SANDIA REPORT

SAND97–8281 • UC–405
Unlimited Release
Printed August 1997

Enhanced Internet Firewall Design Using Stateful Filters Final Report

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

James A. Hutchins, Randall W. Simons

RECEIVED
SEP 2 6 1997
OSTI

MASTER

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Approved for public release; distribution is unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.
NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

UC-405

SAND97-8281

Unlimited Release

Printed August 1997

Enhanced Internet Firewall Design Using Stateful Filters Final Report

James A. Hutchins
Infrastructure and Networking Research Department
Sandia National Laboratories/California
and
Randall W. Simons
Decision Support Systems Architectures
Sandia National Laboratories/New Mexico

ABSTRACT

The current state-of-the-art in firewall design provides a lot of security for company networks, but normally at the expense of performance and/or functionality. Sandia researched a new approach to firewall design which incorporates a highly stateful approach, allowing much more flexibility for protocol checking and manipulation while retaining performance. A prototype system was built and multiple protocol policy modules implemented to test the concept. The resulting system, though implemented on a low-power workstation, performed almost at the same performance as Sandia's current firewall.

Table of Contents

Introduction

With the explosive growth of public networks, such as the Internet, has come a corresponding increase in the number of people and resources accessible to the research and commercial sectors. This growth has also created opportunities for hackers to attack systems on an international scale. To prevent attackers from gaining access, many sites protect their computers by placing them on an internal network and connecting to the public network through a "firewall", a device designed to only allow trusted and well-controlled connections to pass. Most designs today are primitive, using one of two mechanisms: simple packet filtering functions supplied in many of today's routers, or a proxy machine through which all traffic must pass. These firewalls only provide users with a limited functionality and generally fail when communicating with other sites that also have a firewall. A new approach to firewall design needs to be found that increases the available functionality while maintaining the required level of protection.

The biggest difficulty in using a filtering router in a firewall is that it must make the decision of whether to allow a packet to pass based solely upon the information contained in that one packet. This means the filter is "stateless". Using a proxy machine allows state to be kept, but at the cost of transparency and significantly more processing overhead, and hence a decrease in performance. An approach needs to be found that combines these two systems.

Background Study

A survey of current firewall products was conducted. (See Appendix A) The purpose of this study was to determine the state-of-the-art in firewall technology, and also to explore the possibility of identifying a commercial company that would be interested in building a prototype system that incorporatef our design into their product.

We found that while many companies sold firewall devices, almost all of these commercial products were simple enhancements, mostly in the area of the management interface, to the two basic firewall components: packet filters and proxies. A few of the more recently introduced commercial devices were more than a simple enhancement, though not by much. These enhancements were mostly in the form of attempting to employ some state-knowledge at a packet filter level - one of the techniques we were employing, though at a much more simplistic level than our approach. The state information maintained was limited to enhancing the handling of UDP packets and the watching of the FTP control channel to selectively enable FTP data transfers. No other state was kept; therefore if any more complex data manipulation was required a separate device from the firewall would have to be installed and employed.

We also found that only a couple of the commercial firewall devices made any attempt to improve, or even measure, their performance. This lack of interest in performance unfortunately appears common since almost all of the companies connected to the Internet today have a T1 (1.5 Mb/s) or slower connection, and these firewall devices generally can operate up to about a T1 speed. This leaves any company with a connection faster than a T1 with few or no options, other than compromising speed for the ability to buy a commercial product. We believe it is primarily for this reason that we found that most large companies prefer to build their own firewall. Such a custom firewall can be built from publicly available software and hardware components and allows a company to tailor the firewall to their specific environment, and thereby maximize its performance.

In general, it appeared that no commercial firewall devices were available that would be as robust from a security standpoint as our suggested approach while also maintaining good performance. It did appear though that the industry was slowly moving in the general direction of parts of our approach by beginning to employ state information into packet filtering systems.

