Building Firewalls with Intelligent Network Interface Cards

David Friedman and David Nagle

May 2001

CMU-CS-00-173

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The primary method for protecting networks today is to use a firewall: a boundary separating the protected network from the untrusted Internet. However, these firewalls offer no protection from internal attacks, scale poorly due to limited firewall processing capacity, and do not support mobile computing. Distributing a firewall to each network host avoids many of these problems, but weakens the security guarantees of the network since it places the firewall under the control of the host OS. Leveraging the increasing capability of embedded-VLSI, including network-specific processors, we propose a Network Interface Card (NIC) based distributed firewall. Supporting the same (and more) functions as a centralized firewall, NIC-based firewalls provide significant benefits including: scalability, easier client customization, sharing application/OS state to enable application-level filtering, and the ability to block misbehaving hosts at the source, the host itself. We describe the architecture of a Network Interface Card-based distributed firewall and our implementation, which uses an i960-based NIC and IPsec for management and policy distribution. The firewall currently supports basic packet filtering and some application policies as well as secure policy distribution.

This research is supported by the member companies of the Parallel Data Consortium. At the time of this writing, these companies include EMC Corporation, Hewlett-Packard Labs, Hitachi, IBM corporation, Intel Corporation, LSI Logic, Lucent Technologies, Network Appliance, PANASAS, L.L.C., Platys Communications, Seagate Technology, Snap Appliances, Sun Microsystems and Veritas Software Corporation.

Keywords: firewalls, security, network interface cards, distributed systems

1 Introduction

For years, centralized firewalls have been the primary defense for computer systems by blocking unauthorized traffic and creating safe networking zones [33]. To achieve this level of security, firewalls inspect every network packet, blocking inbound and outbound traffic that could create security holes. This inspection is performed at various stages in the protocol stack in order to protect a network.

At the lowest level, firewalls examine fields in individual packets [28], for example, to block ICMP pings from outside the protected network. In some cases, however, simple packet inspection is insufficient due to its stateless nature [10, 42]. For instance, UDP traffic may require that IP fragments be reassembled for effective filtering. This requires slightly more computational resources than basic packet filtering, as well as buffer space for the fragments. The firewall may also provide a proxy which understands the specific requirements of an application. A web proxy, for instance, intercepts outbound requests for HTML documents and rewrites their responses to remove potentially dangerous Java applets or Javascript [26]. This is the most demanding task for a firewall, since the entire protocol stream must be reassembled in addition to the action of filtering the contents of that stream.

Unfortunately, firewalls suffer serious limitations. Inspecting all of the traffic traversing a firewall may not be possible at the line rate of the Internet connection. In addition, a firewall provides no protection to mobile users who take their laptops outside of the protected network. Finally, firewalls often must make decisions about filtering without knowledge of the end applications, which can lead to an overly conservative and pessimistic application of policy.

One solution to these limitations is to distribute the firewall to every node in the network [6, 20], by implementing a firewall in the operating system of the Internet host to be protected. Unfortunately these host-based distributed firewalls introduce new security and management problems. They require that the host operating system performing policy enforcement be trusted, though users may have control over it. Distributed firewalls may also be more susceptible to certain denial of service (DoS) attacks, since filtering is only performed at the end systems. Finally, distributed firewalls may be more difficult to manage than a single centralized firewall.

To overcome these limitations, we propose embedding the firewall and other network functionality in an intelligent Network Interface Card (iNIC). An iNIC may be implemented simply as the combination of a basic processing unit, such as a CPU or custom ASIC, in addition to a standard network interface. This additional processing allows the network interface to quickly filter both inbound and outbound data so that network administrators can block unwanted traffic and isolate misbehaving or suspect hosts from the rest of the network. We also discuss how these iNICs can be used to provide a solution to the problem of filtering IPsec traffic with a firewall. Further, unlike conventional NICs that are managed by the host OS, the filtering policy performed by the iNIC is managed by the network; the host need only have an interface to send and receive data on the network. The iNICs are effectively extensions of the ports on the inside of the centralized firewall, as both are managed by the network administrators.

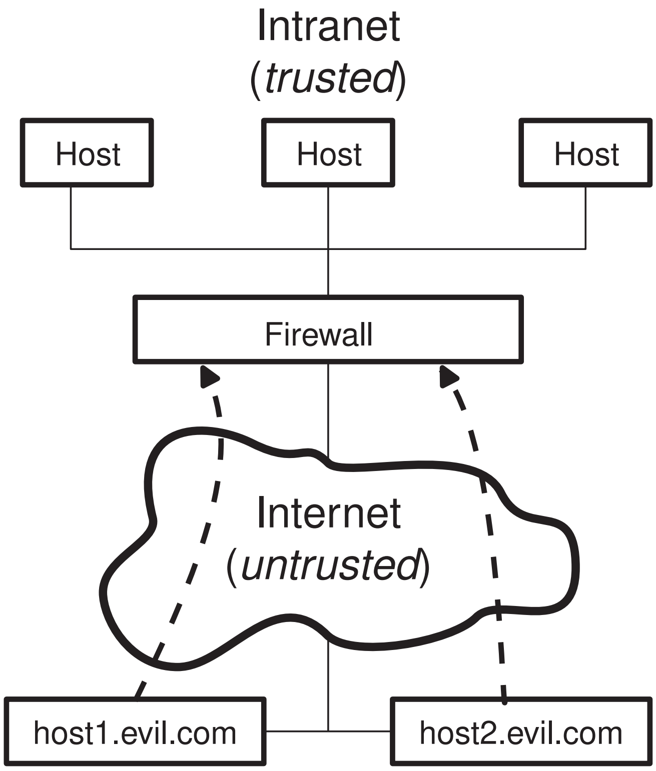

Figure 1: A conventional firewall: The network is partitioned into a trusted private network (or Intranet) and the untrusted public network, the Internet.

Many of the observations in this work are also discussed by Ioannidis [20], research performed concurrently. Our contributions include: an exploration of an iNIC-based distributed firewall capable of blocking both inbound and outbound traffic, an architecture for an iNIC-based firewall that employs a separation between policy enforcement (at the iNICs) and a policy manager, and a proposed interface between the iNIC, network, and host OS. Section 2 outlines previous work in firewalls including policy management and enforcement and further discusses the trends that motivate building distributed firewalls with iNICs. Section 3 describes our iNIC-based distributed firewall architecture and Section 4 discusses an initial implementation. In Section 5, we evaluate our prototype. Section 6 expands the notion of distributed firewalls and some other applications that could be future work. Section 7 discusses related work and Section 8 concludes.

