TCP-connections and tell users to use passive-FTP clients. You can also block outgoing ICMP echo-reply and destination-unreachable messages to hide your network and to prevent use of network scanners. Cisco.com use to have an archive of examples for building firewalls using Cisco routers, but it doesn't seem to be online anymore. There are some notes on Cisco access control lists, at least, at `ftp://ftp.cisco.com/pub/mibs/app_notes/access-lists`.

## 3.7 What are the critical resources in a firewall?

It's important to understand the critical resources of your firewall architecture, so when you do capacity planning, performance optimizations, etc., you know exactly what you need to do, and how much you need to do it in order to get the desired result.

What exactly the firewall's critical resources are tends to vary from site to site, depending on the sort of traffic that loads the system. Some people think they'll automatically be able to increase the data throughput of their firewall by putting in a box with a faster CPU, or another CPU, when this isn't necessarily the case. Potentially, this could be a large waste of money that doesn't do anything to solve the problem at hand or provide the expected scalability.

On busy systems, *memory* is extremely important. You have to have enough RAM to support every instance of every program necessary to service the load placed on that machine. Otherwise, the swapping will start and the productivity will stop. Light swapping isn't usually much of a problem, but if a system's swap space begins to get busy, then it's usually time for more RAM. A system that's heavily swapping is often relatively easy to push over the edge in a denial-of-service attack, or simply fall behind in processing the load placed on it. This is where long email delays start.

Beyond the system's requirement for memory, it's useful to understand that different services use different system resources. So the configuration that you have for your system should be indicative of the kind of load you plan to service. A 1400 MHz processor isn't going to do you much good if all you're doing is netnews and mail, and are trying to do it on an IDE disk with an ISA controller.

| Service | Critical Resource |
|---|---|
| Email | Disk I/O |
| Netnews | Disk I/O |
| Web | Host OS Socket Performance |
| IP Routing | Host OS Socket Performance |
| Web Cache | Host OS Socket Performance, Disk I/O |

Table 1: Critical Resources for Firewall Services

## 3.8 What is a DMZ, and why do I want one?

"DMZ" is an abbreviation for "demilitarized zone". In the context of firewalls, this refers to a part of the network that is neither part of the internal network nor directly part of the Internet. Typically, this is the area between your Internet access router and your bastion host, though it can be between any two policy-enforcing components of your architecture.

A DMZ can be created by putting access control lists on your access router. This minimizes the exposure of hosts on your external LAN by allowing only recognized and managed services on those hosts to be accessible by hosts on the Internet. Many commercial firewalls simply make a third interface off of the bastion host and label it the DMZ, the point is that the network is neither "inside" nor "outside".

For example, a web server running on NT might be vulnerable to a number of denial-of-service attacks against such services as RPC, NetBIOS and SMB. These services are not required for the operation of a web server, so blocking TCP connections to ports 135, 137, 138, and 139 on that host will reduce the exposure to a denial-of-service attack. In fact, if you block everything but HTTP traffic to that host, an attacker will only have one service to attack.

This illustrates an important principle: never offer attackers more to work with than is absolutely necessary to support the services you want to offer the public.

## 3.9 How might I increase the security and scalability of my DMZ?

A common approach for an attacker is to break into a host that's vulnerable to attack, and exploit trust relationships between the vulnerable host and more interesting targets.

If you are running a number of services that have different levels of security, you might want to consider breaking your DMZ into several "security zones". This can be done by having a number of different networks within the DMZ. For example, the access router could feed two Ethernets, both protected by ACLs, and therefore in the DMZ.

On one of the Ethernets, you might have hosts whose purpose is to service your organization's need for Internet connectivity. These will likely relay mail, news, and host DNS. On the other Ethernet could be your web server(s) and other hosts that provide services for the benefit of Internet users.

In many organizations, services for Internet users tend to be less carefully guarded and are more likely to be doing insecure things. (For example, in the case of a web server, unauthenticated and untrusted users might be running CGI, PHP, or other executable programs. This might be reasonable for your web server, but brings with it a certain set of risks that need to be managed. It is likely these services are too risky for an organization to run them on a bastion host, where a slip-up can result in the complete failure of the security mechanisms.)

By putting hosts with similar levels of risk on networks together in the DMZ, you can help minimize the effect of a breakin at your site. If someone breaks into your web server by exploiting some bug in your web server, they'll not be able to use it as a launching point to break into your private network if the web servers are on a separate LAN from the bastion hosts, and you don't have any trust relationships between the web server and bastion host.

Now, keep in mind that this is Ethernet. If someone breaks into your web server, and your bastion host is on the same Ethernet, an attacker can install a sniffer on your web server, and watch the traffic to and from your bastion host. This might reveal things that can be used to break into the bastion host and gain access to the internal network. (Switched Ethernet can reduce your exposure to this kind of problem, but will not eliminate it.)

Splitting services up not only by host, but by network, and limiting the level of trust between hosts on those networks, you can greatly reduce the likelihood of a breakin on one host being used to break into the other. Succinctly stated: breaking into the web server in this case won't make it any easier to break into the bastion host.

You can also increase the scalability of your architecture by placing hosts on different networks. The fewer machines that there are to share the available bandwidth, the more bandwidth that each will get.

## 3.10 What is a 'single point of failure', and how do I avoid having one?

An architecture whose security hinges upon one mechanism has a single point of failure. Software that runs bastion hosts has bugs. Applications have bugs. Software that controls routers has bugs. It makes sense to use all of these components to build a securely designed network, and to use them in redundant ways.

If your firewall architecture is a screened subnet, you have two packet filtering routers and a bastion host. (See question 3.2 from this section.) Your Internet access router will not permit traffic from the Internet to get all the way into your private network. However, if you don't enforce that rule with any other mechanisms on the bastion host and/or choke router, only one component of your architecture needs to fail or be compromised in order to get inside. On the other hand, if you have a redundant rule on the bastion host, and again on the choke router, an attacker will need to defeat *three* mechanisms.

