Protecting a Private Network: The AltaVista Firewall

J. Mark Smith
Sean G. Doherty
Oliver J. Leahy
Dermot M. Tynan

Connecting an organization's private network to the Internet offers many advantages but also exposes the organization to the threat of an electronic break-in. The AltaVista Firewall 97 for DIGITAL UNIX protects a private network from malicious attack or casual infiltration by screening all internetwork communication. It enforces the organization's network security policy so that only allowed network traffic can cross the firewall. When installed on a dual- or multihomed host, the AltaVista Firewall applies the principle "that which is not expressly permitted is denied" and uses patented technology to screen each IP packet that attempts to cross it. A highly flexible access control grammar and a comprehensive reporting and alarm system enable the AltaVista Firewall to detect and react to harmful or dangerous events. The AltaVista Firewall also includes an HTML-based user interface to ease configuration and management of the firewall.

The advent of electronic commerce as a means of conducting business globally has resulted in an increasing number of organizations connecting their internal private networks to the Internet. Most users of the Internet and the World Wide Web (WWW) view the technologies involved as leading edge, but many are unaware that the foundations on which these technologies are built are quite old.

The Transmission Control Protocol and Internet Protocol (TCP/IP) were first developed in 1979. The primary focus then was to ensure reliable communications between groups of networks connected by computers acting as gateways.[1] At that time, security was not an issue because the size of this Internet was small and most of the users knew each other. The base technologies used to construct this network contained many insecurities, most of which continue to exist today.[2,3]

Due to a number of well-reported attacks on private networks originating from the Internet, security is now a primary concern when an organization connects to the Internet.[4] Organizations need to conduct their business in a secure manner and to protect their data and computing resources from attack. Such needs are heightened as businesses link geographically distant parts of the organization using private networks based on TCP/IP.

An organization implementing a secure network must first develop a network security policy that specifies the organization's security requirements for their Internet connection. A network security policy specifies what connections are allowed between the private and external networks and the actions to take in the event of a security breach. A firewall placed between the private network and the Internet enforces the security policy by controlling what connections can be established between the two networks. All network traffic must pass through the firewall, which ensures that only permitted traffic passes and is itself immune to attack and penetration.

This paper comprises two parts. The first part provides an overview of firewalls, describes why an organization needs a firewall, and reviews the different types of firewalls. The second part focuses on the AltaVista Firewall 97 for the DIGITAL UNIX operating system.

It discusses the product's requirements and describes its architecture. It then describes the important aspects of the product's implementation. Finally, future enhancements for the AltaVista Firewall are discussed.

Firewall Overview

Any organization that connects to the Internet should implement an appropriate mechanism to protect the private network against intrusion from the external network and to control the traffic that passes between the two networks. The mechanism used depends on the value of the asset being protected and the impact of damage or loss to the business involved. Typical reasons for using a firewall to protect a private network include the following:

- To prevent unauthorized external users from accessing computing resources on the internal private network. This is necessary because it is extremely difficult and costly to attempt to secure all the hosts within a private network.
- To control internal user access to the external network to prevent the export of proprietary information.
- To avoid the negative public relations impact of a break-in.
- To provide a dependable and reliable connection to the Internet, so that employees do not implement their own insecure private connections.

A firewall is a device or a collection of devices that secures the connection between a private, trusted network and another network. All the traffic between these two networks must pass through the firewall; this enables the firewall to control the traffic. The firewall permits only authorized traffic to pass between the two networks. The organization's network security policy defines what traffic is authorized. The firewall is immune to attack and penetration and provides a reliable and dependable connection between the two networks. It also provides a single point of presence for the organization when, for example, connecting to a public network such as the Internet.

The fundamental role of a firewall is to provide a control mechanism for the IP traffic between two connected networks. Firewalls provide two types of controls and are categorized either as packet-filtering (packet-screening) or application-level implementations.

Packet-filtering or packet-screening firewalls control whether individual packets are forwarded or denied based on a set of rules.[5] These rules specify the action to take for packets whose header data matches the rule criteria. Typically, a rule specifies source and destination IP addresses and ports and the packet type (for example, TCP and User Datagram Protocol [UDP]). The actions are usually to allow or deny the packet. Packet-filtering firewalls provide a basic level of control over traffic at the IP level. The majority of firewalls of this type are custom-configured routers.