2 Firewall Policy Management and Enforcement

Traditionally, firewalls are centralized network nodes that create a boundary between two domains. We first describe the basic operation of conventional centralized firewalls as well as their weaknesses. We then discuss distributed firewalls and how they can address some of these problems.

2.1 Conventional Firewalls

Figure 1 shows a centralized firewall, with the trusted intranet and untrusted Internet. Policy is specified by the administrator at the centralized firewall, and the enforcement of that policy occurs at the firewall as well. All hosts within the trusted domain (behind the firewall) are presumed to be secure, so that only traffic crossing the boundary between the two domains is inspected by the firewall. Simple packet inspection, such as checking for a ping flood attack, is relatively straightforward. However, more complex protocol or application-level processing is much more costly. In Section 5 we present results that underscore the scalability limitations of centralized firewalls.

In Figure 1, the firewall cannot enforce policy on the internal traffic. Thus, there is an inherent conflict: the firewall must trust the internal hosts not to attack each other, but it has no control over them. Furthermore, firewalls have many shortcomings: scalability, fault tolerance, location dependence, and lack of information.

Scalability. All traffic must flow through the firewall, making it a central bandwidth bottleneck. Moreover, the computational cost of processing packets quickly grows as more sophisticated filtering is performed. Thus, it is difficult to satisfy both security and performance requirements simultaneously. Furthermore, while some environments may currently permit taking the network offline during a firewall upgrade, new scenarios such as IP storage [9] and Internet data centers make downtime unacceptable.

Single point of failure. Breaking through the firewall exposes all protected resources. A single flaw in the policy of a firewall may enable attackers to defeat it and gain access to the protected network.

Location dependence. Conventional firewalls can only provide protection to machines that are located in the trusted domain. Machines that physically move outside the firewall are no longer protected. "Road Warriors", for instance, who bring laptops home from work are afforded no protection once they leave the private network. Furthermore, centralized firewalls cannot protect attacks from within and a significant number of all attacks now come from within the firewall [12].

Lack of information. Firewalls must reconstruct information about traffic in order to make intelligent decisions. Such reconstruction may require performing significant protocol processing, as well as application knowledge. In addition, emerging security technologies, such as IPsec [23], prevent centralized firewalls from enforcing policy.

2.2 Distributed Firewalls

Moving firewall enforcement to each host machine can address many of the above problems. Distributed firewalls greatly increase scalability [6]. Firewall enforcement cost is now born by each host and is based on the host's bandwidth requirements. Thus, one host's bandwidth consumption does not penalize the other hosts.

More importantly, the domain of trust shrinks to protect each individual host, providing a high degree of security independent of physical location. The host is responsible for enforcing firewall policy, which it obtains by contacting a firewall policy manager, usually on a separate machine that can be accessed by multiple clients.

Additionally, host-based firewalls can access significantly more information than centralized firewalls. For example, standard FTP specifies ports for receiving data dynamically [32]. Thus, in order to cooperate with a firewall, it is necessary to implement an application proxy which understands when to open the ports necessary for FTP. E-mail presents another problem since it is extremely difficult for a firewall to detect viruses within the contents of a message. Application-level proxies solve this problem at the expense of an additional process and/or machine. With host-based firewalls it is possible to perform the application-level proxy functions directly on the host.

In building a distributed firewall, it is important to maintain the same guarantees that security policy is being enforced. However, in using a host enforced distributed firewall, the network administrators must place even greater trust upon the individual host OSes and their users, since the end machine must enforce policy for communication between both trusted and untrusted machines. Users may deliberately or accidentaly modify or disable firewall policies. Even if the host OS is trustworthy, it is relatively easy to install a new OS that circumvents security. In addition, managing a large number of distributed firewalls can be difficult, especially if users can reconfigure the local firewall. Finally, denial of service attacks may be even more difficult to screen than in a centralized firewall. For example, a ping flood attack causes numerous interrupts to the host OS, slowing down the host processor and consuming available network bandwidth. However, in a centralized firewall, users are only affected if they interact with the untrusted network because this attack would be blocked at the perimeter of the network. This makes network DoS attacks much more lethal.

One alternate possibility for implementing a distributed firewall is to run the host operating system in a virtual machine [37]. The virtual machine may be managed by the administrators, while the operating system is under control of the end user. However, the virtual machine may require supporting several different processor architectures. In addition, virtual machines have historically provided virtual interfaces to all system resources. This adds a significant performance penalty, as normal operating systems functions must now traverse an additional protection boundary.

While it may be possible to build a virtual machine which protects only the network interface, there are several trends that make an iNIC-based distributed firewall preferable. Commodity network processors and ASICs enable low cost processing on each NIC, including TCP/IP protocol processing [1, 2]. Secondly, the increasing bandwidth requirements of applications such as the web and IP based storage [9] necessitate high-bandwidth networks that can easily saturate centralized firewalls.

iNIC-based firewalls can also better prevent DoS attacks. For example, a ping-flood DoS attack against host-based distributed firewalls would force the host OS to spend much (if not all) of the host CPU's resources servicing network interrupts – effectively blocking all computing resources. iNIC-based firewalls block the attack at the network interface. While still preventing access to network resources, the DoS attack stops

