A History and Survey of Network Firewalls

KENNETH INGHAM
Kenneth Ingham Consulting
and
STEPHANIE FORREST
University of New Mexico

Firewalls are network devices which enforce an organization's security policy. Since their development, various methods have been used to implement firewalls. These methods filter network traffic at one or more of the seven layers of the ISO network model, most commonly at the application, transport, and network, and data-link levels. In addition, researchers have developed some newer methods, such as protocol normalization and distributed firewalls, which have not yet been widely adopted.
Firewalls involve more than the technology to implement them. Specifying a set of filtering rules, known as a *policy*, is typically complicated and error-prone. High-level languages have been developed to simplify the task of correctly defining a firewall's policy. Once a policy has been specified, the firewall needs to be tested to determine if it actually implements the policy correctly. Little work exists in the area of firewall theory; however, this article summarizes what exists. Because some data must be able to pass in and out of a firewall, in order for the protected network to be useful, not all attacks can be stopped by firewalls. Some emerging technologies, such as Virtual Private Networks (VPN) and peer-to-peer networking pose new challenges for firewalls.

Categories and Subject Descriptors: C.2.0 [**COMPUTER-COMMUNICATION NETWORKS**]: General

General Terms: security

Additional Key Words and Phrases: Firewalls, Network Security

The University of New Mexico Computer Science Department Technical Report 2002-37.
Author's addresses: K. Ingham, Kenneth Ingham Consulting, 1601 Rita Dr NE, Albuquerque, NM 87106-1127, ingham@i-pi.com. S. Forrest, Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, forrest@cs.unm.edu.

1. INTRODUCTION

The idea of a wall to keep out intruders dates back thousands of years. For example, over two thousand years ago, the Chinese built the Great Wall as protection from neighboring northern tribes. A second example is that of European kings who built castles with high walls and moats to protect themselves and their subjects, both from invading armies and from marauding bands intent on pillaging and looting. The term "firewall" was in use by Lightoler as early as [1764] to describe walls which separated the parts of a building most likely to have a fire (e.g., a kitchen) from the rest of a structure. These physical barriers prevented or slowed a fire's spread throughout a building, saving both lives and property. A related use of the term arose in connection with steam trains, as described by Schneier [2000]:

> Coal-powered trains had a large furnace in the engine room, along with a pile of coal. The engineer would shovel coal into the engine. This process created coal dust, which was highly flammable. Occasionally the coal dust would catch fire, causing an engine fire that sometimes spread into the passenger cars. Since dead passengers reduced revenue, train engines were built with iron walls right behind the engine compartment. This stopped fires from spreading into the passenger cars, but didn't protect the engineer between the coal pile and the furnace.

In this paper we will be concerned with firewalls in a more modern setting—computer networks. The predecessors to firewalls for network security were the routers used in the late 1980s to separate networks from one another. A network misconfiguration which caused problems on one side of the router was largely isolated from the network on the other side. In a similar vein, so-called "chatty" protocols on one network (which used broadcasts for much of their configuration) would not affect the other network's bandwidth [Avolio 1999; Schneier 2000]. From these historical examples we can see how the term "firewall" came to describe a device or collection of devices which separates its occupants from potentially dangerous external environments (e.g., the Internet). A firewall is designed to prevent or slow the spread of dangerous events.

For the purposes of this paper, we define a firewall as a machine or collection of machines between two networks, meeting the following criteria:

—The firewall is at the boundary between the two networks;

—All traffic between the two networks must pass through the firewall;

—The firewall has a mechanism to allow some traffic to pass while blocking other traffic. The rules describing what traffic is allowed enforce the firewall's *policy*.

Additional desirable criteria include:

—Resistance to security compromise;

—Auditing and accounting capabilities;

—Resource monitoring;

—No user accounts or direct user access;

—Strong authentication for proxies (e.g., smart cards rather than simple passwords);

—Fail-safety. If it fails, the protected system(s) is(are) still secure because no traffic is allowed to pass through the firewall.

Our review of firewall technologies proceeds roughly in historical order. Interestingly, the historical development of firewalls parallels the descending levels of the protocol stack:

ISO level	**Internet example**
Application	File Transfer Protocol (FTP), Telnet
Presentation	e.g., Common Object Request Broker Architecture (CORBA)
Session	*no directly corresponding protocol*
Transport	Transport Control Protocol (TCP), User Datagram Protocol (UDP)
Network	Internet Protocol (IP)
Data link	Ethernet or Asynchronous Transfer Mode (ATM)
Physical	twisted pair or fiber optic cable

The protocols used on the Internet for these layers, as well as all other Internet standards are specified by documents known as Requests for Comments (RFCs). RFCs often provide information beyond the bare specifications of the standard, and can be useful network administrators, and they appear frequently as citations in this article. For more information about RFCs, read the RFC frequently-asked question list [The RFC Editor 2001].

Following the order of much of the historical development of firewalls, this paper is organized around the levels of the protocol stack, describing approaches which filter or are otherwise directed at each level. Section 2 sets the stage by describing the many reasons why organizations and individuals need firewalls. The next section, Section 3, describes other surveys of firewall technologies. Sections 4-7 descend through the protocol stack, discussing firewall technologies at each of the four relevant levels. Figure 1 provides a graphical view of several of the architectures that we cover in remainder of the paper. Many of the firewalls we discuss use or have used proxies as a major part of how they protect networks. A proxy is a program that receives the traffic destined for another computer. Proxies sometimes require user authentication; they then verify that the user is allowed to connect to the destination, and then they connect to the destination service on behalf of the user.

Section 8 considers firewall designs which do not fit cleanly into one of the various network protocol layers, for example, methods addressing multicast services, distributed firewalls, and protocol normalization. Firewalls are a large part of the commercial network security market. Many companies market products which filter network traffic one of the ISO levels. Although not the primary focus of this paper, in Section 9 we consider important examples of these commercial products and show how they fit into the existing literature. Specifying firewall policies can be a complex task, and consequently, some higher-level languages have been developed in which to express policies (Section 10). Once a policy has been specified, the next question is, "How can it be determined that a given firewall correctly implements the policy?" Firewall testing is one way to address this question, Section 11 reviews the literature on this topic. Although little theory has been developed about firewalls, Section 12 describes the existing theoretical literature.

Very few computer networks can afford to be completely isolated from external networks. Thus, an important job of a modern firewall is controlling what traffic is allowed through the firewall. Detecting hostile data in this traffic is an important problem confronted by modern firewalls, one that we discuss in Section 13. This discussion leads to the related topics of expected future challenges for firewalls (Section 14) and some speculations about how firewalls have influenced recent protocols and how they are likely to change in the future (Section 15).

2. THE NEED FOR FIREWALLS

In the early years, the Internet supported a relatively small community of compatible users who valued openness for sharing and collaboration. This view was challenged by the Morris Worm [Spafford 1988; Eichin and Rochlis 1989; Denning 1989; Spafford 1991]. However, even without the Morris worm, the end of the open, trusting community would have come soon through growth and diversification. Examples of successful or attempted intrusions around the same time include: Clifford Stoll's discovery of German spies tampering with his system [Stoll 1988; 1989], and Bill Cheswick's "Evening with Berferd" [1992] in which he set up a simple electronic "jail" for an attacker. In this jail, the attacker was unable to affect the real system but was left with the impression that he or she had successfully broken in. Cheswick was able to observe everything the attacker did, learning from these actions, and alerting system administrators of the networks from where the attacks were originating. Such incidents clearly signaled the end of an open and benign Internet. By [1992] Steve Bellovin described a collection of attacks that he had noticed while monitoring the AT&T firewall and the networks around it. The

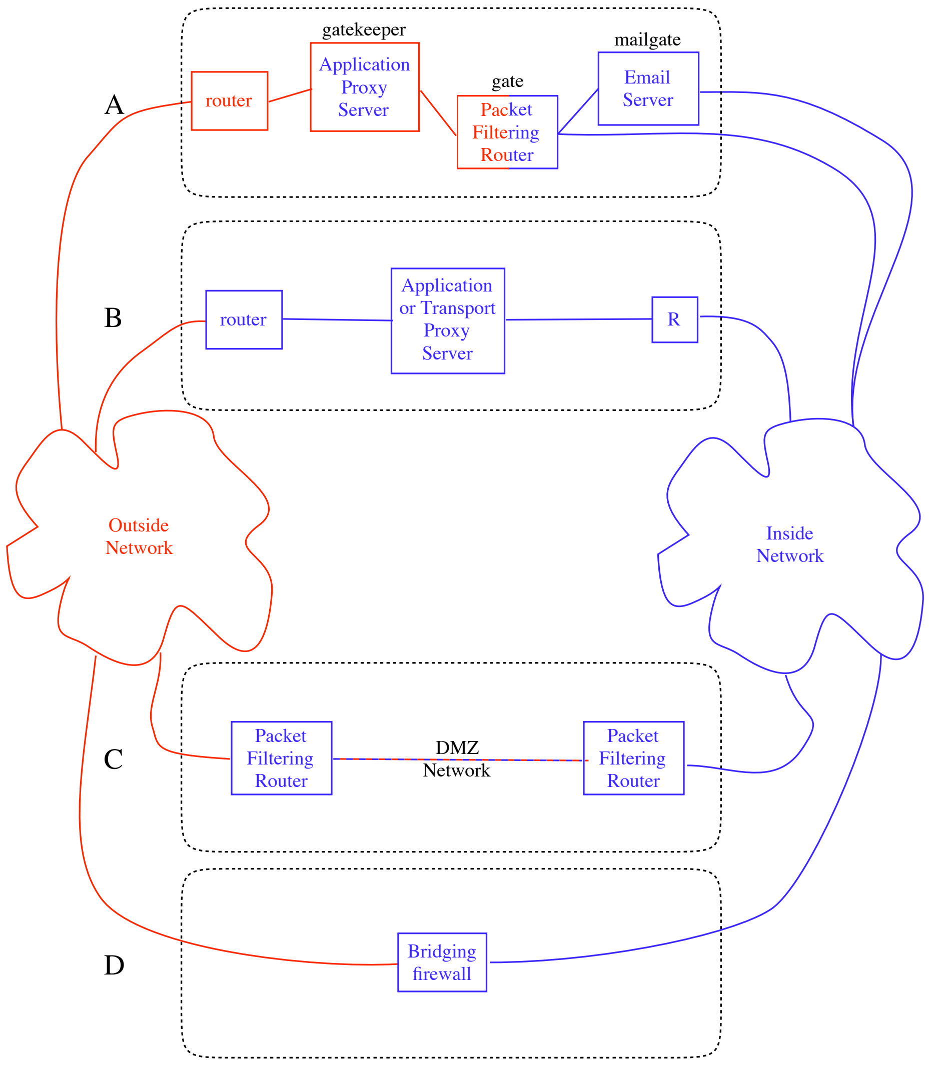

Fig. 1. Common firewall architectures, referred to throughout this paper. The blue portion of the picture represents the trusted parts of the network, and the red portion indicates machines or networks which are not trusted; these colors came from the original DEC documentation for their firewall described in Section 4.1. A) The DEC SEAL from Section 4.1. B) A firewall using only application- or transport-level proxies, a more common architecture than the DEC SEAL architecture due to the reduced cost of hardware. The box containing just the "R" is an optional router in this architecture. This architecture with the transport-level proxy server is similar to the AT&T firewall described in Section 5.1. C) A firewall using two packet filtering routers (described in Section 6.1) and illustrating one way of creating a DMZ, sometimes also known as a screened network. A DMZ network consists of a network of machines offering services to the Internet. If a machine here is compromised, the inside network remains safe. D) A bridging firewall similar to that described in Section 7.

result was clear—there were many untrustworthy and even malicious users on the Internet.

Because not everybody can be trusted, when networks are connected together, a different level of trust often exists on the different sides of the connection. “Trust” in this sense means that an organization believes that the both the software and the users on its computers are not malicious. Firewalls can be used to enforce trust boundaries, which are imposed for a variety of reasons:

Security problems in operating systems:. Operating systems have a history of insecure configurations. For example, Windows 95 and Windows 98 were widely distributed with windows file sharing turned on by default; a collection of viruses exploited this vulnerability (e.g., [Computer Emergency Response Team (CERT) 2000b; 2000c]). A second example is Red Hat Linux versions 6.2 and 7.0, which were vulnerable to three remote exploits when installed using default options [Computer Emergency Response Team (CERT) 2001a]. It is an on-going and expensive process to secure every user’s machine, and many organizations make a conscious decision not to secure the machines inside their firewall. If a machine on the inside is ever compromised, the remaining machines are likely as vulnerable [Muffett 1995], a situation that has been described by Cheswick as “a sort of crunchy shell around a soft, chewy center” [1990].