Design Approach

Our design utilizes a modular approach where information can be processed at multiple levels depending on which level is the lowest, and hence most efficient, but also high enough to fully analyze the data from a security perspective. The block diagram is shown in Figure 1.

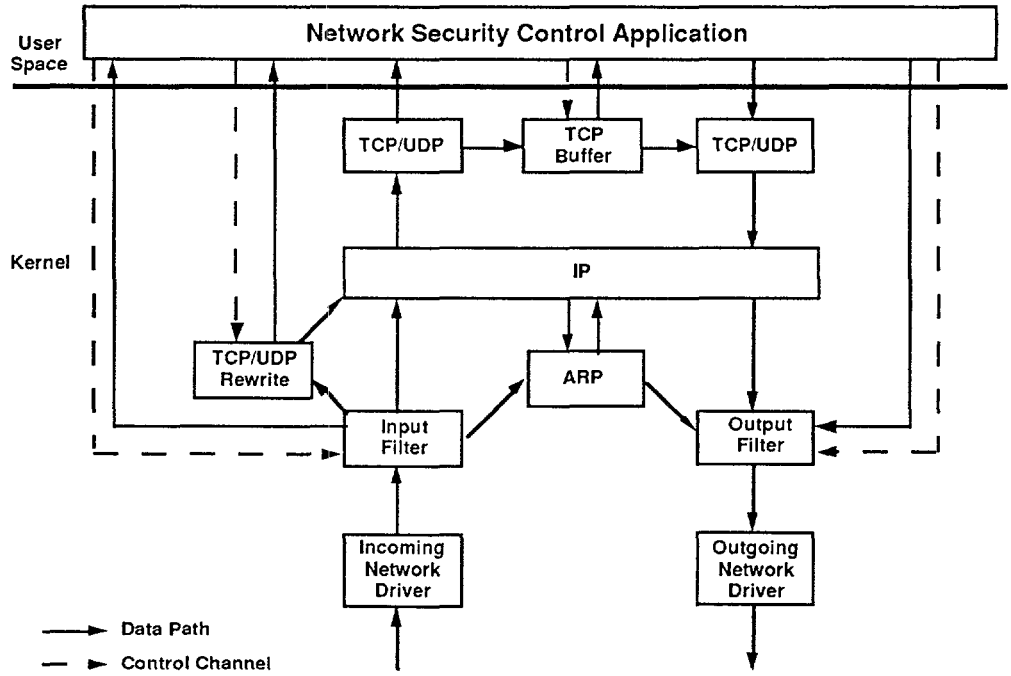

Figure 1: Block diagram of stateful filter design

The flow of data through this structure depends on the state of the connection as determined by the Network Security Control Application (NSCA). It is the NSCA's job to test all data flows and determine whether to trust them, modify them, or reject them. It does this by employing multiple modules, each of which performs a specific function.

Input Filter

The Input Filter is the first component of the firewall to act upon the packet. Its job is to inspect the packet according to a ruleset table defined by the NSCA and determine where to send the packet next. It may send it to IP to be processed normally, to a TCP/UDP rewriting module, to the NSCA, or any combination of these. The ruleset used by the

Input Filter can be updated as needed by the NSCA: for example to unblock an FTP data port when an FTP file transfer is initiated, and then block it again when the transfer is completed.

TCP/UDP Rewrite

The TCP/UDP Rewrite module takes a packet and modifies certain header fields. These changes allow some or all source or destination IP addresses to be hidden from one or both sides of the firewall, to provide a means of transparently gluing two different TCP or UDP connections together, and to allow the firewall to transparently intercept a connection, modify it, then allow it to resume.

To do this requires the module to be able to modify the IP source and destination addresses, the TCP and UDP port numbers, the TCP sequence and ACK numbers, and both the TCP/UDP and IP checksums. Which fields need to be modified and how they will be modified is determined by a table supplied to the module by the NSCA.