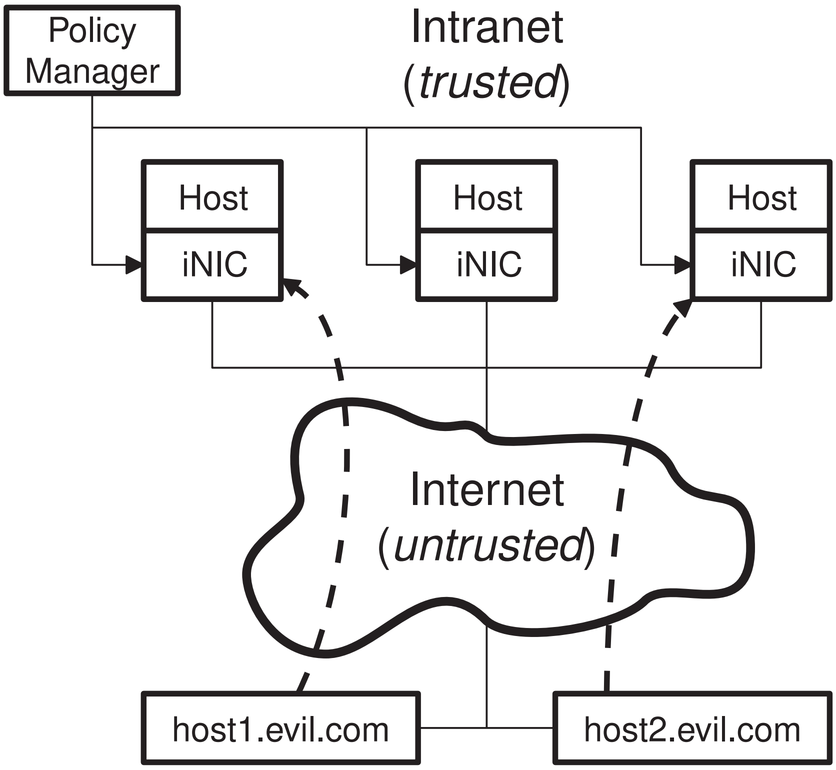

Figure 2: An iNIC-based distributed firewall: Policy is distributed to iNICs from a centralized location. The enforcement of that policy is executed at the end systems by the iNICs.

there, allowing a user to continue to access local computing resources.

3 iNIC-based Distributed Firewall Architecture

The iNIC-based distributed firewall overcomes many of the limitations of host-based distributed firewalls by enforcing policy at a tamper resistent network interface. The iNIC hardware can partition the iNIC's processor(s) and memory from host OS access and implements an iNIC-based embedded OS that provides guarentees for specific CPU and memory resources. The host OS may supplement firewall policy, but cannot prevent any of the network-managed firewall functions from executing. Therefore, administrators do not need to trust the host OS. Essentially, this architecture seperates the interface to transmit data on the network, which the host requires, from the interface to policy management, which is owned and managed by the network administrators. Figure 2 depicts our iNIC-based distributed firewall architecture. Since the firewall is managed by the network administrators, and not by the hosts, the protected network now has control over the entities that it conventionally has to trust. Thus, the network administrators have the same strong guarantee that security policy is being enforced on traffic from both internal and external hosts.

In designing a distributed firewall using an iNIC, there are several questions to be addressed. There are many alternatives for the interfaces between the iNIC and the host machine in which it resides, and also between the iNIC and the network management tools. We first discuss the network management interface.

3.1 Network/iNIC interface

There are several requirements for designing the management interface between the iNIC and the network. As discussed previously, the policies enforced by the iNIC should be defined centrally as in a conventional firewall. Thus, it is the network administrator, not the end user of a particular system, that controls the policies enforced by each iNIC. Therefore, the administrator must be able to install arbitrary filtering policies, though the end user and the operating system of the host cannot. This is enforced by hardware on the iNIC, and this functionality is commonly implemented by the I/O processors in I_2O [19].

In addition, the policies must be distributed in an efficient manner in order to keep the management of this system comparable to that of a centralized system. Furthermore, we need to ensure that any automatic method of policy distribution is authenticated. A distributed firewall could be easily defeated if each host could arbitrarily control the policy of another. We also require that the policies distributed be sufficiently flexible to enable customization to the needs of specific hosts.

One concern with distributed firewalls is the incredible work that may be required if these firewalls are compromised. In the worst case, an administrator may need to physically visit each compromised firewall. In this case, it is likely that the host is compromised as well, requiring the machine to be reinstalled and old data restored from backup. If the compromised iNIC stores its basic operating system in read only memory, the iNIC need only be reset, and a new key negotiated for distributing policy, which can be provided by the Internet Key Exchange (IKE) [17].

3.2 Host/iNIC Interface

We distinguish between two separate interfaces among the host and the iNIC: the data interface and the policy control interface. The data interface provides the host with access to the standard network services, while the control interface may allow hosts to supplement the policies specified by the administrators.

In designing these interfaces, it is perhaps most important to maintain simplicity. By keeping the interfaces simple, there is less likelihood of a security hole in the iNIC. The interfaces should be simpler than those of a host OS, making the stronger security guarantees of an iNIC practical. We first discuss the data interface.

3.2.1 Host/iNIC Data Interface

There are many possibilities for how to implement the data interface for the host and iNIC. The interface may be as low as Ethernet. However, some NICs now implement higher level network functions [1, 2], such as TCP/UDP checksums or even the entire fast path in TCP/IP. The interface could be even higher than

TCP, extending all the way to the application or sockets layer.

In a conventional Ethernet NIC the data interface is only the transmission and reception of Ethernet frames. By retaining this interface, the implementation of an iNIC-based firewall is perhaps more simple, since it is functionally equivalent to a traditional firewall, though it is physically located at the endpoint of the system.

By implementing IP and transport protocols on the iNIC, we have the opportunity to address the traversal problem of IPsec in traditional firewalls [23]. IPsec provides authentication or encryption for network traffic at the IP layer, enabling transparent use by any protocol using IP. However, both authentication and encryption present problems for traditional firewalls [6]. Clearly, encryption presents a problem since the firewall usually does not have the necessary keys to examine the encrypted data. Authentication may be a problem if the data requires modification. To see this, consider a web proxy which rewrites web pages containing Javascript by removing the offending HTML tags. The proxy must modify the data destined for the end system, however, the data is protected by an authentication hash. Thus, any modification will be detected by the end system, and the document will be rejected since this appears to be an attack. In some cases the firewall may correctly handle authenticated traffic if the policy requires only that certain packets be dropped. For example, this may be possible if each packet of a ping flood is authenticated. If the iNIC implements IPsec for the host, then it will have the necessary keys to modify the data according to network policy. Thus, in the above example the iNIC can perform the filtering on the cleartext after TCP and IPsec processing on a HTML data stream.

It may not be desirable for the iNIC to implement IPsec due to privacy issues, however, since the data could be read by the iNIC, and therefore, potentially by the network administrators as well. In this case, the user could use some alternative encryption method such as secure HTTP [36] or PGP [43], so that the iNIC would not have the necessary encryption keys. By allowing the iNIC to perform IPsec, we can make it possible to distinguish between data which is private to individuals, and data which is private for a workgroup or an entire company.

There are further benefits from implementing TCP on the iNIC. More complex filtering such as virus detection or blocking Java applets requires performing the protocol processing necessary in TCP/IP. Thus, a normal firewall may need to perform TCP/IP protocol operations twice: once for reassembling a stream of data, and then again disassembling it after filtering the contents. In Section 5 we present results which show that the computational cost of performing TCP processing is significantly higher than simple IP packet forwarding. An iNIC performing TCP could avoid half of the double traversal of the TCP stack performed by application proxies if it implements TCP for the host.

