Distributed Firewalls

Steven M. Bellovin
smb@research.att.com

Abstract

Conventional firewalls rely on the notions of restricted topology and controlled entry points to function. More precisely, they rely on the assumption that everyone on one side of the entry point--the firewall--is to be trusted, and that anyone on the other side is, at least potentially, an enemy. The vastly expanded Internet connectivity in recent years has called that assumption into question. We propose a "distributed firewall", using IPSEC, a policy language, and system management tools. A distributed firewall preserves central control of access policy, while reducing or eliminating any dependency on topology.

This paper appeared in the November 1999 issue of ;login:, pp. 37-39.

---

1 Introduction

Conventional firewalls [CB94] rely on the notions of restricted topology and control entry points to function. More precisely, they rely on the assumption that everyone on one side of the entry point--the firewall--is to be trusted, and that anyone on the other side is, at least potentially, an enemy. The vastly expanded Internet connectivity in recent years has called that assumption into question. So-called "extranets" can allow outsiders to reach the "inside" of the firewall; on the other hand, telecommuters' machines that use the Internet for connectivity need protection when encrypted tunnels are not in place.

Other trends are also threatening firewalls. For example, some machines need more access to the outside than do others. Conventional firewalls can do this, but only with difficulty, especially as internal IP addresses change. End-to-end encryption is another threat, since the firewall generally does not have the necessary keys to peek through the encryption.

Some people have suggested that the proper approach is to discard the concept of firewalls. Firewalls, they feel, are obsolete, or are not needed if cryptography is used. We disagree.

Firewalls are still a powerful protective mechanism. Most security problems are due to buggy code--in 1998, 9 of 13 CERT advisories concerned buffer overflows, and two of the rest were cryptographic bugs--and cannot be prevented by encryption or authentication. A firewall shields most such applications from hostile connections.

Firewalls are also useful at protecting legacy applications. While applications that require strong authentication should provide their own, there are too many older protocols and implementations that do not. Saying that strong cryptography should be used is true but irrelevant; in the context of such applications, it is simply unavailable.

More subtly, firewalls are a mechanism for policy control. That is, they permit a site's administrator to set a policy on external access. Just as file permissions enforce an internal security policy, a firewall can enforce an external security policy.

To solve these problems while still retaining the advantages of firewalls, we propose a distributed solution. In such a scheme, policy is still centrally defined; enforcement, however, takes place on each endpoint. We thus retain the advantages of firewalls while avoiding most of the problems we have described, most notably the dependency on topology.

Distributed firewalls rest on three notions: (a) a policy language that states what sort of connections are permitted or prohibited, (b) any of a number of system management tools, such as Microsoft's SMS or ASD [Koe84], and (c) IPSEC [KA98], the network-level encryption mechanism for TCP/IP.

The basic idea is simple. A compiler translates the policy language into some internal format. The system management software distributes this policy file to all hosts that are protected by the firewall. And incoming packets are accepted or rejected by each "inside" host, according to both the policy and the cryptographically-verified identity of each sender.

```
inside_net = x509{name="*.example.com"};
mail_gw = x509{name="mailgw.example.com"};
time_server = IPv4{10.1.2.3};

allow smtp(*, mail_gw);
allow smtp(mail_gw, inside_net);
allow ntp(time_server,inside_net);
allow *(inside_net, *);
```

Figure 1 A sample policy configuration file. SMTP from the outside can only reach the machine with a certificate identifying it as the mail gateway; it, in turn, can speak SMTP to all inside machines. NTP--a low-risk protocol that has its own application- level protection--can be distributed from a given IP address to all inside machines. Finally, all outgoing calls are permitted.

2 Policies and Identifiers

Many possible policy languages can be used, including file-oriented schemes similar to Firmato [BMNW99], the GUIs that are found on most modern commercial firewalls, and general policy languages such as KeyNote [BFK99, BFIK99]. The exact nature is not crucial, though clearly the language must be powerful enough to express the desired policy. A sample is shown in Figure 1.