Application-level firewalls disable packet forwarding and provide application gateways (also known as proxies or relays) for each protocol that can cross the firewall. The application gateway relays traffic that crosses the firewall. It can impose protocol-specific and user-specific controls on each connection and can record all operations performed by the user of the connection. This type of gateway therefore allows an organization to control (1) which individuals can establish a connection, (2) when a user can establish a connection, and (3) what operations a user can perform. It also keeps a record of the session for tracking and reporting purposes. Most application-level firewalls are dual- or multi-homed hosts that run a modified operating system and special-purpose software to implement the firewall.

These two approaches to implementing a firewall can be compared, using the following criteria:

- Operating philosophy
- Level of control over connections
- Level of logging and reporting
- Ease of use
- Flexibility
- Ease of administration and configuration
- Private network information made available

Operating Philosophy

The operating philosophy that a firewall implements is the fundamental element of the security of the network connection. For maximum security, firewalls must apply the principle "that which is not expressly permitted is denied." That is, unless the firewall permits a connection, that connection is not allowed. Many packet filters allow any connection that is not expressly prohibited (screend is an important exception).[6,7]

Packet filters are unsuitable for use as a firewall because they require the operator to specify carefully what traffic is allowed and what is denied. Application-level firewalls are specifically designed to implement this philosophy, so that each application gateway allows only those connections specified by its configuration and denies all other connections by default.

Level of Control over Connections

Application-level firewalls allow a significantly greater level of control over who can establish a connection and what operations can be performed over that connection. For example, a File Transfer Protocol (FTP) application gateway, with the assistance of a user authentication system, can identify a user who wishes to establish a connection and can control what FTP operations that user performs. For example, the gateway can permit GET operations and deny PUT operations.

Packet filters can only support host-level control over who can establish a connection, with no restrictions on the individuals who can connect and what operations can be performed once a connection has been established.

Level of Logging and Reporting

Typical packet filters provide basic data logging, and where there is the ability to log traffic, the information available is at the packet level. This makes it difficult to identify and track individual connections when many take place at the same time. No information is available on what operations were performed during a connection. Reporting information is limited to counts of packets passed and dropped and other relatively useless traffic statistics.

Application-level firewalls can log data on each connection. They can identify the individuals who establish connections and what operations they perform. Reports can then list what connections an individual established, what operations were performed, and when they were performed. This facilitates monitoring and maintaining the security of the installation.

Ease of Use

Packet filters tend to be easier to use because they are effectively transparent for those connections that are permitted. Application-level firewalls often require the user to connect first to the firewall host, and then to their destination on the other network. Recently, transparent application gateways have been developed to address this issue.

When support is needed for a new protocol, it is much easier to reconfigure a packet filter than to develop and distribute a new application gateway. For this reason, users behind an application-level firewall may become frustrated when they cannot connect to the latest Internet developments.

Flexibility

Flexibility is important for systems integrators constructing custom network security solutions for large organizations and for those involved in electronic commerce. It is important that the individual firewall components can be configured directly.

Ease of Configuration and Administration

The ease with which an individual can configure and administer a firewall is becoming more important as firewalls are used in organizations that do not possess a high level of technical knowledge. Packet filters are essentially simple but often have a complex rule syntax that makes the task of correctly configuring a packet filter quite difficult. User interfaces could help with the task of configuration, but an expert is usually needed to set up a packet filter properly, because the order of the packet-filtering rules significantly affects the security of the installation.

Application-level firewalls suffer the same disadvantage. The more sophisticated products on the market provide comprehensive user interfaces that guide the user through the task of configuring the firewall, assist in the selection of the appropriate setting for each application gateway, and provide comprehensive firewall management functions.

Private Network Information Made Available

If information about the private network and the hosts that are on it is available to the external network, an attacker may be able to use this information to subvert the system. Packet filters generally do not hide much information from the outside, which increases the risk of a break-in. Application-level firewalls usually appear to be an end node instead of a router to another network; therefore, they can significantly reduce the amount of information that is made available to a potential attacker.

AltaVista Firewall Requirements

This section discusses the functional requirements for firewalls and reviews the product requirements that apply to the AltaVista Firewall.