Further, if the bastion host or the choke router needs to invoke its rule to block outside access to the internal network, you might want to have it trigger an alarm of some sort, since you know that someone has gotten through your access router.

## 3.11 How can I block all of the bad stuff?

For firewalls where the emphasis is on security instead of connectivity, you should consider blocking *everything* by default, and only specifically allowing

what services you need on a case-by-case basis.

If you block everything, except a specific set of services, then you've already made your job much easier. Instead of having to worry about every security problem with everything product and service around, you only need to worry about every security problem with a specific set of services and products.

Before turning on a service, you should consider a couple of questions:

- Is the protocol for this product a well-known, published protocol?
- Is the application to service this protocol available for public inspection of its implementation?
- How well known is the service and product?
- How does allowing this service change the firewall architecture? Will an attacker see things differently? Could it be exploited to get at my internal network, or to change things on hosts in my DMZ?

When considering the above questions, keep the following in mind:

- "Security through obscurity" is no security at all. Unpublished protocols have been examined by bad guys and defeated.
- Despite what the marketing representatives say, not every protocol or service is designed with security in mind. In fact, the number that are is very few.
- Even in cases where security is a consideration, not all organizations have competent security staff. Among those who don't, not all are willing to bring a competent consultant into the project. The end result is that otherwise-competent, well-intended developers can design insecure systems.
- The less that a vendor is willing to tell you about how their system *really* works, the more likely it is that security (or other) problems exist. Only vendors with something to hide have a reason to hide their designs and implementations [2].

## 3.12 How can I restrict web access so users can't view sites unrelated to work?

A few years ago, someone got the idea that it's a good idea to block "bad" web sites, i.e., those that contain material that The Company views "inappropriate". The idea has been increasing in popularity, but there are several things to consider when thinking about implementing such controls in your firewall.

- It is not possible to practically block everything that an employer deems "inappropriate". The Internet is full of every sort of material. Blocking one source will only redirect traffic to another source of such material, or cause someone to figure a way around the block.

- Most organizations do not have a standard for judging the appropriateness of material that their employees bring to work, e.g., books and magazines. Do you inspect everyone's briefcase for "inappropriate material" every day? If you do not, then why would you inspect every packet for "inappropriate material"? Any decisions along those lines in such an organization will be arbitrary. Attempting to take disciplinary action against an employee where the only standard is arbitrary typically isn't wise, for reasons well beyond the scope of this document.

- Products that perform site-blocking, commercial and otherwise, are typically easy to circumvent. Hostnames can be rewritten as IP addresses. IP addresses can be written as a 32-bit integer value, or as four 8-bit integers (the most common form). Other possibilities exist, as well. Connections can be proxied. Web pages can be fetched via email. You can't block them all. The effort that you'll spend trying to implement and manage such controls will almost certainly far exceed any level of damage control that you're hoping to have.

The rule-of-thumb to remember here is that you cannot solve social problems with technology. If there is a problem with someone going to an "inappropriate" web site, that is because someone else saw it and was offended by what he saw, or because that person's productivity is below expectations. In either case, those are matters for the personnel department, not the firewall administrator.

# 4 Various Attacks

## 4.1 What is source routed traffic and why is it a threat?

Normally, the route a packet takes from its source to its destination is determined by the routers between the source and destination. The packet itself only says where it wants to go (the destination address), and nothing about how it expects to get there.

There is an optional way for the sender of a packet (the source) to include information in the packet that tells the route the packet should take to get to its destination; thus the name "source routing". For a firewall, source routing is noteworthy, since an attacker can generate traffic claiming to be from a system "inside" the firewall. In general, such traffic wouldn't route to the firewall properly, but with the source routing option, all the routers between the attacker's machine and the target will return traffic along the reverse path of the source route. Implementing such an attack is quite easy; so firewall builders should not discount it as unlikely to happen.

In practice, source routing is very little used. In fact, generally the main legitimate use is in debugging network problems or routing traffic over specific links for congestion control for specialized situations. When building a firewall, source routing should be blocked at some point. Most commercial routers incorporate the ability to block source routing specifically, and many versions of Unix

that might be used to build firewall bastion hosts have the ability to disable or to ignore source routed traffic.

## 4.2 What are ICMP redirects and redirect bombs?

An ICMP Redirect tells the recipient system to override something in its routing table. It is legitimately used by routers to tell hosts that the host is using a non-optimal or defunct route to a particular destination, i.e., the host is sending it to the wrong router. The wrong router sends the host back an ICMP Redirect packet that tells the host what the correct route should be. If you can forge ICMP Redirect packets, and if your target host pays attention to them, you can alter the routing tables on the host and possibly subvert the security of the host by causing traffic to flow via a path the network manager didn't intend. ICMP Redirects also may be employed for denial of service attacks, where a host is sent a route that loses it connectivity, or is sent an ICMP Network Unreachable packet telling it that it can no longer access a particular network.

Many firewall builders screen ICMP traffic from their network, since it limits the ability of outsiders to ping hosts, or modify their routing tables.

Before you decide to block all ICMP packets, you should be aware of how the TCP protocol does "Path MTU Discovery", to make certain that you don't break connectivity to other sites. If you can't safely block it everywhere, you can consider allowing selected types of ICMP to selected routing devices. If you don't block it, you should at least ensure that your routers and hosts don't respond to broadcast ping packets.

## 4.3 What about denial of service?

Denial of service is when someone decides to make your network or firewall useless by disrupting it, crashing it, jamming it, or flooding it. The problem with denial of service on the Internet is that it is impossible to prevent. The reason has to do with the distributed nature of the network: every network node is connected via other networks which in turn connect to other networks, etc. A firewall administrator or ISP only has control of a few of the local elements within reach. An attacker can always disrupt a connection "upstream" from where the victim controls it. In other words, if someone wanted to take a network off the air, he could do it either by taking the network off the air, or by taking the networks it connects to off the air, ad infinitum. There are many, many, ways someone can deny service, ranging from the complex to the trivial brute-force. If you are considering using Internet for a service which is absolutely time or mission critical, you should consider your fallback position in the event that the network is down or damaged.

