The Multilayer Firewall

Dan Nessett
3Com Corporation, Technology Development Center
5400 Bayfront Plaza
Santa Clara, CA. 95052
(408) 326-1169
dan_nessett@3com.com

Polar Humenn
BlackWatch Technologies, Inc.
2-212 CASE Center - Syracuse University
Syracuse, NY. 13244
(315) 443-3171
polar@blackwatch.com

Abstract

We present a new security technology called the Multilayer Firewall. We argue that it is useful in some situations for which other approaches, such as cryptographically protected communications, present operational or economic difficulties. In other circumstances a Multilayer Firewall can compliment such security technology by providing additional protection against intruder attacks. We first present the operational theory behind the Multilayer Firewall and then describe a prototype that we designed and implemented.

1. Introduction

The economic case for designing, implementing and deploying network and distributed system security mechanisms is now well established. Recent estimates of worldwide annual financial losses during 1995-1996 due to improperly protected information assets range from the hundreds of millions of dollars [1] to as high as 30+ billion dollars [2]. While actual losses may be less than the higher figure, losses in the billions of dollars annually are likely.

Even though there is general agreement that security mechanisms are necessary to protect information assets, there is less agreement on the specific technology to use. It is a thesis of this paper, justified in the next section, that many factors, including economic, legal and social constraints, affect whether a particular technology should be employed in a given situation. Different circumstances force different tradeoffs, implying that no technology is optimal for solving all security problems.

Working under this premise, we present a new technique for securing networks, called the multilayer firewall, which is useful in many circumstances. This approach extends the concept of a firewall as a device or devices that secure the border of a network to include the coordinated and selective restriction of traffic within a network, thereby protecting internal network resources. One of the innovations of our work is the use of a combination of high-level policy statements, network topology and a description of which devices are capable of enforcing security policy to automatically calculate the filter sets for each enforcing device. Since these sets are in general different, this relieves the system administrator from the arduous task of creating and downloading them to each enforcing device, the number of which may be large.

Another innovation is the potential to utilize network devices at both layer 2 and layer 3 to implement the firewall filtering activity. Traditional security filtering normally takes place in packet filtering routers or application level proxy gateways. However, to accommodate the performance requirements of internal network traffic, a multilayer firewall can use filtering functionality in layer two devices, such as 802.x and ATM switches, in order to achieve acceptable performance objectives for internal network traffic.

This paper is organized as follows. In the next section, we present an analysis of several network security technologies and suggest situations in which their use is unattractive. This motivates the presentation of the multilayer firewall, which is given in section 3. In section 4 we describe a prototype that we designed and implemented to test the multilayer firewall concept. Section 5 presents performance data that demonstrates the utility of using a combination of layer 3 and layer 2 devices for security filter enforcement. In section 6 we survey related work. Finally, in section 7 we present our conclusions from this research.

2. Motivation

Network security researchers and implementors have focused a great deal on how to protect networks from external attack. Traditional firewalls [3, 4] are designed to protect the borders of a network, preventing unauthorized access to internal resources by outside agents. Secure virtual private networks (VPNs) [5] have been used mainly to protect communications between private networks communicating over a public facility, such as the internet, or between a remote client and a border device positioned at a private network. In many cases VPNs are implemented using a tunneling protocol, such as PPTP [6], operating over a lower layer security protocol, such as IPSEC [7, 8, 9].

There has been some work to protect the internal communications of network management and control software, but it either has been inadequate, such as the use of SNMP community strings for protecting SNMP requests, is still being developed, such as SNMPv3, or is still in the research stage, such as mechanisms to protect routing protocols [10, 11, 12]. The use of encryption at the network layer and below to protect communications in non-classified networks until recently has been limited to the banking and financial industries. Work within the IEEE to standardize encryption for 802 based layer 2 communications has not seen significant implementation and deployment. While implementations of IPSEC and IPSEC-based [13] protocols are making significant progress, prior implementations of standardizing network layer encryption protocols [14, 15] were not deployed widely, at least in the commercial market.

The greatest success in protecting communications within the interior of a network uses application based security. Kerberos [16], DCE [17], and security mechanisms for the world wide web [18] have seen significant deployment. Security mechanisms for distributed object systems [19] utilize these and similar technologies for access control and protected communications.

So, the two best candidates for protecting resources internal to a network are IPSEC or application based security mechanisms. Yet, there are reasons why using either of these is sometimes inappropriate. This follows from certain characteristics that do not match the requirements of some common deployment situations. These requirements are as follows.

2.1 Performance

Both IPSEC and the common application based security solutions use cryptography to protect communications. While this provides significant protection, there is a performance penalty to pay. The use of cryptography for message integrity and authentication does not severely degrade communication and processor performance in most cases. However, its use for message confidentiality, when implemented in software, can significantly degrade CPU intensive application performance.

The attendant loss of performance becomes more serious as longer key lengths and stronger algorithms are required to meet the continuing decrease in the ratio of cost to computing performance. Thus, while DES was once considered sufficient for protecting high asset value unclassified data, this is no longer true. Most high asset value applications, for example, those in the financial sector, now require the use of triple-DES. However, existing desktop systems and those projected for the next few years have inadequate performance to support communications using triple-DES in software over common and emerging fabrics (e.g., 10 and 100 Mbps ethernet). For example, Bart Preneel of the Catholic University Leuven in Belgium reports that an optimized triple-DES implementation running on a 90 Mhz Pentium achieves 6.2 Mbps [20]. This result was computed when no other computation was running on the test system. Projecting, a 200 Mhz Pentium running nothing but triple-DES software should achieve approximately 13.8 Mbps. Systems running applications that use even a moderate percentage of CPU cycles (i.e., in excess of 30%) could not sustain a rate of 10 Mbps without users noticing a slow down. Applications requiring much higher bandwidth (e.g., those running over 100 Mbps ethernet, such as medical imaging applications) will not be able to use software based triple-DES in the foreseeable future. Consequently, hardware accelerators are probably necessary for systems running these applications. Such hardware introduces cost and legacy support issues that are discussed below.

Even for applications that access moderately valuable assets, for which single DES may be appropriate, confidentiality protection can be a problem. Preneel reports that an optimized single DES implementation can achieve 16.9 Mbps on a 90 Mhz Pentium [20]. Projecting, a 200 Mhz Pentium running only single DES software should achieve approximately 37.5 Mbps. Thus, the performance of single DES in software is acceptable for communications at 10 Mbps only for desktop systems deployed in the past several years. Many legacy desktop systems cannot support this rate. Furthermore, even the best desktop systems available cannot support 100Mbps communications with software based single DES.

The conclusion is cryptographic approaches for confidentiality will be useful only for a subset of applications, e.g., those valuable enough to justify the acquisition of new high performance end systems or hardware acceleration for existing systems, or those applications with low bandwidth requirements. Since some applications, such as medical imaging to the practitioner’s desktop and a large number of CAD applications, do not fall into these categories, a non-cryptographic approach to protected communications is justified in those situations.