Individuals can protect a single machine connected to the Internet by purchasing a personal firewall. Rather than trying to secure the underlying operating system, these firewalls simply prevent some types of communication. Such firewalls are often used in homes and on laptops when they are outside their normal firewall. In this case, the trust boundary is the network interface of the machine.

Preventing access to information:. A second example of protecting a network is the use of national firewalls, for example, China [McKay 1998]. This firewall exists not to protect them from attack, but instead to (attempt to) limit the activities of their users on the Internet. A similar idea is the use of filtering mandated in the US by the Children’s Internet Protection Act (CHIPA). This law requires that schools and libraries which receive federal funding put certain filters on all web content.

Preventing Information Leaks. Because all traffic leaving a network must pass through the firewall, it can be used to reduce information leaks, as in [Ranum 1992]:

> The key criterion for success for the Digital corporate gateways is preventing an unauthorized or unnoticed leak of data to the outside.

Enforcing Policy. Firewalls are one part of an overall security policy; they enforce the policy of network traffic allowed to enter or leave a network. These policies may limit the applications used, remote machines which may be contacted, and/or the bandwidth allowed [Treese and Wolman 1993; Hambridge and Sedayao 1993].

Auditing. If a security breach (which does not include the firewall) occurs, audit trails can be used to help determine what happened. Audit trails have also been used to take employment actions against employees using work network resources for non-work purposes.

3. OTHER SURVEYS

Firewalls have existed since about 1987, and several surveys and histories have already been written. However, none of them provide both the depth and breadth of this survey, nor do they focus on the peer-reviewed literature describing firewall technology.

In [1994], Alec Muffett wrote a paper which provided an excellent review of the firewall policies and architectures of the time. This paper was aimed at people considering implementing a firewall, describing the technologies which they might select, their tradeoffs, and how to maintain a firewall.

One section of the Internet standards document RFC 1636 [Braden et al. 1994] is about the status of firewalls as of February, 1994. In this section, they discuss the problem of false security that a firewall often provides to an organization behind one. They also review the concepts of application- and transport-level proxies, as well as simple packet filtering. They discuss the problems of authentication and enforcing policy, and provide some solutions from the time. One of the most important parts of the firewall section is a discussion of how to design firewall-friendly protocols in the future.

A review of firewalls and their technology appeared in Spectrum [Lodin and Schuba 1998]. This paper is an excellent description of firewalls and their technology at the time it was written. However, it has no references to peer-reviewed literature.

Several books have been written which describe how to build a firewall (e.g., [Cheswick and Bellovin 1994; Zwicky et al. 2000]). These books are excellent for people wanting to either evaluate a commercial firewall or who are implementing their own firewall. However, neither spends much time on firewall history, nor do they provide references to peer-reviewed literature.

In [1997], John Schimmel wrote a historical review of firewall technologies aimed at technical people working in the field of system administration. This review contains good history about early packet filters and a brief overview of proxies. Schimmel also mentions limitations of firewalls, many of which remain to this day and are discussed in this paper in Section 13. Unfortunately, this paper has no references to the original sources of the works described.

A survey of existing firewall technology appeared in Schuba's Ph.D. dissertation, [Schuba 1997]. In this dissertation, Schuba cites many key papers, but he reviews the literature as it relates to technology, and does not provide as comprehensive a collection of firewall-related references as we do in this paper. Steph: is this too harsh? His work is not bad, and his entire thesis has 148 references. However, his review of firewall literature is only 23 references. His review is really weak compared to this paper.

Also in [1998], Rik Farrow wrote a firewall product analysis which was related to the CSI firewall comparison for that year. This analysis is aimed at management and people just arriving at firewalls, and provides them with the background information they would need to talk with a firewall vendor intelligently.

More recent publications include Frederic Avolio's history of firewalls published in the Cisco Internet Protocol Journal [1999]. Avolio is well-qualified to write such a document, as he was involved with some of the first firewalls. His history describes some of the beginnings of firewalls, from a technical point of view, and aimed at technical people and technically-oriented management. He provides a short description of firewalls which use proxies at the application or transport levels, as well as packet filtering and firewalls which may be a mix of technologies. Rather than providing details, he refers the reader to Cheswick and Bellovin's [1994] book on firewalls. As a contrast with Avolio's history, this paper places emphasis on the academic literature and as a result has substantially more references than Avolio's history.

Habtamu Abie wrote an overview of current firewall technology options and emerging trends in [2000]. He discusses the technology, but does not cite the papers by the people who originally developed this technology. Also, Yakomba Yavwa wrote a similar but less in-depth overview [2000]. Like Abie, Yavwa cites none of the original papers where the technology was first described.

Finally, several trade and consumer magazines have reviewed firewalls; [Markus 2001] is a web site with links to many of them; the Computer Security Institute (CSI) in San Francisco, California has a search engine for comparing different commercial firewall products [Computer Security Institute 2001].

Several related topics which we do not address thoroughly in our review include intrusion detection systems [Northcutt and Novak 2001; Axelsson 1998], honeypots [Cheswick 1992; Honeynet Project 2001], IPsec implementations [Doraswamy and Harkins 1999], and commercial products.

4. THE APPLICATION LEVEL

The application layer contains programs that interact directly with a user. Many application-level protocols exist, including FTP, the Simple Mail Transport Protocol (SMTP), the HyperText Transport Protocol (HTTP), Telnet, etc. The first published paper to describe filtering network traffic for access control, by Jeffrey

Mogul in [1989], described a system which worked at the application level.

4.1 The DEC Securing External Access Link

Mogul described a user-level[1] solution to deciding whether or not to pass packets though a router running the UNIX operating system [Mogul 1989]. The packet filter monitored the protocol, source and destination addresses and ports to decide which packets were allowed to continue. Mogul's implementation did not keep track of the state of TCP connections, nor the pseudo-state that can be applied to some UDP requests and responses. However, his implementation could have been expanded to handle these; performance limitations of the computers of the day might have prevented this.

The technology described by Mogul was applied to Digital Equipment Corporation's (DEC) firewall by Paul Vixie [Ranum 1992] and further extended by Marcus Ranum when it became the Securing External Access Link (SEAL)[2], which was the first commercial firewall [Ranum 2001; Avolio 1999], delivered on June 13, 1991 [Avolio 1999]. The DEC SEAL consisted of three components (pictured in Figure 1, part A):

Gatekeeper. The application proxy[3] server for users who were allowed to access external services. **Gatekeeper** was also an externally-accessible machine for uses such as anonymous FTP, the Domain Name System (DNS), etc.

Gate. A packet filtering router limiting what traffic was allowed to pass between the local and external network. This router was configured so that all traffic to/from the inside went to a proxy on **gatekeeper**. *screend*, as described in Mogul's paper [Mogul 1989], performed the packet filtering.

Mailgate. The internal mail server/mail router; this machine was not accessible from the outside. Instead, mail from the outside is delivered to **gatekeeper**, which passes it to **mailgate**.

External connections are permitted only to **gatekeeper**, and it may (depending on the configuration) also be the only way internal users are allowed to access services external to the network.

Gatekeeper ran application-level proxies. These proxies also ensured that the proper protocol was used (e.g., that FTP traffic is what went to port 20, and not a Telnet session). The proxies on **gatekeeper** were [Ranum 1992; Treese and Wolman 1993]:

[1] As opposed to a kernel-level.
[2] Some documentation referred to it as the Screening External Access Link.
[3] Proxies were described in detail on page 3.

—email and USENET news (since these are store-and-forward protocols, no special handling was needed)

—Telnet

—FTP

—WHOIS (a protocol for looking up network contact information)

—*sup* (a software distribution system developed at Carnegie Mellon University)

—the X window system. X is an especially hard protocol to secure, since the client and server are "backwards": the server is on the desktop and the client is a remote machine. Treese and Wolman at DEC Cambridge succeeded in developing a proxy which needed no changes to the X client or server code at all [Treese and Wolman 1993].

All access across the firewall was through these application proxies. The proxies provided several services [Digital Equipment Corporation 1992]:

—They validated that the source and destination machines were allowed to communicate with each other.

—They ensured that the user was allowed to access the service and remote machine specified.

—They maintained an audit trail—all successful and unsuccessful connections were logged.

—They enforced that the protocol behaved as expected (e.g., that *telnet* was not used to connect to an interactive access terminal server which happened to be running on the FTP port).

When a user wanted to send traffic outside of the local network, the traffic was passed through **gate** to the proxy on **gatekeeper**. If the traffic was allowed, **gatekeeper** then sent the traffic on to the destination machine. No traffic was allowed to pass directly from the local machine to the remote machine (and correspondingly, no traffic was allowed to pass directly from the remote machine to the local machine).

The DEC firewall did not allow any UDP traffic to pass through it at all. This prohibition was probably for two reasons:

—Some of the UDP protocols at the time (e.g., NTP, DNS) could have a server running on **gatekeeper** to which the clients connected.

—Other UDP-based protocols, such as the Network File System (NFS), would have security risks if they were allowed through the firewall.

The DEC SEAL is a classic example of a firewall which uses application-level proxies. Proxies at this level provide excellent security and auditing. Also, because they are themselves generating the connection to the remote machine, a proxy has no problems knowing which connections are real and which are being spoofed; this is in contrast to some packet filtering firewalls (Section 6).

One drawback of an application proxy is that it must be designed for the specific protocol. New protocols are developed frequently, requiring the development of new proxies; if there is no proxy, there is no access. Often this time lag creates friction with the users, who may be demanding access to the latest applications, an example of the classic tradeoff between security and user convenience.

Application proxies have other drawbacks. In order to use them, the client program must be changed to accommodate the proxy, telling it about the actual packet destination and authenticating directly to the proxy. Because source code is not publicly available for some applications, in many cases these changes can be made only by the application's vendor, a significant bottleneck. A final drawback of application proxies is performance related, because each packet requires two trips through the complete network protocol stack. None of the papers about the DEC firewall discussed performance directly.

Over time, the DEC SEAL evolved into the AltaVista firewall. The architecture appears to have remained similar, although it was collapsed onto one machine, probably to stay cost-competitive with its competition [Mariet 1997; Smith et al. 1997], resulting in an architecture hybrid of Figure 1, parts B and D. Today, the AltaVista firewall appears to no longer exist.

5. THE TRANSPORT LEVEL

In the Internet, the transport level consists of only two protocols, TCP and UDP. This small number of protocols makes writing a proxy much easier—one proxy suffices for all protocols which use TCP. Contrast this with the application-level proxies, where a separate proxy is required for each service, e.g., Telnet, FTP, HTTP, SMTP, etc.

Transport-level proxies retain the advantage that a machine outside of the firewall cannot send packets through the firewall which claim to be a part of an established connection (some of the packet filters described in Section 6 have this problem). Because the state of the TCP connection is known by the firewall, only packets which are a legitimate part of a communication are allowed inside of the firewall.

5.1 AT&T's Transport-Level Firewall

While DEC was building SEAL, Dave Presotto wrote and Bill Cheswick re-wrote a firewall for AT&T [Cheswick 1990]. AT&T's firewall also relied on proxies, a term first used in the context of firewalls by Bill Cheswick in [Cheswick 1990]. The resulting firewall was similar in effect to that of Figure 1, part B, although it was composed of multiple machines.

In contrast to the application-level proxies in the DEC SEAL, the AT&T proxies worked at the transport level [Cheswick 2001]. This difference arose from the different policies at DEC and AT&T. The DEC policy limited outbound connections, with the goal of preventing unauthorized leaks of information to the outside. AT&T did not limit outbound access other than requiring it to pass through the proxy.

As with application-level proxies, users paid a performance penalty because each datagram had to traverse the network protocol stack twice. Cheswick noted that file transfers without the proxy ran at 60-90 Kb/sec, while through the proxy they ranged from 17-44 Kb/sec. Like application-level proxies, transport-level proxies require that the client program be modified to use the proxy (however, modifying a program to use a transport-level proxy is easier than modifying it to use an application-level proxy; see SOCKS in Section 5.2).

5.2 Later Proxies

Kolbas and Kolbas wrote SOCKS [Koblas and Koblas 1992; Leech et al. 1996; Leech 1996; McMahon 1996] with the goal of simplifying the use of proxies. A SOCKS call replaced a normal socket call, which resulted in all outbound traffic using the proxy. This approach was a nice, clean solution, and worked well if one had the source code for the relevant operating system utilities. Some commercial applications (e.g., Netscape) also were written to accommodate SOCKS. A system using SOCKS and TCP connections could be transparent to the user (assuming the proxy allowed them access to their destination host). In [2000], Fung and Chang describe an enhancement to SOCKS to allow it to handle UDP streams, such as that used by RealNetworks' RealPlayer.