Functional Requirements

The major functional requirement of a firewall is that it protects a private network from unauthorized external access. A firewall itself must be resistant to subversion and must also ensure that other, less secure hosts within the private network cannot be subverted by an external host.

A firewall must provide a central location for controlling network traffic and for implementing an organization's network security policy for its external network connection. Because many Internet-based services are inherently insecure, a firewall must provide an organization with the means to disable some services and restrict others in accordance with their security policy.

It is also critical that the firewall logs all network traffic, so that a record is retained of all connections established between the private and external networks. Support for the management and retention of network traffic logs is also required to assist tracking of potential and actual break-ins and other security-related activity.

A firewall must be reliable so that users are not inconvenienced by sudden losses of connectivity. As the Internet becomes another means of conducting business, an organization's firewall must ensure that loss of service due to security lapses or other failures is minimized. It must also provide a point of presence for an organization on the Internet.

A firewall must be easy to use. Historically, firewall administrators were required to have in-depth knowledge of Internet protocols; they constructed their firewalls manually. Today's organizations find it difficult to obtain the necessary expertise; they require that their firewall be easy to configure and manage, with-

out sacrificing quality of security and service. Users also require protection without modifying their usage of the Internet. They expect the firewall to protect them without obstructing their work. A firewall must be as transparent as possible to users establishing connections between the private and external networks.

Product Requirements

In addition to firewall products, DIGITAL also offers a custom firewall installation service as part of its network services. The AltaVista Firewall has some of its origins in technology developed for systems integration engineers constructing custom firewall solutions for large customers. These engineers continue to build custom solutions using the AltaVista Firewall to provide the basic firewall technology. The AltaVista Firewall must also be designed to accommodate the needs of integrators delivering custom firewalls.

As described above, there are two types of firewall. Typically, filter-based firewalls perform better. Most detailed product evaluations compare the performance of the products under review, so the AltaVista Firewall must equal or better the performance of its leading competitors.

Most firewalls require that the administrator log on to the host at the console—remote logins are not enabled for security reasons. However, as organizations centralize their network management operations, a key requirement for the AltaVista Firewall is to provide secure remote management functionality.

The AltaVista Firewall must itself be secure, and the host it is installed on must also be secure. It is a product requirement that, while being installed, the product secures the operating system against attack.

AltaVista Firewall Architecture

This section describes the constraints that applied to the design of the AltaVista Firewall. It then introduces the major product components and summarizes the main functions they provide. Then it lists the steps taken to establish a connection through an application gateway.

The following constraints were applied to the design of the AltaVista Firewall:

- In every design decision, the security of the firewall must be the primary concern.
- The basic installation and use of the firewall must be simple and must be guided by a simple user interface.
- All firewall component functionality must be configured using text-based configuration files so that systems integrators can install custom solutions. These files must be consistent across platforms, so that the same set of configuration files can be used on different platforms to configure heterogeneous firewalls.
- The performance of the firewall is an important concern. Application gateways must run as daemons to improve performance and avoid process start-up delays.
- It must be possible for engineers in the field to extend the functionality of the AltaVista Firewall without requiring code changes to the product.

The AltaVista Firewall comprises the following three major components:

- The graphical user interface (GUI) provides the functionality for the user to configure and manage the firewall and manages the configuration files that specify how the firewall will operate.
- The application gateways and support daemons provide the network security specified by the configuration files.
- The kernel is hardened and modified to provide the extra security functionality required for the firewall's operation.

The Graphical User Interface

The GUI is based on a set of hypertext markup language (HTML) template files, computer graphic interface (CGI) scripts, and an HTML browser. It is menu-driven and guides the user through the configuration and management of the firewall. The design of the GUI involved many trade-offs between simplicity of the UI and access to the firewall functionality. We often chose to maintain the simplicity of the UI at the expense of functionality for two reasons:

1. We must protect unskilled installers from installing the firewall in an insecure manner.
2. Skilled installers must still have access to the functionality through the configuration files.

The CGI scripts are written in the C language and use a set of HTML templates to generate the GUI pages. Few static pages are used, so the firewall can be modified in the field to add more application gateways, authentication methods, alarms, and alarm actions.

Because the GUI is HTML-based, it can be ported easily to any platform that supports an HTML browser.

The Kernel

The DIGITAL UNIX kernel has been hardened to protect the firewall host and modified to add functionality required by the firewall. The following modifications were made to the kernel to improve security and to support the operation of the firewall:

- Two mechanisms have been added for protection against routing attacks.