2.2 Cost

Since cryptographic services may require hardware acceleration or desktop upgrade in order to achieve acceptable performance, its use introduces cost factors that may be unacceptable in certain circumstances. Generally, the acquisition of capital equipment is budgeted several years in advance and may replace only a portion of deployed computer systems. There is still a large number of relatively old systems in use today in many environments. Replacing all of these systems with newer ones is generally infeasible. Even when this is possible, replacement systems may not provide the highest available performance. A similar situation exists for hardware acceleration of cryptography.

An important consideration in regards to cost is the computing capacity available to an adversary in relation to the computing capacity available on an average desktop. After new desktops have replaced old ones over several years, the computing power available to an adversary will have increased. So, when systems considered state-of-the-art today are commonly available on the desktop, they will not be state-of-the-art in regards to their encryption support capabilities. Bandwidth capacity will have increased; computing capacity available to an adversary will be greater; and applications will arise requiring higher bandwidths.

The conclusion is there will never be a point when cryptographic solutions will be sufficient to address all application security requirements. Desktop computing capacity will always lag the cryptographic requirements of some applications. Consequently, non-cryptographic approaches to protecting network communications will always be useful.

2.3 Policy Enforcement

There are numerous situations for which communication between components in a distributed system must conform to a centrally administered policy. Examples include restrictions on the use of non-mission-critical applications during normal business hours in a financial institution, and restrictions on the information that a particular individual or organization may legitimately access, such as company financial or product planning data. The enforcement of such policy may occur proactively by preventing unauthorized communications, or retroactively by monitoring communications in order to detect policy violations.

When message traffic is confidentiality protected, the enforcement or monitoring activity must take place while the monitored data is in the clear. For application based security services, this requires the policy enforcement logic to reside between the application and the security service libraries. Since commonly deployed application-based security systems do not have such policy enforcement capabilities, this functionality must be retrofitted to the applications either by creating a "glue" layer library implementing policy, or by retrofitting the applications themselves with policy management support. In either case, the expense is potentially high due to engineering, manufacturing and redeployment costs. Furthermore, those administering central policy may not trust end systems to carry out policy enforcement unless there are hardware guarantees that such enforcement cannot be tampered with by the end user. Currently, there is no commonly deployed hardware with this capability.

There are more options for policy enforcement when a network layer security protocol, such as IPSEC, is used. If IPSEC is implemented in a network device, such as a router, which is used as one end of a confidentiality protected association, policy enforcement can be implemented in the network by observing the traffic either before it is encrypted or after it is decrypted. This configuration requires other security technology to implement the policy enforcement function.

If IPSEC is implemented at both ends of an association within the end systems, policy enforcement must occur in the end systems themselves. This may require the use of a policy "shim" inserted into the end system's protocol stack, a "glue" layer library located between the application and the protocol stack interface, or modified applications that are retrofitted with policy management software. As with application based security services, administrators may not find end system policy enforcement acceptable without guarantees that are difficult to achieve with existing end system hardware.

Thus, policy enforcement in a network protected by cryptography may require other security functionality, which examines cleartext data. This functionality would enhance the services provided by cryptography.

2.4 Legacy Support

Protecting communications with cryptographic services requires the use of software or hardware capable of using and providing cryptography. There are many deployed applications that are not designed to use cryptographic services nor could they be easily retrofitted to do so. There also are fielded computer systems running legacy operating systems or based on legacy hardware that cannot economically be retrofitted with the necessary cryptographic features.

To address these situations, the designers of IPSEC have included a tunneling mode, which provides protection for legacy system communications between tunneling endpoints. However, IPSEC tunneling only protects data while it is in the protected tunnel. At either end of the tunnel, the data is unprotected and susceptible to intruder attack. Other security measures may be necessary to protect this data as it moves in the clear over unprotected sections of a network.

2.5 Legal Issues

The use of cryptographic services is complicated by legal constraints. The world-wide promulgation of cryptographic functionality is currently constrained by export control restrictions and in some countries by import and usage restrictions. This has impeded the deployment of cryptographic solutions.

While there is evidence that certain countries are relaxing restrictions on cryptographic technology, it is unlikely that all legal restrictions will be removed in the near future. Thus, other approaches to network and communication security will remain valuable simply because they are more easily deployed.

2.6 Summary

The use of application or network based cryptographic services to protect distributed resources is useful in many important situations. However, there are other circumstances in which these approaches do not meet other system goals. Performance requirements, cost constraints, policy management considerations, legacy systems and legal issues may either render cryptographic solutions undesirable or limit their applicability, creating the opportunity to utilize other security technology.

3. The Multilayer Firewall

3.1 Background

Using firewalls to protect networks from external attack is a mature and widely deployed technique. The term "firewall" identifies a number of different equipment configurations. The most elaborate of these is constructed from several systems [3, 4], such as interior and exterior routers, a DMZ, and one or more bastion hosts located within the DMZ. However, less complicated configurations also qualify as firewalls, such as a single packet filtering router.

Administrators rely on the physical security of firewall equipment in order to prevent the movement of unauthorized traffic through it. They also depend on the integrity of message data, in particular source addresses and for IP based firewalls, source ports, for correct firewall operation. In situations where these assumptions are too risky, the use of firewalls is unwise. However, there are environments, such as certain corporate or institutional networks, some classified networks, and some carrier networks where these assumptions are reasonable. In such cases, the use of firewalls may provide an acceptable alternative to or enhancement of cryptographic based approaches.

Normally, firewalls are placed at the borders of a network in order to protect it against attack by external intruders. The positioning of current generation firewalls within a network to control internal traffic has the disadvantage of significantly reducing overall communications performance. Consequently, when deployed in this manner, firewalls are generally placed only at a very small number of points within the network where the traffic density is low.

3.2 Packet Filtering Firewalls

A simple class of firewall utilizes packet filtering to control the traffic allowed to pass between different networks. Virtually all of these firewalls use packet filtering routers or packet filtering engines that run in the kernel of an operating system.

Packet filtering devices, when properly configured, can prevent data from flowing through inappropriate portions of a network. This provides a limited form of confidentiality and integrity protection, since data is kept out of the reach of unauthorized individuals who might modify or view it. The strength of protection is not as great as that provided by cryptography, but it can increase the level of effort required by an intruder to access information.

The general architecture of a packet filtering firewall consists of the following components : 1) a user interface for specifying packet filtering rules, 2) persistent storage for retaining the current configuration of filtering rules, 3) a filter compiler that accepts a high-level description of filter rules (policy statements) and produces low-level commands or configuration data for the enforcement engine, and 4) an enforcement engine that implements the filtering mechanisms. In addition to these components there may be other optional components, such as audit trail functionality, which records anomalous events, or testing functionality, which allows an administrator to test the filter rules with traffic generated within the firewall.