TCP Buffer

The TCP Buffer module is similar to the TCP Rewrite module in that it allows two separate TCP streams on the firewall to be connected after creation, but with a buffer in between. It is as if the creating application were reading from one of the TCP connections then immediately writing the data to the other connection unmodified.

Network Drivers, ARP, IP, and TCP/UDP

These modules are identical to those in a normal UNIX workstation except they have been modified to pass the data to the special modules shown in the block diagram instead of following the normal data flow.

Services Investigated

We implemented a prototype system on a Sun workstation. The kernel portion of the prototype software is dynamically loaded as a loadable kernel module, which allowed us to load and test changes during development without having to reboot the machine. The NSCA interfaces to the kernel modules through a special device file /dev/ipl. Data can be retrieved from and sent to modules through read and write system calls, and filter manipulations are performed through ioctl system calls. The interface specification is included in Appendix B.

A few NSCA modules were written to allowing testing. These modules included FTP and Telnet. A module for HTTP was also begun but was not completed before the end of the LDRD project. Other protocols, normally easily handled by typical filtering firewalls, such as ICMP and WHOIS, were also included in the base NSCA program.

The FTP module watched traffic on outbound FTP control connections (TCP post 21). When it saw a PORT command, meaning a file transfer was about to take place at that address and TCP port, it modified the filter to enable that port to get through. When the transaction was complete, it then disabled the port. The ability of an FTP data connection to connect from an external machine to an internal one has in general been one of the hardest problems for firewalls to adequately address due to FTP's popularity and requirement for high throughput.

The Telnet module intercepted all incoming telnet connections from an outside machine to an inside machine. It then presented a SecurID challenge to the originating machine on the telnet connection. When the challenge was completed successfully, the filter was modified to make all future packets on this connection pass through the TCP Rewrite kernel module without needing to go to the NSCA. Since the prompting caused the TCP sequence numbers to become desynchronized, the TCP Rewrite module had to be used to modify all future packets so they became resynchronized.

The function of the HTTP module is to intercept all HTTP requests and check them against local policy rules. For example, if the user is attempting to access a forbidden web server or URL, an appropriate error message can be returned. Also, returned header information and data can be inspected for potential trouble. One such use is to block MIME types that are considered unsafe, such as JAVA. Another possibility is to inspect incoming binaries which appear to be executables and susceptible documents for viruses.

Through these implementations, certain limitations of the original design were identified and addressed. One such limitation was that the original implementation had a floating filter set, so the deletion of one filter rule cause the identifier of the other later ones to change. If only one program was performing the filter manipulation, then this was not a problem, but it was decided to segment the functions of the NSCA using modules. For modules to be able to operate fairly independently, a unique filter identifier had to be assigned to each rule at creation. This feature was added.

Results

The implementations were tested for functionality and performance. Even though the test machine used was a low performance Sun Sparc IPC workstation, in the throughput test with FTP it was able to sustain a 3-5 Mb/s transfer rate. For comparision, a pure filtering Cisco router in the same test configuration was able to sustain a consistent 5-6 Mb/s. The FTP connection and command responses showed no noticeable delays. It is reasonable that a faster machine would be able to handle these transfers at full media rates, and probably be able to keep up with faster media, such as FDDI.

The telnet module was tested and found to perform the intended function, again with no noticeable impact on delay. Telnet is normally not a high throughput operation, though a test of the prototypes throughput also found a 3-5 Mb/s rate after the SecurID challenge was completed.

We speculate that the occasional drop in performance from the 5 Mb/s that a router achieves to 3 or 4 Mb/s is due to the workstation's need to periodically perform general kernel operations, such as flushing its disk cache. If these operations occur during a transfer, the momentary stutter impacts the throughput. If these operations could be performed on a dedicated processor, such as if they were built into a router, then the problem should go away.

