FFI PUBLICATION

FIREWALL TECHNOLOGY

HALLINGSTAD Geir, JAATUN Martin Gilje,
WINDVIK Ronny

FFI/PUBLICATION-2002/01741

This page is intentionally left blank.

FFIE/780/113

FIREWALL TECHNOLOGY

HALLINGSTAD Geir, JAATUN Martin Gilje, WINDVIK Ronny

FFI/PUBLICATION-2002/01741

FORSVARETS FORSKNINGSINSTITUTT
Norwegian Defence Research Establishment
P O Box 25, NO 2027 Kjeller, Norway
APRIL 2002

This page is intentionally left blank.

FORSVARETS FORSKNINGSINSTITUTT(FFI)
Norwegian Defence Research Establishment

P O BOX 25
NO 2027 KJELLER, NORWAY

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

REPORT DOCUMENTATION PAGE

1) PUBL/REPORT NUMBER FFI/PUBLICATION-2002/01741 1a) PROJECT REFERENCE FFIE/780/113	2) SECURITY CLASSIFICATION UNCLASSIFIED 2a) DECLASSIFICATION/DOWNGRADING SCHEDULE	3) NUMBER OF PAGES 23

4) **TITLE**
FIREWALL TECHNOLOGY

5) **NAMES OF AUTHOR(S) IN FULL (surname first)**
HALLINGSTAD Geir, JAATUN Martin Gilje, WINDVIK Ronny

6) **DISTRIBUTION STATEMENT**
Approved for public release. Distribution unlimited
(Offentlig tilgjengelig)

7) **INDEXING TERMS**

	IN ENGLISH		IN NORWEGIAN
a)	Computer Security	a)	Datasikkerhet
b)	Firewalls	b)	Brannmurer
c)	Boundary Control	c)	Perimeterkontroll
d)	Packet Filtering	d)	Pakkefiltrering
e)	Buffering Strategy	e)	Bufringsstrategi
f)	Proxy	f)	Proxy

THESAURUS REFERENCE: None

8) **ABSTRACT**

Although the study of firewalls by now has evolved into a mature field, documentation and publications are still rife with ambiguities with respect to the terms used. This document ventures to describe today's firewall technology in its various component parts, using a consistent terminology. Current firewall research is also presented, along with an example of a possible future firewall design.

9) DATE	AUTHORIZED BY This page only	POSITION
29 April 2002	Paul Narum	Director General

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE
(when data entered)

This page is intentionally left blank.

PREFACE

This document does not attempt to present any revolutionary new ideas. Rather, it ventures to describe current firewall technology in a consistent manner, to enable researchers, system administrators and users to compare existing products based on their component features.

This page is intentionally left blank.

CONTENTS

PAGE

1 INTRODUCTION

When the Internet was created, security was not a prime issue. However, in later years, as the Internet has become practically ubiquitous, many organisations have been forced to rethink their attitude toward security. The prodigious amount of sensitive information currently available on private networks connected to the Internet makes security essential. Increased availability of electronically stored information translates into increased opportunity for mischief, and thus the amount of computer related crime is on the rise.

Some of the problems are software bugs, ambiguous protocol use, misconfiguration, and user error. On large intranets, it is difficult to ensure that all software is updated, configured and used correctly at all times. To avoid having to do this on all hosts, a centralised point of control was created: the firewall (1, 4, 6, 7).

The purpose of a firewall is to analyse and filter traffic between a private network and other networks at one single point. The idea is to stop illegal traffic before it enters the private network. Different types of firewalls exist, but they all share the common goal of stopping potential security breaches before they reach the destination host.

While the first firewalls were simple filters, today's firewall implementations contain a great amount of complex functionality. As new types of firewalls emerged, new terms were taken into use without any underlying basis. The abundance of such terms have made it difficult to comprehend what a product is doing by looking at its name. This document presents a concise terminology, breaking the functionality of a firewall down to axiom functions, so that firewall products can be described in a concise manner.

The document is organised as follows: Section 2 presents the technology that different types of firewalls utilise, section 3 looks at ongoing research and section 4 concludes. Appendix A presents some commonly used terminology, and ties it with the terms defined in this document.