3.2.2 Host/iNIC Policy Control Interface

To implement system wide policies, a distributed firewall must allow the network policies to be controlled by the administrator as in a centralized system. The end users of the network should not have arbitrary

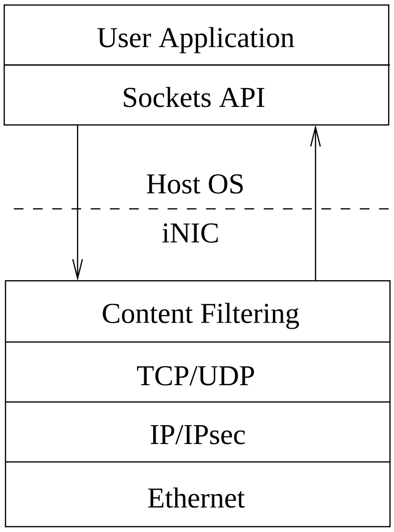

Figure 3: Host/iNIC interface: The user level application running on the host uses the usual sockets API. However, the host OS passes the socket calls directly to the iNIC. The iNIC performs any content filtering on this data above the transport layer, and above IPsec, enabling filtering on cleartext.

control over these policies. However, administrators may wish to allow the hosts the ability to supplement the policy. This supplement may be performed either by the end users or by the applications running on the end system. For example, the iNIC could first apply any system wide policies as specified by the network administrator. The iNIC could then apply user specified policies on any packets not rejected by the system wide policies.

However, such an interface may not be very useful to applications. We expect that a network administrator would specify policy for commonly used applications. Certainly any custom applications requiring some filtering could make use of the ability to supplement policy. Perhaps any custom filtering may be best performed on the host itself rather than on the iNIC, since this eases the problem of delegation and the strong guarantees required by general network policy are not necessary for user specified filters. In addition the iNIC can remain transparent to applications in order to minimize complexity.

This interface could allow the delegation of policy enforcement, either to the host OS or to other hosts as in Keynote [7]. The network administrators would need to decide the degree of control delegated, and in addition, the language for specifying policy must be sufficiently descriptive to allow such delegation. Thus, delegation adds significant complexity, but the additional ability to supplement user policies does not require it.

4 Implementation

We have implemented a prototype of a distributed firewall using an intelligent network adapter. The host interface to the iNIC uses a standard network sockets API. However, the host operating system passes these socket requests as messages directly to the iNIC, which implements TCP/IP and IPsec for the host as shown in Figure 3.

We experimented with various interfaces for the host OS and iNIC. Initially we experimented with the approach used by conventional Ethernet NIC software drivers. These drivers often use a combination of a simple message passing service and direct memory access (DMA). Small packets are copied to reduce latency, while DMA is used for larger packets to free up the CPU to perform other tasks. Additionally, we also considered an interface similar to that of VIA [25], which requires zero data copies between user space and the network. However, this requires changes to user space programs, which is undesirable. While we did experiment with a zero-copy host OS to iNIC interface, the complexity of the implementation led us to choose the approach taken by conventional NIC drivers. We do not discuss the specifics of this interface, however, since it is not essential to the distributed firewall work.

Our early attempts to build the iNIC used a prototype consisting of a StrongARM coprocessor PCI card in addition to a NIC in a common PCI bus. The host OS communicated only with the operating system running on the StrongARM, which controlled the NIC. However, the StrongARM had severe performance problems receiving packets due to the combination of the NIC and the 14-byte Ethernet header. The StrongARM does not support unaligned 4-byte accesses natively, so the IP and TCP headers must be aligned on a 4-byte boundary. However, many NICs also cannot copy incoming data to unaligned 4-byte boundaries. This combination necessitated copying the entire contents of each packet received.

Due to this problem, we instead used a single board iNIC which has a 100MHz i960, 128 MB of RAM, and four 100Mbit Ethernet ports [13]. The iNIC runs a very simple operating system and the network stack based on the implementation in FreeBSD. The host runs Windows NT.

Firewall rules are pushed to the iNIC from a single administration machine via an IPsec authenticated connection. As discussed earlier IPsec provides authentication at the IP layer, and therefore, enables applications to transparently achieve authenticated networking. IPsec authentication is achieved through the use of the Authentication Header (AH), which includes an authentication hash over the contents of the packet, as well as some of the fields in the IP header. Thus, any communication using AH can verify that the payload is authenticated.

The policy for this particular service is set by the iNIC kernel at the time of system boot. The authentication policy specification is currently based only on IP address, however, it could easily be extended to use an alternative credential. Therefore, the kernel will reject any packets destined for this service which do not contain AH or those that contain AH but have an incorrect hash.

The iNIC runs a daemon which listens on this authenticated connection for policy updates. The authentication keys for IPsec are manually distributed and are loaded into the kernel of the iNIC. However, this could easily be extended to use automatic key distribution using IKE [17], which allows keys to be negotiated and exchanged dynamically.

Note that it may be desirable to encrypt the policy distribution as well, since many networks achieve "security through obscurity." This can easily be achieved by adjusting the IPsec policy for this service.

Policy rules are specified using application filters based on regular expressions, which describe how data should be rewritten or dropped. The rules are applied above the sockets layer on the iNIC, as shown in Figure 3. Currently, specific knowledge required to interpret the application protocol is implemented in the iNIC operating system kernel. We describe these application specific filters to explain the operation of the system, however, a general policy system is preferable, though the authors are not aware of such a system that is freely available [1]. For example, the iNIC understands the POP3 mail message transport protocol, and it can apply filters similar to those in procmail. Note, however, that the filters should be restricted to rewriting, rather than dropping messages, as such behavior may cause the clients on the host to wait indefinitely for a dropped message.

One recent email virus was the life stages virus [14], which did not delete any sensitive data, but sent itself to all addresses in the user's address book, placing a great burden on Internet mail servers. The filter shown below rewrites the body of a message containing the words "life stages" in the subject line to remove the offending virus.

```
:0
* ^Subject:.*life.*stages
@rewrite "Message contains life stages virus."
```

The first line indicates the start of a filter. In the second line, the first character specifies the start of a regular expression. The regular expression specifies that any line starting with the word "Subject:" and containing the words "life" followed by "stages" matches. The final line specifies that the message shall be rewritten in the event of a match, and that the only remaining text in the body of the message should be "Message contained life stages virus." The sender's address, subject, and date of the message are replaced with pre-specified values to indicate to the user that the message was rewritten. The original headers are also included so that the user may contact the sender if necessary.