Packet filtering firewalls require the retention of state in order to handle certain protocols. For example, FTP uses both a control and a data association. The control association carries the information specifying which port to use for data transmission. In order to allow the data association traffic to pass, a packet filtering firewall must snoop on the FTP control traffic, looking for the appropriate command containing the data association port. It then establishes a temporary filter rule, or its equivalent, which allows the FTP data to pass.

3.3 Multilayer Firewall Architecture

Traditional firewalls normally protect a network against external attack. An extension of this idea places firewall functionality within a network to protect it against internal attack. As mentioned above, this strategy is presently limited, since systems used to implement firewalls are generally slow. Placing them in the interior of a network dramatically degrades performance.

However, filtering is used for many purposes. Layer 2 devices, such as 802.x and ATM switches, filter traffic to enhance network performance by containing broadcasts. The implementation of filtering in these devices is highly optimized, providing significantly better communications performance than that available in routers.

It is possible to use layer 2 filtering to implement firewall functionality in addition to broadcast containment. This leads to the idea of a multilayer firewall, i.e., a firewall that uses filtering functionality at layer 3 and layer 2 to implement security policy.

A multilayer firewall (MLF) is constructed from the following elements :

- An MLF management system, consisting of one or more stations from which the MLF is controlled. This system provides an appropriate user interface for entering MLF security policy. Depending on its implementation, MLF policy could be expressed as a table of policy statements, as predicates specifying enforcement conditions on firewall traffic, or in some other way. The security policy language should be designed for clear and concise specification of desired network behavior. The language should be human-oriented rather than machine-oriented.
- A set of enforcement devices, which may be layer 3 routers, layer 2 switches or any other device that supports packet filtering or application proxying. An enforcement device is located within the interior of the network and connected to other devices through layer 2 links. Its filtering activity is specified either by commands or by configuration data.
- An MLF policy compiler, which accepts policy expressed in the high-level language and produces low-level data for the enforcement devices (see below). The compiler should be able to transform high-level policy statements into commands or configuration data for a number of different devices. To ensure extensibility, the compiler architecture should allow device translators to be plugged in, allowing the addition of new enforcement device types to the MLF.
- Persistent storage, which stores both the high-level policy and the low-level device data. The MLF architecture allows this persistent storage to be distributed. Such storage includes persistent storage subsystems, such as directory services and distributed database systems, as well as persistent storage on suitably equipped enforcement devices.
- Enforcement data transport, which is used to move the low-level device data from the MLF management system to the enforcement devices.
- A description of network topology, which includes information about the interconnection of nodes (i.e., network devices and end systems (hosts)) and which specifies the devices capable of and trusted to enforce MLF security policy. A more detailed explanation of the network topology information and how it is used by the MLF is given below.

3.4 Multilayer Firewall Operation

Traditional firewalls are normally configured by a firewall administrator entering filtering rules for a particular enforcement engine. Some firewalls [21] allow the administrator to configure multiple engines from a single user interface. The administrator specifies in each rule the set of enforcement engines that the rule affects.

Since the MLF may potentially control a large number of enforcement engines and since the filter sets for these engines (to ensure the highest possible efficiency) are normally different, relying on the administrator to decide which high-level policy statements affect which enforcement devices could lead to misconfigurations. Therefore, the MLF determines which devices are the target of a policy statement. The administrator does not identify these devices.

Devices are selected in the following manner. The high-level policy statements specify which hosts are allowed to inter-communicate using a specified set of application protocols. For example, the high-level language of the prototype (described below) allows the firewall administrator to collect hosts into host groups and then specify policy statements using these or an individual host tag as identifiers. A statement consists of a source host or host group, a destination host or host group, a protocol (e.g., FTP, Telnet), an action (i.e., allow or disallow) and an enforcement point (source, destination or both; this is explained below). For each statement in the high-level policy specification, the MLF : 1) determines the set of enforcement devices that must implement the statement, 2) compiles the statement into low-level commands or configuration data for each type of device (there may be more than one device type represented in the set), and 3) accumulates the low-level data for later delivery to the device.

The MLF decides which devices are affected by a particular statement by consulting the topology information, represented as follows. Each enforcement device is an "active" node. All hosts that do not enforce policy are "passive" nodes. The physical topology of the network is used to determine the passive nodes "associated" with an active node. A passive node is associated with an active node if traffic from the passive node may reach the active node without passing through another active node. Note that passive nodes may be associated with more than one active node. Finally, all active nodes are considered to be associated with themselves (this is important only if the active node can be the destination of network traffic, rather than always acting as a transit point).

When the MLF processes a high-level policy statement, it first determines whether there is a path between any host in the source set and any host in the destination set that does not pass through an active node. If so, the statement is flagged as unenforceable, and the administrator notified.

If the rule is enforceable, the MLF examines the source and destination hosts or host groups and determines a cut vertex set in the network topology graph that separates the source from the destination. Only active nodes may be members of the set. For each device type represented in the cut vertex set, the MLF translates the high-level policy statement into low-level data for that type and stores it in a file for the appropriate devices. When all statements are processed, the MLF transports each file to its associated devices.

Using a cut-vertex set of active nodes allows the MLF to operate in a heterogeneous environment. That is, only the active nodes must be capable of communicating with the management station and enforcing policy. Legacy devices and those without MLF functionality require no modifications to work in an MLF environment.

The best possible efficiency is obtained if the MLF computes a minimum cut vertex set for each high-level policy statement. However, computing cut vertex sets for large graphs can be computationally intensive. Consequently, an MLF may utilize heuristics to compute a cut vertex set that is not guaranteed to be minimal, but which is likely to be in many cases. For example, our prototype utilizes the last field of the policy rule (enforcement point) to quickly compute a cut vertex set associated either with the source, the destination or the union of these two sets (when "both" is specified) to accommodate cases for which double enforcement is desirable.

Distribution of the low-level device information is achieved in one of two ways. If the device is not capable of persistently storing its filtering data, the MLF management station stores it in the persistent store, then signals the device to update its enforcement data. The device then retrieves the data from the persistent store.

If the device is capable of persistently storing its filtering data, the MLF management station contacts it and moves the data to it directly. The MLF could use SNMP, a combination of Telnet and FTP (or TFTP), when the device supports these protocols, or some other configuration data transport mechanism. Use of these protocols for management requires initial device filter configuration data that permits traffic of this type to reach it.

Some enforcement devices, such as remote access concentrators, are able to establish filtering data based on a user identity. For example, some support user authentication and authorization through a server such as RADIUS. As part of the authorization step, filters associated with that user are loaded into the concentrator and then used to enforce security policy for the user's connection. For these devices, low-level enforcement data may be retrieved without prompting by the MLF management station.