2 FIREWALL TECHNOLOGY

Although firewall design and implementation has been studied for at least two decades, there is still a lot of ambiguity with respect to the terms used to describe the various components. In this document, we describe the technologies used in firewalls by breaking firewall components into basic functions that are precisely described using a consistent terminology.

When examining firewall technology, one can identify three main functions, where at least one always is present in an implementation of a firewall: *Packet filtering*, *Network Address Translation (NAT)* and *Proxies*. Packet filtering mainly inspects packets and forwards or blocks them, network address translation is often used in conjunction with packet filters to achieve address hiding, and proxies are used when each connection is to be examined more closely. In the following, these technologies will be discussed in more detail.

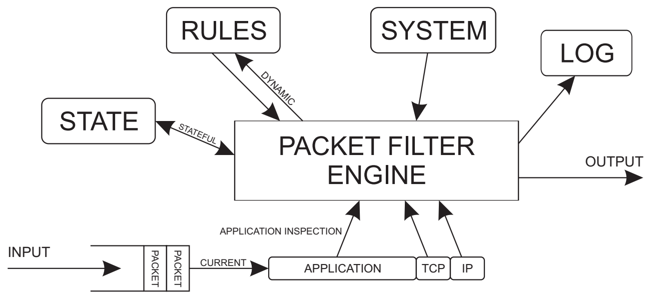

Figure 2.1: A packet filter consisting of the packet filter engine and components that provide information to the filter, including some which are updated by the filter engine.

2.1 Packet Filters

Packet filters are protection measures that can allow, block or take other actions on packets. A packet filter, shown in Figure 2.1, consists of the packet filter engine and several supporting modules. The packet filter engine will load a rule-set from the rule module and process the relevant information about the current package. This information could include header information from the packet itself, state information from a state module, and system information (such as which network interface the packet arrived on, or the time of day). Based on the rule-set, the packet filter engine will perform an action on the packet, such as forwarding the packet to the destination network.

Packet filters usually run on the same host as the router process. This makes filtering transparent to all users, who address their destinations in the standard way. When packets are leaving or entering a private network, the router will pass the packet to a packet filter process or will run it through an integrated packet filter. If the packet is accepted, it is forwarded to the proper network. If the packet is deemed illegal, it is most often simply dropped.

Routers introduced basic packet filtering early, but in a very simple form, and with a limited number of packet properties available for filtering decisions. When it was realised that packet filtering could be effectively used as a security measure, more properties became available (5). However, the formulation of rules that make up the filtering criteria, is often difficult for humans to comprehend. Flexible languages and user friendly interfaces for specification of filtering criteria have therefore been proposed (15).

Packet filters implement a complex functionality which is best described by examining individual properties. The following sections will give a fairly detailed overview of such possible properties.

2.1.1 Header Inspection

Header inspection is the primary source of information for most packet filters. By inspection of protocol information contained in the network and transport layer headers, basic information about the connection can be derived. The source and destination address is available in the network level header. The source and destination port, often representing the service that is requested, is available in the transport level header. Flags that can effectively be used for filtering are present at both layers, such as the TCP SYN flag, which is set when a new connection is about to be established; and the TCP MF flag, which is set when packets are fragmented and more fragments are iminent.

2.1.2 Application Inspection

It is often difficult for a packet filter to have a complete understanding of the packet flow only by looking at the headers. One example is the FTP service where a transfer connection is established and the port number for this connection is communicated over the application layer of an existing control connection. Packet filters could therefore benefit from scanning the payload of the packets received. Such scanning can also be more advanced, examining payloads spanning several buffered packets and applying, for example, URL filtering or virus signature scanning.

2.1.3 Stateful Filtering

The simplest packet filters make decisions based on each packet individually. This is sometimes difficult, for example with IP fragmentation, where port numbers are included only in the first packet, or with the connectionless UDP protocol. To combat these problems, packet filters can keep state tables that track packets and connections. This will allow the filter to take into account what has happened before, e.g. a connection request was sent out.

