MINI PROJECT TITLE

PROJECT REPORT

18CSE383T – INFORMATION ASSURANCE AND SECURITY

(2018 Regulation)

III Year/ V Semester

Academic Year: 2022 -2023

By

Utkarsh Ajmani(RA2011030010130)

Under the guidance of

Dr. SAVARIDASSAN.P

Assistant Professor

Department of Networking and Communications

COLLEGE OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Kancheepuram

NOVEMBER 2022

BONAFIDE

This is to certify that **18CSE383T – INFORMATION ASSURANCE AND SECURITY mini project report** titled "**Network Firewalls"** is the bonafide work of **Utkarsh Ajmani(RA2011030010130)** who undertook the task of completing the project within the allotted time.

Signature of the Faculty

Dr. Savaridassan.P

Assistant Professor

Department of NWC

SRM Institute of Science and Technology

Signature of the II Year Academic Advisor

Dr. Annapurani Panaiyappan .K

Professor and Head

Department of NWC

SRM Institute of Science and Technology

Contents

Abstract

Firewalls are network devices that 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, network, and data-link levels. Newer methods, which have not yet been widely adopted, include protocol normalization and distributed firewalls.

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, testing is required to determine if the firewall correctly implements the policy.

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 existing firewall technology.

1 Introduction

The idea of a wall to keep out intruders dates back thousands of years. Over two thousand years ago, the Chinese built the Great Wall as protection from neighboring northern tribes. European kings 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 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 [40]. These physical barriers prevented or slowed a fire's spread throughout a building, saving both lives and property. A related use of the term is described by Schneier [60]:

> 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.

This chapter is 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 [2]. These historical examples illustrate 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.

Firewalls have existed since about 1987, and several surveys and histories have been written (e.g., [34, 52, 11, 42]). In this chapter, we give an updated and more comprehensive survey of firewall technology. Several books have been written which describe how to build a firewall (e.g., [15, 71]). 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.

For the purposes of this chapter, 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 (this is often called filtering). 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.

The fact that a firewall is at the boundary between two networks has also led to firewalls being called "perimeter security"—see, for example, Figure 1.

Firewalls function by filtering traffic at one or more (today, normally multiple) layers in the network protocol stack. These layers are described using the ISO seven-layer model for networking [36]:

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) [67].

This chapter is divided into several sections. Section 2 describes the history and rationale for organizations adopting firewalls. Security professionals build firewalls using many different architectures, depending on the security needs of the organization, and Section 3 describes several of these choices. Section 4 reviews the ISO protocol layers, describing firewall technology at each relevant layer. Section 5 considers alternative approaches to firewall construction; these approaches are typically more experimental, but they represent technology that could appear in common firewalls in the near future. Once a firewall is constructed, it must be tested to show that it actually enforces the organization's security policy; testing is the subject of Section 6. Although firewalls are an important tool for securing an organization's network, they have limitations which are discussed in Section 7. Section 8 discusses some projected challenges for firewalls in the face of technicological change, and Section 9 concludes the chapter.

1 The Need for Firewalls

In the early years, the Internet supported a small community of compatible users who valued openness for sharing and collaboration. This view was challenged by the Morris Worm [22]. 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 [63], and Bill Cheswick's "Evening with Berferd" [13] 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 [7]. The result was clear—there were many untrustworthy and even malicious users on the Internet.

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 both the software and the users on its computers are not malicious. Firewalls enforce trust boundaries, which are imposed for several 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 enabled by default; many viruses exploited this vulnerability (e.g., [16, 17]). A second example is Red Hat Linux versions 6.2 and 7.0, which were vulnerable to three remote exploits when the operating was installed using default options [18]. It is an on-going and expensive process to secure every user's machine, and many organizations

consciously decide not to secure the machines inside their firewall. If a machine on the inside is ever compromised, the remaining machines are likely also vulnerable [53], a situation that has been described as "a sort of crunchy shell around a soft, chewy center" [14].

Individuals can protect a single machine connected to the Internet with 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.

Organizations often use firewalls to prevent a compromised machine inside from attacking machines outside. In this case, the firewall protects the organization from possible liability due to propagating an attack.