3.5 MLF Partitioning

Many organizations are divided into separate divisions, departments or business units that control their own computing and networking assets. In addition, some networks may be too large to manage as a single MLF. Finally, some networks are naturally partitioned into independent units based on classification level, physical security, or other characteristics. To accommodate such cases it is necessary to create MLF partitions that consist of a subset of the nodes in a network and manage each partition as a single MLF. In order for the hosts in these MLFs to communicate with one another, an MLF must support the specification of "external nodes," which represent other MLFs. The identifiers for these external nodes should be allowed to appear in high-level policy statements wherever hosts or host groups appear.

MLF partitioning introduces several management issues. First, connections between MLF partitions may only occur at active nodes, otherwise, unauthorized traffic from one MLF partition could enter another. Secondly, policies specified in both MLF partitions controls the communications between them. Since each MLF management station only displays its own policy, an administrator cannot determine from either MLF console how inter-MLF partition traffic is controlled (unless some auxiliary protocol is defined for the exchange of policy data between directly connected MLFs). Finally, since MLF partitions can be interconnected in a general graph, it may be difficult to determine what actual policy is enforced within the federated MLF partitioned network. This difficulty arises from transitivity considerations that cannot be analyzed from the data of a single MLF policy database.

3.6 MLF Applications

An MLF is useful in a number of different situations. We present two examples.

A significant security problem for many corporations is allowing business partners access to the corporation's internal network in order to share information vital to the partnership. This must be done in a way that doesn't give the partner access to information unrelated to the relationship [22]. Protecting communications between the partners' networks using protocols such as IPSEC does not achieve this objective, since once connected to an end system, an individual can use it to connect to other systems in the network through protocols such as telnet or rlogin.

One way to limit such access is to create an MLF partition consisting of the systems that contain the data to be shared. Policy can then be established within this partition that allows telnet, rlogin and other remote terminal session protocols to enter the partition, but prevents their use from systems within the partition to those outside it. Establishing this kind of directionality for other protocols, such as http, ftp, nfs, and so forth, can further tighten the protection provided.

Another situation in which an MLF is useful occurs when a large organization formed from smaller departments requires protection against an insider threat. If the volume of traffic within departments is significantly greater than that between them, host groups comprised of departmental systems may be used to specify inter-departmental security policy. Such policy can limit the kind of traffic moving between departments, thus, providing limited protection against insider initiated intrusions. Other security functionality, such as auditing and intrusion detection, would further increase the network's ability to thwart insider attacks.

4. A Prototype

To test the concepts described above, we designed and implemented an MLF prototype. It has the ability to manage, analyze, and distribute high level firewall filtering policy in a network.

4.1 General Architecture

The MLF concept is implemented by using a network traffic analyzer and monitoring tool called *Traffix* along with *Tartan*, the MLF policy management tool. *Tartan* consists of a graphical user interface to create and edit policy and a policy engine that compiles the high level MLF policy, generates configuration information for the active nodes in the physical topology, and performs the configuration on the active nodes in the network.

4.2 Theory of Operation

A network administrator, Bob, uses *Traffix* to divide hosts on his network into logical groups. These groups can be semantic in nature, such as hosts grouped by the *Marketing* Department, the *Engineering* Department, and so forth. Using *Traffix* as an initial front end to set up the logical groups, the administrator can then invoke *Tartan* from a button on the *Traffix* console. He then is able to create and enforce MLF policy based on the logical group topology he created with *Traffix*. Through *Tartan*'s graphical user interface, the administrator can add, delete, and change firewall policy between groups of hosts. *Tartan*'s policy engine performs enforceability analysis and then distributes the policy throughout the network by reconfiguring the active nodes.

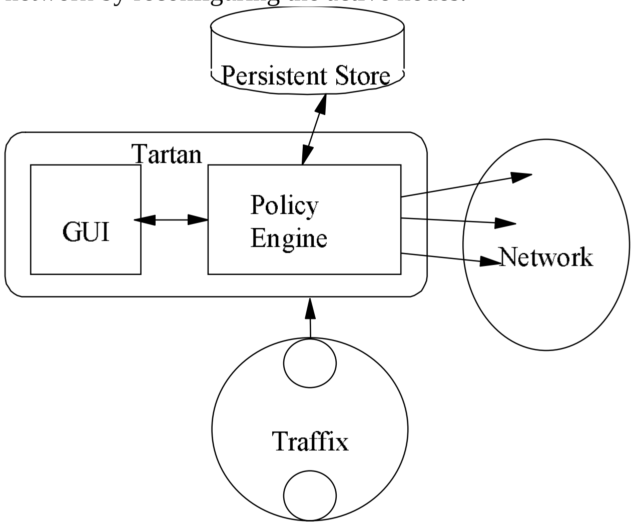

Figure 1. General Architecture of Prototype

4.3 Physical Network Topology

An MLF not only needs information on logical topology, it also needs information on the physical layout of the network topology. *Tartan* views the network hardware topology as a collection of active nodes and passive interfaces. Active nodes are filter enforcing and remotely configurable devices, such as routers. Passive interfaces are network interfaces that are not on policy enforcing devices, such as workstations.

Figure 2 shows a sample physical network layout. The boxes labeled *X* and *Z* represent active nodes each with two *ports* that are labeled 1 and 2, e.g., ethernet interface cards. The circles labeled *A, B, C, D, E, F,* and *G* are workstations. The larger ellipses labeled *Net1*, *Net2*, and *Net3* represent actual local area networks.

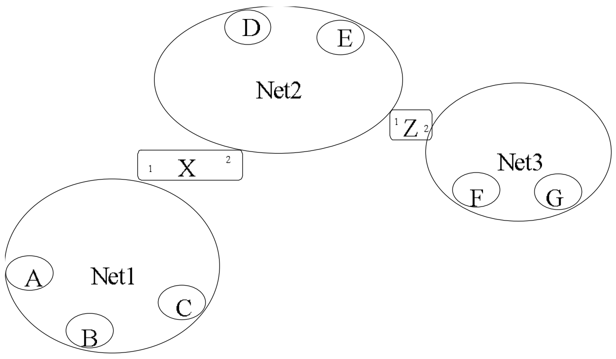

Figure 2. Physical Network Topology

A specification of the physical topology of the network that *Tartan* can work with should be provided automatically by some network administration tool. However, *Traffix* does not presently supply this information. Finding and integrating a suitable tool was beyond the realizable scope of the prototype. Therefore, network topology is presently specified in a configuration file by an administrator (see section 4.6.7 for further discussion of this problem).