TCP/IP's UDP echo service is trivially abused to get two servers to flood a network segment with echo packets. You should consider commenting out unused entries in `/etc/inetd.conf` of Unix hosts, adding `no ip small-servers` to Cisco routers, or the equivalent for your components.

## 4.4 What are some common attacks, and how can I protect my system against them?

Each site is a little different from every other in terms of what attacks are likely to be used against it. Some recurring themes do arise, though.

### 4.4.1 SMTP Server Hijacking (Unauthorized Relaying)

This is where a spammer will take many thousands of copies of a message and send it to a huge list of email addresses. Because these lists are often so bad, and in order to increase the speed of operation for the spammer, many have resorted to simply sending all of their mail to an SMTP server that will take care of actually delivering the mail.

Of course, all of the bounces, spam complaints, hate mail, and bad PR come for the site that was used as a relay. There is a very real cost associated with this, mostly in paying people to clean up the mess afterward.

The Mail Abuse Prevention System[1] Transport Security Initiative[2] maintains a complete description of the problem, and how to configure about every mailer on the planet to protect against this attack.

### 4.4.2 Exploiting Bugs in Applications

Various versions of web servers, mail servers, and other Internet service software contain bugs that allow remote (Internet) users to do things ranging from gain control of the machine to making that application crash and just about everything in between.

The exposure to this risk can be reduced by running only necessary services, keeping up to date on patches, and using products that have been around a while.

### 4.4.3 Bugs in Operating Systems

Again, these are typically initiated by users remotely. Operating systems that are relatively new to IP networking tend to be more problematic, as more mature operating systems have had time to find and eliminate their bugs. An attacker can often make the target equipment continuously reboot, crash, lose the ability to talk to the network, or replace files on the machine.

Here, running as few operating system services as possible can help. Also, having a packet filter in front of the operating system can reduce the exposure to a large number of these types of attacks.

And, of course, chosing a stable operating system will help here as well. When selecting an OS, don't be fooled into believing that "the pricier, the better". Free operating systems are often much more robust than their commercial counterparts

[1] `http://mail-abuse.org/`
[2] `http://mail-abuse.org/tsi/`

# 5 How Do I...

## 5.1 Do I really want to allow everything that my users ask for?

It's entirely possible that the answer is "no". Each site has its own policies about what is and isn't needed, but it's important to remember that a large part of the job of being an organization's gatekeeper is *education*. Users want streaming video, real-time chat, and to be able to offer services to external customers that require interaction with live databases on the internal network.

That doesn't mean that any of these things can be done without presenting more risk to the organization than the supposed "value" of heading down that road is worth. Most users don't want to put their organization at risk. They just read the trade rags, see advertisements, and they want to do those things, too. It's important to look into what it is that they really want to do, and to help them understand how they might be able to accomplish their real objective in a more secure manner.

You won't always be popular, and you might even find yourself being given direction to do something incredibly stupid, like "just open up ports foo through bar". If that happens, don't worry about it. It would be wise to keep all of your exchanges on such an event so that when a 12-year-old script kiddie breaks in, you'll at least be able to separate yourself from the whole mess.

## 5.2 How do I make Web/HTTP work through my firewall?

There are three ways to do it.

1. Allow "established" connections out via a router, if you are using screening routers.

2. Use a web client that supports SOCKS, and run SOCKS on your bastion host.

3. Run some kind of proxy-capable web server on the bastion host. Some options include Squid[3], Apache[4], Netscape Proxy[5], and *http-gw* from the TIS firewall toolkit. Most of these can also proxy other protocols (such as gopher and ftp), and can cache objects fetched, which will also typically result in a performance boost for the users, and more efficient use of your connection to the Internet. Essentially all web clients (Mozilla, Internet Explorer, Lynx, etc.) have proxy server support built directly into them.

[3] `http://squid.nlanr.net/`
[4] `http://www.apache.org/docs/mod/mod_proxy.html`
[5] `http://home.netscape.com/proxy/v3.5/index.html`

## 5.3 How do I make SSL work through the firewall?

SSL is a protocol that allows secure connections across the Internet. Typically, SSL is used to protect HTTP traffic. However, other protocols (such as telnet) can run atop SSL.

Enabling SSL through your firewall can be done the same way that you would allow HTTP traffic, if it's HTTP that you're using SSL to secure, which is usually true. The only difference is that instead of using something that will simply relay HTTP, you'll need something that can tunnel SSL. This is a feature present on most web object caches.

You can find out more about SSL from Netscape[6].

## 5.4 How do I make DNS work with a firewall?

Some organizations want to hide DNS names from the outside. Many experts don't think hiding DNS names is worthwhile, but if site/corporate policy mandates hiding domain names, this is one approach that is known to work. Another reason you may have to hide domain names is if you have a non-standard addressing scheme on your internal network. In that case, you have no choice but to hide those addresses. Don't fool yourself into thinking that if your DNS names are hidden that it will slow an attacker down much if they break into your firewall. Information about what is on your network is too easily gleaned from the networking layer itself. If you want an interesting demonstration of this, ping the subnet broadcast address on your LAN and then do an "arp -a." Note also that hiding names in the DNS doesn't address the problem of host names "leaking" out in mail headers, news articles, etc.

This approach is one of many, and is useful for organizations that wish to hide their host names from the Internet. The success of this approach lies on the fact that DNS clients on a machine don't have to talk to a DNS server on that same machine. In other words, just because there's a DNS server on a machine, there's nothing wrong with (and there are often advantages to) redirecting that machine's DNS client activity to a DNS server on another machine.