  We added two mechanisms to the kernel to prevent attackers from using routing information to launch an attack through the firewall. The first mechanism

allows the firewall to be configured to ignore requests for it to change its routing tables. These requests come in the form of Internet Control Message Protocol (ICMP) redirect messages,[2] which the firewall ignores. The second mechanism allows the firewall to be configured to drop all IP packets that have source routing options set in the packet header. Source routing can force a packet to take a specified route through the Internet and is not normally used. IP packets with source routing options specified are considered to be evidence of an attack, so it is valid for the firewall to ignore these packets.[8]

- Interface access filtering has been implemented to prevent IP spoofing.

  Interface access filtering provides a mechanism to specify what IP packets are to be accepted on a particular network interface. The source address for each packet is inspected and compared against a filter list. The packet is passed through to the kernel or dropped, depending on the corresponding filter list action. This provides the ability to prevent attackers on the external network from forging IP packets so that they appear to come from trusted hosts on the private network.[8]

- Interface trust group support has been implemented to support packet filtering.

  Trust group support provides a mechanism to specify a color for a network interface. When a packet arrives on a given interface, the kernel marks the packet with the color of the interface. The application gateways and the packet-filtering daemon can use this information to apply filtering rules to allow or deny the packet.

- Transparent proxy support has been implemented to support transparent application gateways. Kernel support for transparent proxying is described in the section on Packet Filtering.

The Firewall

The main functionality of the firewall is provided by the application gateways that provide connection services to users on each side of the firewall. A number of daemons also implement common services to support the operation of the application gateways. These daemons are described here:

- screend: The packet-screening daemon provides packet filtering and transparent proxying functionality.
- alarmd: The alarm daemon provides logging and alarm functionality to the application gateways and other firewall components.
- authd: The authentication service daemon provides user authentication services to the application gateways.
- dnsd: The name service daemon provides the Domain Name Service (DNS) to the application gateways and to the users of the firewall.
- fwcond: The firewall control daemon monitors the application gateway and support daemons that must run on the firewall and restarts any that stop operating.

In addition to these support daemons, the following statically linked libraries provide common services to the application gateways and the other firewall components:

- The access control list (ACL) library allows or denies access to clients based on the appropriate access control list.
- The firewall logging (fwlog) library provides a uniform logging format for all firewall components. This library also provides the interface between the other firewall components and the alarm daemon.

The main features of the application gateways, support daemons, and libraries are described in more detail below. Figure 1 and the following steps show how these components interact when a network connection is established through an application gateway.

1. The application gateway receives the connection request.
2. The application gateway sends requests to the name service (dnsd) to get the host name of the client and the host name or IP address of the server.
3. When the application gateway gets all the information available to it from the name service, it sends a request to the ACL system to ask whether to allow the connection. At this stage, the gateway does not have an authenticated name for the user attempting to connect, so it asks the ACL system whether unknown users are allowed to make the requested connection.
4. If the connection is allowed, the application gateway makes a connection with the server. Data can now flow between the client and the server through the firewall. The gateway also generates a log message for the connection.
5. If the connection is denied by the ACL system, it is still possible that the connection is allowed for certain users, so the application gateway prompts the user for a user identifier so that it can be authenticated.
6. The user specifies an identifier to the application gateway. The gateway sends a request to the authentication service (authd) to authenticate the user and initiates an authentication sequence.
7. The authentication service responds to the application gateway with a challenge for the user. The gateway displays the challenge to the user.