Overall, the project was a success. The concept of the stateful filter was implemented and tested. It performed the functions we were trying to achieve, and did so with good performance, especially considering the CPU performance rating of the prototype platform.

The development showed that the interface specification for the NSCA was general enough to be able to handle a wide range of application, though special tweaking for certain applications may become necessary. It also showed that once an approach for addressing a protocol was developed, implementing a module to perform that approach was fairly straight-forward, though more time consuming than originally anticipated.

Appendix A - Background Study Results

The Current State of Firewall Technology

1. Abstract

As the number of sites connecting to public networks like the Internet increases, so does the concern about protecting these resources from hostile elements that are also connected and growing in number. The primary method of protection today is to install a firewall between a site and the public network. Most firewalls today are built out of a small number of component types, many of which are available in the public domain. This paper discusses the various component types, what commercial products are available, and what they have to offer over the readily available, public domain packages.

2. Introduction

The Internet is a world-wide collection of networks all using the common protocols of the TCP/IP suite. When the Internet was originally created, it was primarily populated by researchers from educational institutions. The need for network security was low since it operated by academic rules and little of real monetary value was attached to it. Since those days, the resources available on the Internet have exploded. Many organizations are in the process of connecting to take advantage of Internet services and resources. Businesses and agencies are now using the Internet for a variety of purposes, including exchanging electronic mail, distributing information to the public, facilitating collaboration with partners, and conducting research. Many organizations are connecting their existing internal local-area networks to the Internet so their workstations can have direct access to Internet services.

Internet connectivity can offer enormous advantages, however security needs to be a major consideration when planning an Internet connection. There are significant security risks associated with the Internet that are often not obvious to new (and existing) users. In particular, intruder activity as well as vulnerabilities that could assist intruder activity are widespread. Such activity is often difficult to detect and correct. Many organizations have lost productive time and money dealing with intruder activity; additionally, some organizations have had their reputations suffer as a result of break-ins at their sites being publicized.

There are two major ways to deal with these security threats. One method is to secure every device that a company has connected to the Internet. This technique is almost always impractical since the investment necessary to accomplish it is very large, or even impossible. A far more practical approach is to build a "firewall" between the company's internal network and the Internet. This device regulates the network traffic that flows between the company and the outside world, restricting it to operations that are allowed and considered safe.

Most firewalls in use today are either designed and built by the organization seeking protection, or by a hired consultant. The mechanisms available for firewall construction are very limited, comprised of packet filters, proxy servers, and circuit relays. These mechanisms are discussed in section 3. A few commercial firewall products exist, though almost all employ these same mechanisms with the only added value being a

better configuration/management interface. A few of the remaining firewall products present some interesting features which are discussed in section 4.

3. Methods of Firewall Construction

Almost all firewalls today are constructed using one or more of these three basic techniques: packet filters, proxy servers, and circuit relays. Each has its advantages and disadvantages, so typically a firewall system is built using a combination of them.

3.1. Packet Filters

Most high-end routers today contain the ability to filter packets while routing, based upon information contained in the packet. Examples of information available include what machine the packet is to or from, what service it is for, or how big it is. Each packet is filtered individually, so only information contained in the current packet is available to make a decision, but due to the way TCP, UDP and IP work, this is often enough. Based upon the packet's content, the router can then route it, drop it, or respond to it (with an ICMP Unreachable for example). Each of these actions can occur with the operation optionally being logged.

The use of a filtering router has certain advantages over proxy servers and circuit relays. Since the filtering is normally performed in special hardware inside the router, it can be done at very high rates so little or no performance penalty is usually paid. Additionally, the filtering does not alter the packets, just inspects them, so its operation is transparent; the user does not have to modify how he uses Internet services in any way. In fact, the user may not even know there is a firewall installed.