Preventing access to information: National firewalls (attempt to) limit the activities of their users on the Internet, for example China [49]. A similar idea in the US is the Children's Internet Protection Act (CHIPA) which mandates that certain information be filtered. This law requires that schools and libraries which receive federal funding block certain classes of 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 [55]:

> 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 rules about which network traffic is allowed to enter or leave a network. These policies restrict the use of certain applications, restrict which remote machines may be contacted, and/or limit the bandwidth.

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 monitor employees, e.g., for using work network resources for non-work purposes.

2 Firewall architectures

Firewalls range from simple machines designed to be purchased "off-the-shelf" and installed by a person unskilled in network security (e.g., as shown in Figure 1) to complex, multiple-machine custom installations used in large organizations. Regardless of their complexity, all firewalls have the concept of "inside" for the protected network, and "outside" for the untrusted network. These terms are used even when a firewall protects the outside world from potentially compromised machines inside.

Another common feature of firewalls is the existence of a DMZ (named for the demilitarized zone separating North and South Korea) or "screened network." Examples of how a DMZ may be constructed are illustrated in Figures 2 and 3. Machines such as email and web servers are often placed on the DMZ. These machines 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. Thus if a server on the DMZ is compromised, the attacker cannot directly attack machines on the inside. Servers are particularly vulnerable because they must be accessed in order to be useful, and current firewalls are largely ineffective against attacks through these services (see Section 5.4). can do little against Examples of attacks on servers include the "Code Red" and "Nimda" worms which attacked Microsoft Windows machines running Microsoft's web server IIS, and in the case of Nimda, several additional routes.

Firewall architectures are constrained by the type of filtering (described shortly) and the presence or absence of a DMZ.

2.1 Packet filtering

Packet filtering is looking at the headers in network packets and deciding whether or not to allow the packet based on the policy enforced by the firewall. Packet filtering for network security began with Mogul's paper describing *screend* in 1989 [50]. Most early work on packet filtering for security emphasized performance (e.g., [4]); later papers continued this trend (e.g., [43, 66]). In addition to its efficiency, packet filtering is appealing because it does not require the cooperation of users, nor does it require any special action on their part like some proxies require (See Section 3.2).

Packet filters use one or more of the following pieces of information to make their decision on whether or not to forward the packet: 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; and whether the packet is inbound or outbound.

Although packet filtering is fast, it has some drawbacks, most importantly the difficulty of writing correct filters. For example, Chapman compares packet filter languages to assembly language [12]. In 1995, Molitor proposed an improved commercial filter language [51].

A second drawback is that packet filtering cannot identify which user is causing which network traffic. It can inspect the IP address of the host from which the traffic originates, but a host is not identical to 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 (see Section 3.2 for one example of how this might be done) 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 still use IP addresses or DNS names for access control.

With packet filters, the local machine directly initiates the connection to the remote machine. A result 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 can potentially exploit weaknesses in the protocol implementation of the local computer (e.g., [61]).

Protocols such as FTP are difficult 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, contrary to the communication patters in other client-server protocols. FTP's lack of encryption protecting user authentication data has led to reduced usage, and eventually it may no longer be used.

2.1.1 Packet Filtering with State

Originally, packet filters ignored 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. Attacks against bugs in the TCP/IP protocol stack (e.g., [61]) can pass through the packet filtering firewalls which do not keep state by claiming to be part of an established TCP session. Some network mapping software (e.g., [24]) can map the inside network as if the firewall did not even exist.

The solution to this problem is to record the state of a connection, a property referred to variously as stateful firewalls, adaptive firewalling and packet inspection. In other words, the packet filter records both the network level and the transport level data. For example, a router can monitor the initial TCP packets with the SYN flag set and allow the return packets only until the FIN packet is sent and acknowledged. 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 stateless packet filtering firewalls [12]. The first peer-reviewed paper to describe adding state to packet filters was by Julkunen and Chow in 1998, which describes a dynamic packet filter for Linux [37]. Today, all major packet filtering firewalls are capable of using connection state.

2.1.2 Improving Packet Filter Specification

Firewalls were originally built and configured by experts. However, firewalls are now commodity products which are sold with the intent that nearly anyone can be responsible for their network's security. Typically a graphical user interface (GUI) is used to configure packet filtering rules. Unfortunately, this GUI requires the user to understand the complexities of packet filters, complexities originally pointed out by Chapman in 1992 [12]. In many cases, the only advance since then is the GUI. The prevalence of transparent proxies only increases the complexity of the administrator's task because he or she must understand the advantages and drawbacks of using proxies compared to packet filtering.