First, you set up a DNS server on the bastion host that the outside world can talk to. You set this server up so that it claims to be authoritative for your domains. In fact, all this server knows is what you want the outside world to know; the names and addresses of your gateways, your wildcard MX records, and so forth. This is the "public" server.

Then, you set up a DNS server on an internal machine. This server also claims to be authoritative for your domains; unlike the public server, this one is telling the truth. This is your "normal" nameserver, into which you put all your "normal" DNS stuff. You also set this server up to forward queries that it can't resolve to the public server (using a "forwarders" line in /etc/named.boot on a Unix machine, for example).

Finally, you set up all your DNS clients (the `/etc/resolv.conf` file on a Unix box, for instance), including the ones on the machine with the public

[6] `http://developer.netscape.com/docs/manuals/security/sslin/contents.htm`

server, to use the internal server. This is the key.

An internal client asking about an internal host asks the internal server, and gets an answer; an internal client asking about an external host asks the internal server, which asks the public server, which asks the Internet, and the answer is relayed back. A client on the public server works just the same way. An external client, however, asking about an internal host gets back the "restricted" answer from the public server.

This approach assumes that there's a packet filtering firewall between these two servers that will allow them to talk DNS to each other, but otherwise restricts DNS between other hosts.

Another trick that's useful in this scheme is to employ wildcard PTR records in your IN-ADDR.ARPA domains. These cause an an address-to-name lookup for any of your non-public hosts to return something like "unknown.YOUR.DOMAIN" rather than an error. This satisfies anonymous FTP sites like ftp.uu.net that insist on having a name for the machines they talk to. This may fail when talking to sites that do a DNS cross-check in which the host name is matched against its address and vice versa.

## 5.5 How do I make FTP work through my firewall?

Generally, making FTP work through the firewall is done either using a proxy server such as the firewall toolkit's ftp-gw or by permitting incoming connections to the network at a restricted port range, and otherwise restricting incoming connections using something like "established" screening rules. The FTP client is then modified to bind the data port to a port within that range. This entails being able to modify the FTP client application on internal hosts.

In some cases, if FTP downloads are all you wish to support, you might want to consider declaring FTP a "dead protocol" and letting you users download files via the Web instead. The user interface certainly is nicer, and it gets around the ugly callback port problem. If you choose the FTP-via-Web approach, your users will be unable to FTP files out, which, depending on what you are trying to accomplish, may be a problem.

A different approach is to use the FTP "PASV" option to indicate that the remote FTP server should permit the client to initiate connections. The PASV approach assumes that the FTP server on the remote system supports that operation. (See "Firewall-Friendly FTP" [1].)

Other sites prefer to build client versions of the FTP program that are linked against a SOCKS library.

## 5.6 How do I make Telnet work through my firewall?

Telnet is generally supported either by using an application proxy such as the firewall toolkit's tn-gw, or by simply configuring a router to permit outgoing connections using something like the "established" screening rules. Application proxies could be in the form of a standalone proxy running on the bastion host, or in the form of a SOCKS server and a modified client.

## 5.7 How do I make Finger and whois work through my firewall?

Many firewall admins permit connections to the finger port from only trusted machines, which can issue finger requests in the form of: finger user@host.domain@firewall. This approach only works with the standard Unix version of finger. Controlling access to services and restricting them to specific machines is managed using either tcp_wrappers or netacl from the firewall toolkit. This approach will not work on all systems, since some finger servers do not permit user@host@host fingering.

Many sites block inbound finger requests for a variety of reasons, foremost being past security bugs in the finger server (the Morris internet worm made these bugs famous) and the risk of proprietary or sensitive information being revealed in user's finger information. In general, however, if your users are accustomed to putting proprietary or sensitive information in their *.plan* files, you have a more serious security problem than just a firewall can solve.

## 5.8 How do I make gopher, archie, and other services work through my firewall?

The majority of firewall administrators choose to support gopher and archie through web proxies, instead of directly. Proxies such as the firewall toolkit's `http-gw` convert gopher/gopher+ queries into HTML and vice versa. For supporting archie and other queries, many sites rely on Internet-based Web-to-archie servers, such as ArchiePlex. The Web's tendency to make everything on the Internet look like a web service is both a blessing and a curse.

There are many new services constantly cropping up. Often they are misdesigned or are not designed with security in mind, and their designers will cheerfully tell you if you want to use them you need to let port xxx through your router. Unfortunately, not everyone can do that, and so a number of interesting new toys are difficult to use for people behind firewalls. Things like RealAudio, which require direct UDP access, are particularly egregious examples. The thing to bear in mind if you find yourself faced with one of these problems is to find out as much as you can about the security risks that the service may present, before you just allow it through. It's quite possible the service has no security implications. It's equally possible that it has undiscovered holes you could drive a truck through.

## 5.9 What are the issues about X11 through a firewall?

The X Windows System is a very useful system, but unfortunately has some major security flaws. Remote systems that can gain or spoof access to a workstation's X11 display can monitor keystrokes that a user enters, download copies of the contents of their windows, etc.

While attempts have been made to overcome them (E.g., MIT "Magic Cookie") it is still entirely too easy for an attacker to interfere with a user's X11 display.

Most firewalls block all X11 traffic. Some permit X11 traffic through application proxies such as the DEC CRL X11 proxy (FTP crl.dec.com). The firewall toolkit includes a proxy for X11, called x-gw, which a user can invoke via the Telnet proxy, to create a virtual X11 server on the firewall. When requests are made for an X11 connection on the virtual X11 server, the user is presented with a pop-up asking them if it is OK to allow the connection. While this is a little unaesthetic, it's entirely in keeping with the rest of X11.

## 5.10 How do I make *RealAudio* work through my firewall?

RealNetworks maintains some information about how to get RealAudio working through your firewall[7]. It would be unwise to make *any* changes to your firewall without understanding what the changes will do, exactly, and knowing what risks the new changes will bring with them.