2.1.4 Dynamic Filtering

While most packet filters use filtering rules that are static, modern filters can change filtering rules based on input to the filter. This is very useful, for example, when FTP is used, where a connection from the outside to the inside on an ephemeral port can be allowed based on previous negotiation on the standard FTP port.

2.1.5 Buffering Strategy

When a packet filter receives a packet, it will buffer the packet, then run it in the filtering engine applying rules and make a decision, which, in turn, leads to one or more actions. More advanced packet filters could use more advanced buffering, and we can imagine two such strategies: *Buffer and release*, and *reflection*.

Buffer and release will buffer packets per connection, to enable the filter to scan the payload of several packets belonging to the same stream, and release them when it has been verified that they are admissible.

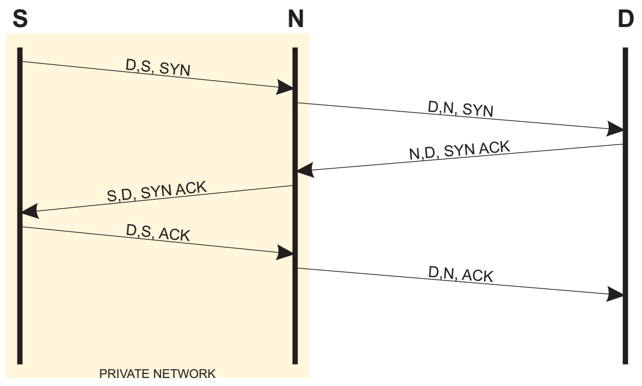

Figure 2.2: Source Network Address Translation. When a source initiates a connection to an outside host, NAT will exchange the source IP address. On the way back, the connection will be mapped by NAT to the originating host. This creates a routing function at the IP layer.

Reflection could be either internal buffer reflection which will copy packets to an internal buffer for processing, or an external reflection technique where packets will be copied and sent to another host. After reflection, the packets will be processed independently of the actual packet flow. If an anomaly is found, some action, e.g. termination, could be performed on the already established connection.

2.1.6 Actions

After a packet filter has applied a set of rules to the inputs, one or more actions will be taken. Basic packet filters will either stop or forward the packet. More advanced packet filters can provide a wider range of additional actions such as rejection, which in addition to stopping the packet will send a message to the source saying the destination is unreachable; reflection, which will copy the packet in addition to forwarding or stopping it; logging, which provides the ability to log various types of events; or modification, which will edit the content of the packet to allow e.g. IP options to be removed.

2.2 Network Address Translation (NAT)

While packet filters control the flow of packets between two networks, there is still direct communication between clients and servers. The addresses of the internal servers are publicly available in order for outside clients to contact the right server. A less revealing approach would be to have one or more publicly available addresses that were mapped to internal services.

There are generally two types of NAT: source NAT and destination NAT. Source NAT is used when internal clients are accessing the outside network. A connection setup using source NAT is shown in Figure 2.2. When leaving the internal network, the packets pass through NAT, and the source address is changed to the address of the host running NAT. The NAT host will then create a mapping between outgoing and incoming address and port.

When the external server responds, NAT checks the mapping and changes the destination address to the address of the requesting client. This way, the client's IP address is never seen outside the internal network.

Destination NAT is used when an external host contacts a specific IP/port in order to reach some service. The NAT service will then, based on some criteria, change the destination address to the appropriate internal server, and send the packet out on the internal network. In selecting the appropriate server, the criteria can simply be the type of service required, but can also take into account features such as load balancing.

One benefit of using NAT is that the private network address space can be hidden from outside hosts. This enables an organisation to use addresses from the private range (17) on the internal network and have a limited set of public IP addresses for use when communicating between the private and the public network, reducing the need for public IP addresses considerably.

Another advantage of NAT is that it causes minimal communication interference. There is only one TCP connection, set up and controlled by the source and destination without interference from the NAT host. NAT only serves as an additional node on the path between the source and the destination, changing the routing by substituting IP addresses.