What is important is how the inside hosts are identified. Today's firewalls rely on topology; thus, network interfaces are designated "inside", "outside", "DMZ", etc. We abandon this notion (but see Section 5), since distributed firewalls are independent of topology.

A second common host designator is IP address. That is, a specified IP address may be fully trusted, able to receive incoming mail from the Internet, etc. Distributed firewalls can use IP addresses for host identification, though with a reduced level of security.

Our preferred identifier is the name in the cryptographic certificate used with IPSEC. Certificates can be a very reliable unique identifier. They are independent of topology; furthermore, ownership of a certificate is not easily spoofed. If a machine is granted certain privileges based on its certificate, those privileges can apply regardless of where the machine is located physically.

In a different sense, policies can be "pulled" dynamically by the end system. For example, a license server or a security clearance server can be asked if a certain communication should be permitted. A conventional firewall could do the same, but it lacks important knowledge about the context of the request. End systems may know things like which files are involved, and what their security levels might be. Such information could be carried over a network protocol, but only by adding complexity.

3 Distributed Firewalls

In a typical organizational environment, individuals are not necessarily the administrators of the computers they use. Instead, to simplify system administration and to permit some level of central control, a system management package is used to administer individual machines. Patches can be installed, new software distributed, etc. We use the same mechanisms, which are likely present in any event, to control a distributed firewall.

Policy is enforced by each individual host that participates in a distributed firewall. The security administrator--who is no longer necessarily the "local" administrator, since we are no longer constrained by by topology--defines the security policy in terms of host identifiers. The resulting policy (probably, though not necessarily, compiled to some convenient internal format) is then shipped out, much like any other change. This policy file is consulted before processing incoming or outgoing messages, to verify their compliance. It is most natural to think of this happening at the network or transport layers, but policies and enforcement can equally well apply to the application layer. For example, some sites might wish to force local Web browsers to disable Java or Javascript.

Policy enforcement is especially useful if the peer host is identified by a certificate. If so, the local host has a much stronger assurance of its identity than in a traditional firewall. In the latter case, all hosts on the inside are in some sense equal. If any such machines are subverted, they can launch attacks on hosts that they would not normally talk to, possibly by impersonating trusted hosts for protocols such as rlogin. With a distributed firewall, though, such spoofing is not possible; each host’s identity is cryptographically assured.

This is most easily understood by contrasting it to traditional packet filters [Mog89]. Consider the problem of electronic mail. Because of a long-standing history of security problems in mailers, most sites with firewalls let only a few, designated hosts receive mail from the outside. They in turn will relay the mail to internal mail servers. Traditional firewalls would express this by a rule that permitted SMTP (port 25) connections to the internal mail gateways; access to other internal hosts would be blocked. On the inside of the firewall, though, access to port 25 is unrestricted.

With a distributed firewall, all machines have some rule concerning port 25. The mail gateway permits anyone to connect to that port; other internal machines, however, permit contact only from the mail gateway, as identified by its certificate. Note how much stronger this protection is: even a subverted internal host cannot exploit possible mailer bugs on the protected machines.

Distributed firewalls have other advantages as well. The most obvious is that there is no longer a single chokepoint. From both a performance and an availability standpoint, this is a major benefit. Throughput is no longer limited by the speed of the firewall; similarly, there is no longer a single point of failure that can isolate an entire network. Some sites attempt to solve these problems by using multiple firewalls; in many cases, though, that redundancy is purchased only at the expense of an elaborate (and possibly insecure) firewall-to-firewall protocol.

A second advantage is more subtle. Today’s firewalls don’t have certain knowledge of what a host intends. Instead, they have to rely on externally-visible features of assorted protocols. Thus, an incoming TCP packet is sometimes presumed legitimate if it has the "ACK" bit set, since such a packet can only be legitimate if it is part of an ongoing conversation--a conversation whose initiation was presumably allowed by the firewall. But spoofed ACK packets can be used as part of "stealth scanning". Similarly, it is hard for firewalls to treat UDP packets properly, because they cannot tell if they are replies to outbound queries, and hence legal, or if they are incoming attacks. The sending host, however, knows. Relying on the host to make the appropriate decision is therefore more secure.