5 Evaluation

In this section we evaluate the performance of our prototype and compare it to a single centralized web server. We note that there are no standards for firewall benchmarking, so we developed our own. The

[1] We hypothesize that protocol specific knowledge could be specified in the filters themselves rather than requiring special support in the iNIC. This could be encoded in a simple state-machine language, specifying actions to take when certain patterns are sent or received. Even more flexibility is provided by the N-code language used by Network Flight Recorder [35], though it was intended for intrusion detection rather than general filtering. However, our primary goal is to understand how to build a distributed firewall with iNICs, rather than to create a generic policy language.

	Bandwidth
TCP transmit	2.2 Mbytes/s
TCP receive	6.5 Mbytes/s

Table 1: TCP performance using an i960-based iNIC: The performance of the i960 is disappointing, as caused by problems with its cache and low speed.

difficulty is in determining the relevant traffic which is to be filtered. Not only are the types and sizes of packets relevant, but the payload of those packets may also determine the filtering cost. We assumed that Web traffic would be the most common and developed a hybrid of SpecWeb and WebBench, to use a Zipf distribution of HTML and graphics. The workload uses 40% HTML files, and transfers a total of 200 Mbytes of data.

The experiments conducted in this section were carried out using one to five web clients connected to a 100 Mbit Ethernet switch. For the centralized firewall, the switch was connected to a single firewall, which was in turn connected via a second switch to a web server. For the distributed firewall, the web server was connected directly to the switch with the web clients. The firewall, web server, and clients were 300 Mhz Pentium IIs and ran Linux 2.2, except for the distributed firewall clients, which ran Windows NT. We first attempt to quantify the cost of various firewall operations. We then compare the centralized and distributed systems.

5.1 Firewall operations

In order to compare the cost of basic firewall functions, we used a single client running our hybrid workload with the centralized firewall topology described above. The firewall ran either simple in kernel packet forwarding routines, or a user level web proxy performing various operations. For the web proxy, we used http-gw from the TIS Firewall Toolkit [34], which is capable of rewriting HTML documents to remove Javascript or references to Java. Non-HTML documents are not examined.

Figure 4 shows the results of this experiment. The results show that while simple packet forwarding is easily performed at line rate, the addition of performing TCP on the stream adds a significant cost. The further addition of parsing HTML documents adds a still greater burden.

5.2 Centralized and Distributed Firewalls

Unfortunately, the performance of the i960 is less than satisfactory. It can only perform TCP at the speed of the network if TCP checksums are disabled (such computations are performed in hardware by many modern NICs, but not by ours). Table 1 shows the performance of sending maximum sized packets over TCP using our iNIC.

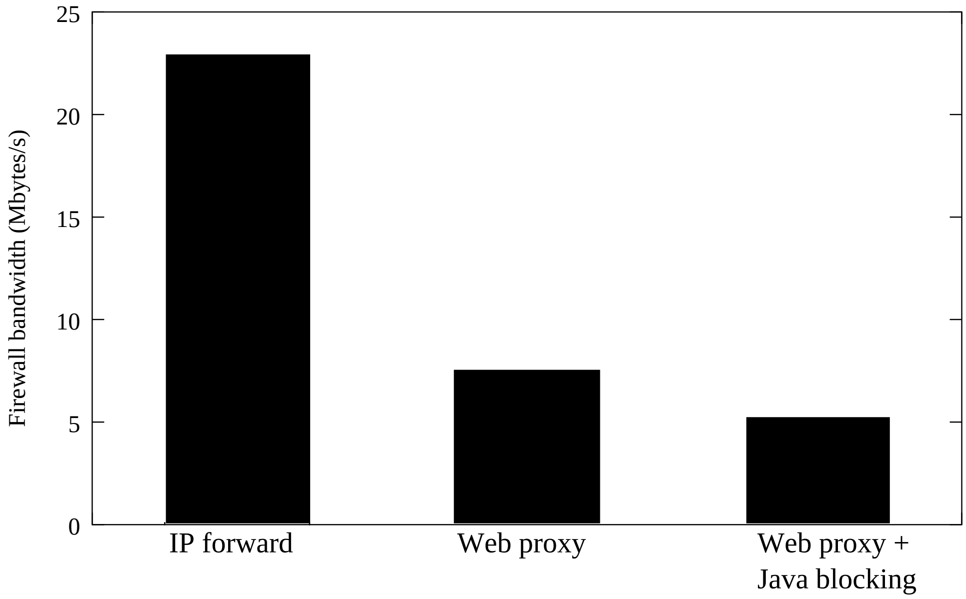

Figure 4: Firewall performance: IP Forward shows the performance of a centralized firewall using only the IP router of the Linux kernel with no packet filtering. Web proxy uses a user level web proxy to make requests on behalf of a user. Web Proxy + Java also examines the contents of HTML documents. The bandwidth achieved through the firewall is normalized to full CPU utilization.

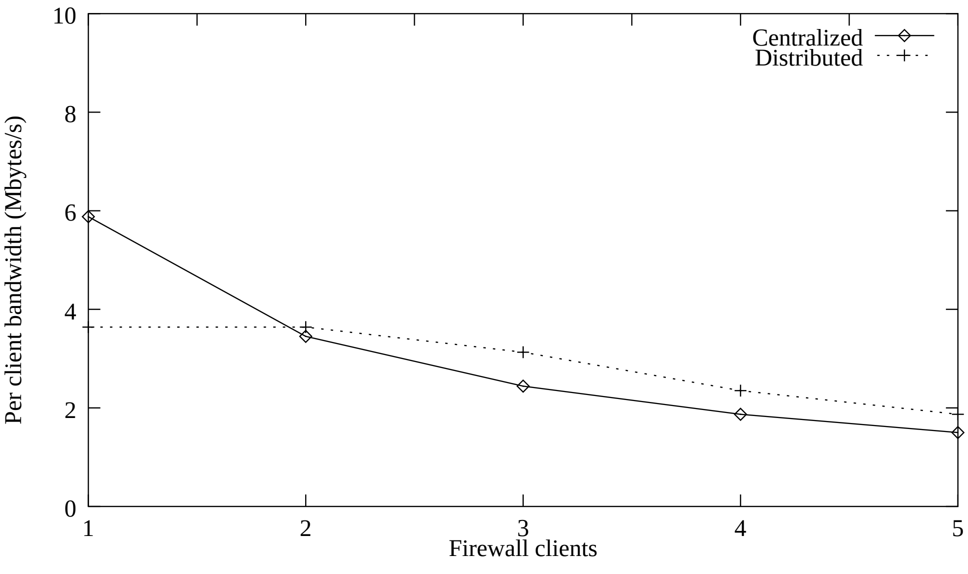

Figure 5: Web proxy performance: The firewalls each performed the proxy operations, making requests on behalf of the users, but did not examine the contents of the requests. Despite its performance limitations, the iNIC-based distributed firewall outperforms the centralized firewall with only a few clients, and is limited by the link capacity with only three clients.