Marcus Ranum and Frederick Avolio worked on the Trusted Information Systems (TIS) Firewall Toolkit (FWTK), a collection of proxies for building firewalls [Ranum and Avolio 1994; Avolio and Ranum 1994]. This freely-available toolkit provided SMTP, the Network News Transport Protocol (NNTP), FTP and Telnet application proxies as well as a generic circuit-level proxy. To improve security, the proxies used the UNIX system call *chroot* to limit how much of the system became visible; this way if a proxy were compromised, the rest of the firewall was more likely to remain trustworthy. The TIS FWTK had no proxies for UDP services; instead, the firewall machine ran DNS and the Network Time Protocol (NTP). The inside machines used the firewall for those services. When Trusted Information Systems and Network Associates, Inc. (NAI) merged in February 1998, the TIS firewall became NAI's Gauntlet Internet Firewall.

6. THE NETWORK LEVEL

If a limited number of protocols at the transport level is an advantage, then it would seem that the single protocol, IP, at the network level would be even better. In some ways, it is. Packet filtering is fast, and it does not require the knowledge or cooperation of the users. However, a packet filter working only at the network level means that it cannot determine whether a packet belongs to an established communication or not. And, similarly to transport-level proxies, it cannot enforce which application-level protocol is used. For organizations trying to prevent an information leak, this is a serious limitation. It is relatively straightforward to use allowed protocols to send arbitrary data; for an example, see [daemon9 and Alhambra 1996] which describes communication using the Internet Control Message Protocol (ICMP) echo ("ping") and ICMP echo reply ("pong") packets. In spite of these impediments, packet filtering at the IP level has been successful when state information about the connection is used (Section 6.2).

6.1 Packet Filtering

Packet filtering for network security began with Mogul's paper about *screend* [1989]. Whereas application- and transport-level proxies require each datagram to make a trip up and down the whole protocol stack, packet filtering can be much faster. Most of the early papers on packet filtering for security stressed the performance benefits [Corbridge et al. 1991; Chapman 1992; Bailey et al. 1994; Molitor 1995]; later papers continued this trend [Suri and Varghese 1999; Lyu and Lau 2000]. Besides the performance benefits, packet filtering does not require the cooperation of the users, nor does it require any special action on their part [Weber 1999].

Packet filters use one or more of the following pieces of information to make their decision on whether or not to forward the packet [Reed 2002a]:

—source address

—destination address

—options in the network header

—transport-level protocol (i.e., TCP, UDP, ICMP, etc.)

—flags in the transport header

—options in the transport header

—source port or equivalent if the protocol has such a construct

—destination port or equivalent if the protocol has such a construct

—the interface on which the packet was received or will be sent

—whether the packet is inbound or outbound

Although packet filtering is fast, it does have drawbacks. The most important limitation is the difficulty of writing correct filters. Several researchers have pointed out this problem and attempted to solve it. Mogul in [1991] discusses the difficulty of setting up packet filtering rules and provides correct examples as illustrations. In [1992], Chapman makes the point that most packet filter languages are similar to assembly language. In [1995], Molitor describes an improved commercial filter language.

Another drawback of packet filtering is that it cannot determine which user is causing which network traffic. It can inspect the IP address of the host where the traffic originates, but a host is not the same as a user. If an organization with a packet-filtering firewall is trying to limit the services some users can access, it must either implement an additional, separate protocol for authentication or use the IP address of the user's primary machine as a weak replacement for true user authentication.

Also, because IP addresses can be spoofed, using them for authentication can lead to other problems. If the router is running a properly configured filter, remote attackers should not be able to spoof local addresses, but they could spoof other remote addresses. Local machines can spoof other local machines easily. In spite of these problems, many organizations are using IP addresses or DNS names for access control.

When a proxy is used, the connection to the remote machine comes from the machine running the proxy. With packet filters, the local machine directly initiates the connection to the remote machine. A result of this difference is that the entire internal network is potentially reachable from external connections; otherwise the reply packets from the remote host would not be delivered properly. As a consequence, hostile remote computers may be able to exploit weaknesses in the protocol implementation of the local machine (e.g., [Securityfocus.com 1997]).

Protocols such as FTP [Postel and Reynolds 1985] also cause problems for packet filters. FTP uses a control channel opened from the client to the server for commands. However, when getting a file, one method of using FTP (active FTP) has the server open a connection back to the client. This connection is initiated from the FTP data port (TCP port 20) to a random client port. Since malicious programs could use the FTP data port as their source port, packet filter writers cannot assume that a connection from TCP port 20 is a FTP transfer. One solution is to use passive FTP, where the client opens the connection for the data transfer to the server. However, not all FTP clients support passive transfers. RFC 1579 [Bellovin 1994] suggests that FTP client writers include support for the passive method of file transfer, but this solution has yet to be universally adopted.

6.2 Packet Filtering with State

Originally, packet filters did not keep track of the state of a connection. This means that a remote host could send in packets which appeared to be part of an established TCP connection (with the TCP ACK flag set), but which in reality were not. This allowed a remote attacker to map the local network as if the firewall did not even exist. Attacks against bugs in the TCP/IP protocol stack (e.g., [Securityfocus.com 1997]) can pass through the firewall by claiming to be part of an established TCP session.

The solution to this problem is to keep track of the state of a connection, a property referred to variously as stateful firewalls, adaptive firewalling and packet inspection. In other words, the packet filter looks at both the network level and the transport level data. The router monitors the initial TCP packets with the SYN flag set and allows the packets to return through only until the FIN packet is sent and acknowledged. A similar pseudo-state can be kept for most UDP (e.g., DNS, NTP) and some ICMP communication (e.g., ping)—a request sent out opens a hole for the reply, but only for a short time. In [1992], Chapman was one of the first to point out the problem of the lack of state in packet filtering firewalls. In [1995], Molitor identified the problem of implementing state in his packet filtering architecture as a "future direction." The first peer-reviewed paper to describe adding state to packet filters was by Julkunen and Chow in [1998], where they describe a dynamic packet filter for Linux. This paper was proceeded by a technical report in [1997].

Although Julkunen and Chow were the first to publish a peer-reviewed paper about keeping state in packet-filtering firewalls, they were not the first to keep track of connection state. IP Filter (IPF) version 3.0 in 1996 by Darren Reed predated their work [Reed 2002b; 2002c]. The first peer-reviewed description of this work appeared much later in a discussion of state in IPF [van Rooij 2001].

6.3 Improving Packet Filter Specification

In [1992], Chapman makes the point that writing packet filter rules is similar to writing in assembly language. Some researchers have therefore developed higher-level languages for specifying packet filters. These improved packet filter languages are simpler than the policy specifications mentioned in Section 10, and they do somewhat ease the writing of filter rules. Specific examples include:

—In [2000], Hazelhurst proposed binary decision diagrams (BDDs) for visualizing router rule sets. Since BDDs represent boolean expressions, they are ideal for representing the block/pass rules which occur in packet filters. BDDs also make automated analysis of packet filter rules easier, as well as providing better performance than the table lookups used in many routers.

—The filter language compiler, *flc* [Reed 19], allows the use of the C preprocessor, specification of a default block or pass policy for various directions of traffic flow, and provides a simple if-then-else facility. *flc* also generates rules for several different packet filters (IPF, *ipfw*, *ipfwadm*, *ipfirewall*, Cisco extended access lists, and *screend*).

6.4 Network Address Translation (NAT)

Because the Internet is short of IPv4 addresses, many people use Network Address Translation (NAT) to gain more mileage out of a single IP address [Egevang and Francis 1994; Srisuresh and Egevang 2001; Hain 2000]. When a router uses NAT, it changes the source address of outbound packets to its own address (or one from a pool of addresses which it controls). It chooses a local, unused port for the upper layer protocol (TCP or UDP), and stores in a table the association between the new address and port and the real sender's address and port. When the reply arrives, it looks up the real destination in this table, rewrites the packet, and passes it to the client. When the connection is finished (or after a timeout period for UDP packets), the entry is removed from the table.

NAT provides a similar form of protection to that of proxies. In NAT, all connections originate from the router performing the address translation. As a result, someone outside the local network cannot gain access to the protected local machines unless the proper entry exists in the table on the router. The network administrator can manually install such an entry, causing all traffic destined for a specific port to be forwarded to a server for that service (in effect, providing an Internet-accessible service on an inside machine.[4]).

RFC 2663 [Srisuresh and Holdrege 1999] notes that NAT is not without its problems. For example, NAT may prevent IPsec from working correctly. One feature of IPsec is the ability to know that a packet was not modified during transit. However, one of the purposes of NAT is to modify the packet—the source address and possibly the source port must be modified for NAT to work. DNS problems can also occur. A machine behind a router using NAT has a name and an IP address. However, most networks using NAT also use RFC1918 [Rekhter et al. 1996] private IP addresses, which are not globally unique. Therefore, DNS inside the network is not meaningful outside the network.

[4]Setting up such an entry is usually a bad idea from a security standpoint. Maintaining a server inside a firewall is risky because if it is compromised, the attacker then has access inside the network, which as noted in Section 2 is likely to be insecure.

7. THE DATA-LINK LAYER

A bridge is a network device which works at the ISO data-link layer. Because it operates at this level, it does not need to access to routing information. A bridging firewall uses the information listed in Section 6.1 to decide whether or not to block a packet. As a result, a bridging firewall can look at data in several other levels of the Internet Protocol suite, including the network and transport layers. Because a filtering bridge is still a filter, the disadvantages of packet filtering still apply to it.

What makes a bridging firewall different from a packet filtering router is that it can be placed anywhere because it is transparent at the network level (see Figure 1, part D). It can be used to protect a single machine or a small collection of machines that would normally not warrant a separate network required when using a router. As it does not need its own IP address, the bridge itself can be immune to any attack which makes use of IP (or any of the protocols on top of IP). And, no configuration changes are needed in the protected hosts when the firewall is installed. Installation times can be minimal (for example, Limoncelli claims three second install times [Limoncelli 1999]), so users are minimally disrupted when the bridge is installed.

The first bridging firewalls were described by Kahn et al. in [1997] and developed for computers running DOS. They refer to earlier unpublished research concerning packet filtering filtering on a bridge. Later work which also explores the idea of a bridging firewall includes Jianbing Liu and Yan Ma in [Liu and Ma 1999]. Keromytis and Wright discuss using IPsec on a bridging firewall to set up secure, virtual LANs [2000].

8. OTHER APPROACHES TO FIREWALLS

Some firewall designs do not fit cleanly into a single network protocol layer. Some of them (e.g. Normalization, discussed in Section 8.2) are ideas that can be implemented in a firewall using one of the technologies already discussed. These proposals are just now being incorporated in firewall implementations. Others, such as the distributed firewalls discussed in Section 8.5 are more revolutionary, changing the entire architecture of the firewall. Although a few organizations may be using firewalls based on these more revolutionary ideas, they are not yet in widespread use.

8.1 Transparent Proxies

As mentioned in Sections 4.1 and 5.1, the primary problem with proxies has been that the client software must be modified and/or the user must work differently when using the. Transparent proxies address this limitation. With a transparent proxy the client sends packets to the destination in a normal fashion. However, when the packets reach the firewall, access control checks and logging are performed as in a classical proxy system. The "magic" is implemented by the firewall, which notes the destination address and port, opens up a connection to it and then replies to the client, as if the proxy were the remote machine. This relaying can take place at either the transport level or the application level. RFC 1919 [Chatel 1996] compares classical proxies with transparent proxies. Bonachea and McPeak use transparent proxies to improve the security of FTP in [2001]. Rodriguez et al. describe what they call translucent proxying of TCP in [2001].

Transparent proxies are demanding because the firewall must operate both at the network and application levels. Therefore, performance has remained a concern. One solution proposed by Spatscheck et al. [2000] and Maltz and Bhagwat [1998] is that of "splicing." In splicing, after the proxy verifies that communication is allowed to proceed, the firewall converts to a network-level packet filtering firewall for that communication. Splicing provides the extra control of proxies but maintains performance closer to that of packet filters.

8.2 Normalization

Attackers often abuse protocol specifications, e.g., by sending overlapping IP fragments or out-of-order TCP byte sequences. In [2001], Handley et al. stressed that a firewall is a good location for enforcing tight interpretation of a protocol. Besides protecting the computers behind the firewall from attacks based on protocol abuses, this so-called "normalization" also makes signature-based intrusion detection systems more reliable because they see a consistent data stream. Handley et al. provide a list of possible normalizations, ranging from those guaranteed to be safe to others that are potentially too strict in their interpretation of the standard. They were not the first to suggest normalization, however. In [2000], Malan et al. describe "transport scrubbing," and more recently the idea is elaborated in [Watson et al. 2001]. At about the same time, Strother [Strother 2000] proposed a similar idea. Her solution involved different rings of trust, in which a network packet must pass through one ring before proceeding to the next. Many of her rings achieve the same effect as normalization.

8.3 Signature-based Firewalls

Malan et al. discuss "application scrubbing" in [2000]. In this approach, a user-level program is established as a transparent proxy (see Section 8.1) which monitors the data stream for strings known to be hazardous (and presumably to prevent these strings from reaching the client). Watson et al. refer to the same concept as a "fingerprint scrubber" in [2001].