## 5.11 How do I make my web server act as a front-end for a database that lives on my private network?

The best way to do this is to allow very limited connectivity between your web server and your database server via a specific protocol that only supports the level of functionality you're going to use. Allowing raw SQL, or anything else where custom extractions could be performed by an attacker isn't generally a good idea.

Assume that an attacker is going to be able to break into your web server, and make queries in the same way that the web server can. Is there a mechanism for extracting sensitive information that the web server doesn't need, like credit card information? Can an attacker issue an SQL `select` and extract your entire proprietary database?

"E-commerce" applications, like everything else, are best designed with security in mind from the ground up, instead of having security "added" as an afterthought. Review your architecture critically, from the perspective of an attacker. Assume that the attacker knows everything about your architecture. Now ask yourself what needs to be done to steal your data, to make unauthorized changes, or to do anything else that you don't want done. You might find that you can significantly increase security without decreasing functionality by making a few design and implementation decisions.

Some ideas for how to handle this:

- Extract the data you need from the database on a regular basis so you're not making queries against the full database, complete with information that attackers will find interesting.
- Greatly restrict and audit what you do allow between the web server and database.

[7] `http://www.real.com/firewall/`

### 5.12 But my database has an integrated web server, and I want to use that. Can't I just poke a hole in the firewall and tunnel that port?

If your site firewall policy is sufficiently lax that you're willing to manage the risk that someone will exploit a vulnerability in your web server that will result in partial or complete exposure of your database, then there isn't much preventing you from doing this.

However, in many organizations, the people who are responsible for tying the web front end to the database back end simply do not have the authority to take that responsibility. Further, if the information in the database is about people, you might find yourself guilty of breaking a number of laws if you haven't taken reasonable precautions to prevent the system from being abused.

In general, this isn't a good idea. See question 5.11 for some ideas on other ways to accomplish this objective.

### 5.13 How Do I Make IP Multicast Work With My Firewall?

IP multicast is a means of getting IP traffic from one host to a set of hosts without using broadcasting; that is, instead of every host getting the traffic, only those that want it will get it, without each having to maintain a separate connection to the server. IP unicast is where one host talks to another, multicast is where one host talks to a set of hosts, and broadcast is where one host talks to all hosts.

The public Internet has a multicast backbone ("MBone") where users can engage in multicast traffic exchange. Common uses for the MBone are streams of IETF meetings and similar such interaction. Getting one's own network connected to the MBone will require that the upstream provider route multicast traffic to and from your network. Additionally, your internal network will have to support multicast routing.

The role of the firewall in multicast routing, conceptually, is no different from its role in other traffic routing. That is, a policy that identifies which multicast groups are and aren't allowed must be defined and then a system of allowing that traffic according to policy must be devised. Great detail on how exactly to do this is beyond the scope of this document. Fortunately, RFC 2588 [4] discusses the subject in more detail. Unless your firewall product supports some means of selective multicast forwarding or you have the ability to put it in yourself, you might find forwarding multicast traffic in a way consistent with your security policy to be a bigger headache than it's worth.

## 6 TCP and UDP Ports

*by Mikael Olsson*

This appendix will begin at a fairly "basic" level, so even if the first points seem childishly self-evident to you, you might still learn something from skipping ahead to something later in the text.

## 6.1 What is a port?

A "port" is "virtual slot" in your TCP and UDP stack that is used to map a connection between two hosts, and also between the TCP/UDP layer and the actual applications running on the hosts.

They are numbered 0–65535, with the range 0–1023 being marked as "reserved" or "privlileged", and the rest (1024–65535) as "dynamic" or "unprivileged".

There are basically two uses for ports:

- "Listening" on a port.
  This is used by server applications waiting for users to connect, to get to some "well known service", for instance HTTP (TCP port 80), Telnet (TCP port 23), DNS (UDP and sometimes TCP port 53).

- Opening a "dynamic" port.
  Both sides of a TCP connection need to be identified by IP addresses and port numbers. Hence, when you want to "connect" to a server process, your end of the communications channel also needs a "port". This is done by choosing a port above 1024 on your machine that is not currently in use by another communications channel, and using it as the "sender" in the new connection.

Dynamic ports may also be used as "listening" ports in some applications, most notably FTP.

Ports in the range 0–1023 are almost always server ports. Ports in the range 1024–65535 are usually dynamic ports (i.e., opened dynamically when you connect to a server port). However, *any* port may be used as a server port, and *any* port may be used as an "outgoing" port.

So, to sum it up, here's what happens in a basic connection:

- At some point in time, a server application on host 1.2.3.4 decides to "listen" at port 80 (HTTP) for new connections.

- You (5.6.7.8) want to surf to 1.2.3.4, port 80, and your browser issues a connect call to it.

- The connect call, realising that it doesn't yet have local port number, goes hunting for one. The local port number is necessary since when the replies come back some time in the future, your TCP/IP stack will have to know to what application to pass the reply. It does this by remembering what application uses which local port number. (This is grossly simplified, no flames from programmers, please.)

- Your TCP stack finds an unused dynamic port, usually somewhere above 1024. Let's assume that it finds 1029.
- Your first packet is then sent, from your local IP, 5.6.7.8, port 1029, to 1.2.3.4, port 80.
- The server responds with a packet from 1.2.3.4, port 80, to you, 5.6.7.8, port 1029.
- This procedure is actually longer than this, read on for a more in-depth explanation of TCP connect sequences.

## 6.2 How do I know which application uses what port?

There are several lists outlining the "reserved" and "well known" ports, as well as "commonly used" ports, and the best one is: `ftp://ftp.isi.edu/in-notes/iana/assignments/port-number` For those of you still reading RFC 1700 to find out what port number does what, STOP DOING IT. It is horribly out of date, and it won't be less so tomorrow.

Now, as for trusting this information: These lists do not, in any way, constitute any kind of holy bible on which ports do what.