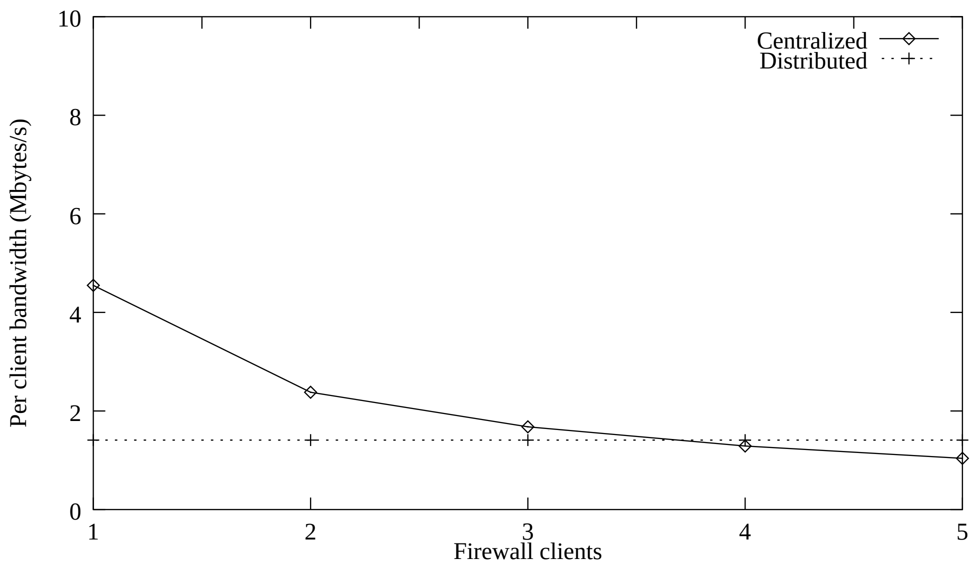

Figure 6: Java blocking performance: The firewalls each performed the proxy operations and also examined the contents of the requests and responses. The iNIC-based distributed firewall outperforms the centralized firewall with only a few clients.

Figures 5 and 6 show the performance of the centralized and distributed firewalls with simple proxy operations and blocking Java applets, respectively. The figures show the per client bandwidth with an increasing number of clients. Due to testbed limitations, the performance of the distributed firewall is extrapolated from that of one client. While the performance of the centralized firewall is initially better, with only a few clients, the distributed firewall achieves greater bandwidth.

6 Future Work

There are many future directions to be taken in this work. We plan to explore the partitioning of security functions in a hybrid network, which could include iNICs as well as some kind of centralized firewall. The results in Section 5 show that application filtering should not be performed on the centralized firewall. The placement of intrusion detection is much less clear, however, since this often requires inspecting state across hosts. Finally, some DoS attacks are clearly more effectively blocked at the entry points of the network. In a centralized firewall, such an attack would be blocked at the connection to the Internet, enabling communication among the machines behind the firewall to continue unaffected. In a strictly distributed firewall, at least some of the internal machines would also be denied the service of the network.

In addition, we hypothesize that some basic packet filtering may be best performed at a centralized location. For example, packet filters which are applied universally to all network traffic may be easily implemented in a centralized fashion. We have recently completed experiments which support this. Figure 7

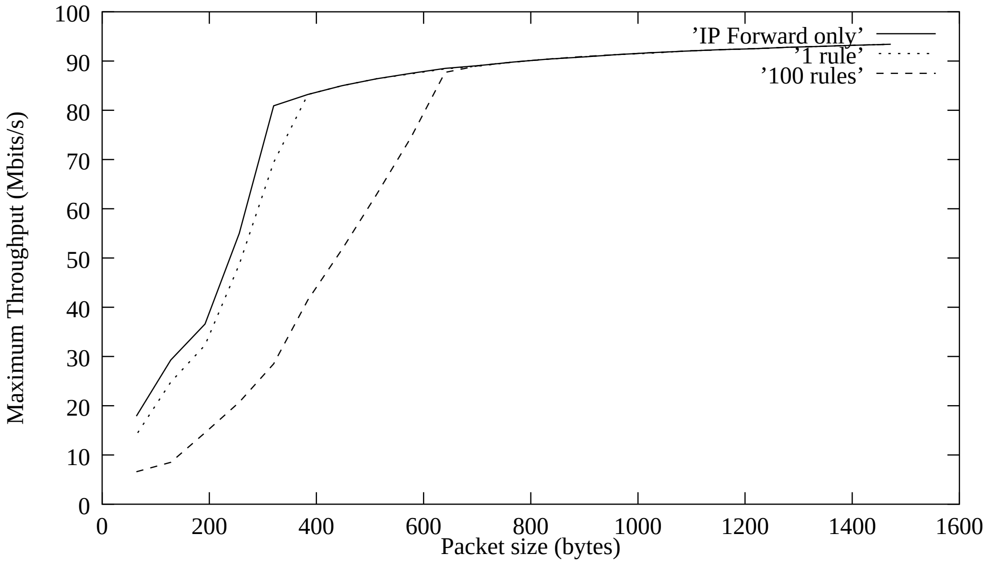

Figure 7: Network performance of basic packet filtering: IP Forward only presents performance with no rules at all. For 1 rule, the policy was set to allow only packets for the port used for the traffic measurements. For 100 rules, the rules included blocking specific IP addresses and ports, and represents a realistic firewall. For relatively large packet sizes, the bandwidth is comparable across all three cases.

shows the cost of applying simple packet filters. The tests were conducted by sending UDP packets on the same testbed as in Section 5, using IPchains [41], which allows very basic filtering rules using ports or IP addresses. This demonstrates that for sufficiently large packets, a large number of rules (and hence, complex policies) may be feasible for a centralized firewall.