2.3 Proxies

A proxy is used to avoid direct communication between hosts on a private network and hosts on an outside network. When a client requests communication with a host outside its private network, the proxy will act on behalf of the client towards the external server. This way, all traffic passes through the proxy, which relays data between the connections.

When a connection is initiated from inside the private network, a host contacts the proxy and a TCP connection will be established between the source host and the proxy using the TCP connection sequence (Figure 2.3). Upon completion of the setup of this connection, the source will inform the proxy of its intended destination. The proxy will proceed by establishing a separate TCP connection between the destination and itself. When this connection is also complete, the proxy will relay data from one connection to the other.

In order for this approach to work, the client software has to be proxy aware. Instead of contacting the destination server, the source must contact the proxy and inform the proxy of the destination address.

Proxies are not only used for security purposes. One very common application is a web caching proxy. While this type of proxy can greatly enchance network utilisation by locally caching recently used data, this document will only consider proxies used for security purposes (the interested reader will find many references to web cache proxying in (8)).

If a proxy is contacted from the outside with a request, a TCP connection will be set up between the requesting party and the proxy. The proxy will then look at the request and set up another TCP connection with the appropriate server on the inside. The selection of the appropriate server can be based purely on service, but can also include such features as load balancing.

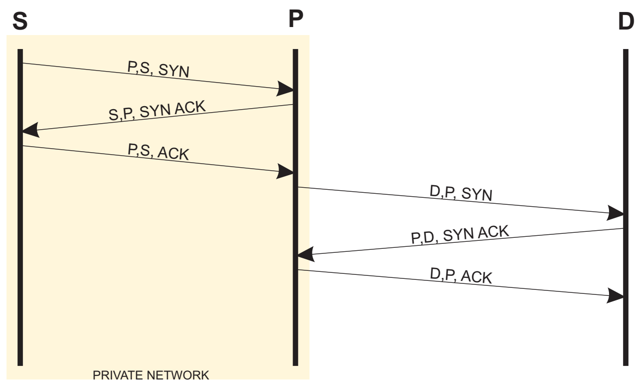

Figure 2.3: Conncetion setup with proxy. When a proxy is used, the initiating client will contact the proxy, who will set up a TCP connection with the requesting client. Upon completion of the setup of this connection, the proxy will set up another TCP connection to the requested destination. When both connections are set up, the proxy will simply relay packets between the two connections.

When the connection is initiated from the outside, the proxy is transparent to all parties. The requesting client on the outside will believe that the proxy is in fact the server. Servers on the inside will receive requests from the proxy, which is then the requesting party. In other words, both parties believe that the proxy is the endpoint of their connection, thus no proxy aware software is necessary.

Two main types of proxies exist: Generic proxies and dedicated proxies. Generic proxies operate on TCP data, and can be used for most applications. Dedicated proxies are specific for each application protocol, so that a separate dedicated proxy must exist for each application protocol that is to be checked. Both types of proxies generally require that hosts inside the private network direct service requests to the outside to the proxy, i.e. the client software has to be proxy-aware. To overcome this limitation, both generic and dedicated proxies can be made transparent as described later.

2.3.1 Generic Proxy

Generic proxies terminate outgoing connections at the proxy, and establish a new connection from the proxy to the outside destination. The private address space is therefore hidden from outside networks, achieving the same benefits as running source NAT. Another benefit with generic proxies is that during the setup phase of the TCP connections, added processing can be done. An example is when a client requests a service, the proxy can require proper authentication of the client before the outside connection is set up. Furthermore, co-operating proxies could perform mutual authentication of each other, and set up an encrypted link between two networks. The flexibility of setting up two separate connections is one of the greatest advantages of a generic proxy.

2.3.2 Dedicated Proxy

While a generic proxy only decouples the TCP connection between client and server, a dedicated proxy will have an understanding of the protocol that is used for that service. After the TCP connections are set up in the same way as for the generic proxy, a dedicated proxy will process the data and can, among other things, verify that the correct protocol is being used.

A dedicated proxy has the ability to interpret data at the application level. An HTTP proxy understands the HTTP protocol, and can for example verify that the protocol is being used correctly, log all accesses to URLs in Sweden, inspect embedded JAVA code or disable JAVA scripts from untrusted locations.