Snort [Roesch 1999] is a common intrusion detection system. *Hogwash* [Larsen 2001] is a firewall that blocks packets matching the *snort* rules. It runs on a bridging firewall (Section 7) and the authors claim it can handle network speeds of up to 100Mbps on hardware which is not state-of-the-art.

8.4 Firewalls at Other ISO Network Levels

The Common Object Request Broker (CORBA) allows applications written in one language to make requests of objects which may be written in a different language and which may be available on a different machine. The CORBAgate by Dotti and Rees in [1999a; 1999b] is a presentation-level proxy. When a request is made to an object which is on the other side of the firewall, the proxy transparently changes the references. The result is that objects on either side of the firewall end up referring to an object on the firewall.

8.5 Distributed Firewalls

There are several limitations to the firewall technology that we have presented so far. One common assumptions is that all the hosts inside a firewall are trustworthy. This assumption is not always valid—for example, see the discussion about the problems with virtual private networks (VPNs) in Section 14.1. A related problem is that firewalls ignore internal traffic, which may violate security policy. Because firewalls are typically centralized in one location, they can become performance bottlenecks as well as providing a single point of failure. A further limitation of conventional firewalls arises because in some cases the local machines know the context of the communication not available to the firewall. For example, a file transfer may be allowed or denied based on what file is being transferred and by whom. The firewall does not have this local, contextual knowledge.

One solution to these problems, proposed by Bellovin [1999], is a distributed firewall. This was implemented by Ioannidis et al. in [2000] and by Markham and Payne in [2001]. In this firewall, the network administrator has a single policy specification, loaded onto all machines. Each host runs its own local firewall implementing the policy. Machines are identified by cryptographic certificates, a stronger form of authentication than IP addresses. With a distributed firewall, the common concept of a DMZ or screened network, where servers accessible to the outside world are located, is no longer necessary (for an example of a DMZ or screened network, see Figure 1, part C).

Hwang and Gangadharan [Hwang and Gangadharan 2001; Gangadharan and Hwang 2001] propose using firewalls on all devices attached to the network, where they can be combined with an intrusion detection system (IDS). When the IDS detects an anomalous event, it modifies the firewall to react to the threat. Lower overhead can be achieved with this approach than that reported for the distributed firewall developed by Ioannidis [2000]. However, the architecture still uses a conventional firewall.

Distributed firewalls have a different set of problems than centralized ones. The most significant is that a distributed firewall relies on its users (with physical access to the machine) not to override or replace the policy. In other words, the network administrator must trust his or her users. Additionally, if the firewall is running as a part of the operating system, then the operating system must protect the firewall software. However, the local firewall is protecting the operating system, creating a circular set of dependencies. In [2001], Markham and Payne propose implementing the distributed firewall on a programmable network interface card (NIC) to reduce reliance on the operating system for protection.

Around the same time that Bellovin proposed the distributed firewall, Ganger and Nagle at Carnegie Mellon University also proposed a distributed approach to security in [2000], in which each device was responsible for its part of the security policy. Ganger and Nagle argue that if each device were more secure, then an attacker who succeeds in passing the outer defenses (the firewall) would not find vulnerable targets inside. They propose installing security devices on many parts of a network, such as NICs, storage devices, display devices, and networking hardware such as routers and switches. The idea is that the devices would dynamically adjust their approach to security based on the overall network defense level. As with Bellovin's proposal, programmable NICs are an important part of the overall strategy.

8.6 Protection against denial of service attacks

In a denial of service attack, the attacker's goal is to reduce or eliminate an authorized user's access to a service, machine or network by disabling the service or machine, or by overloading some aspect of the system with voluminous but legal activities. Because all traffic for the network travels through the firewall, some denial-of-service attacks can be stopped at the firewall. For example, protocol-based attacks (e.g., [Computer Emergency Response Team (CERT) 1997b] and [Computer Emergency Response Team (CERT) 1998]), which cause the remote machine to crash, can be stopped by protocol normalization (Section 8.2). When attackers are attempting to saturate available bandwidth, Ioannidis and Bellovin proposed the idea of pushback—the ability for a router to detect and preferentially drop packets which are part of an attack, and also to notify upstream routers of the attack [Ioannidis and Bellovin 2002].

In [2000], Balasubramanian described an architecture in which network- and host-based intrusion detection systems are combined with a policy-driven coordinator to respond to denial of service attacks. This response is a job that traditionally would be given to the firewall. As presented, the response could be anything a process can do; it is up to the network administrator to determine the proper response to the attacks. Balasubramanian recognizes but does not address the problem of false positives or spoofed network packets being used to cause a reaction desired by the attacker.

In [1997], Schuba et al. review various approaches to SYN-flood denial-of-service attacks and present a new approach (not limited only to firewalls). They suggest using a program to monitor all traffic on the LAN. They describe a program which categorizes SYN packets into four classes: never seen, belonging to correctly behaving hosts, potentially spoofed, and certainly spoofed. In addition to the program's classification, the administrator can provide a collection of addresses known to be good or bad. When the program sees SYN packets from bad hosts (dynamically characterized by their behavior), it sends a RST packet to kill the half-open connection.

8.7 Multicast

On the Internet, multicast is often used for various forms of multimedia. In contrast with traditional unicast communication, in multicast the sender does not necessarily know the identities of all the participants in the session. And, this is also true for the recipients, who do not know in advance all the possible people who might be sending to them. This difference also makes proxies such as SOCKS difficult to implement unless they change the multicast into a collection of unicasts, a change that defeats the benefits of multicast—with multicast, once one client inside of the firewall has joined a group, others may join without needing to authenticate. Additionally, the multicast routing protocol, the Internet Group Management Protocol (IGMP), specifies only multicast groups and not UDP ports; in a default configuration, a multicast source has access to the complete set of UDP ports on client machines. If a source has access to all UDP ports, then it could potentially attack other services, (e.g. Microsoft networking) which are unrelated to the service it is providing.

The classic paper on multicast and firewalls was published by Djahandari and Sterne in [1997]. In this paper they describe an application proxy for the TIS Firewall Toolkit. The proxy has the following features:

—It allows authentication and auditing;

—It prevents multicast traffic from reaching hosts which did not request it;

—It allows the multicast traffic to be sent only to "safe" ports.

The proxy converts multicast traffic into unicast traffic. Unfortunately, this approach also means that it does not scale well, as a collection of N users all receiving the same multicast stream increases the traffic inside the firewall by a factor of N over what it would have been if multicast had been retained. On the other hand, they do solve all of the security problems mentioned in the previous paragraph and later in this section.

RFC 2588 [Finlayson 1999] suggests several possible solutions to the problem of multicast and firewalls. For example, communication between external and internal machines could be tunneled through the firewall using the UDP Multicast Tunneling Protocol (UMTP). This protocol was designed to connect clients to the Multicast Backbone (the MBone), but would work for tunneling through multicast-unaware firewalls.

RFC 2588 also mentions the possibility of dynamic firewall rules, and in [1999], Oria describes in further detail how they can be implemented. A program runs on the router, which monitors multicast session announcements. The program reads the announcements, and if the specified group and UDP port are allowed by the policy, it generates the necessary rules permitting the data to pass through the firewall. When a client informs the router that it wishes to join a multicast group, it sends an IGMP join message to the router. The dynamically generated rules permit or deny this access. This approach to multicast on the firewall assumes that session announcements can be trusted. Unfortunately, this is not a valid assumption because they can be spoofed.

8.8 Transient Addressing

Many protocols, such as FTP, RealAudio, and H.323 (a protocol used for programs such as Microsoft's NetMeeting), open secondary channels for additional communication. These additional channels are a problem for firewalls unless the firewall makes use of a stateful proxy. In [2001] Gleitz and Bellovin propose a solution to this problem by taking advantage of version 6 of the Internet Protocol (IPv6), which has 128 bits of address space. This is large enough for each host to have multiple addresses. A client initiating a connection to a FTP server uses an address which includes the process group ID of the FTP client process. The firewall sees a connection from a specific IPv6 address going to a FTP server at a remote site, and then allows all communication from the server back to the client's address. On the client side, this address is only used for the FTP process; connections from the FTP server to other ports on the client will not be accepted, because only the FTP client is using that specific address.

9. COMMERCIAL FIREWALL PRODUCTS

It is difficult to separate completely advances in firewall technology from the commercial products that implement them. There is a large market for commercial firewall products, which has driven many important recent developments. At the same time, without direct inspection of the source code, it can be quite difficult to ascertain exactly how or why a particular product operates. Further, the success or failure of commercial products depends as much on business issues as on technical merit. In spite of these problems, no review of firewall technology would be complete without some discussion of commercial products. In the following, we highlight a few of the many products currently on the market, emphasizing those that we believe are interesting intellectually, have had unusual impact on the development of firewalls, or are particularly clear examples of a particular methodology.

9.1 Network Associates, Inc. (NAI)

As mentioned in Section 5.2, the TIS firewall toolkit was an early firewall based on proxies. After the merger with NAI, it became the basis for the Gauntlet firewall. This firewall continued to evolve and now incorporates the following features [Network Associates Technology, Inc. 2001b; 2001c; 2001d]:

—Packet filtering with state;

—Transport-level proxies;

—Transparent application-level proxies for common protocols such as RealAudio, QuickTime, HTTP, SMTP, and FTP;

—Groups to ease the packet filter specification;

—Splicing like was described in Section 8.1 to improve performance;

—Content scanning for known viruses and hostile applications.

9.2 Cisco

Cisco has two firewall products (IOS firewall and Private Internet Exchange (PIX)) which share many features, but also have some differences. Both are based on packet filtering with state. They also perform what is known as "inspection," in which the packet filter inspects the data inside the application layer and modifies the packet filtering rules to adapt to what the application is expected to do. For example, the inspection algorithm could determine that a FTP client will need a port opened, changing the filtering rules appropriately. The result of such packet inspection is similar to that provided by a transparent application proxy, but with less overhead. It also suffers from the same problems—as protocols evolve, the inspection code must also evolve in order to take advantage of the new features in the protocol.

Both firewalls can require authentication against Terminal Access Controller Access Control System (TACACS)+ or Remote Authentication Dial-In User Service (RADIUS) before passing packets from or to specified hosts and/or services. The IOS firewall can also use Kerberos for authentication [Cisco Systems, Inc. 2001].

The PIX is especially notable because it was the first commercially available implementation of NAT [Cisco Systems, Inc. 2000]. It also regenerates TCP sequence numbers, using strong random numbers. This is important because machines which have weak TCP sequence number generation algorithms are vulnerable to session hijacking and spoofed one-way connections.

The IOS firewall protects against pre-identified malicious Java applets, presumably by matching against a static signature file. The IOS firewall defends and protects against SYN floods and other closely related TCP attacks. As explained in [Cisco Systems, Inc. 2001], SYN-floods are detected by

> comparing the rate of requests for new connections and the number of half-open connections to a configurable threshold level to detect SYN flooding. When the router detects unusually high rates of new connections, it issues an alert message and takes a preconfigured action

The preconfigured action either instructs the protected host to drop half-open connections (presumably by sending it a TCP reset packet), or temporarily filters all TCP SYN packets destined for the protected host. Unfortunately, this action disables all inbound connections to the protected host. The IOS firewall also monitors TCP sequence numbers and drops all packets with suspicious numbers.

9.3 Lucent's firewall family

Lucent's firewall family [Lucent Technologies 2001] is based on packet filtering with state. Similar to Cisco, packets are inspected, and the packet filter examines at the higher-level data (including some protocols at the application layer). Unlike the Cisco products, the firewall can be a bridging firewall.

9.4 Sun's Sunscreen Secure Net

Sun's Sunscreen Secure Net 3.1 [Sun Microsystems 2000] is based on packet filtering with state and a collection of application proxies. Its HTTP proxy can be configured to pass or drop Java applets. It can also filter:

—Java applets based on the cryptographic signature included in the applet,

—Java applets based on a hash of the applet,

—Cookie requests, and

—ActiveX content.

The FTP proxy is a true application proxy and therefore can limit access to certain file transfer commands such as `put` or `get` based on source and destination addresses and/or user authentication.

Sunscreen Secure Net can also be configured as a bridging firewall.

10. FIREWALL POLICY SPECIFICATION

Firewalls were originally built and configured by experts. Now, firewalls are commodity products which are sold with the intent that nearly anyone can be responsible for their network's security. The network administrator uses a graphical user interface (GUI) to configure packet filtering rules. Unfortunately, this GUI still requires the user to understand the complexities of packet filters. These complexities were originally pointed out in [1992] by Chapman. In many cases, the only difference between then and now is that the user interacts with the packet filter rules through a GUI. The common use of transparent proxies only increases the complexity of the administrator's task.