We also plan to explore the requirements of building a heterogeneous network, where some hosts have distributed firewalls but others do not. A trivial solution would be to partition the network, but a centralized firewall may inspect traffic to decide whether to perform filtering on it. The problem of identifying which machines have iNICs and which do not must also be addressed. This could be accomplished with link layer authentication, similar to IEEE 802.1x [21]. For example, the NICs could authenticate to the upstream switch, which in turn authenticates to another upstream device. The authentication could continue up to the centralized firewall, which could then mark the individual machines as being trusted.

We believe that the iNIC is a useful tool for implementing generic network policy. For example, the policy distribution scheme could also easily support cryptographic policy using IPsec policies. It may also be possible to implement a traffic shaper on the iNIC which enforces policies for quality of service. The iNICs could mark packets with priorities for drop precedence, using an approach similar to that used by Stoica [38].

Finally, it may be possible to establish "collectives of trust" among iNICs. For example, a group of users may decide that they trust each other, so filtering may be unnecessary in this case.

7 Related Work

Firewalls have been the subject of research for many years [11, 28]. In that time, many problems with firewalls have been discovered as well [10, 42]. Effective techniques for blocking Java applets are described by Martin [26], though the authors note that blocking Javascript is more difficult.

Recent work has argued for a shift from firewalls to IPsec [31]. However, Bellovin [6] notes that encryption and authentication, (and thus, IPsec) do not prevent some attacks.

As discussed previously, firewall management has become a significant challenge. The administration of many independent firewalls is discussed by Miller [27] and it is argued that the firewalls should all be the same, though the management of different policies is not mentioned. Firmato [5] describes a toolkit for specifying policy independent of specific devices. This approach could be useful in a heterogenous environment.

The multilayer firewall [29] is quite similar to our work, promoting the addition of firewall functionality within a network, such as at a switch. By placing a firewall at a switch, it is effectively equivalent to an intelligent NIC with the restriction that the interface is strictly Ethernet. Thus, there is no possibility to inspect IPsec traffic, and more importantly, no protection afforded for mobile hosts.

Pathfinder explored work in both software and hardware-based packet classification [4]. Packet filters were specified using a tuple consisting of offset, length, mask, and value. The hardware implementation operated statelessly on packets, and it would need to be extended for application filtering, however, such functionality could be used to build an ASIC for use in an iNIC. More recently there has been work showing that high speed packet classification is feasible [24]. In addition, Netscreen uses specialized hardware for its firewall [30].

As discussed earlier, there has been other recent work in distributed firewalls. Ioannidis [20] describes a distributed firewall implemented in the operating system of a host. In the precursor to that work, the idea of a distributed firewall is discussed [6], where the firewall is based primarily on IPsec support in the operating systems of the end hosts. The work on TCP wrappers [39], which also runs on a host, uses pattern based access control to network services.

Previous work on intelligent I/O cards and NICs has focused on performance benefits [3]. Much of this work has used Myrinet NICs which have a low performance LanAI processor and little memory [8, 16, 40]. The SPINE project [15] described offloading multimedia functionality to a processor on a NIC, although the focus is on safe code execution. The Auspex NFS server [18] handles NFS caching and communication on its Ethernet processors. The VMP network adapter board [22] discusses the possiblity of using the adapter as a firewall. However, the adapter was intended only to prevent the host from processing unwanted packets, rather than distributing security policy enforcement from a centralized location, and therefore, the trust issues between the host and the adapter are not explored.

8 Conclusions

Firewalls are an important defense in an increasingly hostile world of networked computing devices. Unfortunately, technology and application trends are decreasing the effectiveness of conventional centralized firewalls. Increasing bandwidth needs and a diverse set of applications require firewalls to provide a significant amount of bandwidth and processing capacity. A significant percentage of attacks now also come from inside the firewall, where no protection can be provided. Worse, mobile devices cannot be tied to the strict topology requirements of centralized firewalls.

We have shown that an iNIC-based distributed firewall can overcome the limitations of centralized firewalls. Distributing the firewall to every host/device delivers scalablity, allows firewalls to exploit host-based application-level information, and enables security in a mobile environemnt. Central to this work is the notion that the firewall remain independent from the host, allowing network administrators to retain control, preventing breaches through a compromised OS, and preventing attacks such as DoS from denying access to all computing resources.

References

[1] Akamba. Akamba's Velobahn TM Web Server Accelerator - Keeping Pace with Network Growth, October 2000. http://www.alacritech.com/html/whitepapers.html.

[2] Alacritech. Accelerating server and application technology performance, 2000. http://www.alacritech.com/html/whitepapers.html.

[3] Emmanuel A. Arnould, François J. Bitz, Eric C. Cooper, H. T. Kung, Robert D. Sansom, and Peter A. Steenkiste. The design of Nectar: a network backplane for heterogeneous multicomputers. *Architectural Support for Programming Languages and Operating Systems* (Boston, MA). Published as *Operating Systems Review*, **23**(Special Issue):205–216, April 1989.

[4] Mary L. Bailey, Burra Gopal, Michael A. Pagels, Larry L. Peterson, and Prasenjit Sarkar. PATHFINDER: a pattern-based packet classifier. *Symposium on Operating Systems Design and Implementation* (Monterey, CA), pages 115–123. Usenix Association, 14–17 November 1994.

[5] Y. Bartal, A. Mayer, K. Nissim, and A. Wool. Firmato: a novel firewall management toolkit. *IEEE Symposium on Security and Privacy*, pages 17–31, 1999.

[6] S. Bellovin. Distributed firewalls. *;login: Magazine, Special Issue on Security*, November 1999.

[7] M. Blaze, J. Feigenbaum, and J. Ioannidis. *The Keynote trust managment system version 2*. RFC–2704. September 1999.

[8] Nannette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, Charles E. Seitz, Jakov N. Seizovic, and Wen-King Su. Myrinet: a Gigabit-per-second local area network. *IEEE Micro*, pages 29–36, February 1995.

[9] Mark A. Carlson, Stish Mali, Milan Merhar, Charles Monia, and Murali Rajagopal. A Framework for IP Based Storage, November 2000. http://www.ietf.org/internet-drafts/draft-ietf-ips-framework-00.txt.

[10] D. Brent Chapman. Network (in)security through IP packet filtering. *Computer*, **31**(9):34–41. IEEE Computer Society, September 1998.

[11] B. Cheswick and S. Bellovin. *Firewalls and Internet security: repelling the wily hacker*. Addison-Wesley, Reading, Mass. and London, 1994.

[12] Computer Security Institute. 2000 CSI/FBI Computer Crime and Security Survey, 2000.