Wait, let me rephrase that: THERE IS NO WAY OF RELIABLY DETERMINING WHAT PORT DOES WHAT SIMPLY BY LOOKING IN A LIST.

## 6.3 What are LISTENING ports?

Suppose you did "netstat -a" on your machine and ports 1025 and 1030 showed up as LISTENing. What do they do?

Right, let's take a look in the assigned port numbers list.

```
blackjack       1025/tcp   network blackjack
iad1            1030/tcp   BBN IAD
```

Wait, what's happening? Has my workstation stolen my VISA number and decided to go play blackjack with some rogue server on the internet? And what's that software that BBN has installed?

This is NOT where you start panicking and send mail to the firewalls list. In fact, this question has been asked maybe a dozen times during the past six months, and every time it's been answered. Not that THAT keeps people from asking the same question again.

If you are asking this question, you are most likely using a windows box. The ports you are seeing are (most likely) two listening ports that the RPC subsystem opens when it starts up.

This is an example of where dynamicly assigned ports may be used by server processes. Applications using RPC will later on connect to port 135 (the netbios "portmapper") to query where to find some RPC service, and get an answer back saying that that particular service may be contacted on port 1025.

Now, how do we know this, since there's no "list" describing these ports? Simple: There's no substitute for experience. And using the mailing list search engines also helps a hell of a lot.

## 6.4 How do I determine what service the port is for?

Since it is impossible to learn what port does what by looking in a list, how do i do it?

The old hands-on way of doing it is by shutting down nearly every service/daemon running on your machine, doing `netstat -a` and taking note of what ports are open. There shouldn't be very many listening ones. Then you start turning all the services on, one by one, and take note of what new ports show up in your netstat output.

Another way, that needs more guess work, is simply telnetting to the ports and see what comes out. If nothing comes out, try typing some gibberish and slamming Enter a few times, and see if something turns up. If you get binary garble, or nothing at all, this obviously won't help you. `:-)`

However, this will only tell you what listening ports are used. It won't tell you about dynamically opened ports that may be opened later on by these applications.

There are a few applications that might help you track down the ports used.

On Unix systems, there's a nice utility called `lsof` that comes preinstalled on many systems. It will show you all open port numbers and the names of the applications that are using them. This means that it might show you a lot of locally opened files aswell as TCP/IP sockets. Read the help text. `:-)`

On windows systems, nothing comes preinstalled to assist you in this task. (What's new?) There's a utility called "Inzider" which installs itself inside the windows sockets layer and dynamically remembers which process opens which port. The drawback of this approach is that it can't tell you what ports were opened before inzider started, but it's the best that you'll get on windows (to my knowledge). `http://ntsecurity.nu/toolbox/inzider/`.

## 6.5 What ports are safe to pass through a firewall?

ALL.

No, wait, NONE.

No, wait, uuhhh... I've heard that all ports above 1024 are safe since they're only dynamic??

No. Really. You CANNOT tell what ports are safe simply by looking at its number, simply because that is really all it is. A number. You can't mount an attack through a 16-bit number.

The security of a "port" depends on what application you'll reach through that port.

A common misconception is that ports 25 (SMTP) and 80 (HTTP) are safe to pass through a firewall. *meep* WRONG. Just because everyone is doing it doesn't mean that it is safe.

Again, the security of a port depends on what application you'll reach through that port.

If you're running a well-written web server, that is designed from the ground up to be secure, you can probably feel reasonably assured that it's safe to let

outside people access it through port 80. Otherwise, you CAN'T.

The problem here is not in the network layer. It's in how the application processes the data that it receives. This data may be received through port 80, port 666, a serial line, floppy or through singing telegram. If the application is not safe, it does not matter how the data gets to it. The application data is where the real danger lies.

If you are interested in the security of your application, go subscribe to bugtraq[8] or or try searching their archives.

This is more of an application security issue rather than a firewall security issue. One could argue that a firewall should stop all possible attacks, but with the number of new network protocols, NOT designed with security in mind, and networked applications, neither designed with security in mind, it becomes impossible for a firewall to protect against all data-driven attacks.

## 6.6 The behavior of FTP

Or, "Why do I have to open all ports above 1024 to my FTP server?"

FTP doesn't really look a whole lot like other applications from a networking perspective.

It keeps one listening port, port 21, which users connect to. All it does is let people log on, and establish ANOTHER connection to do actual data transfers. This second connection is usually on some port above 1024.

There are two modes, "active" (normal) and "passive" mode. This word describes the server's behaviour.

In active mode, the client (5.6.7.8) connects to port 21 on the server (1.2.3.4) and logs on. When file transfers are due, the client allocates a dynamic port above 1024, informs the server about which port it opened, and then the server opens a new connection to that port. This is the "active" role of the server: it actively establishes new connections to the client.

In passive mode, the connection to port 21 is the same. When file transfers are due, the SERVER allocates a dynamic port above 1024, informs the client about which port it opened, and then the CLIENT opens a new connection to that port. This is the "passive" role of the server: it waits for the client to establish the second (data) connection.

If your firewall doesn't inspect the application data of the FTP command connection, it won't know that it needs to dynamically open new ports above 1024.

On a side note: The traditional behaviour of FTP servers in active mode is to establish the data session FROM port 20, and to the dynamic port on the client. FTP servers are steering away from this behaviour somewhat due to the need to run as "root" on unix systems in order to be able to allocate ports below 1024. Running as "root" is not good for security, since if there's a bug in the software, the attacker would be able to compromise the entire machine. The

[8] `http://www.securityfocus.com`

same goes for running as "Administrator" or "SYSTEM" ("LocalSystem") on NT machines, although the low port problem does not apply on NT.

To sum it up, if your firewall understands FTP, it'll be able to handle the data connections by itself, and you won't have to worry about ports above 1024.