Some researchers have therefore developed higher-level languages for specifying packet filters. Specific examples include using binary decision diagrams (BDDs) to specify the policy, a compiler for a higher-level language that produces packet-filtering rules, a LISP-like language describing policy, and the Common Open Policy Service (COPS) protocol standard.

In 2000, Hazelhurst proposed BDDs for visualizing router rule sets [31]. 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* [58], 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*).

Guttman described a LISP-like language for expressing access control policies for networks where more than one firewall router is used to enforce the policy [28]. The language is then used to compute a set of packet filters which will properly implement the policy. He also describes an algorithm for comparing existing 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.

The Internet standards RFC2748, RFC3060, and RFC3084 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 likely have a significant impact on how firewalls are constructed.

Stone et al. survey policy languages through 2000 and describe a new approach to policy specification [64]. 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 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 called Ponder [19]. Ponder is a declarative, object-oriented language, which uses its structures to represent policies. Constraints on a policy can also be represented in Ponder. Although Ponder appears to be a rich and expressive language for expressing policies, there is not yet an automated policy implementation path.

Bartal et al. describe *firmato* [5]. *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. [48]. Fang reads the firewall configurations and discovers what policy is described. The network administrator can then verify that the actual policy on 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 the case in which authorized packets succeed and the one in which 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 queries and to display the results.

A recent example of this family of firewall test and analysis tools is the Lumeta Firewall Analyzer (LFA) [70]. LFA is a commercial product which extends Fang to synthesize its own "interesting" queries based only on the firewall configuration. The result is a system that hides the complexities of underlying router configurations, providing a much more comprehensible picture of

the resulting policy.

Other tools for analyzing packet filter rules and highlighting problems (in some cases with proposed solutions) include those by: Hari et al. [30], Al-Shaer and Hamed [1].

2.2 Proxies

A proxy is a program that receives traffic destined for another computer. Proxies sometimes require user authentication; they can verify that the user is allowed to connect to the destination, and then connect to the destination service on behalf of the user. One example of a firewall architecture which makes use of a proxy server is shown in Figure 4.

When a proxy is used, the connection to the remote machine comes from the machine running the proxy instead of the original machine making the request. Because the proxy generates the connection to the remote machine, it has no problems determining which connections are real and which are spoofed; this is in contrast to stateless packet filtering firewalls (described in Section 3.1).

Proxies appear in firewalls primarily at the Transport and Application ISO network levels. In the Internet, the transport level consists of only two protocols, TCP and UDP. This small number of protocols makes writing a proxy easy—one proxy suffices for all protocols that use TCP. Contrast this with the application-level proxies (covered below), where a separate proxy is required for each service, e.g., Telnet, FTP, HTTP, SMTP, etc.

Transport-level proxies have 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 3.1 have this problem). Because the state of the TCP connection is known by the firewall, only packets that are a legitimate part of a communication are allowed inside the firewall.

Proxies at the application level provide the benefits of transport-level proxies, and additionally they can enforce the proper application-level protocol and prevent the abuses of the protocol by

either client or server. The result is excellent security and auditing. Unfortunately, application proxies are not without their drawbacks:

- The proxy must be designed for a specific protocol. New protocols are developed frequently, requiring new proxies; if there is no proxy, there is no access.
- To use an application proxy, the client program must be changed to accommodate the proxy. The client needs to understand the proxy's authentication method and it must communicate the actual packet destination to the proxy. Because source code is not publicly available for some applications, in these cases the required changes can be made only by the application's vendor, a significant bottleneck.
- Each packet requires two trips through the complete network protocol stack which adversely affects performance. This is in contrast to packet filtering, which handles packets at the network layer.

One of the most common proxies is SOCKS, by Kolbas and Kolbas [38]. SOCKS simplifies the changes needed to the source code of the client application—A SOCKS call replaces a normal socket call, which results in all outbound traffic using the proxy. This approach is a clean solution, and it works well if one has the source code for the relevant operating system utilities. Some commercial applications (e.g., Netscape) were written to accommodate SOCKS. A system using SOCKS and TCP connections is transparent to the user (assuming the proxy allows access to the destination host). In 2000, Fung and Chang described an enhancement to SOCKS for UDP streams, such as that used by RealNetworks' RealPlayer [23].