[13] Cyclone Microsystems. PCI Quad-Ethernet Controller, 1999. http://www.cyclone.com/product/pci981.html.

[14] F-Secure Corporation. F-Secure Virus Information Pages: Stages, June 2000. http://www.datafellows.com/v-descs/stages.htm.

[15] Marc E. Fiuczynski, Richard P. Martin, Tsutomu Owa, and Brian N. Bershad. On using intelligent network interface cards to support multimedia applications. *International Workshop on Network and Operating System Support for Digital Audio and Video* (Cambridge,England, UK, 8–10 July, 1998), IEEE, July 1998.

[16] A. Gallatin, J. Chase, and K. Yocum. Trapeze/IP: TCP/IP at Near-Gigabit Speeds. *Annual USENIX Technical Conference* (Monterey, CA, June 1999). USENIX Association, June 1999.

[17] D. Harkins and D. Carrel. *The Internet Key Exchange (IKE)*. RFC–2409. November 1998.

[18] David Hitz, Guy Harris, James K. Lau, and Allan M. Schwartz. Using Unix as one component of a lightweight distributed kernel for multiprocessor file servers. *Winter USENIX Technical Conference* (Washington, DC), 23-26 January 1990.

[19] Intelligent I/O Special Interest Group. Intelligent I/O (I2O) Architecture Specification, March 1999. Available from http://www.i2osig.org/.

[20] Sotiris Ioannidis, Angelos D. Keromytis, Steve M. Bellovin, and Jonathan M. Smith. Implementing a distributed firewall. *ACM Conference on Computer and Communications Security* (Athens, Greece, 1–4 November 2000), pages 190–199, 2000.

[21] Tony Jeffree. 802.1x - Port Based Network Access Control, Draft 11, March 2001. Available from http://www.ieee802.org/1/pages/802.1x.html.

[22] Hemant Kanakia and David R. Cheriton. The VMP network adapter board (NAB): high-performance network communication for multiprocessors. *SIGCOMM '88 Symposium: Communications Architectures and Protocols* (Stanford, CA, 16–19 August 1988). Published as *Computer Communication Review*, **18**(4):175–187. ACM, August 1988.

[23] Stephen Kent and Randall Atkinson. *Security architecture for the Internet Protocol*, RFC–2401, November 1998.

[24] T. V. Lakshman and D. Stiliadis. High-speed policy-based packet forwarding using efficient multi-dimensional range matching. *ACM SIGCOMM Conference* (2–4 September 1998, Vancouver, BC, Canada). Published as *Computer Communications Review*, **28**(4):203–214, 1998.

[25] Rajesh S. Madukkarumukumana, Hemal V. Shah, and Calton Pu. Harnessing user-level networking architectures for distributed object computing over high-speed networks. *USENIX Windows NT Symposium* (Seattle, Washington), pages 127–135, 13–5 August 1998.

[26] David M. Martin Jr, Sivaramakrishnan Rajagopalan, and Aviel D. Rubin. Blocking java applets at the firewall. *Symposium on Network and Distributed Systems Security* (San Diego, CA, 10–11 February 1997), pages 16–26, 1997.

[27] Mark Miller and Joe Morris. Centralized administration of distributed firewalls. *Systems Administration Conference* (Chicago, IL, 29 September – 4 October 1996), pages 19–23. USENIX, 1996.

[28] Jeffrey C. Mogul, Richard F. Rashid, and Michael J. Accetta. The packet filter: an efficient mechanism for user-level network code. *ACM Symposium on Operating System Principles* (Austin, TX, 9–11 November 1987). Published as *Operating Systems Review*, **21**(5):39–51, 1987.

[29] Dan Nessett and Polar Humenn. The multilayer firewall. *Symposium on Network and Distributed Systems Security* (San Diego, CA, 11–13 March 1998), 1998.

[30] NetScreen. Securing Enterprise Networks With VPN & Firewall Solutions, January 2000. http://www2.netscreen.com/pub/solutions/enterprise_app.PDF.

[31] Rolf Oppliger. Internet Security: Firewalls and Beyond, May 1997. Available from http://www.i2osig.org/.

[32] J. Postel and J. Reynolds. *File transfer protocol (FTP)*, RFC–959, October 1985.

[33] Marcus J. Ranum. A Network Firewall. *World Conference on System Administration and Security* (Washington, D.C.), July, 1992.

[34] Marcus J. Ranum and Frederick M. Avolio. A toolkit and methods for internet firewalls. *Summer USENIX Technical Conference* (Boston, MA, 6–10 June 1994), pages 37–44. USENIX, 1994.

[35] Marcus J. Ranum, Kent Landfield, Mike Stolarchuk, Mark Sienkiewicz, Andrew Lambeth, and Eric Wall. Implementing a generalized tool for network monitoring. *Systems Administration Conference* (San Diego, CA, 26 – 31 October 1997), pages 1–8. USENIX, 1997.

[36] E. Rescorla and A. Schiffman. *The Secure HyperText Transfer Protocol*, RFC–2660, August 1999.

[37] Mendel Rosenblum, Edouard Bugnion, Stephen Alan Herrod, Emmett Witchel, and Anoop Gupta. The impact of architectural trends on operating system performance. *ACM Symposium on Operating System Principles* (Copper Mountain Resort, CO, 3–6 December 1995). Published as *Operating Systems Review*, **29**(5), 1995.

[38] Ion Stoica, Scott Shenker, and Hui Zhang. Core-stateless fair queueing: achieving approximately fair bandwidth allocations in high-speed networks. *ACM SIGCOMM Conference* (Vancouver, British Columbia, Canada, 2–4 September 1998). Published as *Computer Communication Review*, **28**(4):118–130, October 1998.

[39] Wietse Venema. TCP WRAPPER: network monitoring, access control, and booby traps. *UNIX Security Symposium* (Baltimore, MD, 14–16 September 1992), pages 85–92, 1992.

[40] Kenneth G. Yocum, Jeff G. Chase, Andrew J. Gallatin, and Alvin R. Lebeck. Cut-through delivery in trapeze: an exercise in low-latency messaging. *IEEE International Symposium on High-Performance Distributed Computing* (Portland, OR, 5–8 August 1996), pages 243–252. IEEE Computer Society Press, 1996.

[41] Robert L. Ziegler. *Linux Firewalls*. New Riders, 2000.

[42] G. Ziemba, D. Reed, and P. Traina. *Security considerations for IP fragment filtering*. RFC–1858. October 1995.

[43] P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, Cambridge, Massachusetts, 1995.