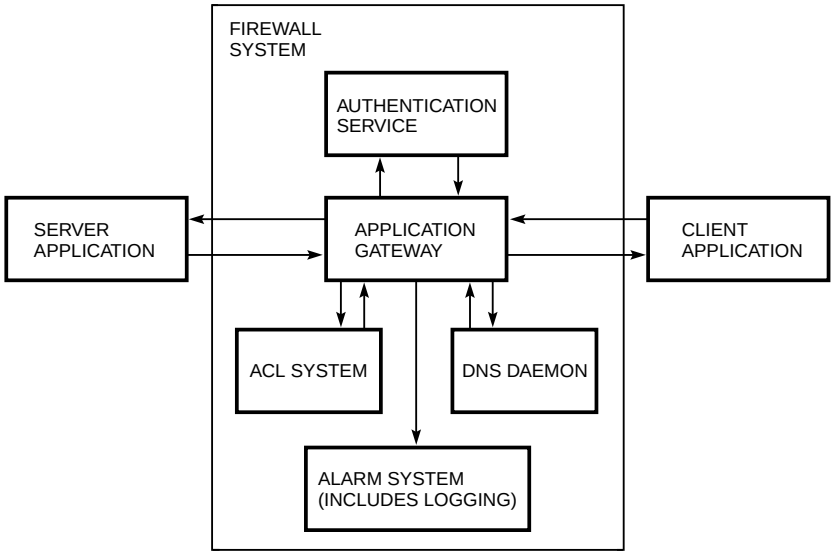

Figure 1
Interaction between Application Gateway, Support Daemons, and Libraries

8. The user uses this challenge to generate a response. The user then enters this response. This is passed by the application gateway back to the authentication service. The authentication service uses the response to authenticate the user and confirms or denies that the user is authenticated to the gateway. If the identifier specified is unknown, the authentication service fakes an authentication sequence and then denies authentication. When authentication is denied, the authentication service logs an event that may result in an alarm being triggered.
9. If the user is authenticated successfully, the application gateway sends another request to the ACL system, this time with the additional information of the authenticated user identifier.
10. If the connection is denied, the application gateway logs an event, which may result in an alarm being triggered. The user can try to authenticate again.
11. If the connection is allowed, the application gateway logs a message specifying that the connection has been accepted and makes a connection with the server. The client and server may now exchange data.
12. When either the client or the server terminates the connection, the application gateway logs two messages: one specifies that the connection has terminated, and another reports statistics such as duration and amount of data exchanged.

Packet Filtering

The AltaVista Firewall for the DIGITAL UNIX operating system provides both application-level and packet-filtering functionality. Packet-filtering functionality is provided by the well-known packet screening daemon, screend,[6,7] which is shipped with DIGITAL UNIX. The AltaVista Firewall replaces the standard DIGITAL UNIX version of screend with a modified version that extends its packet-filtering functionality and provides support for transparent proxying. This section provides an overview of how screend operates as part of the packet routing in a dual- or multi-homed host. It then outlines why a firewall requires transparent proxying and describes how screend is used to implement transparent proxying.

Screend is a daemon that runs in user space. It examines every IP packet that is being forwarded by a firewall and decides whether that packet should be forwarded or dropped. Screend bases its decision on a set of filter rules specified in a configuration file that it reads at start-up. When an IP packet is received by a DIGITAL UNIX host, the first check that the DIGITAL UNIX kernel performs is whether or not the packet is destined for this host. If the packet is destined for another host, then most hosts discard the packet. However, a DIGITAL UNIX host can be configured (by using iprsetup to switch on ipforwarding) to route packets that are destined for other hosts. This is useful when hosts that are not on the same physical network need to communicate. In this case, a number of routers must exist between the communicating hosts that can pass packets from one physical network to another. These routers have at least two network interfaces, as shown in Figure 2. The router can then be configured to pass packets between the networks on each of its interfaces. In Figure 2, for example, if the

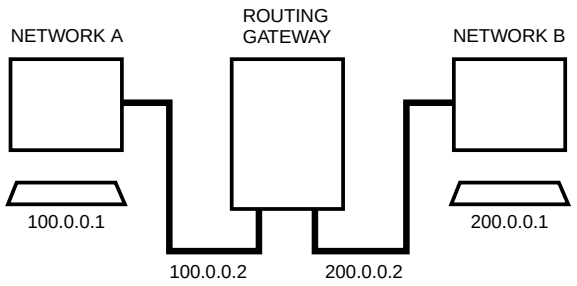

Figure 2
Routing Packets between Networks

host on network A with address 100.0.0.1 wants to send a packet to a host on network B, the packet is initially sent to the routing host. The router receives the packet on interface 100.0.0.2. The routing software in the kernel then sends the packet to the host on network B.