The topology file representing the above figure is as follows:

```
Topology “BlackWatch Technology 13 June 1997”
   128.230.32.11 128.230.59.12
Active NetBuilderII  X1  “userid” “pw1234”
   Port  1  X1
       Passive  A
       Passive  B
       Passive  C
    Port  2  X2
       Passive  D
       Passive  E
Active NetBuilderII Z1 “userid” “pw1234”
   Port  1  Z1
       Passive  D
       Passive  E
   Port  2  Z2
       Passive  F
       Passive  G
```

The topology file contains the necessary information for *Tartan* to determine the lower level policy directives that must go to each active node within the physical topology. The keywords are: **Topology**, **ManagementStation**, **Active**, **Port**, and **Passive**. The first line in the file labels the topology with some identifying information. The second line labels the IP address(es) of the management station(s). The subsequent lines layout the physical topology.

The physical topology specification is structured primarily by active node. Active node lines contain a type identifier and authentication information for configuring the active node. For the prototype, the authentication information consists of a user name and password. Use of a more secure authentication technique would be preferable for an MLF implementation intended for deployment.

Passive interfaces are considered to be “behind” a particular active node’s port and are organized by port under each active node. For instance, in the above network, the passive interfaces, *A*, *B*, *C* are considered to be behind *X*’s port 1 (represented by the symbol) because they are directly connected to that port via an ethernet cable. Likewise, passive interfaces *D* and *E* are considered to be behind *X*’s port 2 and *Z*’s port 1. Similarly, the passive interfaces *F* and *G* are considered to be behind *Z*’s port 2.

NB: The topology file actually must contain only IP addresses to label the different active nodes and passive interfaces. However, for purposes of explaining this example in an understandable manner, please read the labels *A, B, C, D, E, F, G, X1, X2, Z1, Z2* as visual replacements for unique IP addresses.

4.4 Creating an MLF Policy

The administrator Bob uses *Traffix* to group the hosts in a semantic manner disregarding physical network boundaries. For example, Bob organizes the hosts *A* through *G* by department, such as *Sales*, *Engineering*, and *Management*. Bob organizes the groups such that hosts *A* and *D* belong to *Management*; hosts *B*, *C*, and *E* belong to *Sales*; and hosts *F* and *G* belong to *Engineering*.

Once Bob specifies the topology and groups the hosts, he can use *Tartan* to create an MLF policy. An MLF policy is an ordered list of policy statements. Policy statements are described in *Tartan* similarly to most traditional firewall products. Traditional firewalls describe policy between two hosts or networks by IP addresses. Policy rules are described in a source to destination directed manner. *Tartan* takes advantage of the *Traffix* grouping mechanism to specify a higher level filtering policy between source and destination host groups.

A policy statement that is specified between two entities contains three other attributes. It contains the name of the protocol to be filtered, such as FTP, Telnet, etc.; the policy, whether the traffic is *allowed* or *disallowed*; and whether the filter rule is enforced at active nodes near the source, near the destination, or near both end points.

Source	**Dest**	**Proto**	**Policy**	**Enforce**
Sales.C	Eng	FTP	Allow	Both
Sales	Eng	FTP	Disallow	Source
Mgmt	Sales	SMTP	Allow	Both
Mgmt	Eng	SMTP	Allow	Both

The order of the rules in *Tartan* is important. Each policy statement that is placed earlier in the list takes precedence over subsequent ones. Therefore, more general rules should be listed last, and the exceptions should be placed first.

For example, assume there are two logical groups *Sales* and *Engineering*, and a host, *C* from *Sales*. A more general rule is one disallowing FTP traffic from *Sales* to *Engineering*, whereas a more specific rule is allowing FTP traffic from host *C* to *Engineering*. Ordering the more general rule in front of the more specific rule effectively renders the more specific rule ineffective. Putting the rules in reverse order ensures that no FTP traffic should flow from *Sales* to *Engineering* except from host *C* (see section 4.6.9 for a discussion of problems with this rule ordering approach).

Tartan displays the policy statements in a graphical user interface using a vertically ordered list. Bob can insert, delete, and modify rules using the functions of the Tartan graphical user interface.

Once Bob has finished modifying or creating rules, he can run a check on policy enforceability. Due to a network hardware topology, a rule that Bob introduces might be unenforceable, such as a rule between two hosts that are directly connected without any active node between them. These checks help Bob define a comprehensive, sound policy.

When Bob is satisfied with his creation he can then ask *Tartan* to deploy the policy by the press of a button on *Tartan*'s graphical user interface. *Tartan*'s policy engine takes into account the physical topology and the logical group topology and delivers the necessary administration commands to the various active nodes on the network.

At the time of writing, the MLF prototype only compiles low-level data for NetBuilder II routers. However, the software architecture of the prototype is designed it to accept “plug-ins” that translate the high-level policy statements into low-level data for an arbitrary device, such as high-performance layer 2 switches.

4.5 Algorithm for Determining Enforceability

The prototype uses the following algorithm to determine the enforceability of policy statements.

1. For each rule in the policy:
1.1 Determine the set of source hosts and the set of destination hosts according to the logical grouping of hosts by *Traffix*
1.2 Create a set of all active node ports from the physical topology that contain the IP address of the source hosts.
1.3 Create a set of all active node ports from the physical topology that contain the IP address of the destination hosts
1.4 If the intersection of the two sets is non-empty, the rule is unenforceable. Therefore, the policy is unenforceable. Return with error.
2. If all rules have been processed without error, the policy is enforceable. Return with success.

4.6 Unresolved Issues

The prototype illustrates the concept of a multilayer firewall and demonstrates its feasibility. However, there are a number of issues that it does not address.

4.6.1 Layer 2 or Layer 3 Address Masquerade

Security policy that is defined using source addresses is susceptible to address masquerade attacks. Both network layer (IP) and link layer (802.x) addresses are easily spoofed. An intruder who wishes to defeat security policy need only discover which source addresses are allowed to send certain traffic and then masquerade as one of those addresses. While not specific to an MLF, layer 2 and layer 3 address masquerade lessens the attractiveness of firewalling as a security policy implementation strategy in environments susceptible to this attack.

There are several ways to deal with this problem. Some layer 2 devices, such as certain repeaters, provide a learning phase that allows a device to learn all legal MAC addresses. After the learning period, the arrival of a frame using a source MAC address not in the learned set causes the repeater to partition the port, preventing its traffic from being forwarded. This feature can be used to limit the number of MAC addresses available to an intruder for masquerade purposes.

Another approach to MAC address masquerade is to use a MAC address authentication technique. One approach is to secure source addresses by means of a cryptographic protocol such as that defined in the IEEE 802.10 standard [23]. While this reintroduces cryptography, the scope of cryptographic protection is local, allowing an administrator to protect only those parts of the network for which MAC address masquerade is considered a viable threat. Furthermore, only integrity protection is required, so the performance problems alluded to in section 2 do not pertain.