Ranum and Avolio developed the Trusted Information Systems (TIS) Firewall Toolkit (FWTK), a collection of proxies for building firewalls [3, 57]. 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 is exposed; this way if a proxy were compromised, the rest of the firewall would more likely remain trustworthy. The TIS FWTK had no proxies for UDP services; instead, the firewall machine ran DNS and the Network Time Protocol (NTP). The internal 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.

A limitation of proxies is that client software must be modified and/or the user must work differently when using the proxy. Transparent proxies address this limitation. With a transparent proxy the client sends packets to the destination as usual. 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 compares classical proxies with transparent proxies.

Transparent proxies are demanding because the firewall must operate both at the network and application levels, affecting performance. One solution proposed by Spatscheck et al. [62] and Maltz and Bhagwat [45] 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.

3 Firewalls at various ISO network levels

3.1 Physical layer

The physical layer of the network is usually covered by an organization's physical security—conventional locks, keys, and other forms of physical access control. Untrusted persons

must not have access the physical cables and other network hardware which make up the network.

Wireless communication, especially radio, introduces new complications. For example, radio waves travel through most walls easily. This feature necessitates the use of encryption. Wired Equivalent Privacy (WEP) was the first attempt at providing security on wireless links. However, Borisov et. al [10] discovered a weakness in key management, with the result that after an attacker had received a sufficient number of packets, she could see all traffic and inject fake packets. The new standard is Wi-Fi Protected Access (WPA), which, as of this writing, has no known flaws.

3.2 Data link layer

At the data link layer, two types of firewall technology are used. Filtering based on the media access control (MAC) layer address (in most cases, the MAC address is the 48-bit Ethernet address) determines which machines are allowed to communicate with whom. Bridging firewalls are more traditional firewalls, but with the advantage that they can be placed anywhere in a network.

3.2.1 Filtering on MAC address

The MAC address of a machine uniquely identifies it on the local network. Some switches and firewalls are able to use this address to decide what communication to allow. This form of filtering has three limitations:

1. The MAC address is not routed; therefore any filtering must occur at or before the first router.
2. Some Ethernet cards can have a MAC address programmed into them via software running on the machine. Therefore, the MAC address must be verified at the connection to the network for it to provide security.
3. A machine is not a person; determining who is actually operating the machine is not possible with the MAC address.

3.2.2 Bridging firewalls

A bridge is a network device which works at the ISO data-link layer. Operating at this level, it does not need access to routing information. A bridging firewall uses the information listed in Section 3.1 to decide whether or not to block a packet. As a result, a bridging firewall can examine 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—it is transparent at the network level. It can be used to protect a single machine or a small collection of machines that would not normally warrant the 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 [41]), so users are minimally disrupted when the bridge is installed.

3.3 Network

At the network level, addresses indicate routing information, and hosts can be grouped together into networks. These differences from the data link layer provide important filtering options. An additional firewall feature at this level is Network Address Translation (NAT), in which an address on one side of the router is changed to a different one on the other side. In addition, multicast protocols—sending packets to a collection of hosts—operate at this level. Multicast presents a new set of problems: 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.

3.4 Network- and host-based filtering

Sometimes, all machines attached to a network can be assigned a similar trust level; for example, consider a DMZ network as in Figure 2 or 3. In this case, packet filtering rules can be developed which enforce the trust (or lack thereof). Two problems must be addressed:

1. IP version 4 does not contain authentication (unless IPsec is in use, which is rare for non-Virtual Private Network (VPN) communication), and it is not required in IP version 6. Many programs exist which can "spoof" another host—they put packets on the network claiming to have originated with the spoofed host. Any IP-based authentication faces the problem of not knowing that the correct host generated the packets. Blocking spoofed packets generated on a **remote** network is easy with packet filters—add a rule that says any packet with a source address cannot arrive on a network interface attached to any other network. However, preventing one machine on the **local** network from impersonating another is more difficult; a firewall which is not on the offending machine cannot help.

2. IP has a feature known as "source routing," where the source indicates the routing the packet should take (instead of allowing the routing algorithms on the intervening routers to determine the route). Return packets take the reverse route to return. The specified source route may be bogus, or it may be valid and allow a spoofed IP address to communicate with a remote machine. The result is that most firewalls block all source routed packets.