When a DIGITAL UNIX machine with ipforwarding switched on receives a packet that is not destined for itself the packet is passed to a forwarding module in the kernel. This module decides whether the packet can be forwarded and what network interface to send it to. When screend is running on the host, a third operation is inserted after the host has decided that the packet is not for itself and before the packet is sent to the forwarding module. This intermediate function sends the packet to the screend process, which decides whether the host is allowed to forward the packet, based on its set of rules. If screend rejects the packet, it is discarded. If screend accepts the packet, it is sent to the forwarding module as normal.

The AltaVista Firewall is primarily an application gateway firewall. This means that all hosts interacting with the firewall route packets to the application gateways on the firewall. The application gateway then opens a connection to the remote service if the requested connection is allowed. Because the firewall never has to forward packets, IP forwarding could be switched off at the firewall, and screend need not run. If a user attempts to connect to a server on the other side of a firewall, however, the user must understand that the application gateway is present. Instead of connecting directly to the server wanted, the user must first connect to the firewall and then request an application gateway to establish a connection to the remote server. This can be especially awkward with some GUI-based clients.

Transparent proxying was developed to overcome this problem. When an application gateway that supports transparent proxying is running on the firewall, the user can make what appears to be a direct connection to the remote server. In reality, the connection from the client machine is routed through the firewall, which intercepts the connection and redirects the packet to the appropriate gateway. Then, if the gateway allows the connection, it builds a new connection between the firewall and the server. From the user's perspective, the firewall is transparent and has played no part in the establishment of the connection. However, the gateway processes all the user's requests. When a connection is made by a transparent gateway between a client and a server, two TCP connections exist. The first is between the client machine and the firewall. The client host address and the server host address are the source and destination addresses of packets on this connection. The second TCP connection is between the firewall and the server host. The firewall and server host address are the source and destination addresses of packets on this connection.

The AltaVista Firewall uses screend to implement transparent proxying. To do this, changes were made to the DIGITAL UNIX kernel and to the screend application. The most significant change to screend was to add a third option to how screend deals with a packet. Although the standard screend implementation can only accept or reject packets, the implementation of screend that is shipped with the AltaVista Firewall can also proxy a packet. In this case, the packet is not passed on to the IP forwarding module in the DIGITAL UNIX kernel on the firewall. Instead, the kernel sets a flag in the packet to indicate that it is being proxied, and the packet is sent back into the IP input module. When the IP input module detects that the packet is being proxied, it treats the packet as if it were addressed to itself, and passes the packet to its own TCP input module. This then passes the packet on to the application gateway that is listening on the appropriate port. The application gateway determines what the intended destination for the packet is and, subject to the access control specified for the gateway, creates a new connection to the requested server.

Screend imposes an overhead on packet forwarding. Most of this overhead occurs because of the need for a system call from screend to deal with every packet traversing the AltaVista Firewall. The designers of the AltaVista Firewall decided that this overhead would seriously degrade performance, and a cache for screend was implemented in the DIGITAL UNIX kernel. This cache maintains the ways in which screend deals with most open connections, so that only new connections suffer the overhead of a system call from screend.

Screend can also be used as a packet filter for connections to pass through the firewall at the IP level. Packet-screening routers are not as safe as application gateways, so the AltaVista Firewall does not use screend to filter packets through the firewall and does not provide a feature in the user interface to configure screend. Experienced firewall administrators, however, can configure screend to allow IP-level connections to pass across the firewall when they need to support protocols for which application gateways are not available.

Application Gateways

As previously stated, the AltaVista Firewall is primarily an application gateway firewall. As the name implies, application gateways operate in user space at the application layer of the open system interconnection (OSI) model, controlling the traffic between directly connected networks. A separate gateway listens on the appropriate TCP / UDP port on the firewall for each protocol that the firewall relays. This approach provides a high level of control over all major TCP / IP services and allows extensive logging, neither of which packet-filtering techniques can provide. For example, it is possible to allow only authenticated users to copy files, using FTP, out of a private network using the FTP PUT command, while also allowing anyone to copy files from external servers into the private network using the FTP GET command. This high level of control is also apparent in the log files, which show the source and destination addresses (both in IP format and as a fully qualified domain name [FQDN]), as well as the commands executed, names of files transferred, and the number of bytes transferred in each direction across the firewall.