The level of protocol interpretation can vary from implementation to implementation. A very simple proxy could verify that the given protocol starts with the appropriate keyword. A more advanced proxy would have a full state-machine to interpret the protocol. Full interpretation of a given protocol would provide powerful logging and content checking possibilities, as well as verifying the protocol use.

2.3.3 Transparent Proxy

Traditional proxies often constitute a problem in that all clients have to be proxy-aware, i.e. modified to connect to the proxy instead of the external destination server. This problem can be avoided by employing a transparent proxy. When transparent proxying is performed, the communication between the two hosts appear to both parties to be a direct link, while in reality they are separated by the proxy.

A transparent proxy will intercept packets and process them before forwarding. Upon interception of a client request originating in the private network, the transparent proxy will set up a TCP connection with the requesting client pretending to be the destination. After this setup is complete, the proxy will set up a TCP connection to the destination. If the proxy is in address hiding mode, the connection will be set up originating from the proxy (Figure 2.4a). If address hiding is unnecessary, the transparent proxy will set up the connection pretending to be the client (Figure 2.4b). Either way, the proxy is fully transparent to both parties.

There is a fine line between the definition of a transparent proxy and an application inspecting packet filter. For the service to be a proxy, two TCP connections have to exist. This is clearly the case for the transparent proxy. For a packet filter using NAT as shown in Figure 2.2, this is not the case. Packet filters can buffer packets and scan the payload across buffered packets, however, timing constraints in the TCP protocol limit the processing time available.

To the best of our knowledge, no monolithic implementation of a transparent proxy currently exists. However, it turns out that it is possible to implement transparent proxy behavior using NAT redirection and a regular proxy in tandem. This approach is used by e.g. the Squid Web Cache system (19) .

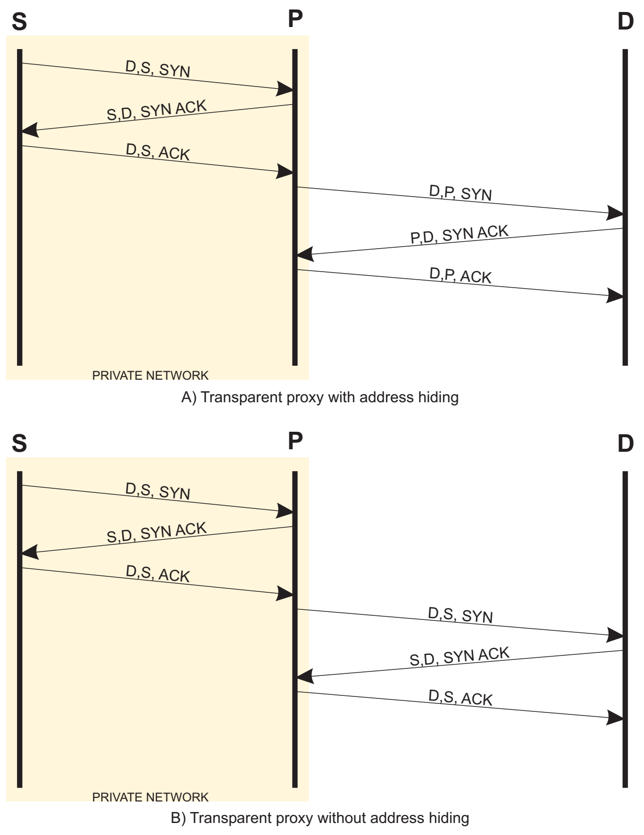

Figure 2.4: Connection setup with a transparent proxy A) with address hiding, and B) without address hiding. In both cases, the proxy is transparent to both client and destination.

2.3.4 UDP Proxies

The concept of a UDP proxy presents a cognitive problem - on one hand, it makes eminent sense to be able to process UDP packets on an application level; on the other hand, since our definition of a proxy requires the establishment of two connections, a connectionless protocol such as UDP has no way of satisfying our requirement!