Some researchers have tried to improve the way that firewalls are managed. Guttman [1997] described a LISP-like language for expressing access control policies for networks where more than one firewall router is used to enforce the policy. The language is then used to compute a set of packet filters which will properly implement the policy. He also presents an algorithm for comparing previously generated filters to the policy to identify any policy breaches. However, the automatically generated filters are not expressed in the language of any router; the network administrator must build them manually from the LISP-like output.

In [2000], Hazelhurst et al. describe binary decision diagrams (BDDs) for visualizing router rule sets. BDDs have the benefit that they can represent boolean expressions—this makes them ideal for representing the block/pass rules which occur in packet filters. Once a set of packet filter rules has been converted to BDDs, it becomes easy to apply automated analysis. BDDs are also an efficient method for storing and using the rules; they can be used by a router to provide better performance than the simple table-based lookup which is used in many routers in 2002.

The Internet standards RFC2748 [Durham et al. 2000], RFC3060 [Moore et al. 2001] and RFC3084 [Chan et al. 2001] describe the Common Open Policy Service (COPS) protocol. This protocol is used between a policy server (Policy Decision Point or PDP) and its clients (Policy Enforcement Points or PEPs). The basic idea is that the policy is specified at a different location from the firewall (a PEP), and the policy server ensures that the various policy enforcers have and use the correct policy. The policy may relate to Quality of Service (QoS), it may relate to security, or it may relate to some other part of network policy (e.g., IPsec); the COPS protocol is extensible. The network is modeled as a finite state machine and a policy is modeled as a collection of policy rules. These rules have a logical expression of conditions and a set of actions. If the logical expression is true, then the actions are executed. These actions may cause a state change in the network finite state machine. The policy rules can be prioritized, allowing conflict resolution when two or more rules match but specify conflicting actions. As these proposed standards are adopted, they will have a significant impact on how firewalls are constructed.

Stone et al. survey policy languages through late 2000 and describe a new approach to policy specification [Stone et al. 2001]. In addition to security concerns, their approach also takes into account Quality of Service (QoS). In specifying policies, they note that some policies are static, i.e., for security reasons, all access to certain network addresses are prohibited. Other policies, however, are dynamic, i.e., if the available bandwidth is too low, streaming video is no longer allowed. Finally, different users may receive different levels of service (e.g., customers in the company web store have priority over employees browsing the web). Their policy language is called the Path-Based Policy Language (PPL), and it corrects some of the deficiencies in the other policy languages.

Damianou et al. describe a policy language, Ponder, in [2001]. Ponder is a declarative, object-oriented language, which through its structures can represent policies. Constraints on a policy can also be represented in Ponder. Although Ponder appears to be a particularly rich and expressive language for expressing policies, there is not yet an automated policy implementation path.

Bartal et al. describe *firmato* [Bartal et al. 1999]. *firmato* has an underlying entity-relationship model which specifies the global security policy, a language in which to represent the model, a compiler which translates the model into firewall rules, and a tool which displays a graphical view of the result to help the user visualize the model.

A module for use with *firmato* is the firewall analysis engine, Fang (Firewall ANalysis enGine) by Mayer et al. [2000]. Fang reads the firewall configurations and discovers what policy is described. The network administrator can then verify that the policy being implemented by the various routers matches the desired policy. For example, the network administrator can ask questions such as "From which machines can our DMZ be reached, and with which services?" Fang builds a representation of the policy; it is not an active testing program. This difference allows Fang to test both that authorized packets actually succeed as well as that unauthorized packets are blocked. It also allows testing before the firewall is deployed; by contrast, active test tools require the firewall to be up and running to test it. Also, active testing cannot test the network's vulnerability to spoofing attacks, whereas Fang can. Fang provides a GUI to collect the queries from the network administrator and to display the results.

A recent example of this family of firewall test and analysis tools is the Lumeta Firewall Analyzer (LFA) [Wool 2001]. LFA is a commercial product which goes beyond Fang by automatically generating the "interesting" queries. The network administrator needs only to provide the firewall configuration. The result is a system that hides the complexities of the underlying router configurations, providing a much more comprehensible picture of the resulting policy.

11. FIREWALL TESTING

Firewall testing was originally an ad-hoc exercise, the thoroughness being determined by the skill of the person running the tests. A second phase of testing methodology included security scanners such as the Security Administrator Tool for Analyzing Networks (SATAN) [Venema 1995] and the Internet Security Systems (ISS) Internet scanner [Internet Security Systems, Inc. 2002]. These scanners provided the basis for the National Computer Security Association (NCSA) certification [Vigna 1997] for a period of time. Vigna extended this approach by defining a formal model of a network's topology in [1997]. His model can also represent the TCP/IP protocol stack up through the transport level. Using this model, he was able to generate logical statements describing the requirements for the firewall. Given these requirements, he then generated a series of locations for probes and packets to attempt to send when testing the real firewall. From a formal standpoint, this work is promising, but it fails to address the common problem of how to develop a correct formal description. Producing complete formal descriptions for realistic networks represents a significant amount of work and is difficult to do correctly. Additionally, the test generator must have a complete list of vulnerabilities for which to generate tests.

Marcus Ranum took a different approach to firewall testing in [Ranum 1995]; he notes that firewalls are (or at least should be) different for different organizations. After a firewall is deployed, an expert can study the policy specification for the firewall and decide which tests will verify that the firewall properly implements the policy, using a top-down approach. He emphasizes the importance of testing both the security of the firewall itself (that the firewall is secure from attack) and the correctness of the policy implementation. Unfortunately, such testing is both expensive and time-consuming.

Some of the tools for firewall policy specification (Section 10) also provide testing or guidance for testing.

12. FIREWALL THEORY

Firewalls have evolved in an ad-hoc fashion, reacting to various attacks. Lyles and Schuba [1996b; 1996a] and Schuba et al. [1997] developed the first formal model for firewalls. Their model described firewall technology in terms of a policy P, communication traffic T, and a network policy domain D. This model allows firewall functions to be distributed across multiple machines. It also allows for repeated application of the theory; for example, at the link, network, and transport layers of a network. Schuba expanded the model in his Ph.D. dissertation [1997], developing a theoretical basis for firewalls based on Hierarchical Colored Petri Nets (HCPN).

A different and less detailed formal model of firewalls is presented by Vigna in [Vigna 1996; 1997]. Here, a network is modeled as a hypergraph, protocols such as IP, UDP, and TCP are modeled as tuples (possibly containing other tuples). A vulnerability is a boolean expression of quantified boolean expressions relating the tuples and hypergraphs. These vulnerabilities can be used to generate tests for firewalls.

Packet classification is an important part of filtering traffic. Gupta and McKeown wrote a review of packet classification algorithms in [2001]. They provide examples of four different types of algorithms, and discuss when the algorithms might be most useful.

13. WHAT FIREWALLS DO NOT PROTECT AGAINST

No firewall provides perfect security. Several problems exist which are not addressed by the current generation of firewalls. In the event that a firewall does try to provide protection for the problems discussed in this section, either it is not in widespread use or it has problems with the protection it provides.

13.1 Data Which Passes Through the Firewall

A firewall is probably best thought of as a permeable membrane. That is, it is only useful if it allows some traffic to pass through it (if not, then the network could be physically isolated from the outside world and the firewall not needed). Unfortunately, any traffic passing though the firewall is a potential avenue of attack. For example, most firewalls have some provision for email, but email is a common method of attack; a few of the many email attacks are described in [Cohen et al. 2001; Computer Emergency Response Team (CERT) 1999; 2000a; 2001b; 2001c; 2001d]. The serious problem of email-based attacks has resulted in demand for some part of the firewall to check email for hostile code. Products such as AMaViS [amavis.org 2002] and commercial email virus scanners (e.g., [Network Associates Technology, Inc. 2001a]) have responded to this problem. However, they are only as good as the signatures for which they scan; novel attacks pass through without a problem.

If web traffic is allowed through the firewall, then network administrators must cope with possibility of malicious web sites. With scripting languages such as Java, JavaScript, and ActiveX controls, malicious web administrators can read arbitrary files on client machines (e.g., [Hernan 2000]) and execute arbitrary code on the client (e.g., [Cohen and Hernan 2000; Computer Emergency Response Team (CERT) 1997a]). ActiveX controls are of particular concern, because they do not run in any form of "sandbox" the way Java applets do [Bellovin et al. 2000]. ActiveX controls can be digitally signed, and if properly used, can be used to authenticate the author, if not the author's intentions.

In [1997], Martin et al. describe some attacks written in Java. They advocate the draconian solution of blocking all applets, on the grounds that it cannot be determined which Java applets are dangerous. They suggest the following methods of blocking Java applets at the firewall:

(1) Using a proxy to rewrite `<applet>` tags. This requires that the proxy be smart enough to rewrite only the tags in HTML files and not if they appear in other file types, such as image files. This requires that the proxy parse the HTML documents in the same manner as the browser.

(2) Java class files always begin with four byte hex signature CAFE BABE. A firewall could block all files that begin with this sequence. A possibility of false positives exists with this scheme, but Martin et al. believe that this problem is less likely to occur than the `<applet>` tag appearing in non-HTML files.

(3) Block all files whose names end in `.class`. This solution is weak because Java classes can come in files with other extensions, for example, packing class files in a `.zip` file is common.

Their suggestion is to implement all three of these, and they write a proxy which does everything except look inside of `.zip` files.

13.2 Servers on the DMZ

Because the networks inside of a firewall are often not secure, servers which must be accessible from the Internet (e.g., web and mail servers) are often placed on a screened network, called the DMZ (for demilitarized zone; for a picture of one way a DMZ may be constructed, see Figure 1, part C). Machines on the DMZ are not allowed to make connections to machines on the inside of the firewall, but machines on the inside *are* allowed to make connections to the DMZ machines. The reason for this architecture is that if a server on the DMZ is compromised, the attacker cannot directly attack the other machines inside. Because a server must be accessible to be of use, current firewalls other than signature-based ones (Section 8.3) can do little against attacks through the services offered. Examples of attacks on servers include worms such as those described in [Danyliw and Householder 2001; Danyliw et al. 2001].

13.3 Insider Attacks

In spite of the fact that early firewalls such as the DEC SEAL were initially set up to prevent information leaks, they cannot protect against insiders intent on getting information out of an organization. Consider a hostile employee with access to a CD burner. The resulting CD will not be traveling through the firewall, so the firewall cannot prevent this data loss. In [1995], Muffett also points out that inside a firewall, security tends to decrease over time unless the internal machines are continually updated. Therefore, a hostile insider can generally penetrate other internal machines, and since these attacks do not go through the firewall, it cannot stop them. To reduce this threat, some organizations have set up internal firewalls.

14. FUTURE CHALLENGES FOR FIREWALLS

All of the topics discussed in the prior section pose serious challenges for firewalls. In addition, two emerging technologies will further complicate the job of a firewall, Virtual Private Networks (VPNs) and peer-to-peer networking.

14.1 VPNs

Because firewalls are deployed at the network perimeter, if the network perimeter is expanded the firewall must somehow protect this expanded territory. VPNs provide an example of how this can happen. A laptop being used by a traveling employee in an Internet cafe or a home machine which is connected to an ISP via a DSL line or cable modem must be inside the firewall. However, if the laptop or home machine's security is breached, the entire internal network becomes available to the attackers.

Remote access problems are first mentioned in [Avolio and Ranum 1994]. Due to the fact that VPNs had not yet been invented, it is easy to understand why Avolio and Ranum failed to discuss the problem of a remote perimeter which includes hosts always-connected to the Internet (via DSL or cable modems) and which are also allowed inside through a VPN tunnel.

14.2 Peer-to-peer Networking

The music sharing system Napster is the most famous example of peer-to-peer networking. However, several other peer-to-peer systems exist as well, including Gnutella (e.g., [Wego Systems, Inc 2001]) and AIMster (file sharing over AOL Instant Messenger). When not used for music sharing, peer-to-peer file sharing is used to support collaboration between distant colleagues. However, as Bellovin points out in [2000], these systems raise serious security concerns. These include the possibility of using Gnutella for attacks, buggy servents (server+client programs), and the problems of web and email-based content in yet another form. Current firewalls are unable to provide any protection against these types of attacks beyond simply blocking the peer-to-peer networking.

15. CONCLUSION

The need for firewalls has led to their ubiquity. Nearly every organization connected to the Internet has installed some sort of firewall. The result of this is that most organizations have some level of protection against threats from the outside. Attackers still probe for vulnerabilities that are likely to only apply to machines inside of the firewall. They also target servers, especially web servers. However, these attackers are also now targeting home users (especially those with full-time Internet connections) who are less likely to be well protected.

Because machines inside a firewall are often vulnerable to both attackers who breach the firewall as well as hostile insiders, we will likely see increased use of the distributed firewall architecture. The beginnings of a simple form of distributed firewalls are already here, with personal firewalls being installed on individual machines. However, many organizations will require that these individual firewalls respond to configuration directives from a central policy server. This architecture will simply serve as the next level in a sort of arms race, as the central server and the protocol(s) it uses become special targets for attackers.