This advantage is even clearer when it comes to protocols such as FTP [PR85]. By default, FTP clients use the PORT command to specify the port number used for the data channel; this port is for an incoming call that should be permitted, an operation that is generally not permitted through a firewall. Today’s firewalls--even the stateful packet filters--generally use an application-level gateway to handle such commands. With a distributed firewall, the host itself knows when it is listening for a particular data connection, and can reject random probes.

The most important advantage, though, is that distributed firewalls can protect hosts that are not within a topological boundary. Consider a telecommuter who uses the Internet both generically and to tunnel in to a corporate net. How should this machine be protected? A conventional approach can protect the machine while tunneled. But that requires that generic Internet use be tunneled into the corporate network and then back out the Internet. Apart from efficiency considerations, such use is often in violation of corporate guidelines. Furthermore, there is no protection whatsoever when the tunnel is not set up. By contrast, a distributed firewall protects the machine all of the time, regardless of whether or not a tunnel is set up. Corporate packets, authenticated by IPSEC, are granted more privileges; packets from random Internet hosts can be rejected. And no triangle routing is needed.

4 Hybrid Firewalls

4.1 Remote Nodes and IPSEC

Although a full implementation of distributed firewalls is the most secure and the most flexible, hybrid implementations can exist. That is, one can combine the techniques described here with traditional firewalls, achieving adequate functionality at lower cost, especially until IPSEC support becomes ubiquitous.

In a hybrid implementation, some hosts are behind a traditional firewall, while other hosts live on the outside. An IPSEC gateway at the central site provides connectivity to the outside machines. (Whether this gateway is inside the traditional firewall, outside it, in parallel with it, or even integrated with it is largely irrelevant to this discussion.) This configuration is common at companies with a major central site and some number of telecommuters.

As in ordinary virtual private networks (VPNs), remote hosts have full access to the inside, by virtue of the IPSEC tunnel. Traffic from inside machines to the remote nodes is similarly protected. What is distinct is that traffic from remote nodes to the rest of the Internet is governed by the central site's security policy. That is, the firewall administrator distributes a security policy to the remote nodes, as we have described. Ideally, of course, this same policy statement is used to control the traditional firewall, thus ensuring a consistent security policy.

4.2 Distributed Firewalls and Topological Knowledge

Another variety of hybrid implementation ignores IPSEC entirely. In this situation, address-dependent policy rules are distributed to, and enforced by, every individual host within a site. (Many newer systems support such functionality in the kernel.) While address-based authentication is quite weak, if a simple router prevents address-spoofing from the outside the security should comparable to that of traditional firewalls.

Here, we use system management techniques to ensure consistent policy. We also rely on topology, thus forfeiting the ability to protect remote hosts. However, we still eliminate the single chokepoint and point of failure.

A final hybrid scheme combines the two previous hybrid schemes. Again, a simple router prevents address-spoofing by outside machines that talk to inside nodes. IPSEC is used to tunnel traffic from inside notes to remote nodes. On these systems, the IPSEC module provides anti-spoofing protection. Finally, all protected machines, whether local or remote, receive and enforce an address-based firewall policy.

4.3 Application-specific Tunnels

With the proper sort of routing, triangle routing can be used for a few protocols, while distributed firewall techniques are used for most. Suppose, for example, that some protocol requires an application-level proxy for proper firewalling. Packets for that protocol can be routed through an IPSEC-protected tunnel to the inside of the firewall. From there, they can be forwarded to the Internet, but only after proper processing by the firewall.

In the extreme case, of course, this reduces to a conventional firewall. But if few enough protocols need this sort of filtering, it becomes an attractive variant.