3.4.1 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 the recipients, and recipients who do not know in advance who might be sending data to them. This difference 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 [20]. 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 that 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 subsection.

RFC 2588 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 Oria describes in further detail how they can be implemented [54]. 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.

3.4.2 Network Address Translation

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. 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 form of protection similar 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.[1]).

RFC 2663 notes some limitations of NAT. For example, NAT may prevent IPsec from working correctly. One feature of IPsec is the ability to ensure that a packet is not modified in transit.

[1]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.

However, one of the purposes of NAT is to modify packets—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 private IP addresses, which are not globally unique. Therefore, DNS inside the network is not meaningful outside.

3.5 Transport

When they can be used, transport-level proxies (from Section 3.2) work well. Since a transport-level proxy initiates the connection, it cannot be spoofed by a packet claiming to be part of an established communication. A problem analogous to the authentication problem of the data link and network layers exists here: one cannot guarantee that the expected application is running on its "well known" port. The solution to this problem lies in using an application-level proxy.

Note that packet filtering is faster than using proxies, so performance considerations may dictate which to use.

3.6 Presentation

Little exists in the Internet at the Presentation layer, and even less exists in terms of firewalls. The Common Object Request Broker (CORBA) allows applications written in one language to make requests of objects possibly written in different languages or running on a different machine. CORBAgate by Dotti and Rees [21] 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.

3.7 Application

If the performance needs can be met, application-level proxies offer the best security. They can:

- Avoid being fooled into accepting spoofed packets.
- Ensure that both sides follow the expected application-level protocol.
- Limit the communication to an approved subset of the application-level protocol.
- Authenticate users and limit their communication according to their authorization.
- Monitor traffic for known problems, such as worms in email or hostile web server attacks against vulnerable clients.

With the advent of transparent proxies, network administrators can achieve most of these benefits without the awareness or cooperation of the users. The primary drawbacks were described above in Section 3.2—performance concerns and each protocol requires a separate proxy, and the development of proxies lags the development of new protocols.

4 Other approaches

Although filtering and proxies are the most common approaches to firewalls, they are not the only ones. Researchers have experimented with dynamic and/or distributed firewalls. Because attackers abuse protocol specifications, protocol normalization can also be beneficial. Since some communication is known to be hazardous, signature-based firewalls might help improve security against already-known attacks. Transient addressing provides the security benefits of network address translation to a single machine. This section will discuss all of these approaches in more depth.

4.1 Distributed Firewalls

There are several limitations to the firewall technology that we have presented so far. One common assumption is that all the hosts inside a firewall are trustworthy. This assumption is not always valid—for example, see Section 8.1. A related problem is that firewalls are unaware of internal traffic which violates the security policy. Because firewalls are typically centralized in one location, they can become performance bottlenecks and provide a single point of failure. A further limitation of conventional firewalls arises because in some cases the local machines know context that is 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 [8], is a distributed firewall. This was implemented by Ioannidis et al. in 2000 [35], and by Markham and Payne in 2001 [46]. 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 examples of a DMZ or screened network, see Figures 3 or 2).

Hwang and Gangadharan [33, 25] propose using firewalls on all devices attached to the protected network, where the firewalls 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 [35].

Distributed firewalls have a different set of problems from 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. 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. Markham and Payne propose implementing the distributed firewall on a programmable network interface card (NIC) to reduce reliance on the operating system for protection [46].

Around the same time that Bellovin proposed the distributed firewall, Ganger and Nagle also proposed a distributed approach to security [26], in which each device is 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, including NICs, storage devices, display devices, 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.

4.2 Dynamic firewalls

Dynamic firewalls change their rules depending on the traffic passing through them. The simplest approach is to just block traffic deemed as bad. However, this approach leaves one open to attacks where an attacker spoofs an attack from an important site (e.g., google), causing the important site to get blocked. Better systems do more than just block, e.g. throttle traffic. e.g. [32]. Others that can be dynamic include OpenBSD's *pf*, Linux *iptables*, and some commercial products.

4.3 Normalization

Attackers often abuse protocol specifications, e.g., by sending overlapping IP fragments or out-of-order TCP byte sequences. Handley et al. stressed that a firewall is a good location for enforcing tight interpretation of a protocol [29]. 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. Malan et al. describe "transport scrubbing" [44], and more recently the idea is elaborated in [69]. At about the same time, Strother [65] 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.