Filtering routers also have certain disadvantages over proxy servers and circuit relays. Since it can only decide the fate of a packet based upon the information contained in that one packet, any service whose security cannot be determined with that limited amount of information cannot be safely filtered. Some applications, such as outgoing telnet connections, contain all the required information in each packet, but many applications, such as incoming telnet connections, do not. Another concern with filtering routers is the filter's programming interface. The interfaces tend to be very terse and difficult to understand. The programmer is required to be knowledgeable in the operation of all the protocols the filter will handle, including any possible interactions. This step has proven to be highly prone to errors, with these errors resulting in undetected vulnerabilities to the site.

3.2. Proxy Servers

A proxy server is a device that accepts network connections for a service, processes the data, then passes the data on to the actual destination over a new network connection. The data is processed by an application, which may make forwarding decisions for the data, or require and process additional information such as for authentication. An SMTP mail relay is an example of a proxy server. It accepts mail from a remote host, processes the mail header to determine where the real destination is (or at least the next hop to it), then forwards the message on. Mail relays, like other proxy servers, are necessary since the destination may not be directly accessible from the source, or the destination may not be able to handle direct connections from the source for a variety of reasons, security being one.

The advantage offered by a proxy server over other firewall component types is its ability to examine at the application layer all the information passed and to base decisions on it. Since the information is from the entire data stream and not just one packet, the entire context of the operation is available for the decision. Furthermore, the application can modify the data stream to request additional security-related information, or to edit out dangerous operations when possible.

Proxy servers do suffer from many disadvantages. The one often considered the most significant is the potential performance penalty. All proxied network operations require going through the proxy server, having the data processed by an application, then being sent out on a new connection to the real destination. All of this takes time which can significantly impact the operation's performance. Since all operations go through a proxy machine, and there is normally only one such machine for each type of operation, indeed normally one machine for most or all proxied operations, this machine can become a performance bottleneck. Another disadvantage is the non-transparency of proxy servers. For most operations, the users must know that the proxy server exists and perform special actions to use it. These actions may be as simple as configuring a client at installation time like in Netscape, or as cumbersome as going to a special machine, performing some action like responding to a challenge-response, then telling the proxy where the real destination is every time an operation is desired. An additional disadvantage of proxy servers is the need for a different proxy server application for each service desired. Since the proxy examines the data, it must understand the data and its security implications, which is unique for each service.

3.3. Circuit Relays

A circuit relay is a special kind of proxy server, though different enough that it is normally placed in a class by itself. Its function is to accept connections from inside machines, determine the true destination and connect to it, then pass the data between these two connections. The clients that use it are linked with special libraries that know when and how to talk with the relay machine. The user's machine must have these special client programs installed for each service that will need to use external resources. After installation, the user does not have to do anything special, or even know there is a firewall installed. The circuit relay passes data that an inside machine requested without any interpretation, hence providing nothing more than a data pipe, or circuit. The most popular such package today is the public-domain SOCKS library.

The circuit relay enjoys certain advantages over the other component types, though it generally falls between a filtering router and a normal proxy server. Since it is in essence a stripped done proxy server, it does not have as much overhead as a normal proxy server does and may therefore be faster, though not as fast as a filtering router. It also is transparent to the user after installation like a filtering router. Since the relay is performed in an application running on a workstation, the administrator is able to create more extensive access control capabilities then normally available in filtering routers.

The circuit relay also suffers from certain disadvantages. Since it does not examine the data that passes through it, it cannot protect against malicious data an outside server might send in response to an insiders request. Also, a circuit relay is designed to allow inside machines to get out, but has no provision for outside authorized users to get in. Lastly, to use a circuit relay requires that a special client be created for each platform and service. Some platforms may not be common enough to be in the "ported to" list so some users would be unable to use it.

3.4. Bastion Hosts

A special kind of proxy server is a bastion host. A bastion host is a machine directly connected to both the external "untrusted" network, and to the internal network. The machine is explicitly configured to not act as a router. For an inside user to reach an external resource, he must first login to the bastion host, then run the appropriate utility for the desired service on the bastion. Likewise, for an approved user residing on the external network to access internal resources, he must also first login to the bastion host, however this login normally requires additional authentication actions like correctly responding to a challenge-response.