Firewalls and the restrictions they commonly impose have affected how application-level protocols have evolved. Because traffic initiated by an internal machine is often not as tightly controlled, newer protocols typically begin with the client contacting the server; not the reverse as active FTP did. The restrictions imposed by firewalls have also affected the attacks that are developed. The rise of email-based attacks is one example of this change.

An even more interesting development is the expansion of HTTP and port 80 for new services. File sharing and remote procedure calls can now be accomplished using HTTP. This overloading of HTTP results in new security concerns, and as a result, more organizations are beginning to use a (possibly transparent) web proxy so they can control the remote services used by the protected machines. The future is likely to see more of this co-evolution between protocol developers and firewall designers until the protocol designers consider security when the protocol is first developed. Even then, firewalls will still be needed to cope with bugs in the implementations of these protocols.

REFERENCES

Abie, H. 2000. An overview of firewall technologies. *Telektronikk 96,* 3, 47–52. `http://www.nr.no/publications/FirewallTechnologies.pdf` Accessed 2002 Feb 20.

amavis.org. 2002. AMaViS—a mail virus scanner. `http://www.amavis.org/` Accessed 2002 Feb 20.

Avolio, F. 1999. Firewalls and Internet security, the second hundred (Internet) years. *The Internet Protocol Journal 2,* 2 (June), 24–32. `http://www.cisco.com/warp/public/759/ipj_2-2/ipj_2-2_fis1.html` Accessed 2002 Feb 20.

Avolio, F. M. and Ranum, M. J. 1994. A network perimeter with secure external access. In *Internet Society Symposium on Network and Distributed Systems Security, 3-4 Feb. 1994, San Diego, CA, USA*. Internet Society, Reston, VA, USA, 109–119. `http://www.ja.net/CERT/Avolio_and_Ranum/isoc94.ps` Accessed 2002 Feb 20.

Axelsson, S. 1998. Research in intrusion-detection systems: A survey. Tech. Rep. 98–17, Dept. of Computer Eng. Chalmers Univ. of Tech, SE-412 96 Göteborg, Sweden. December. `http://www.ce.chalmers.se/staff/sax/survey.ps` Accessed 2002 Feb 20.

Bailey, M. L., Gopal, B., Pagels, M. A., Peterson, L. L., and Sarkar, P. 1994. PATHFINDER: A pattern-based packet classifier. In *1st Symposium on Operating Systems Design and Implementation, 14-17 November 1994, Monterey, CA, USA*. USENIX Association, Berkeley, CA, 115–123.

Balasubramanian, S. 2000. An architecture for protection of network hosts from denial of service attacks. M.S. thesis, University of Florida. `http://etd.fcla.edu/etd/uf/2000/ana6124/Master.pdf` Accessed 2002 Feb 20.

Bartal, Y., Mayer, A. J., Nissim, K., and Wool, A. 1999. Firmato: A novel firewall management toolkit. In *1999 IEEE Symposium on Security and Privacy, 9-12 May 1999, Oakland, CA, USA*. IEEE, Los Alamitos, CA, USA, 17–31. `http://www.wisdom.weizmann.ac.il/~kobbi/papers/firmato.ps` Accessed 2002 Feb 20.

Bellovin, S. 1994. Firewall-friendly FTP. RFC 1579. `ftp://ftp.isi.edu/in-notes/rfc1579.txt` Accessed 2002 Feb 20.

Bellovin, S., Cohen, C., Havrilla, J., Herman, S., King, B., Lanza, J., Pesante, L., Pethia, R., McAllister, S., Henault, G., Goodden, R., Peterson, A. P., Finnegan, S., Katano, K., Smith, R., and Lowenthal, R. 2000. Results of the security in ActiveX workshop Pittsburgh, Pennsylvania USA August 22-23, 2000. Tech. rep., CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, Pittsburg, PA 15213, USA. December. `http://www.cert.org/reports/activeX_report.pdf` Accessed 2002 Feb 20.

Bellovin, S. M. 1992. There be dragons. In *UNIX Security Symposium III Proceedings, 14-16 Sept 1992, Baltimore, MD, USA*. USENIX Association, Berkeley, CA, 1–16. `http://www.research.att.com/~smb/papers/dragon.pdf` Accessed 2002 Feb 20.

Bellovin, S. M. 1999. Distributed firewalls. *;login: 24,* Security (November). `http://www.usenix.org/publications/login/1999-11/features/firewalls.htm%l` Accessed 2002 Feb 20.

Bellovin, S. M. 2000. Security aspects of napster and gnutella. Invited talk at the USENIX 2001 Annual Technical Conference, June 25-30, 2001. `http://www.research.att.com/~smb/talks/NapsterGnutella/index.htm` Accessed 2002 Feb 20.

Bonachea, D. and McPeak, S. 2001. SafeTP: Transparently securing FTP network services. Tech. Rep. UCB Tech Report CSD-01-1152, Computer Science Division, University of California, Berkeley, 387 Soda Hall #1776, Berkeley, CA 94720-1776. `http://www.cs.berkeley.edu/~bonachea/safetp/CSD-01-1152.pdf` Accessed 2002 Feb 20.

Braden, R., Clark, D., Crocker, S., and Huitema, C. 1994. Report of IAB workshop on security in the Internet architecture February 8-10, 1994. RFC 1636. ftp://ftp.isi.edu/in-notes/rfc1636.txt Accessed 2002 Feb 20.

Chan, K., Seligson, J., Durham, D., Gai, S., McCloghrie, K., Herzog, S., Reichmeyer, F., Yavatkar, R., and Smith, A. 2001. COPS usage for policy provisioning (COPS-PR). RFC 3084. ftp://ftp.isi.edu/in-notes/rfc3084.txt Accessed 2002 Feb 20.

Chapman, D. B. 1992. Network (in)security through IP packet filtering. In *UNIX Security Symposium III Proceedings, 14-16 Sept 1992, Baltimore, MD, USA*. USENIX Association, Berkeley, CA, 63–76. http://www.greatcircle.com/pkt_filtering.html Accessed 2002 Feb 20.

Chatel, M. 1996. Classical versus transparent IP proxies. RFC 1919. ftp://ftp.isi.edu/in-notes/rfc1919.txt Accessed 2002 Feb 20.

Cheswick, B. 1990. The design of a secure Internet gateway. In *USENIX 1990 Summer Conference*. USENIX Association, Berkeley, CA. http://www.cheswick.com/ches/papers/gateway.ps Accessed 2002 Feb 20.

Cheswick, B. 1992. An evening with Berferd in which a cracker is lured, endured, and studied. In *Winter 1992 USENIX Conference, 20-24 Jan 1992, San Francisco, CA, USA*. USENIX Association, Berkeley, CA, 163–173. http://www.cheswick.com/ches/papers/berferd.ps Accessed 2002 Feb 20.

Cheswick, B. 2001. Personal communication.

Cheswick, W. R. and Bellovin, S. M. 1994. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley, One Jacob Way, Reading, MA, 01867.

Cisco Systems, Inc. 2000. Cisco - Cisco's PIX firewall and stateful firewall security. http://www.cisco.com/warp/public/cc/pd/fw/sqfw500/tech/nat_wp.htm Accessed 2002 Feb 20.

Cisco Systems, Inc. 2001. Cisco - building a perimeter security solution. http://www.cisco.com/warp/public/cc/pd/iosw/ioft/iofwft/tech/firew_wp.h%tm Accessed 2002 Feb 20.

Cohen, C., Danyliw, R., Finlay, I., Shaffer, J., Hernan, S., Houle, K., King, B. B., and van Ittersum, S. 2001. CERT advisory CA-2001-03 VBS/OnTheFly (Anna Kournikova) malicious code. http://www.cert.org/advisories/CA-2001-03.html Accessed 2002 Feb 20.

Cohen, C. and Hernan, S. 2000. CERT advisory CA-2000-12 HHCtrl ActiveX control allows local files to be executed. http://www.cert.org/advisories/CA-2000-12.html Accessed 2002 Feb 20.

Computer Emergency Response Team (CERT). 1997a. CERT advisory CA-1996-07 weaknesses in Java bytecode verifier. http://www.cert.org/advisories/CA-1996-07.html Accessed 2002 Feb 20.

Computer Emergency Response Team (CERT). 1997b. CERT advisory CA-1996-26 denial-of-service attack via ping. http://www.cert.org/advisories/CA-1996-26.html Accessed 2002 Feb 20.

Computer Emergency Response Team (CERT). 1998. CERT advisory CA-1997-28 IP denial-of-service attacks. http://www.cert.org/advisories/CA-1997-28.html Accessed 2002 Feb 20.

Computer Emergency Response Team (CERT). 1999. CERT advisory CA-1999-04 Melissa macro virus. http://www.cert.org/advisories/CA-1999-04.html Accessed 2002 Feb 20.

COMPUTER EMERGENCY RESPONSE TEAM (CERT). 2000a. CERT advisory CA-2000-04 love letter worm. http://www.cert.org/advisories/CA-2000-04.html Accessed 2002 Feb 20.

COMPUTER EMERGENCY RESPONSE TEAM (CERT). 2000b. CERT incident note IN-2000-02 : Exploitation of unprotected windows networking shares. http://www.cert.org/incident_notes/IN-2000-02.html Accessed 2002 Feb 20.

COMPUTER EMERGENCY RESPONSE TEAM (CERT). 2000c. CERT incident note IN-2000-03 : 911 worm. http://www.cert.org/incident_notes/IN-2000-03.html Accessed 2002 Feb 20.

COMPUTER EMERGENCY RESPONSE TEAM (CERT). 2001a. CERT incident note IN-2001-01 : Widespread compromises via "ramen" toolkit. http://www.cert.org/incident_notes/IN-2001-01.html Accessed 2002 Feb 20.

COMPUTER EMERGENCY RESPONSE TEAM (CERT). 2001b. Vulnerability note VU#131569: Microsoft Outlook view control allows execution of arbitrary code and manipulation of user data. http://www.kb.cert.org/vuls/id/131569 Accessed 2002 Feb 20.

COMPUTER EMERGENCY RESPONSE TEAM (CERT). 2001c. Vulnerability note VU#149424 Outlook Web Access (OWA) executes scripts contained in email attachment opened via Microsoft Internet Explorer (IE). http://www.kb.cert.org/vuls/id/149424 Accessed 2002 Feb 20.

COMPUTER EMERGENCY RESPONSE TEAM (CERT). 2001d. Vulnerability note VU#5648: Buffer overflows in various email clients. http://www.kb.cert.org/vuls/id/5648 Accessed 2002 Feb 20.

COMPUTER SECURITY INSTITUTE. 2001. CSI firewall product search center. http://www.spirit.com/CSI/firewalls.html Accessed 2002 Feb 20.

CORBRIDGE, B., HENIG, R., AND SLATER, C. 1991. Packet filtering in an IP router. In *Proceedings of the USENIX Fifth System Administration Conference (LISA V)*. USENIX Association, Berkeley, CA, 227–232. http://www.netsys.com/library/papers/packet_filtering_in_an_ip_router.p%df Accessed 2002 Feb 20.

DAEMON9 AND ALHAMBRA. 1996. Project Loki: ICMP tunneling. *Phrack 7,* 49 (August). http://www.phrack.org/show.php?p=49&a=6 Accessed 2002 Feb 20.

DAMIANOU, N., DULAY, N., LAPU, E., AND SLOMAN, M. 2001. The ponder policy specification language. In *Policies for Distributed Systems and Networks. International Workshop, POLICY 2001. Proceedings, 29-31 Jan. 2001, Bristol, UK*. Springer-Verlag, Berlin, Germany. http://www.doc.ic.ac.uk/~mss/Papers/Ponder-Policy01V5.pdf Accessed 2002 Feb 20.

DANYLIW, R., DOUGHERTY, C., HOUSEHOLDER, A., AND RUEFLE, R. 2001. CERT advisory CA-2001-26 Nimda worm. http://www.cert.org/advisories/CA-2001-26.html Accessed 2002 Feb 20.

DANYLIW, R. AND HOUSEHOLDER, A. 2001. CERT advisory CA-2001-19 "code red" worm exploiting buffer overflow in IIS indexing service DLL. http://www.cert.org/advisories/CA-2001-19.html Accessed 2002 Feb 20.

DENNING, P. J. March-April 1989. The Internet worm. *American Scientist 77,* 2, 126–128.

DIGITAL EQUIPMENT CORPORATION. 1992. Securing external access link (SEAL) introductory guide. This document was part of the documentation provided by DEC to purchasers of SEAL.

DJAHANDARI, K. AND STERNE, D. 1997. An MBone proxy for an application gateway firewall. In

Proceedings of the 1997 Conference on Security and Privacy (S&P-97). IEEE Press, Los Alamitos, 72–81.