If it does NOT, there are four issues that you need to address:

- Firewalling an FTP server in active mode
  You need to let your server open new connections to the outside world on ports 1024 and above

- Firewalling an FTP server in passive mode
  You need to let the outside world connect to ports 1024 and above on your server. CAUTION!!!! There may be applications running on some of these ports that you do NOT want outside people using. Disallow access to these ports before allowing access to the 1024–65535 port range.

- Firewalling FTP clients in active mode
  You need to let the outside world connect to ports 1024 and above on your clients. CAUTION!!!! There may be applications running on some of these ports that you do NOT want outside people using. Disallow access to these ports before allowing access to the 1024–65535 port range.

- Firewalling FTP clients in passive mode
  You need to let your clients open new connections to the outside world on ports 1024 and above.

Again, if your firewall understands FTP, none of the four points above apply to you. Let the firewall do the job for you.

## 6.7 What software uses what FTP mode?

It is up to the client to decide what mode to use; the default mode when a new connection is opened is "active mode".

Most FTP clients come preconfigured to use active mode, but provide an option to use "passive" ("PASV") mode. An exception is the windows command line FTP client which only operates in active mode.

Web Browsers generally use passive mode when connecting via FTP, with a weird exception: MSIE 5 will use active FTP when FTP:ing in "File Explorer" mode and passive FTP when FTP:ing in "Web Page" mode. There is no reason whatsoever for this behaviour; my guess is that someone in Redmond with no knowledge of FTP decided that "Of course we'll use active mode when we're in file explorer mode, since that looks more active than a web page". Go figure.

## 6.8 Is my firewall trying to connect outside?

My firewall logs are telling me that my web server is trying to connect from port 80 to ports above 1024 on the outside. What is this?!

If you are seeing dropped packets from port 80 on your web server (or from port 25 on your mail server) to high ports on the outside, they usually DO NOT mean that your web server is trying to connect somewhere.

They are the result of the firewall timing out a connection, and seeing the server retransmitting old responses (or trying to close the connection) to the client.

TCP connections always involve packets traveling in BOTH directions in the connection.

If you are able to see the TCP flags in the dropped packets, you'll see that the ACK flag is set but not the SYN flag, meaning that this is actually not a new connection forming, but rather a response of a previously formed connection.

Read point 8 below for an in-depth explanation of what happens when TCP connections are formed (and closed)

## 6.9 The anatomy of a TCP connection

TCP is equipped with 6 "flags", which may be ON or OFF. These flags are:

**FIN** "Controlled" connection close

**SYN** Open new connection

**RST** "Immediate" connection close

**PSH** Instruct receiver host to push the data up to the application rather than just queue it

**ACK** "Acknowledge" a previous packet

**URG** "Urgent" data which needs to be processed immediately

In this example, your client is 5.6.7.8, and the port assigned to you dynamically is 1049. The server is 1.2.3.4, port 80.

You begin the connection attempt:

```
5.6.7.8:1049 -> 1.2.3.4:80   SYN=ON
```

The server receives this packet and understands that someone wants to form a new connection. A response is sent:

```
1.2.3.4:80 -> 5.6.7.8:1049   SYN=ON  ACK=ON
```

The client receives the response, and informs that the response is received

```
5.6.7.8:1049 -> 1.2.3.4:80   ACK=ON
```

Here, the connection is opened. This is called a three-way handshake. Its purpose is to verify to BOTH hosts that they have a working connection between them.

The internet being what it is, unreliable and flooded, there are provisions to compensate for packet loss.

If the client sends out the initial SYN without receiving a SYN+ACK within a few seconds, it'll resend the SYN.

If the server sends out the SYN+ACK without receiving an ACK in a few seconds, it'll resend the SYN+ACK packet.

The latter is actually the reason that SYN flooding works so well. If you send out SYN packets from lots of different ports, this will tie up a lot of resources on the server. If you also refuse to respond to the returned SYN+ACK packets, the server will KEEP these connections for a long time, resending the SYN+ACK packets. Some servers will not accept new connections while there are enough connections currently forming; this is why SYN flooding works.

All packets transmitted in either direction after the three-way handshake will have the ACK bit set. Stateless packet filters make use of this in the so called "established" filters: They will only let packets through that have the ACK bit set. This way, no packet may pass through in a certain direction that could form a new connection. Typically, you don't allow outside hosts to open new connections to inside hosts by requiring the ACK bit set on these packets.

When the time has come to close the connection, there are two ways of doing it: Using the FIN flag, or using the RST flag. Using FIN flags, both implementations are required to send out FIN flags to indicate that they want to close the connection, and then send out acknowledgements to these FINs, indicating that they understood that the other end wants to close the connection. When sending out RST's, the connection is closed forcefully, and you don't really get an indication of whether the other end understood your reset order, or that it has in fact received all data that you sent to it.

The FIN way of closing the connection also exposes you to a denial-of-service situation, since the TCP stack needs to remember the closed connection for a fairly long time, in case the other end hasn't received one of the FIN packets.

If sufficiently many connections are opened and closed, you may end up having "closed" connections in all your connection slots. This way, you wouldn't be able to dynamically allocate more connections, seeing that they're all used. Different OSes handle this situation differently.

# A Some Commercial Products and Vendors

We feel this topic is too sensitive to address in a FAQ, however, an independently maintained list (no warranty or recommendations are implied) can be found online.[9]

# B Glossary of Firewall-Related Terms

**Abuse of Privilege** When a user performs an action that they should not have, according to organizational policy or law.

**Access Control Lists** Rules for packet filters (typically routers) that define which packets to pass and which to block.

**Access Router** A router that connects your network to the external Internet. Typically, this is your first line of defense against attackers from the out-

[9] `http://www.thegild.com/firewall/`.

side Internet. By enabling access control lists on this router, you'll be able to provide a level of protection for all of the hosts "behind" that router, effectively making that network a DMZ instead of an unprotected external LAN.