Bastion hosts have the advantage of being the easiest (though in a sense also the hardest) to install and administer. All that is needed is a machine with two network interfaces. No additional hardware or software is necessary. Their major disadvantage is they are the keys to the kingdom, and every authorized user has an unrestricted account on it. Any authorized user is potentially capable of running a process that creates a vulnerability in the machine which an outsider could exploit. Also, any authorized user with malicious intentions could break the machine from inside, then subvert the security measures, again opening the internal network to attack from the outside. This is why a bastion host is considered by many people to actually be the hardest (and hence least secure) firewall device to operate.

4. Special Firewall Products

There are many different definitions of what a firewall is, so there is wide disagreement on how many firewalls are currently installed or what kind they are. If you limit the discussion to IP-to-IP firewalls, and ignore those that act solely as email gateways, then it is safe to say that most firewalls are home-grown. That is, most firewalls are constructed by the site that wishes protection out of the components listed in section 3. There are however many commercial firewall products available. Most of these products are merely a vendor supported version of these same components, many with a more administrator-friendly interface than available with the public-domain versions. Some however, offer additional features designed to increase a firewall's security or usefulness.

4.1. FireWall-1

FireWall-1 is a software package that basically converts a Sun workstation into a filtering router, but one that retains a small amount of state information. One of the major problem areas (holes) for filtering routers is in the handling of outgoing FTP operations. When FTP performs a data transfer, the client dynamically allocates a TCP port and sends this port's address to the server over the FTP control channel. The server then connects to this port from the server's TCP port 20 and begins the data transfer. Since an outgoing FTP results in an outside machine making a TCP connection to a seemingly random port on the protected internal network, a site with a filtering router is faced with either letting such dangerous actions occur, or disallowing outgoing FTP. FireWall-1 watches the commands sent over the FTP control channel, and only lets incoming connections proceed if they correspond to an observed request from an inside machine.

A similar problem exists for RPC, though it has traditionally not been as much of an issue since there has been little need for RPC to pass through a firewall. Servers which will offer RPC services dynamically allocate ports at startup which they listens on for RPC requests. So that other programs can locate them, these ports are registered with a program called the portmapper. When a client wishes to access an RPC server, it can

either send the request to the portmapper and have it forward it, or first ask the portmapper what port the server is using, then send it directly. While RPC can use either TCP or UDP, most developers have chosen to use UDP for their transport protocol. The problem for FireWall-1 is, it is now faced with receiving UDP packets from a seemingly random port, headed for a seemingly random port, and trying to decide whether these packets pose a threat. FireWall-1 attempts to address this issue by watching results from portmapper lookups and only allowing outgoing UDP packets to ports corresponding to approved RPC services. When such a packet is let through, its address information is entered into a table. When an incoming UDP packet is received that does not correspond to an allowed non-RPC service, it is checked against this table to see if it could be an RPC response, and if so, lets it pass.

4.2. JANUS and Raptor

Both the JANUS and Raptor firewall products are software packages that run on workstations. They are unique in their ability to rewrite the TCP/IP headers to alter, and therefore hide, the internal IP addresses from the outside world. Each outgoing packet's TCP/UDP and IP headers are rewritten to make all requests look like they are coming from the firewall, thereby hiding all knowledge about machines on the internal network from the outside world. Whenever a packet is received from an internal host, the firewall first checks a table to see if a mapping already exists for the connection. If it does, then the packet's headers are rewritten and the packet is sent out. If there is no mapping, a new mapping is created. Incoming packets are also checked against this table. If the incoming packet matches an assigned mapping, it is unmapped (the internal machine's IP address and TCP/UDP port are substituted) and allowed in.