Layer 3 source address masquerade is somewhat more problematic than that for layer 2. While cryptographic protocols such as IPSEC provide source address authentication, it is an end-to-end service. An MLF requires end-to-multipoint authentication in order that the enforcement points are not misled by source address masquerade. In addition, these multipoints are internal to the network and are generally unknown to an end system. We are not aware of existing layer 3 protocols with this capability.

One possibility is to restrict all enforcement points to the layer 2 domain of the sending end system (i.e., to that part of the network fabric for which no layer 3 routing devices intervene between an end system and layer 2 devices). A combination of MAC address authentication and secure binding of MAC addresses to layer 3 addresses then provides the necessary service. Secure binding of layer 3 addresses to MAC addresses is problematic using protocols such as ARP, which have no security provisions. However, other binding protocols, such as NHRP, have fields, as yet undefined, that could be used for secure address binding.

4.6.2 Interaction Between the Prototype and DHCP

The current MLF prototype utilizes source and destination addresses within policy statements. However, protocols such as DHCP allocate IP addresses to end systems dynamically. Consequently, using a fixed IP address to identify such an end system is not viable.

There are a number of ways to solve this problem. One is to use a different piece of identifying information for these end systems. For example, the device's MAC address (assuming most such systems are singly-homed) could be used for this purpose. Since MAC addresses are only valid within a broadcast domain and not across routers, the enforcement analysis would become more complicated, since all active devices enforcing a particular rule would have to be in the end system's domain.

Alternatively, if the end system is directly connected to an appropriately equipped device, that device could snoop DHCP packets, associate the allocated IP address to the end system's MAC address and then securely distribute this binding to the MLF management system. The management system could then access the appropriate policy statements (which would specify the device's MAC address) and recompute the data for the enforcement devices. The possible disadvantage of this approach is that it might overload the MLF management system if there are a large number of DHCP managed devices or if those devices all tended to get DHCP addresses at the same time (e.g., when employees arrived in the morning).

A different approach would be to coordinate security policy with DHCP allocated addresses. If end systems were constrained to receive IP addresses in a particular range and all end systems allocated addresses in that range were considered to be equivalent from a security policy viewpoint, then the MLF policy statements could use host groups associated with that range of addresses in its policy statements. This would result in some loss of flexibility in managing security policy, but a DHCP server that allowed flexibility in allocating addresses to end systems might provide sufficient control to meet most security objectives.

4.6.3 Handling Protocols Requiring State Retention in Switches

Switches may not be designed to retain temporary state, such as that needed to properly handle certain application protocols. For example, ftp uses a data association, one port of which is defined by a command in the control association. Firewalls normally handle ftp by snooping the control association and temporarily establishing filtering data that passes data to/from that port. If the firewall notices no traffic on the data association for a configured interval of time, the enabling filter data is removed. Switches may not be capable of establishing such temporary filters.

There are several ways to deal with this problem. One option is to filter on the ftp well-know port and redirect the ftp traffic to a packet filtering router capable of handling ftp. The disadvantage of this approach is performance for ftp and other redirected protocols is severely degraded, since their packets must now transit a slow router, rather than be handled by a fast switch.

Another approach is to configure the switch with policy statements that prevent ftp traffic and then direct users to utilize another bulk data transfer protocol, like NFS or SMB, which does not separate control and data. This approach has the disadvantage that some applications may depend on ftp, and so preventing ftp traffic from transiting switches also prevents these applications running through switches.

4.6.4 Security of the MLF Management Protocols

The protocols to manage existing layer 2 and 3 devices generally are not secure. Using telnet and ftp with cleartext passwords has obvious problems. SNMPv1 is not secure and SNMPv3 is still being worked on. The security of proprietary management protocols is hard to ascertain.

This problem can be solved by computing a cryptographic checksum on the data downloaded to the device. However, this introduces the common problem of how to manage the keys, including managing and enforcing certificate signing policy. For small MLF partitions, this may be tractable in practice, and may provide a practical work around.

In the general MLF architecture, policy is stored in a persistent store, such as that offered by a directory service. The most common protocol for access to this service is LDAP. Since the prototype only works with routers that use telnet and ftp for configuration, we did not investigate the security issues associated with using LDAP for distributing security policy nor did we investigate the potential security problems with using SNMP set commands to notify a device that new policy is available.

4.6.5 Optimization of Enforcement Data

Our prototype uses the most obvious techniques to accumulate enforcement data for a device. Each individual statement in the high-level policy specification is translated independently of the others. However, there may very well be optimizations that are possible by processing rules together. For example, two policy statements may have the same source and destination but differ in the protocols specified. These two high-level statements may be implemented by one enforcement rule. Similarly, high-level policy statements specifying the same protocol might combine into one device enforcement rule. In general any commonality in high-level policy statements might allow optimizations of the low-level enforcement rules.

4.6.6 Interaction With Other Kinds of Filtering

We did not investigate how security filtering interacts with other filtering objectives, such as filtering for Quality of Service (QoS). A casual argument suggests such interaction might be important. Security filtering in an intermediate device would most likely introduce delay in forwarding a packet or frame. This in turn might introduce both delay and jitter into a stream of packets. Thus, calculation of QoS guarantees should accommodate delays introduced by security related filtering.

4.6.7 Security of *Traffix* Data

Traffix relies on SNMP to discover end systems. In particular, it downloads data accumulated in the RMON2 MIB of probes located at various points in the network. This is both an advantage and a disadvantage. It is an advantage in that an administrator need not maintain an ever changing database of end systems. It is a disadvantage in the following ways.

First, unless probes are placed in strategic places, not all end systems may be discovered. Appropriate placement of the probes in the network is critical for the correct operation of the MLF prototype. Secondly, RMON2 data must be securely moved between the probe and Traffix. This is a potential point of attack by an intruder attempting to subvert MLF security policy. Currently, SNMP is used to move RMON2 data, which has well-known security vulnerabilities.

These problems would be solved if there was a secure physical topology discovery protocol available to the MLF. While there has been work within the IETF to develop a physical topology discovery protocol [24], this work seems to have stalled. Furthermore, it's not clear the working group responsible for this work has the development of a secure protocol as an objective.

One work around for this problem is to manage physical topology information manually. To make this a tractable alternative, large networks would have to be divided into MLF partitions, each of moderate size.

4.6.8 Integrating MLF and Cryptographic Policy Management

The MLF prototype could be generalized and used to manage not only filtering policy, but also cryptographic policy (e.g., what services to provide, what algorithms and transforms to use). This would be a natural fit, since cryptographic policy might use the prototype's filter specification capability to advantage (i.e., specify cryptographic policy based on application as well as source and destination address). While the prototype software architecture is adaptable, using it for the specification of cryptographic policy was not a design objective. Consequently, the feasibility of such integration, both using our prototype as well as in general, is untested.

4.6.9 Alternate Policy Specification Approaches

The prototype uses the traditional approach to firewall policy specification, viz., a table of rules with precedence given to those that appear first. This technique has well know problems [25]. We did not experiment with other approaches that might eliminate some of the problems with table driven specification. This is an area for future study.