4.4 Signature-based Firewalls

Malan et al. discuss "application scrubbing" [44]. In this approach, a user-level program is established as a transparent proxy (see Section 3.2) 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" [69].

Snort [59] is a common intrusion detection system. *Hogwash* [39] is a firewall that blocks packets matching the *snort* rules. It runs on a bridging firewall (Section 4.2.2) and the authors claim it can handle network speeds of up to 100Mbps on hardware which is not state-of-the-art.

Commercial products such as web and email anti-virus and anti-spam often make use of signatures. The advantage is high accuracy on known attacks. The disadvantage is that they do not prevent attacks which are not in their database of signatures.

4.5 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. 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 [27]. 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.

5 Firewall Testing

Since no two organizations communications needs and patterns are identical, few if any will have identical firewalls. This leads to the problem of determining whether or not the firewall is correctly enforcing the policy. 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) and the Internet Security Systems (ISS) Internet scanner. These scanners provided the basis for the National Computer Security Association (NCSA) certification [68] for a period of time. Vigna extended this approach by defining a formal model of a network's topology [68]. 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 [56]; 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 3.1.2 on page 12) also provide testing or guidance for testing.

6 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.

6.1 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 include the “I Love You” letter, the “Sobig” worm, VBS/OnTheFly (Anna Kournikova) worm, etc. The serious problem of email-based attacks has resulted in demand for some part of the firewall to check email for hostile code. Open source products such as AMaViS and commercial email virus scanners 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. Additionally, SPAM is turning into a denial-of-service attack due to the volume, causing anti-spam products to be merged into anti-virus email checking systems.

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., when a bug in Netscape allowed Java applets to read protected resources), and execute arbitrary code on the client (e.g., when an ActiveX Control allowed local files to be executed or when a weakness in the Java bytecode verifier allowed applets to do whatever they wanted). ActiveX controls are of particular concern, because they do not run in any form of “sandbox” the way Java applets do [6]. 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 [47]. 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.

6.2 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 see Figures 3 or 2). 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 5.4) can do little against attacks through the services offered. Examples of attacks on servers include worms such as “Code Red” and “Nimda”.

6.3 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 DVD burner. The resulting DVD will not be traveling through the firewall, so the firewall cannot prevent this data loss. Muffett also points out that inside a firewall, security tends to decrease over time unless the internal machines are

continually updated [53]. 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.

7 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.

7.1 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 [3]. 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.

7.2 Peer-to-peer Networking

The music sharing system Napster was the most famous example of peer-to-peer networking. However, several other peer-to-peer systems exist as well, including Gnutella 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 [9], 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.

7.3 HTTP as a "universal transport protocol"

The development of firewalls and the filtering that usually occurs at an organization's perimeter has affected the design of new protocols. Many new protocols are developed on top of HTTP, since it is often allowed through firewalls. In some cases, this piggy backing is a reasonable use of HTTP. In other cases, such as the Simple Object Access Protocol (SOAP), HTTP is used as a remote procedure call protocol. A good proxy is required to determine what HTTP is allowed with whom.

8 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. These attacks are two-fold: 1) to take advantage of the lower security awareness of the home user, and 2) to get through a VPN connection to the inside of an organization.

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 an 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.

Result: We have implemented network firewall successfully

List of Figures

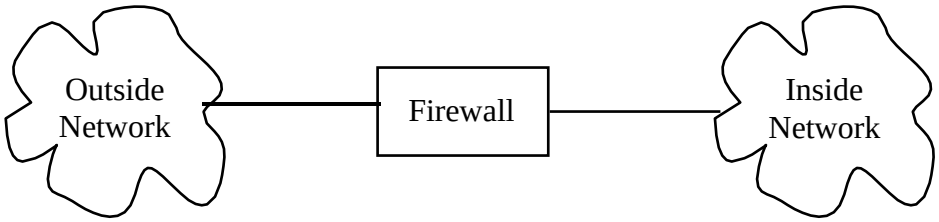

Figure 1: A firewall at the perimeter of an organization's network. The inside network may be as simple as a few machines, or may consist of several divisions located in geographically distant locations connected by telecommunication lines.