Doraswamy, N. and Harkins, D. 1999. *IPSec: the new security standard for the Internet, intranets, and virtual private networks*. Prentice-Hall, Upper Saddle River, NJ.

Dotti, P. and Rees, O. 1999a. Protecting the hosted application server. In *Proceedings of the IEEE 8th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. IEEE Computer Society, Los Alamitos, CA, USA.

Dotti, P. and Rees, O. 1999b. Protecting the hosted application server. Tech. Rep. HPL-1999-54 990413, Hewlett-Packard Labs., Bristol, UK. `http://www.hpl.hp.com/techreports/1999/HPL-1999-54.pdf` Accessed 2002 Feb 20.

Durham, D., Boyle, J., Cohen, R., Herzog, S., Rajan, R., and Sastry, A. 2000. The COPS (Common Open Policy Service) protocol. RFC 2748. `ftp://ftp.isi.edu/in-notes/rfc2748.txt` Accessed 2002 Feb 20.

Egevang, K. and Francis, P. 1994. The IP network address translator (NAT). RFC 1631. `ftp://ftp.isi.edu/in-notes/rfc1631.txt` Accessed 2002 Feb 20.

Eichin, M. W. and Rochlis, J. A. 1989. With microscope and tweezers: An analysis of the Internet virus of November 1988. In *1989 IEEE Computer Society Symposium on Security and Privacy*. IEEE Computer Society, Los Alamitos, CA, USA, 326–343.

Farrow, R. 1998. CSI 1997 firewalls matrix [an analysis of current firewall technologies]. `http://www.spirit.com/CSI/Papers/farrowpa.htm` Accessed 2002 Feb 20.

Finlayson, R. 1999. IP multicast and firewalls. RFC 2588. `ftp://ftp.isi.edu/in-notes/rfc2588.txt` Accessed 2002 Feb 20.

Fung, K. P. and Chang, R. K. C. 2000. A transport-level proxy for secure multimedia streams. *IEEE Internet Computing 4,* 6 (November–December), 57–67.

Gangadharan, M. and Hwang, K. 2001. Intranet security with micro-firewalls and mobile agents for proactive intrusion response. In *Proceedings of the International Conference on Computer Networks and Mobile Computing, 2001*. IEEE Computer Society, Los Alamitos, CA, USA, 325–332.

Ganger, G. R. and Nagle, D. F. 2000. Enabling dynamic security management of networked systems via device-embedded security. Tech. Rep. CMU-CS-00-174, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213-3890. December. `http://reports-archive.adm.cs.cmu.edu/anon/2000/CMU-CS-00-174.pdf` Accessed 2002 Feb 20.

Gleitz, P. M. and Bellovin, S. M. 2001. Transient addressing for related processes: Improved firewalling by using IPV6 and multiple addresses per host. In *Proceedings of the 10th USENIX Security Symposium*. USENIX Association, Berkleley, CA, USA, 99–113. `http://www.usenix.org/publications/library/proceedings/sec01/full_paper%s/gleitz/gleitz.pdf` Accessed 2002 Feb 20.

Gupta, P. and McKeown, N. 2001. Algorithms for packet classification. *IEEE Network 15,* 2 (March-April), 24–32.

Guttman, J. D. 1997. Filtering postures: local enforcement for global policies. In *1997 IEEE Symposium on Security and Privacy, 4-7 May 1997, Oakland, CA, USA*. IEEE Computer Society Press, Los Alamitos, CA, USA, 120–129. `http://www.mitre.org/support/papers/filtering_postures/filtering_postures.pdf` Accessed 2002 Feb 20.

HAIN, T. 2000. Architectural implications of NAT. RFC 2993. ftp://ftp.isi.edu/in-notes/rfc2993.txt Accessed 2002 Feb 20.

HAMBRIDGE, S. AND SEDAYAO, J. C. 1993. Horses and barn doors: Evolution of corporate guidelines for Internet usage. In *Proceedings of the USENIX Seventh System Administration Conference (LISA '93)*. USENIX Association, Berkeley, CA. http://www.usenix.org/publications/library/proceedings/lisa93/full_pape%rs/hambridge.ps Accessed 2002 Feb 20.

HANDLEY, M., PAXSON, V., AND KREIBICH, C. 2001. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *Conference Proceedings: 10th USENIX Security Symposium*. USENIX Association, Berkleley, CA, USA, 115–131. http://www.aciri.org/vern/papers/norm-usenix-sec-01.pdf Accessed 2002 Feb 20.

HAZELHURST, S., ATTAR, A., AND SINNAPPAN, R. 2000. Algorithms for improving the dependability of firewall and filter rule lists. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2000)*. IEEE Computer Society, Los Alamitos, CA, USA. IEEE.

HERNAN, S. 2000. CERT advisory CA-2000-15 Netscape allows Java applets to read protected resources. http://www.cert.org/advisories/CA-2000-15.html Accessed 2002 Feb 20.

HONEYNET PROJECT. 2001. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Addison-Wesley, Boston, MA.

HWANG, K. AND GANGADHARAN, M. 2001. Micro-firewalls for dynamic network security with distributed intrusion detection. In *IEEE International Symposium on Network Computing and Applications, NCA 2001*. IEEE Computer Society, Los Alamitos, CA, USA, 68–79.

INTERNET SECURITY SYSTEMS, INC. 2002. ISS - security products - security assessment - Internet scanner. http://www.iss.net/securing_e-business/security_products/security_asses%sment/internet_scanner/ Accessed 2002 Feb 20.

IOANNIDIS, J. AND BELLOVIN, S. M. 2002. Pushback: Router-based defense against DDoS attacks. In *Proceedings of Network and Distributed System Security Symposium, Catamaran Resort Hotel San Diego, California 6-8 February 2002*. The Internet Society, 1775 Wiehle Ave., Suite 102, Reston, VA 20190. http://www.research.att.com/~smb/papers/pushback-impl.pdf Accessed 2002 Feb 20.

IOANNIDIS, S., KEROMYTIS, A. D., BELLOVIN, S. M., AND SMITH, J. M. 2000. Implementing a distributed firewall. In *ACM Conference on Computer and Communications Security*. Association for Computing Machinery, One Astor Plaza, 1515 Broadway, New York, New York 10036-5701, USA, 190–199. http://www.cis.upenn.edu/~angelos/Papers/df.ps.gz Accessed 2002 Feb 20.

JULKUNEN, H. AND CHOW, C. 1997. Enhance network security with dynamic packet filter. Tech. Rep. EAS-CS-97-2, University of Colorado at Colorado Springs Department of Computer Science, 1420 Austin Bluffs Parkway, P.O. Box 7150, Colorado Springs, CO 80933-7150. April. http://www.cs.auc.dk/~fleury/papers/firewall/julkunen.pdf Accessed 2002 Feb 20.

JULKUNEN, H. AND CHOW, C. 1998. Enhance network security with dynamic packet filter. In *7th International Conference on Computer Communications and Networks, 12-15 Oct. 1998, Lafayette, LA, USA*, K. Makki, I. Chalamrac, and N. Pissinou, Eds. IEEE, Piscataway, NJ, USA, 268–275.

KAHN, A., AL-DARWISH, N., GUIZANI, M., MENTEN, M., AND YOUSSEF, H. 1997. Design and implementation of a software bridge with packet filtering and statistics collection functions.

International Journal of Network Management 7, 5 (September-October), 251–263. `http://www1.acm.org/pubs/articles/journals/ijnm/1997-7-5/p251-khan/p251%-khan.pdf` Accessed 2002 Feb 20.

Keromytis, A. D. and Wright, J. L. 2000. Transparent network security policy enforcement. In *FREENIX Track. 2000 USENIX Annual Technical Conference, 18-23 June 2000, San Diego, CA, USA*. USENIX Association, Berkeley, CA, USA, 215–225. `http://www.cis.upenn.edu/~angelos/Papers/bridgepaper.ps.gz` Accessed 2002 Feb 20.

Koblas, D. and Koblas, M. R. 1992. SOCKS. In *UNIX Security Symposium III Proceedings, 14-16 Sept. 1992, Baltimore, MD, USA*. USENIX Association, Berkeley, CA, 77–83.

Larsen, J. 2001. HogWash. `http://hogwash.sourceforge.net/` Accessed 2002 Feb 20.

Leech, M. 1996. Username/password authentication for SOCKS V5. RFC 1929. `ftp://ftp.isi.edu/in-notes/rfc1929.txt` Accessed 2002 Feb 20.

Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and Jones, L. 1996. SOCKS protocol version 5. RFC 1928. `ftp://ftp.isi.edu/in-notes/rfc1928.txt` Accessed 2002 Feb 20.

Lightoler, T. 1764. *The gentleman and farmer's architect. A new work. Containing a great variety of ... designs. Being correct plans and elevations of parsonage and farm houses, lodges for parks, pinery, peach, hot and green houses, with the fire-wall, tan-pit, &c particularly described ...* R. Sayer, London, UK.

Limoncelli, T. 1999. Tricks you can do if your firewall is a bridge. In *1st Conference on Network Administration, 7-10 April 1999, Santa Clara, CA, USA*. USENIX Association, Berkeley, CA, USA, 47–55. `http://www.bell-labs.com/user/tal/papers/` Accessed 2002 Feb 20.

Liu, J. and Ma, Y. 1999. Packet filtering in bridge. In *1999 Internet Workshop (WS'99), 18-20 Feb. 1999, Osaka, Japan*. IEEE, Piscataway, NJ, USA, 94–98.

Lodin, S. W. and Schuba, C. L. 1998. Firewalls fend off invasions from the net. *IEEE Spectrum 35,* 2 (February), 26–34.

Lucent Technologies. 2001. Lucent - VPN firewall family. `http://www.lucent.com/products/solution/0,,CTID+2012-STID+10080-SOID+12%01-LOCL+1,00.html` Accessed 2002 Feb 20.

Lyles, J. and Schuba, C. 1996a. A reference model for firewall technology and its implications for connection signaling. Tech. Rep. CSD-TR-96-073, Department of Computer Sciences, Purdue University. December.

Lyles, J. B. and Schuba, C. L. 1996b. A reference model for firewall technology and its implications for connection signaling. Tech. Rep. CSD-TR-94-061, Reprinted as Department of Computer Sciences, Purdue University, Purdue University, 1398 Computer Science Building, West Lafayette, IN 47907-1398. Proceedings Open Signaling Workshop, Columbia University, New York, NY, October 1996. `https://www.cerias.purdue.edu/techreports-ssl/public/csd_94-061.pdf` Accessed 2002 Feb 20.

Lyu, M. R. and Lau, L. K. 2000. Firewall security: Policies, testing and performance evaluation. In *Proceedings of The Twenty-Fourth Annual International Computer Software and Applications Conference*. IEEE Computer Society, Los Alamitos, CA, USA.

Malan, G. R., Watson, D., Jahanian, F., and Howell, P. 2000. Transport and application protocol scrubbing. In *IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies,* *26-30 March 2000, Tel Aviv, Israel*. IEEE Computer Society; IEEE Communications Society, Piscataway, NJ, USA, 1381–1390.

Maltz, D. and Bhagwat, P. 1998. TCP splicing for application layer proxy performance. Tech. Rep. RC 21139, IBM. March. `http://domino.watson.ibm.com/library/cyberdig.nsf/a3807c5b4823c53f85256%561006324be/88d1e552b09ffa65852565e6006616f1?OpenDocument` Accessed 2002 Feb 20.

Mariet, P. 1997. AltaVista firewall. In *DECUS Switzerland conference*. Decus Switzerland, Sonnenbergstrasse 11 CH-8610 Uster. `http://elias.decus.ch/presentations/ge_19970415_av/index.htm` Accessed 2002 Feb 20.

Markham, T. and Payne, C. 2001. Security at the network edge: a distributed firewall architecture. In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX '01. Proceedings*. Vol. 1. IEEE Computer Society, Los Alamitos, CA, USA, 279–286.

Markus, H. S. 2001. Firewall guide software reviews. `http://www.firewallguide.com/software.htm` Accessed 2002 Feb 20.

Martin, D.M., J., Rajagopalan, S., and Rubin, A. 1997. Blocking Java applets at the firewall. In *SNDSS '97: Internet Society 1997 Symposium on Network and Distributed System Security, 10-11 Feb. 1997, San Diego, CA, USA*. IEEE Computer Society Press, Los Alamitos, CA, USA.

Mayer, A., Wool, A., and Ziskind, E. 2000. Fang: A firewall analysis engine. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy (S&P 2000)*. IEEE Computer Society, Los Alamitos, CA, USA.

McKay, N. 1998. China: The great firewall. Web publication: `http://www.wired.com/news/politics/0,1283,16545,00.html` Accessed 2002 Feb 20.