4.6.10 MLF Partitioning

The prototype does not support the specification of external nodes (i.e., other MLF partitions). Consequently, we do not know how this concept might work in practice.

5. Performance

We present two aspects of MLF performance. The first is how long it takes to compile and distribute MLF policy. The second is the throughput degradation imposed by security filtering in network devices.

We measured the performance of policy compilation and distribution on the prototype by specifying 13 high-level policy statements, which used 12 separate host groups and 1 host as sources and destinations. In the test network there were two enforcement devices, each NetBuilder II routers, the first with two interfaces and the second with three. The compilation produced 168 filter commands for the first router and 288 filter commands for the second for a total of 456 generated rules.

Our measurements were made on a Pentium 120 based system with 72 Mbytes of memory, running Linux SlackWare 3.2. The measured compilation and distribution times averaged 2 minutes and 20 seconds over 4 independent runs. This figure includes the time to translate high-level policy statements into low-level filter rules, logging on to each router, downloading the files to the routers, restarting the packet filtering firewall on each router (necessary to establish the new filter rules), and logging off each router.

While we were disappointed in the performance of policy compilation and distribution, several factors suggest much better times are possible. Firstly, our measurements were performed on first light prototype code. No code optimizations were available for these tests. Secondly, significantly more than 50% of the time was spent downloading filter rules to the routers and restarting their packet firewalls. There are ways in which the time to complete this phase of the update process could be reduced. Finally, there are several possible optimizations that would greatly reduce the number of filter rules generated by the prototype from a given set of policy statements.

To provide a comparison of router and switch filtering performance, we present performance data for two network devices, a NetBuilder II router and a CoreBuilder 2500 layer 2 Switch.

The current generation NetBuilder II router can route approximately 85,000 packets per second when no filtering is applied. The exact figure depends on a number of factors, including the number of interfaces supported and the bandwidth of its connections. A test of worst case filtering performance degradation was conducted, in which a set of ten filter rules were installed, including ones for telnet, ftp and http. When these filter rules were applied to eight interfaces, the packet throughput dropped by 45%. This implies a filtered aggregate throughput of 46,750 packets per second when these filters are installed.

The CoreBuilder 2500 theoretically can switch 565,000 pkts per second. However, the most loaded configuration is two 100 Mbit per second and sixteen 10 Mbit per second interfaces. With the smallest sized packets possible (64 bytes), this translates to 148,000 packets per second (accommodating frame synchronization and other overhead). Filtering is implemented in hardware in the CoreBuilder 2500, so filtering performance degradation depends on a number of factors, including the probability that a particular filter scores a “hit” (filters are applied serially). The worst case filtering performance on a CoreBuilder 2500 is 75,000 packets per second.

The above data implies that in the worst case the CoreBuilder 2500 switch can achieve approximately 160% better filtering performance than a NetBuilder II router. Since a CoreBuilder 2500 switch is around 1/3 the cost of a similarly configured NetBuilder II, this represents a filtering cost/performance gain of almost 500%.

6. Related Work

An MLF implements security policy by using network devices, persistent storage and a management station to control network resource usage. Security policy management systems are implemented in other ways and have been the subject of prior research.

Traditional firewalls are a major technology supporting security policy management. Research into their architecture and design is fairly advanced; so much so that several comprehensive treatments exist [3, 4]. Firewall policy data specifies the characteristics of the network traffic allowed to transit the firewall boundary. Such data drives the firewall’s access control machinery

[26], which tests attributes of the data to determine whether to admit or drop the traffic.

It was recognized early that firewall configuration data encoded security policy [25]. Experience with the use of firewall configuration suggests specifying policy data for firewall machines is prone to user error [27]. This is a problem that traditional firewalls share with our prototype, which uses a tabular format for presenting security policy statements.

Currently, administrators use simple policy management languages to configure firewalls. Modern firewalls allow an administrator to distribute policy data to multiple firewall machines [21]. However, such systems are generally intended to implement border firewalls, since the policy statements distributed to each firewall machine are a copy of those managed by the administrator on the common firewall console. These security policy management systems are not currently designed to manage policy where the enforcement data is automatically generated based on network topology, which is the case for an MLF.

Guttman [28] describes a specification language allowing administrators to test whether the composition of filters along packet paths conforms to security policy. Developing tools that analyze the global effects of local filtering policies is an important area of future research. Such tools would help administrators understand the global effects of MLF partitioning, for example.

Other policy specification languages include those for the management of distributed applications [29] and for distributed trust management [30]. The former allows an application manager to specify reaction rules for distributed real-time applications. These rules are compiled into enforcement code that is then distributed to the application components. Such distribution allows the components to react only to those conditions that directly affect their behavior. This is a characteristic that they share with an MLF.

A distributed trust management system, known as PolicyMaker [30], supports the use of policy rules, called assertions, to process queries, which specify access requests. Assertions associate a sequence of public keys with a predicate, the latter being used to determine whether a particular access request is affected by the policy represented by the assertion. PolicyMaker presupposes that policy enforcement is a local matter. Policies, even those representing global requirements, are processed locally by each application that utilizes them.

An MLF differs from PolicyMaker in two ways. First, policies do not express the association of public keys with predicates. Instead, they express conditions that a network administrator desires to establish within the network. Secondly, policy enforcement in an MLF is not a local matter. Rather, policies are defined and administered by one or more central management stations, which distribute policy enforcement information to trusted network enforcement devices. This architecture matches the operational environments in which network management normally operates, i.e., centrally administered control within and between administrative domains.

Another area utilizing security policy is network communications. When this service is based on encrypted network traffic, an administrator specifies the cryptographic services to use when two machines in a network communicate [31]. Issues such as the cryptographic algorithms to use, the required strength of the keys and other keying material, the required or desired type of service (e.g., data origin authentication, integrity, confidentiality) and which key distribution mechanism to use are examples of policy data relevant in this problem space. The IP Security protocol [7, 8, 9, 32, 33, 34] is an example of secure network communication technology that requires such policy management services. An MLF focuses on maintaining behavioral invariants in a network, rather than managing cryptographic services. However, it might be used to manage cryptographic services when these are an integral part of network behavioral management. This is an area of future investigation.

Prior research on policy routing [35] investigated how networks can implement resource usage policies. This motivated subsequent research that addressed the certification of policy data driving the routing algorithms [36]. Policy certification ensures that only authorized individuals or organizations control the security characteristics of a network. Such certification services would be useful in a federated set of MLF partitions that might be controlled by more than one administration.

7. Conclusions

The Multilayer Firewall is a new security tool that offers significant benefit to the practitioner. While it is not a panacea that will solve all security problems, when used in conjunction with other security technology, such as cryptographically protected communications, access control and event auditing, it can enhance the protection afforded to networking and distributed system resources.