This address mapping presents certain difficulties for protocols that pass IP addresses in their data. One of these occurs in how FTP handles data connections. Since the data connection for an outgoing FTP will originate on the outside, a connection request will arrive at the firewall which it must handle. When the port address for the data connection is sent over the FTP control channel, the firewall intercepts it and substitutes the firewall's address. It also creates a mapping for it. Then, when the connection occurs, the firewall knows where to send the data. A consequence of this is both JANUS and Raptor maintain the same state information as FireWall-1 about which ports FTP has requested.

4.3. Encryption

A few firewalls, including NSC BDF, Blackhole, and ANS Interlock, have implemented a packet encryption capability. These firewalls allow packets destined for networks behind cooperating compatible firewalls to encrypt the packets as they travel across the untrusted network in between. So far, there is no standard for how the encryption and key management are performed so each such product is incompatible with those from other companies. They all use shared private keys for the establishment of the secure connections, so only sites that have negotiated the use of the encryption through human intervention can use it.

The benefit of data privacy as packets travel over the public network is obvious. A less apparent benefit is the authentication of packets supposedly from a peer-encrypted network. Forged packets masquerading as from another site behind an encrypting firewall will be rejected since they are either unencrypted or encrypted with the wrong key (if an encrypted site tries to masquerade as another). The difficulty with an encrypting firewall is the encryption overhead can force maximum size IP packets to be fragmented, increasing the probability for packet loss and decreasing the network performance of the receiving host.

Appendix B - Kernel Interface Specification

Available operations:

READ If you open the file /dev/ipl, then read from it, it will return a buffer containing chained struct raw_data structures. The actual structure's definition is different, but is effectively defined as:

```
struct raw_data {
        int             rd_totsize;
        u_short         rd_pktlen;
        u_short         rd_flags;
        u_char          rd_srcaddr[6];
        u_char          rd_if;
        u_short         rd_ref;
        struct timeval  rd_tstamp;
        u_char          rd_data[1];
};
```

rd_totsize	the total size of struct including padding
rd_pktlen	the length of packet
rd_srcaddr	the source MAC address (not yet working)
rd_if	the network interface which the packet arrived on (not yet working)
rd_ref	the filter rule that matched causing a capture
rd_tstamp	a timestamp of when the packet arrived
rd_data	the packet data
rd_flags	or'ed together bitmasks as defined below

Flags for rd_flags:

RD_BROADCAST	packet received as a broadcast
RD_CHAINED	another raw_data struct follows this one
RD_HASMACADDR	packet contains a MAC header (unused)
RD_HASIPHDR	packet contains a IP header
RD_HASTCPHDR	packet contains a TCP header
RD_HASUDPHDR	packet contains a UDP header

Note: If you do not pass read a big enough buffer, you may be returned no data. The read will return as many structures (in the order received) as it can fit in the buffer it was passed.

Note: By default, if no filter rules exist, packets are silently discarded (DISCARD).

WRITE Same as read, except will put a packet on the wire. The struct raw_data structures can also be chained.

IOCTL If you open the file /dev/ipl2, then perform an IOCTL on it, the kernel filter driver is notified and filter manipulation is possible. All the IOCTL's take a (struct snl_frentry *) as the third arg. The following ioctl's exist:

SIOC_GET_NUM_FILTERS	takes an int and returns the numbr of rules in the existing filter table

SIOC_PACKET_AT_A_TIME	takes an int and enables/disables whether read returns one packet or multiple chained packets
SIOC_INSERT_FILTER	takes snl_frentry struct and adds the rule after the existing rule with "handle". The special handles of zero and -1 mean to insert at the head and at the end of the list, respectively. Upon successful completion, the handle in the passed struct is filled in to contain a unique handle for this new rule.
SIOC_DELETE_FILTER	takes snl_frentry struct and deletes the rule with "handle".
SIOC_REPLACE_FILTER	takes snl_frentry struct and replaces the rule with "handle". The new rule's "handle" remains the same
SIOC_GET_FILTERS	takes POINTER TO a struct snl_frentry array and returns filter set (see SIOC_GET_NUM_FILTERS for how many)
SIOC_TCP_SHORTCIRCUIT	shortcircuit a TCP connection at the buffer level. This needs a different IOCTL since it works at a different protocol level and isn't a filter rule per se.