In fact, version 5 of the SOCKS generic proxy implementation (12) provides the option of creating a relay process for handling UDP datagrams. However, the UDP relay process must be initiated by a specific TCP connection to the firewall, the termination of which also terminates the relay process. This can thus be viewed as a hybrid approach to UDP proxying.

2.4 Other Mechanisms

Modern firewall products also offer other services besides the ones described above, e.g., virus scanning of incoming traffic, Virtual Private Network (VPN) functionality and Intrusion Detection System features. Although these mechanisms can provide valuable services to the users, we do not consider them integral parts of what constitutes a firewall. The fact that these services are offered on the firewall is more a matter of convenience or the firewall vendor “adding value” to its product as a competitive advantage.

3 RESEARCH AND DIRECTIONS

It might seem that the firewall community is about to run out of steam, with more and more effort apparently directed toward other areas like intrusion detection. There is, however, still being done interesting work in the firewall community, which is presented in the following sections.

3.1 Distributed Firewalls

A distributed firewall (3) is a scheme where each host in a sensitive domain is protected by its own software firewall. This removes the binding between the physical topography of the underlying network and the logical definition of a domain (i.e., a host can be a member of a domain regardless of where it is physically located), and also avoids a central firewall as a possible throughput choke point. Furthermore, since there is no longer a central firewall that one can be physically on the “inside” or “outside” of, there is no longer a need for special considerations for mobile users. The crucial component of the Distributed Firewall concept is a centralised control facility that allows the enforcement of a uniform security policy for all the distributed members of the domain. To accomplish this, the control facility needs to employ a policy enforcing language, in addition to a solution that enables policy changes and other domain-internal communication to be transmitted securely.

A prototype implementation of a distributed firewall system is described in (10).

3.2 Domain and Type Enforcement Firewalls

Traditional firewalls have an inside vs. outside view of the world, and are usually more concerned with regulating information flow from the outside to the inside, than vice versa. This has in later years been considered a shortcoming, and although there have been various Multilevel Security (MLS) based products that regulate information flow in both directions on the market, these have suffered from being too costly and inflexible when compared to more readily available commercial-off-the-shelf (COTS) software. As an alternative, (16) presents a firewall design that employs DTE, an access control mechanism described in (2), to regulate the information flow.

The design of DTE has ventured to avoid the drawbacks of MLS systems, retaining compatibility with existing COTS code. The essence of DTE is the administration of access lists defining which groups of users (or roles) should be allowed access to which resources.

With a DTE firewall, it is possible to enforce a fine-grained access control, allowing remote users restricted access to very specific resources. Furthermore, all information transmitted from local users through the firewall can be verified against the DTE policy. As an additional benefit, proxies on the DTE firewall could run with privileges which are restricted by DTE, and thus it is not possible for an intruder, even if a proxy should be subverted, to compromise the entire firewall. The DTE firewall is most versatile when mediating access between hosts that implement DTE, but non-DTE hosts can also participate, with the restriction that a single role is assigned to each host.

3.3 TCP Splicing

Dedicated proxies normally provide a higher level of security than packet filtering due to the added understanding of the application protocol. However, the former are significantly less efficient, with respect to throughput, than the latter. This has led researchers to explore possibilities for improving the efficiency of firewall proxies. (11, 14, 18) have all presented variations over the concept of TCP splicing or cut-through proxies.

As explained previously, a generic proxy will merely copy data from server to client after having performed the initial connection setup. Since the transmitted TCP application data is not being processed in any way at this stage, this amounts to a lot of unnecessary, time-consuming context switching and copying of data. Realising this, a way was sought to hand off the actual copying of data to lower layers in the TCP/IP stack. This process is usually referred to as TCP splicing, as the communications path up and down the protocol stack is cut, and spliced together again at the TCP level.

In its simplest form, a TCP splicing proxy is merely a generic proxy where the application-level copying of data has been replaced with a spliced TCP-connection. However, this principle can also be applied to dedicated proxies. At a certain point of a connection, a dedicated proxy may deem the remainder of a communication to be safe, and at this point splice the connection at a lower protocol level. Ideally, the connection should be unspliced if the communication somehow returns to a state that requires more stringent security measures; this, however, is an issue that remains to be resolved.