8. Acknowledgments

We gratefully acknowledge the help of various individuals during the course of this research. In the initial stages of the development of the Multilayer Firewall concept, Paul Sherer acted as a much needed sounding board, which clarified a number of the ideas. Robin Iddon provided us with the necessary modifications to *Traffix* so that it could be used as the front-end for the policy definition user interface. Steve Mohr and Nair Venugopal provided us with the performance data for the NetBuilder II. Jim McCarron supplied the performance figures for the CoreBuilder 2500.

We especially thank Fred Dushin for his work on the Tartan GUI and Ender Ozcan for implementing some of the policy compiler functionality of Tartan.

Finally, we acknowledge the discussions on the IETF IPSEC working group mailing list by various authors, which provided some of the ideas presented in section 2.

References

[1] Computer Science Institute, "1997 Computer Crime and Security Survey," Computer Security Institute, 600 Harrison Street, San Francisco, CA 94107.

[2] J. Heffeman and D. T. Swartwood, Trends in Intellectual Property Loss Survey, American Society for Industrial Security, March 1996.

[3] William R. Cheswick and Steven M. Bellovin, Firewalls and Internet Security, Addison-Wesley, Reading, Massachusetts, 1994.

[4] D. Brent Chapman and Elizabeth D. Zwicky, Building Internet Firewalls, O'Reilly and Associates, Sebastopol, CA, 1995.

[5] Fisher, Susan E., "VPNs use tunneling to build private business links," Datamation, Vol 41, No. 23, June 1, 1996.

[6] Kory Hamzeh, Tim Kolar, Morgan Littlewood, Gurdeep Singh Pall, Jeff Taarud, Andrew J. Valencia, William Verthein, "Layer Two Tunneling Protocol 'L2TP'," IETF work in progress.

[7] R. Atkinson, "Security Architecture for the Internet Protocol," Internet RFC 1825, August, 1995.

[8] R. Atkinson, "IP Authentication Header," Internet RFC 1826, August, 1995.

[9] R. Atkinson, "IP Encapsulating Security Payload (ESP)," Internet RFC 1827, August, 1995.

[10] S.L Murphy and M.R. Badger, "Digital signature protection of the OSPF routing protocol," ISOC Symposium on Network and Distributed System Security, San Diego, CA., 1996, pp. 93-102.

[11] Bradley R. Smith, Shree Murthy and JJ. Garcia-Luna-Aceves, "Securing distance-vector routing protocols," ISOC Symposium on Network and Distributed System Security, San Diego, CA., 1997, pp. 85-92.

[12] Karen E. Sirois and Stephen T. Kent, "Securing the nimrod routing architecture," ISOC Symposium on Network and Distributed System Security, San Diego, CA., 1997, pp. 74-84.

[13] Germano Caronni, Hannes Lubich, Ashar Aziz, Tom Markson, Rich Skrenta, "SKIP - securing the internet," Proceedings of the fifth workshop on enabling technologies, (WET ICE '96), IEEE Computer Society Press, 1996.

[14] SDNS Secure Data Network System, Security Protocol 3, SP3, Document SDN.301, Revision 1.5, 15 May 1989, as published in NIST Publication NIST-IR-90-4250, February 1990

[15] ISO/IEC JTC1/SC6, Network Layer Security Protocol, ISO-IEC DIS 11577, International Standards Organisation, Geneva, Switzerland, 29 November, 1992.

[16] J. Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5)," Internet RFC 1510, September, 1993.

[17] Brad Curtis Johnson, "A distributed computing environment framework : an OSF perspective," Open Group Technical Paper DEV-DCE-TP6-1, June 10, 1991.

[18] Alan O. Freier, Philip Karlton, Paul C. Kocher, "The SSL Protocol, Version 3.0," IETF work in progress.

[19] Object Mangement Group, "CORBA Security," OMG Document Number 95-12-1, Dec., 1995, Object Management Group, 492 Old Connecticut Park, Framingham, MA 01701.

[20] Comments seen on the IPSEC Working Group mailing list (ipsec@tis.com), 1997.

[21] Checkpoint Software Technologies, Ltd., "Firewall-1 Architecture and Administration, Version 2.1," Checkpoint Software Technologies, Ltd., 400 Seaport Court, Suite 105, Redwood City, CA 94063.

[22] B. Hartman, D. Nessett, and N. Yialelis, “Scalability of security in distributed object systems : panel session,” ISOC Symposium on Network and Distributed System Security San Diego, 1996 , CA., pp. 40-41.

[23] Institute of Electrical and Electronics Engineers, “Standard 802.10-1992, Interoperable LAN/MAN Security (SILS),” IEEE, Inc., 345 East 47th St., New York, NY 10017.

[24] A. Bierman and K. McCloghrie, “Physical Topology MIB and Discovery Protocol Proposal,” IETF work in progress.

[25] D. Brent Chapman, "Network (in)security through IP packet filtering," In Proceedings Third Usenix UNIX Security Symposium, pp. 63-76, Baltimore, MD, Sept., 1992.

[26] Steffen Stempel, “IpAcess – an internet service access system for firewall installations,” Proceedings 1995 Symposium on Network and Distributed System Security, San Diego, CA., pp. 31–41.

[27] William J. Wied, “Trusted to untrusted network connectivity,” Proceedings 1994 Symposium on Network and Distributed System Security, San Diego, CA, pp. 89–98.

[28] J.D. Guttman, “Filtering Postures: Local Enforcement for Global Policies,” Proceedings of the 1997 IEEE Symposium on Security and Privacy, Oakland, CA., pp. 120-129.

[29] K. Marzullo, R. Cooper, M. Wood, and K. Birman, “Tools for Distributed Application Management,” IEEE Computer, August, 1991, pp. 42–51.

[30] M. Blaze, J. Feigenbaum, and L. Lacy, “Decentralized Trust Management,” IEEE Symposium on Security and Privacy, Oakland, CA., May, 1996.

[31] C. Weissman, “BLACKER: Security for the DDN, Examples of A1 Security Engineering Trades,” IEEE Symposium on Security and Privacy, May, 1992, pp. 286-292.

[32] D. Harkins, D. Carrel, “The resolution of ISAKMP with Oakley,” version 2, IETF work in progress

[33] D. Maughhan, M. Schertler, M. Schneider, and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", version 6, IETF work in progress

[34] H. Orman, "The Oakley Key Determination Protocol", version 1, IETF work in progress.

[35] D. Clark, “Policy routing in internetworks,” Internetworking: Research and Experience, Vol 1., No. 1, Sept., 1990, John Wiley and Sons, pp. 35–52.

[36] D. Nessett and D. Solo, “Policy route certification: requirements and techniques,” Information Security, Proceedings of IFIP TC11, 7th International Conference on Information Security, May, 1991, pp. 87–98.