5 Implementation Techniques

A number of different techniques can be used to to implement distributed firewalls. The strongest is to use end-to-end IPSEC. Each incoming packet can be associated with a certificate; the access granted to that packet is determined by the rights granted to that certificate. Consider a packet destined for port 25 (SMTP) on an inside machine, given the sample policy shown in Figure 1. If the certificate identifies the source as mailgw.example.com--note that this is not necessarily the same as the machine with that domain name--the packet will be accepted. If the certificate name is different (for clarity, we omit any discussion of the certificate's signature chain, though that is clearly important for real use), or if there is no IPSEC protection, the packet will be dropped as unauthorized.

We note that the necessary filtering is prescribed by the IPSEC architecture [KA98]. Specifically, the inbound Security Policy Database (SPD) is used to reject illegal input packets, while the outbound SPD can be used to control outgoing connections. An informal survey showed that most commercial IPSEC implementations either support port number-granularity security associations or will in the near future.

Application-level protection can be achieved by distributing application-specific policy files. Thus, web browsers can be told, by the central site, to reject, for example, all ActiveX controls or Java applets. Note that this is a hard problem for conventional firewalls [MRR97]; doing it on the end hosts is more secure, if the policy distribution problem can be solved.

The hardest problem is handling protocols such as FTP without touching the application. That is done most easily with per-process keying for IPSEC. For example, a policy rule for FTP would indicate that outbound connections to port 21 must be protected with IPSEC, and that all other TCP connections protected by that security association are legal. Since only that process can use that SA, and it would only have the FTP data channel open, an adequate level of protection can be achieved.

```
inside_net = x509{name="*.example.com", ver>19990315};
mail_gw = x509{name="mailgw.example.com"};
time_server = IPv4{10.1.2.3};

allow smtp(*, mail_gw);
allow smtp(mail_gw, inside_net);
allow ntp(time_server,inside_net);
allow *(inside_net, *);
```

Figure 2 A policy configuration file with version information.

KeyNote is an especially attractive choice for a policy language. Indeed, Blaze, Ioannidis, and Keromytis [BFK99, BFIK99] explicitly note its suitability for configuring packet filters. Its advantages include the integration of credentials with policy specification, and an ability to use a single mechanism to specify policy at different levels.

In some sense, distributed firewalls are similar to host-based enforcement as implemented in TCP wrappers [Ven92], a similar package for restricting access to assorted daemons [LeF92], or various commercial PC "personal firewall" packages. There is indeed some similarity. However, those schemes are still dependent on IP addresses (and hence topology), and do not include central policy definition as an integral piece.

6 Change Management

Given that access rights in a strong distributed firewall are tied to certificates, access rights can be limited by changing the set of certificates accepted. Suppose, for example, that a security flaw is discovered in some networked application. A new certificate can be distributed to hosts in the same distribution as the patch. Only hosts with newer certificates are then considered to be "inside"; if the change is not installed, the machine will have fewer privileges (Figure 2).

In some environments, it may even be possible to use certificates to help ensure user compliance. A PolicyMaker certificate [BFL96, BFS98] could contain code that checked the configuration of the user's system. If the check failed--that is, if the proper versions of the proper files were not present--the certificate would not allow itself to be used for authentication. (Clearly, without tamper-resistant hardware this is not an absolute check. But it can help protect against accidental misconfiguration and less-determined malfeasance.)

The certificate version mechanism also protects against new, insecure machines that are installed on an inside network. Until the appropriate filtering software and rulesets are installed (and possibly until the machine is otherwise made secure by the organization's administrators), no certificate is issued. The machine is thus an outside machine, regardless of its physical location.

7 Threat Comparison

Distributed firewalls have both strengths and weaknesses when compared to conventional firewalls. By far the biggest difference, of course, is their reliance on topology. If your topology does not permit reliance on traditional firewall techniques, there is little choice. A more interesting question is how the two types compare in a closed, single-entry network. That is, if either will work, is there a reason to choose one over the other?

7.1 Service Exposure and Port Scanning

Both types of firewalls are excellent at rejecting connection requests for inappropriate services. Conventional firewalls drop the requests at the border; distributed firewalls do so at the host. A more interesting question is what is noticed by the host attempting to connect. Today, such packets are typically discarded, with no notification. A distributed firewall may choose to discard the packet, under the assumption that its legal peers know to use IPSEC; alternatively, it may instead send back a response requesting that the connection be authenticated, which in turn gives notice of the existence of the host.