McMahon, P. 1996. GSS-API authentication method for SOCKS version 5. RFC 1961. `ftp://ftp.isi.edu/in-notes/rfc1961.txt` Accessed 2002 Feb 20.

Mogul, J. 1991. Using *screend* to implement IP/TCP security policies. Tech. Rep. NSL Technical Note TN-2, Digital Network Systems Laboratory, Palo Alto, CA. July. `http://research.compaq.com/nsl/publications/postscript/nsltn2.pdf` Accessed 2002 Feb 20.

Mogul, J. C. 1989. Simple and flexible datagram access controls for Unix-based gateways. In *Proceedings of the USENIX Summer 1989 Conference*. USENIX Association, Berkeley, CA, 203–222. `ftp://ftp.digital.com/pub/Digital/WRL/research-reports/WRL-TR-89.4.ps.g%z` Accessed 2002 Feb 20.

Molitor, A. 1995. An architecture for advanced packet filtering. In *Proceedings of the Fifth USENIX UNIX Security Symposium*. USENIX Association, Berkeley, CA, USA, 117–126. `http://www.usenix.org/publications/library/proceedings/security95/full_%papers/molitor.ps` Accessed 2002 Feb 20.

Moore, B., Ellesson, E., Strassner, J., and Westerinen, A. 2001. Policy core information model—version 1 specification. RFC 3060. `ftp://ftp.isi.edu/in-notes/rfc3060.txt` Accessed 2002 Feb 20.

Muffett, A. 1994. Proper care and feeding of firewalls. In *Proceedings of the UKERNA Computer Security Workshop*. United Kingdom Education and Research Networking Association, Atlas Centre, Chilton, Didcot, Oxfordshire, OX11 0QS UK. `ftp://coast.cs.purdue.edu/pub/doc/firewalls/Muffett_Alec_feeding_firewa%lls.ps.Z` Accessed 2002 Feb 20.

Muffett, A. 1995. Wan-hacking with *AutoHack* - auditing security *behind* the firewall. In *The Fifth USENIX UNIX Security Symposium.* USENIX Association, Berkeley, CA, 21–34.

Network Associates Technology, Inc. 2001a. E-mail protection. `http://www.mcafeeb2b.com/products/email-protection.asp` Accessed 2002 Feb 20.

Network Associates Technology, Inc. 2001b. Gauntlet firewall administrator's guide version 6.0. `http://download.nai.com/products/media/pgp/support/gnt/admin_GNT60.pdf` Accessed 2002 Feb 20.

Network Associates Technology, Inc. 2001c. Gauntlet firewall for UNIX advanced administrator's guide version 6.0. `http://download.nai.com/products/media/pgp/support/gnt/adv_admin_60.pdf` Accessed 2002 Feb 20.

Network Associates Technology, Inc. 2001d. Gauntlet firewall services guide version 6.0. `http://download.nai.com/products/media/pgp/support/gnt/services_60.pdf` Accessed 2002 Feb 20.

Northcutt, S. and Novak, J. 2001. *Network Intrusion Detection: An Analyst's Handbook, Second Edition.* New Riders Publishing, 201 West 103rd St, Indianapolis, IN, 46290, USA.

Oria, L. 1999. Approaches to multicast over firewalls: an analysis. Tech. Rep. HPL-IRI-1999-004 990827, Hewlett-Packard Laboratories. August. `http://www.hpl.hp.com/techreports/1999/HPL-IRI-1999-004.html` Accessed 2002 Feb 20.

Postel, J. and Reynolds, J. 1985. File transfer protocol (FTP). RFC 959. `ftp://ftp.isi.edu/in-notes/rfc959.txt` Accessed 2002 Feb 20.

Ranum, M. J. 1992. A network firewall. In *Proceedings of the First World Conference on System Administration and Security.* SANS Institute, 5401 Westbard Ave. Suite 1501, Bethesda, MD 20816.

Ranum, M. J. 1995. On the topic of firewall testing. `http://web.ranum.com/pubs/fwtest/index.htm` Accessed 2002 Feb 20.

Ranum, M. J. 2001. personal communication.

Ranum, M. J. and Avolio, F. M. 1994. A toolkit and methods for Internet firewalls. In *Conference Proceedings: USENIX Summer 1994 Technical Conference.* USENIX Association, Berkeley, CA, 37–44.

Reed, D. 19?? Filter language compiler. Undated web page. `http://coombs.anu.edu.au/ipfilter/flc.html` Accessed 2002 Feb 20.

Reed, D. 2002a. IP filter. Undated web page. `http://coombs.anu.edu.au/~avalon/ip-filter.html` Accessed 2002 May 15.

Reed, D. 2002b. IP filter. HISTORY file from the IP Filter distribution. `http://coombs.anu.edu.au/~avalon/ip-fil3.4.27.tar.gz` Accessed 2002 May 15.

Reed, D. 2002c. What's new for IP filter. Undated web page. `http://coombs.anu.edu.au/~avalon/ipfil-new.html` Accessed 2002 May 15.

Rekhter, Y., Moskowitz, B., Karrenberg, D., deGroot, G. J., and Lear, E. 1996. Address allocation for private internets. RFC 1918. `ftp://ftp.isi.edu/in-notes/rfc1918.txt` Accessed 2002 Feb 20.

Rodriguez, P., Sibal, S., and Spatscheck, O. 2001. TPOT: translucent proxying of TCP. *Computer Communications 24,* 2, 249–255. `http://www.terena.nl/conf/wcw/Proceedings/S7/S7-3.pdf` Accessed 2002 Feb 20.

Roesch, M. 1999. Snort - lightweight intrusion detection for networks. In *LISA'99: 13th System Administration Conference and Exhibition, 7-12 Nov. 1999, Seattle, WA, USA*. USENIX Association, Berkleley, CA, USA, 229–238. http://www.usenix.org/events/lisa99/full_papers/roesch/roesch.pdf Accessed 2002 Feb 20.

Schimmel, J. 1997. A historical look at firewall technologies. *;login: 22,* 7. http://www.usenix.org/sage/best.of/breakins/firewall.html Accessed 2002 Feb 20.

Schneier, B. 2000. *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons, New York, NY, 188–193.

Schuba, C., Lyles, B., and Spafford, E. 1997. A reference model for firewall technology. In *13th Annual Computer Security Applications Conference (ACSAC), 8-12 Dec. 1997, San Diego, CA, USA*. IEEE Computer Society, Los Alamitos, CA, USA, 133–145.

Schuba, C. L. 1997. On the modeling, design, and implementation of firewall technology. Ph.D. thesis, Purdue University. ftp://ftp.cerias.purdue.edu/pub/papers/christoph_schuba/schuba_phddis.p%df Accessed 2002 Feb 20.

Schuba, C. L., Krsul, I. V., Kuhn, M. G., Spafford, E. H., Sundaram, A., and Zamboni, D. 1997. Analysis of a denial of service attack on TCP. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*. IEEE Computer Society, IEEE Computer Society Press, Los Alamitos, CA, USA, 208–223. https://www.cerias.purdue.edu/techreports-ssl/public/97-06.ps Accessed 2002 Feb 20.

Securityfocus.com. 1997. Multiple vendor "out of band" data (winnuke.c) DoS vulnerability. Vulnerability database. http://www.securityfocus.com/bid/2010 Accessed 2002 Feb 20.

Smith, J. M., Doherty, S. G., Leahy, O. J., and Tynan, D. M. 1997. Protecting a private network: The AltaVista firewall. *Digital Technical Journal 9,* 2 (November), 17–32. http://www.research.compaq.com/wrl/DECarchives/DTJ/DTJQ00/index.html Accessed 2002 Feb 20.

Spafford, E. H. 1988. The Internet worm program: An analysis. Tech. Rep. Purdue Technical Report CSD-TR-823, Department of Computer Science, Purdue University, West Lafayette, IN 47907-2004. ftp://ftp.cs.purdue.edu/pub/reports/TR823.PS.Z Accessed 2002 Feb 20.

Spafford, E. H. 1991. The Internet worm incident. Tech. Rep. Purdue Technical Report CSD-TR-933, Department of Computer Science, Purdue University, West Lafayette, IN 47907-2004.

Spatscheck, O., Hansen, J. S., Hartman, J. H., and Peterson, L. L. 2000. Optimizing TCP forwarder performance. *IEEE/ACM Transactions on Networking 8,* 2, 146–157. http://www.cs.arizona.edu/scout/Papers/TR98-01.ps Accessed 2002 Feb 20.

Srisuresh, P. and Egevang, K. 2001. Traditional IP network address translator (traditional NAT). RFC 3022. ftp://ftp.isi.edu/in-notes/rfc3022.txt Accessed 2002 Feb 20.

Srisuresh, P. and Holdrege, M. 1999. IP network address translator (NAT) terminology and considerations. RFC 2663. ftp://ftp.isi.edu/in-notes/rfc2663.txt Accessed 2002 Feb 20.

Stoll, C. 1989. *The Cuckoo's Egg*. Doubleday, New York, NY.

Stoll, C. May, 1988. Stalking the wily hacker. *Commun. ACM 31,* 5, 484–497.

Stone, G. B., Lundy, B., and Xie, G. G. 2001. Network policy languages: A survey and a new approach. *IEEE Network 15,* 1 (January-February), 10–21.

Strother, E. 2000. Denial of service protection: the nozzle. In *Annual Computer Security Applications Conference, 11-15 Dec. 2000, New Orleans, LA, USA*. IEEE Computer Society, Los Alamitos, CA, USA, 32–41. `http://www.acsac.org/2000/papers/41.pdf` Accessed 2002 Feb 20.

Sun Microsystems, I. 2000. Sunscreen 3.1 reference manual. `http://www.sun.com/software/securenet/ds/` Accessed 2002 Feb 20.

Suri, S. and Varghese, G. 1999. Packet filtering in high speed networks. In *Tenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99)*. SIAM, 3600 University City Science Center, Philadelphia, PA 19104-2688, 969–970. `http://siesta.cs.wustl.edu/~suri/psdir/soda_filter.ps` Accessed 2002 Feb 20.

The RFC Editor. 2001. Request for comments (RFC) frequently asked questions. `http://www.rfc-editor.org/rfcfaq.html` Accessed 2002 Feb 20.

Treese, G. W. and Wolman, A. 1993. X through the firewall and other application relays. In *Proceedings of the USENIX Summer Conference*. USENIX Association, Berkeley, CA. `ftp://crl.dec.com/pub/DEC/CRL/tech-reports/93.10.ps.Z` Accessed 2002 Feb 20.

van Rooij, G. 2001. Real stateful TCP packet filtering in IP-filter. Invited talk at the 10th USENIX Security Symposium August 13-17, 2001 Washington DC. Talk slides available at: `http://www.usenix.org/events/sec01/invitedtalks/rooij.pdf`.

Venema, W. 1995. Project SATAN: UNIX/Internet security. In *COMPSEC International 95. Twelfth World Conference on Computer Security, Audit and Control, 25-27 Oct. 1995, London, UK*. Elsevier, Oxford, UK.

Vigna, G. 1996. A topological characterization of TCP/IP security. Tech. Rep. TR-96.156, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20133 Milano, Italy. December. `http://www2.elet.polimi.it/pub/data/Giovanni.Vigna/www_docs/pub/tr96156%.ps.gz` Accessed 2002 Feb 20.

Vigna, G. 1997. A formal model for firewall testing. Unpublished paper. `http://citeseer.nj.nec.com/279361.html` Accessed 2002 Feb 20.

Watson, D., Smart, M., Malan, G., and Jahanian, F. 2001. Protocol scrubbing: network security through transparent flow modification. In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX '01. Proceedings*. Vol. 2. IEEE Computer Society, Los Alamitos, CA, USA, 108–118.

Weber, W. 1999. Firewall basics. In *4th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services. TELSIKS'99. Proceedings of Papers, 13-15 Oct. 1999, Nis, Yugoslavia*. Vol. 1. IEEE, Piscataway, NJ, USA, 300–305.

Wego Systems, Inc. 2001. Welcome to gnutella. `http://gnutella.wego.com/` Accessed 2002 Feb 20.

Wool, A. 2001. Architecting the Lumeta firewall analyzer. In *Conference Proceedings: 10th USENIX Security Symposium*. USENIX Association, Berkleley, CA, USA, 85–97. `http://www.usenix.org/events/sec01/full_papers/wool/wool.pdf` Accessed 2002 Feb 20.

Yavwa, Y. 2000. The firewall technology. Student paper, available at: `http://www.cs.uct.ac.za/courses/CS400W/NIS/resources.html` Accessed 2002 Feb 20.

Zwicky, E. D., Cooper, S., and Chapman, D. B. 2000. *Building Internet Firewalls, 2nd Edition*. O'Reilly & Associates, 101 Morris St, Sebastopol, CA 95472 USA.

Received Month Year; revised Month Year; accepted Month Year