The IOCTL's take a struct snl_frentry, which is defined as:

```
struct snl_frentry {
        u_short filter_num;       /* which filter rule is this */
        u_char  action;           /* what to do when match occurs */
        u_char  protocol;         /* which protocol is this for */
        u_long  src_addr;         /* src IP address */
        u_long  src_addr_mask;    /* src IP address mask */
        u_long  dst_addr;         /* dst IP address */
        u_long  dst_addr_mask;    /* dst IP address mask */
        u_short src_port;         /* src port (or start of range) */
        u_short dst_port;         /* dst port (or start of range) */
        u_short src_port_top;     /* if non-zero, end of src port range */
        u_short dst_port_top;     /* if non-zero, end of dst port range */

        /* if rewriting (either header or shortcircuit) */
        u_long  new_src_addr;
        u_long  new_dst_addr;
        u_short new_src_port;
        u_short new_dst_port;
};
```

Protocol is which protocol this rule should apply to. The protocol of IP (0) causes the rule to only filter on IP addresses and masks, not on any ports or protocol. A protocol of TCP (6) and UDP (17) cause the ports to also be checked. Any other protocol number causes IP addresses and masks to be checked, as well as the protocol field, but not the ports.

The filters are applied in order, and the first match determines what happens to the packet. Various actions are possible upon a match. They include DISCARD, PERMIT, DISCARD_AND_LOG, PERMIT_AND_LOG and HEADER_REWRITE. One other action is defined (TCP_BUFFER) but is used internally and is not directly setable by the user.

The addr_masks are first applied to the packet's addrs, then these results are compared to src_addr and dst_addr for a match. So, to match any addr in subnet 37 of the 134.252 network, you would set the addr to 134.252.37.0. and the mask to 0xffffff00.

If src_port_top or dst_port_top is non-zero, then the port matches on a range where src_port and/or dst_port is the bottom and src_port_top and/or dst_port_top is the top of the range (inclusive).

For example, the following code fragment will delete the filter at rule position 20.

```
struct snl_frentry filter;

filter.filter_num = 20;
ioctl(fd, SIOC_DELETE_FILTER, &filter);
```

UNLIMITED RELEASE

INITIAL DISTRIBUTION:

1	MS 0806	C. D. Brown, 4621
1	MS 1138	B. N. Malm, 6532
2	MS 1138	R. W. Simons, 6532
1	MS 9003	D. L. Crawford, 8900
1	MS 9011	R. E. Palmer, 8901
1	MS 9011	P. W. Dean, 8910
4	MS 9011	J. A. Hutchins, 8910
1	MS 9012	J. E. Costa, 8920
1	MS 9012	S. C. Gray, 8930
1	MS9019	B. A. Maxwell, 8940
1	MS 9011	J Meza, 8950
1	MS 9011	D. B. Hall, 8970
1	MS 9001	T. O. Hunter, 8000; Attn: J. B. Wright, 2200 J. F. Ney (A), 5200 M. E. John, 8100 L. A. West, 8200 W. J. McLean, 8300 R. C. Wayne, 8400 P. N. Smith, 8500 T. M. Dyer, 8700 P. E. Brewer, 8800
1	MS 1436	LDRD Office,
3	MS 9018	Central Technical Files, 8940-2
4	MS 0899	Technical Library, 4916
1	MS 9021	Technical Communications Dept., 8815/Technical Library, 4916
2	MS 9021	Technical Communications Dept., 8815 For DOE/OSTI