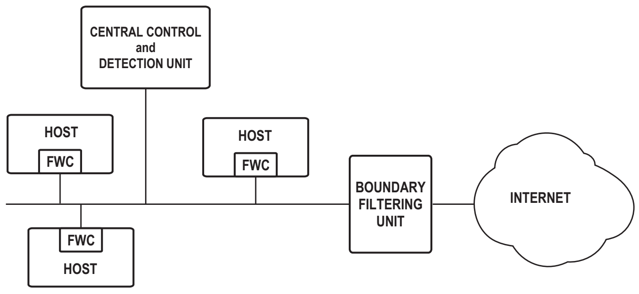

Figure 3.1: Network with distributed firewall components under central control, integrating firewall, IDS, and scrubbing functionality.

An example of where the TCP splicing concept would be applicable, is a dedicated FTP proxy. Specifically, the transport connection for an FTP file transfer operation is an ideal candidate for splicing. When the proxy detects that a file transfer is about to be performed, it will prepare for the transfer connection. When the connection is established, it will be spliced to speed up the transfer.

3.4 Protocol Scrubbing

Ambiguities in protocol implementations often cause different network components to interpret identical traffic differently. With an aim towards ensuring unique interpretation of traffic by all applicable systems, a novel approach to thwarting protocol-level attacks is presented in (13). Observing that many protocol-level attacks succeed due to ambiguities in TCP, IP, and application protocol specifications, security can be enhanced by requiring all protocols to conform to a single interpretation of their respective specifications. The authors claim that for the vast majority of legitimate connections, such a restriction will have no negative impact, since although implementations vary, they all tend to interpret well-behaved flows consistently.

The protocol scrubber is implemented as an in-kernel modification to the protocol stack, typically on a firewall, and the traffic flow will be homogenised to an unambiguous flow. This is primarily done by rejecting traffic that does not adhere to the protocol scrubber's definition of well-behaved traffic.

Another effect of forcing all traffic to become well-behaved is that it simplifies the tasks of a network intrusion detection system (9), but that will not be discussed further here.

3.5 An Example of a Possible Future System

Trends indicate that in the future we will see a merging of security-related technologies and thus a tight integration of Intrusion Detection Systems (IDS), packet filters and proxy-based firewalls.

One likely solution combines many of the elements previously presented in this article, as illustrated in Figure 3.1. We envision a distributed firewall environment, where each host on a private network contains a firewall component. Additionally, a boundary unit placed at the private network's point of connection to the Internet, could contribute towards providing defence in depth and ward off typical Denial of Service attacks like network flooding, apart from homogenising the network traffic by protocol scrubbing. The host firewall components and the boundary unit would be controlled by a central control unit, which could also incorporate an advanced intrusion detection system. The central control unit should maintain and distribute a uniform security policy, and the hosts' firewall components would enforce it. Furthermore, the individual hosts could also contain host-based IDS functionality, which would communicate incidence data to central control unit for collation. The central control unit would adapt and redistribute the security policy as appropriate, based on the IDS results. All communication to and from the central control unit must be cryptographically protected.

The advantages of this scheme are legion. By avoiding a central choke point with dedicated proxies, there is less trade-off between throughput and security. By distributing the intrusion detection capabilities, a more complete picture of the network state can be assembled and a central control unit can update the security policies of connected hosts, adapting to dynamic threats. By retaining a boundary firewall component, the likelihood of successful DoS attacks is reduced.

4 CONCLUSION

With the proliferation of the Internet, securing private networks have become a must for all corporations that care about their integrity. The concept of a firewall has evolved from early perimeter filtering into a melting pot of terms and technologies. This article has presented a concise description of the functions that firewalls can implement, which could serve as a basis for a firewall taxonomy.

New research in the area tries to overcome some of the problems with current firewalls, such as performance penalties and the limited amount of grain in the protection schemes provided. We predict that future implementations of firewalls are integrated solutions providing proxies, protocol scrubbing and intrusion detection, while maintaining centralised control over these components. Such an approach would give a better picture of the actual traffic and events that occur, and could provide better technical security. However, firewall security is only as good as the security policy it implements, and no matter how good the technical solution is, everything depends on good administration.