Firewalls built on pure packet filters cannot reject some "stealth scans" very well. One technique, for example, uses fragmented packets that can pass through unexamined because the port numbers aren't present in the first fragment. A distributed firewall will reassemble the packet and then reject it.

On balance, against this sort of threat the two firewall types are at least comparable.

7.2 Application-level Proxies

Some services require an application-level proxy. Conventional firewalls often have an edge here; the filtering code is complex and not generally available on host platforms. As noted, a hybrid technique can often be used to overcome this disadvantage.

In some cases, of course, application-level controls can avoid the problem entirely. If the security administrator can configure all Web browsers to reject ActiveX, there is no need to filter incoming HTML via a proxy.

In other cases, a suitably sophisticated IPSEC implementation will suffice. For example, there may be no need to use a proxy that scans outbound FTP control messages for PORT commands, if the kernel will permit an application that has opened an outbound connection to receive inbound connections. This is more or less what such a proxy would do.

7.3 Denial of Service Attacks

It is impossible to lump all denial of service attacks into one basket. However, some statements can be made about particular known attacks.

The "smurf" attack (CERT Advisory CA-98.01, 5 January 1998) primarily consumes the bandwidth on the access line from an ISP to the target site. Neither form of firewall offers an effective defense. If one is willing to change the topology, both can be moderately effective. Conventional firewalls can be located at the ISP’s POP, thus blocking the attack before it reaches the low-bandwidth access line. Distributed firewalls permit hosts to be connected via many different access lines, thus finessing the problem. For now, a distributed intrusion detection systems would be useful.

It may be possible to chew up CPU time by bombarding the IKE process with bogus security association negotiation requests. While this can affect conventional firewalls, inside machines would still be able to communicate. Distributed firewalls rely much more on IKE, and hence are more susceptible. It is an open question if a different key exchange protocol will be needed to resist such attacks.

Conversely, any attack that consumes resources on conventional firewalls, such as many email attachments that must be scanned for viruses, can bog down such firewalls and affect all users. For that matter, too much legitimate traffic can overload a firewall. As noted, distributed firewalls do not suffer from this effect.

7.4 Intrusion Detection

Many firewalls detect attempted intrusions. If that functionality is to be provided by a distributed firewall, each individual host has to notice probes and forward them to some central location for processing and correlation.

The former problem is not hard; many hosts already log such attempts. One can make a good case that such detection should be done in any event. Collection is more problematic, especially at times of poor connectivity to the central site. There is also the risk of co-ordinated attacks in effect causing a denial of service attack against the central machine.

Our tentative conclusion is that intrusion detection is somewhat harder than with conventional firewalls. While more information can be gathered, using the same techniques on hosts protected by conventional firewalls would gather the same sort of data.

7.5 Insider Attacks

At first glance, the biggest weakness of distributed firewalls is their greater susceptibility to lack of cooperation by users. What happens if someone changes the policy files on their own?

Although there are technical measures that can be taken, as discussed earlier, these are limited in their ability to cope with serious misbehavior. That said, we assert that this problem is not a real differentiator. Even conventional firewalls are easily subverted by an uncooperative insider. SSH [Ylo96] can be used to tunnel TCP ports, external Web proxies such as www.anonymizer.com can bypass destination restrictions, end-to-end encryption can hide traffic, etc. In other words, an insider who wishes to violate firewall policy can do so, with either type of firewall. As Marcus Ranum put it, "You can't solve social problems with software."

On the other hand, distributed firewalls can reduce the threat of actual attacks by insiders, simply by making it easier to set up smaller groups of users. Thus, one can restrict access to a file server to only those users who need it, rather than letting anyone inside the company pound on it.