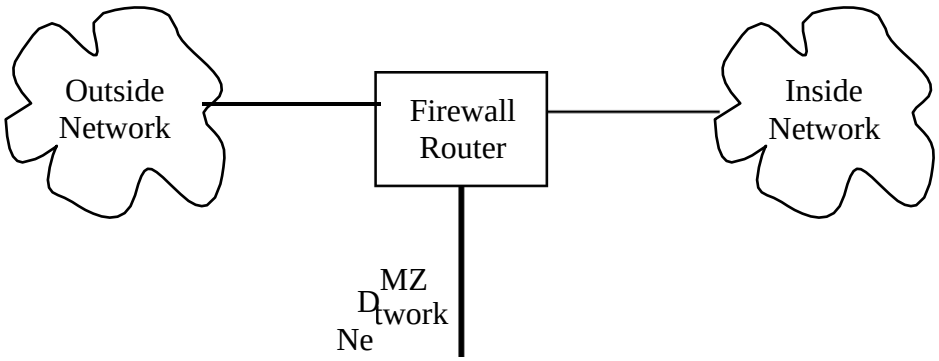

Figure 2: A firewall with a DMZ on a third network attached to the firewall router. Some commercial products are configured this way, as well as custom firewalls.

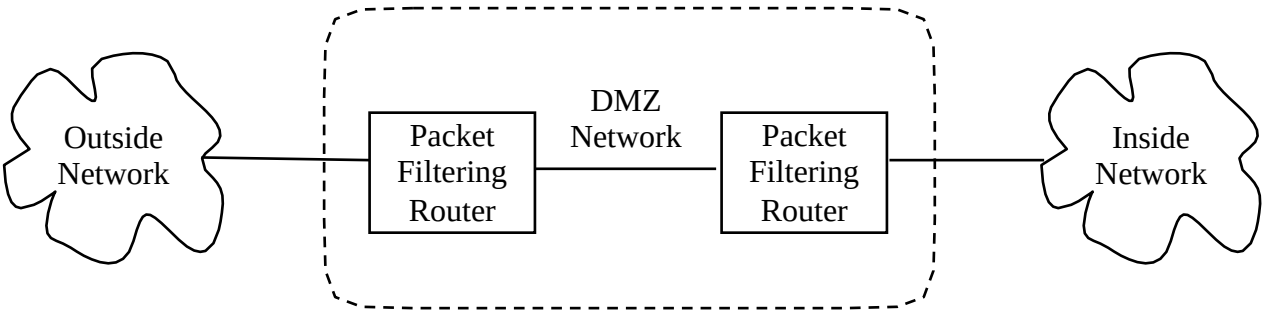

Figure 3: A screened network as a DMZ. The “firewall” is enclosed by the dashed line.

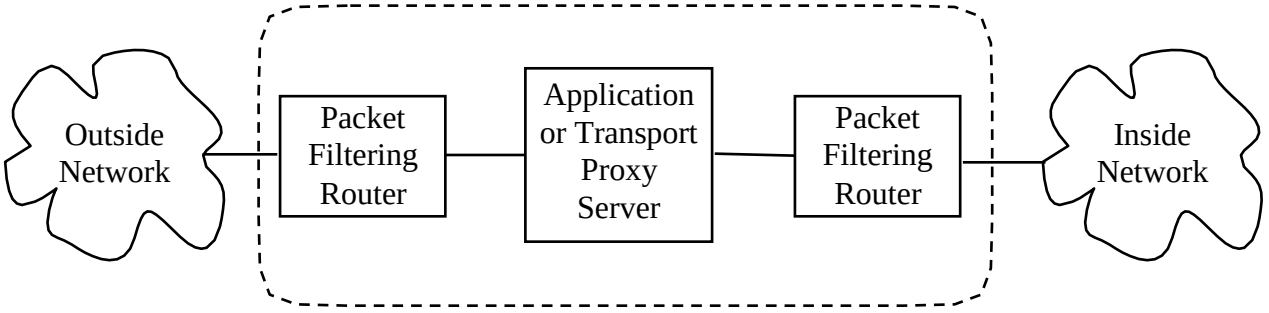

Figure 4: A network using a proxy server. Some commercial products combine all of the machinesshown in the dashed lines into one to reduce the cost.