REFERENCES

(1) F. M. Aviolo and M. J. Ranum (1994): A Network Perimeter with Secure External Access. In *Proceedings of the 2nd Symposium on Network and Distributed System Security*. Internet Society (ISOC).

(2) L. Badger, D. F. Sterne, D. L. Sherman, K. M. Walker and S. A. Haghighat (1995): Practical Domain and Type Enforcement for UNIX. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, pages 66–77.

(3) S. M. Bellovin (1999): Distributed Firewalls. *;login:*, pages 37–39.

(4) S. M. Bellovin and W. R. Cheswick (1994): Network Firewalls. *IEEE Communications Magazine*, pages 50–57.

(5) D. B. Chapman (1992): Network (In)Security Through IP Packet Filtering. In *Proceedings of the Third USENIX UNIX Security Symposium*, Baltimore.

(6) D. B. Chapman and E. D. Zwicky (1995): *Building Internet Firewalls*. O'Reilly and Associates.

(7) W. R. Cheswick (1990): The Design of a Secure Internet Gateway. In *Proceedings of the USENIX Summer '90 Conference*.

(8) B. D. Davison (1999): Simultaneous Proxy Evaluation. In *Proceedings of the 4th International Web Caching Workshop*.

(9) M. Handley, V. Paxson and C. Kreibich (2001): Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. In *Proceedings of the 10th USENIX Security Symposium*.

(10) S. Ioannidis, A. D. Keromytis, S. M. Bellovin and J. M. Smith (2000): Implementing a Distributed Firewall. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*.

(11) R. Knobbe, A. Purtell and S. Schwab (2000): Advanced Security Proxies: An Architecture and Implementation for High-Performance Network Firewalls. *Advanced Security Research Journal - NAI Labs*, pages 15–23.

(12) M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas and L. Jones (1996): SOCKS Protocol Version 5. Network Working Group Request for Comments. RFC 1928.

(13) G. R. Malan, D. Watson, F. Jahanian and P. Howell (2000): Transport and Application Protocol Scrubbing. In *Proceedings of the IEEE INFOCOM 2000 Conference*.

(14) D. Maltz and P. Bhagwat (1998): TCP Splicing for Application Layer Proxy Performance. Technical Report RC 21139, IBM Research Division.

(15) A. Molitor (1995): An Architecture for Advanced Packet Filtering. In *Proceedings of the Fifth USENIX UNIX Security Symposium*, Salt Lake City, Utah.

(16) K. A. Oostendorp, L. Badger, C. D. Vance, W. G. Morrison, M. J. Petkac, D. L. Sherman and D. F. Sterne (1997): Domain and Type Enforcement Firewalls. In *Proceedings of the 13th Annual Computer Security Applications Conference*. Trusted Information Systems, Inc.

(17) Y. Rekhter, R. G. Moskowitz, D. Karrenberg and G. J. d Groot (1994): Address Allocation for Private Internets. Network Working Group Request for Comments. RFC 1597.

(18) O. Spatscheck, J. S. Hansen, J. H. Hartman and L. L. Peterson (2000): Optimizing TCP Forwarder Performance. *IEEE/ACM Transactions on Networking*.

(19) D. Wessels and K. Claffy (1998): ICP and the Squid Web Cache. *IEEE Journal on Selected Areas in Communication*, 16(3):345–357.

This page is intentionally left blank.

APPENDIX

A COMMON TERMINOLOGY

Stateful Inspection A stateful, dynamic, application-inspecting packet filter called Checkpoint Firewall-1.

Circuit-Level Proxy Identical to generic proxy.

Circuit-Level Gateway Network gateway with a generic proxy.

Application-Level Proxy Identical to dedicated proxy.

Application-Level Gateway Network gateway with one or more dedicated proxies.

Plug Gateway Identical to generic proxy.

Transport-level Gateway Identical to Circuit-Level Gateway.

Masqueradeing Source NAT used with dynamicly allocated IP addresses.