It is also worth expending some effort to prevent casual subversion of policies. If policies are stored in a simple ASCII file, a user wishing to, for example, play a game could easily turn off protection. Requiring the would-be uncooperative user to go to more trouble is probably worthwhile, even if the mechanism is theoretically insufficient. For example, policies could be digitally signed, and verified by a frequently-changing key in an awkward-to-replace location. For more stringent protections, the policy enforcement can be incorporated into a tamper-resistant network card.

A more serious issue concerns inadvertent errors when using application-level policies. An administrator might distribute configuration restrictions for Netscape Navigator or Microsoft's Internet Explorer. But what would happen if a user, in all innocence, were to install the Opera browser? The only real solution here is a standardized way to express application-level policies across all applications of a given type. We do not see that happening any time soon.

8 Conclusions

Firewalls are sometimes described as incompatible with the modern network environment. Furthermore, there are conflicts between a strong security technology--end-to-end IPSEC--and firewalls, since the firewall can no longer examine the traffic. We have shown that the two are actually synergistic: IPSEC can be used to implement stronger firewalls, while eliminating many of the limitations of today's firewalls. We retain protection and centralized policy control; we eliminate dependency on topology, IP addresses, and the conflict with IPSEC.

9 Acknowledgments

Angelos Keromytis, Matt Blaze, and John Ioannidis made a number of useful suggestions, especially with regard to KeyNote.

References

[BFIK99] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The KeyNote Trust-Management System Version 2, September 1999. RFC 2704.

[BFK99] M. Blaze, J. Feigenbaum, and A. Keromytis. KeyNote: Trust Management for Public-Key

Infrastructures. In Proceedings of the 1998 Cambridge Security Protocols International Workshop, pages 59-63. Springer, LNCS vol. 1550, 1999.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In IEEE Symposium on Security and Privacy, pages 164-173, 1996.

[BFS98] Matt Blaze, Joan Feigenbaum, and Martin Strauss. Compliance checking in the PolicyMaker trust management system. In Proceedings of the 2nd Financial Crypto Conference, Lecture Notes in Computer Science, 1998.

[BMNW99] Yair Bartal, Alain Mayer, Kobbi Nissim, and Avishai Wool. Firmato: A novel firewall management toolkit. In Proc. IEEE Computer Society Symposium on Security and Privacy, 1999.

[CB94] William R. Cheswick and Steven M. Bellovin. Firewalls and Internet Security: Repelling the Wily Hacker. Addison-Wesley, Reading, MA, 1994.

[KA98] S. Kent and R. Atkinson. Security architecture for the internet protocol. Request for Comments (Proposed Standard) 2401, Internet Engineering Task Force, November 1998.

[Koe84] Andrew Koenig. Automatic software distribution. In USENIX Conference Proceedings, pages 312-322, Salt Lake City, UT, Summer 1984.

[LeF92] William LeFebvre. Restricting network access to system daemons under SunOS. In UNIX Security III Symposium, pages 93-103, Baltimore, MD, September 14-17, 1992. USENIX.

[Mog89] Jeffrey C. Mogul. Simple and flexible datagram access controls for UNIX-based gateways. In USENIX Conference Proceedings, pages 203-221, Baltimore, MD, Summer 1989.

[MRR97] David M. Martin, Sivaramarkrishnan Rajagopalan, and Aviel D. Rubin. Blocking Java applets at the fireawll. In Proceedings of the Symposium on Network and Distributed System Security, pages 16-26, San Diego, February 1997.

[PR85] J. Postel and J. Reynolds. File transfer protocol. Request for Comments (Standard) STD 9, 959, Internet Engineering Task Force, October 1985. (Obsoletes RFC765); (Updated by RFC2228).

[Ven92] Wietse Venema. TCP WRAPPER: Network monitoring, access control and booby traps. In Proceedings of the Third Usenix UNIX Security Symposium, pages 85-92, Baltimore, MD, September 1992.

[Ylo96] Tatu Ylonen. SSH - secure login connections over the internet. In Proceedings of the Sixth Usenix UNIX Security Symposium, pages 37-42, July 1996.