**Application-Layer Firewall** A firewall system in which service is provided by processes that maintain complete TCP connection state and sequencing. Application layer firewalls often re-address traffic so that outgoing traffic appears to have originated from the firewall, rather than the internal host.

**Authentication** The process of determining the identity of a user that is attempting to access a system.

**Authentication Token** A portable device used for authenticating a user. Authentication tokens operate by challenge/response, time-based code sequences, or other techniques. This may include paper-based lists of one-time passwords.

**Authorization** The process of determining what types of activities are permitted. Usually, authorization is in the context of authentication: once you have authenticated a user, they may be authorized different types of access or activity.

**Bastion Host** A system that has been hardened to resist attack, and which is installed on a network in such a way that it is expected to potentially come under attack. Bastion hosts are often components of firewalls, or may be "outside" web servers or public access systems. Generally, a bastion host is running some form of general purpose operating system (e.g., Unix, VMS, NT, etc.) rather than a ROM-based or firmware operating system.

**Challenge/Response** An authentication technique whereby a server sends an unpredictable challenge to the user, who computes a response using some form of authentication token.

**Chroot** A technique under Unix whereby a process is permanently restricted to an isolated subset of the filesystem.

**Cryptographic Checksum** A one-way function applied to a file to produce a unique "fingerprint" of the file for later reference. Checksum systems are a primary means of detecting filesystem tampering on Unix.

**Data Driven Attack** A form of attack in which the attack is encoded in innocuous-seeming data which is executed by a user or other software to implement an attack. In the case of firewalls, a data driven attack is a concern since it may get through the firewall in data form and launch an attack against a system behind the firewall.

**Defense in Depth** The security approach whereby each system on the network is secured to the greatest possible degree. May be used in conjunction with firewalls.

**DNS spoofing** Assuming the DNS name of another system by either corrupting the name service cache of a victim system, or by compromising a domain name server for a valid domain.

**Dual Homed Gateway** A dual homed gateway is a system that has two or more network interfaces, each of which is connected to a different network. In firewall configurations, a dual homed gateway usually acts to block or filter some or all of the traffic trying to pass between the networks.

**Encrypting Router** see Tunneling Router and Virtual Network Perimeter.

**Firewall** A system or combination of systems that enforces a boundary between two or more networks.

**Host-based Security** The technique of securing an individual system from attack. Host based security is operating system and version dependent.

**Insider Attack** An attack originating from inside a protected network.

**Intrusion Detection** Detection of break-ins or break-in attempts either manually or via software expert systems that operate on logs or other information available on the network.

**IP Spoofing** An attack whereby a system attempts to illicitly impersonate another system by using its IP network address.

**IP Splicing / Hijacking** An attack whereby an active, established, session is intercepted and co-opted by the attacker. IP Splicing attacks may occur after an authentication has been made, permitting the attacker to assume the role of an already authorized user. Primary protections against IP Splicing rely on encryption at the session or network layer.

**Least Privilege** Designing operational aspects of a system to operate with a minimum amount of system privilege. This reduces the authorization level at which various actions are performed and decreases the chance that a process or user with high privileges may be caused to perform unauthorized activity resulting in a security breach.

**Logging** The process of storing information about events that occurred on the firewall or network.

**Log Retention** How long audit logs are retained and maintained.

**Log Processing** How audit logs are processed, searched for key events, or summarized.

**Network-Layer Firewall** A firewall in which traffic is examined at the network protocol packet layer.

**Perimeter-based Security** The technique of securing a network by controlling access to all entry and exit points of the network.

**Policy** Organization-level rules governing acceptable use of computing resources, security practices, and operational procedures.

**Proxy** A software agent that acts on behalf of a user. Typical proxies accept a connection from a user, make a decision as to whether or not the user or client IP address is permitted to use the proxy, perhaps does additional authentication, and then completes a connection on behalf of the user to a remote destination.

**Screened Host** A host on a network behind a screening router. The degree to which a screened host may be accessed depends on the screening rules in the router.

**Screened Subnet** A subnet behind a screening router. The degree to which the subnet may be accessed depends on the screening rules in the router.

**Screening Router** A router configured to permit or deny traffic based on a set of permission rules installed by the administrator.

**Session Stealing** See IP Splicing.

**Trojan Horse** A software entity that appears to do something normal but which, in fact, contains a trapdoor or attack program.

**Tunneling Router** A router or system capable of routing traffic by encrypting it and encapsulating it for transmission across an untrusted network, for eventual de-encapsulation and decryption.

**Social Engineering** An attack based on deceiving users or administrators at the target site. Social engineering attacks are typically carried out by telephoning users or operators and pretending to be an authorized user, to attempt to gain illicit access to systems.

**Virtual Network Perimeter** A network that appears to be a single protected network behind firewalls, which actually encompasses encrypted virtual links over untrusted networks.

**Virus** A replicating code segment that attaches itself to a program or data file. Viruses might or might not not contain attack programs or trapdoors. Unfortunately, many have taken to calling *any* malicious code a "virus". If you mean "trojan horse" or "worm", say "trojan horse" or "worm".

**Worm** A standalone program that, when run, copies itself from one host to another, and then runs itself on each newly infected host. The widely reported "Internet Virus" of 1988 was not a virus at all, but actually a worm.

# References


[1] Steven M. Bellovin. Firewall-friendly FTP. RFC 1579.

[2] Matt Curtin. Snake oil warning signs: Encryption software to avoid. USENET FAQ, 1996–1997.

[3] Matt Curtin, Gary Ellison, and Doug Monroe. Why anti-virus software cannot stop the spread of email worms, May 2000.

[4] R. Finlayson. Ip multicast and firewalls. RFC 2588, May 1999.

[5] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address allocation for private internets. RFC 1918, February 1996.

[6] R. Thayer, N. Doraswamy, and R. Glenn. IP Security Document Roadmap. RFC 2411, November 1998.