By default, the AltaVista Firewall is configured without any screend packet filter rules that might allow network traffic to cross the firewall at the IP level. Therefore, traffic traveling in both directions must be directed to the firewall where it will be relayed by the appropriate application gateway. This highlights another significant advantage of application-level firewalls—the ability to perform network address hiding. This allows customers to use RFC 1918 (Address Allocation for Private Internets)[9] addresses on their internal network, since the gateway hides the IP addresses of the hosts on the private network inside the firewall.

Historically, application-level firewalls performed poorly, because a new process was forked to handle each connection. For FTP and Telnet, this is not a serious issue; however, with Web traffic, a single URL request may result in numerous other requests for in-line images. The overhead involved in forking a new process for each connection is not acceptable. The application gateways used in the AltaVista Firewall have been designed to address this limitation, without sacrificing security.

Each application gateway is implemented using a single process to handle all connections. To process each connection, the gateway multiplexes between open I / O file descriptors using the select() system call, performing nonblocking reads and writes, and buffering all data until it can be passed on to the client or server. This nonblocking functionality allows the gateway to handle other connections without having to wait for the client / server to process the data already sent to it, or to wait for the name service daemon or the authentication service daemon to return with a response.

This requires that each of the support daemons, such as the alarm daemon, the name service daemon, and the authentication service daemon, must also process requests from the application gateways without blocking. This design resulted in significant improvements in performance and in memory usage. Performance has also been improved by caching name service lookups in the name service daemon. Independent tests have demonstrated that the AltaVista Firewall 97 performs better than packet-filter firewalls, even at Fast Ethernet speeds.[10]

The AltaVista Firewall currently provides the following application-level gateways:

- WWW (including Hypertext Transfer Protocol (HTTP), gopher, FTP, and secure socket layer (SSL) forwarding)
- Simple Mail Transfer Protocol (SMTP)
- FTP
- Telnet
- RealAudio / RealVideo
- SQL*Net
- Finger
- Generic TCP
- Generic UDP

Web traffic comprises the majority of activity on a typical firewall. The WWW application gateway includes specific functionality for

- Blocking Java class files to protect users
- URL filtering to deny access to certain Web sites
- Caching Web documents to improve performance

Alarm System

The AltaVista Firewall uses a dedicated alarm system to monitor the security of the firewall and to respond to attempts to circumvent the security of the firewall. It can also respond to less serious anomalies such as incorrect passwords. The alarm system is implemented by the alarm daemon alarmd, which monitors events generated by application gateways or other servers on the firewall. These are communicated to alarmd by means of log messages.

The alarm system associates one or more alarms with each event. Each alarm includes one or more actions to take when the event occurs. The alarm that is triggered when an event occurs depends on the state of the firewall, that is, the current level of security awareness of the firewall. The state of the firewall is represented using the colors green, yellow, orange, and red. Each color represents a different security awareness level, ranging from under no threat to under serious threat.

The following list describes each state and the level of awareness associated with the state.

- Green: The firewall has not detected any events, and all appears normal.
- Yellow: The firewall has detected one or more events that may indicate a malicious attempt to compromise the firewall or the private network. If no further event occurs in the next two hours, the firewall returns to the green state.
- Orange: The firewall has detected events that are construed as malicious. Events that cannot be created by harmless or accidental operation cause escalation to this state. For example, if the DEBUG command appears during a mail (SMTP) session, the firewall cannot revert from the orange state to the yellow state; the firewall administrator must explicitly set the state back to a lower level once the threat has been analyzed and appropriate action taken.
- Red: The firewall has detected events that may result in a breach of the security of the firewall or of the private network if not addressed immediately. When an escalation to this state occurs, the interface to the external interface is usually disabled immediately, preventing any further IP traffic from that network.

The current firewall state is constantly displayed in the background of the GUI and on the console monitor. The alarm system can be configured manually or using the GUI and allows the firewall administrator to specify the alarm to be triggered for each event occurring at one or more firewall states. Each alarm specification includes one or more actions. For example, an alarm can advance the firewall state, shut down an application gateway or the firewall, and notify the firewall administrator. The administrator can write shell scripts for additional specific actions to take. This can enable the firewall to actively counter a threat to its security or that of the private network.

Authentication Service

When users log on to most computer systems, they specify a password to prove their identity to the system. User authentication is the process by which a computer system verifies the identity of a user or entity through a unique password.

The authentication service on the firewall provides a mechanism by which the identity of a user can be verified. This allows organizations to implement security policies that specify that only certain users are allowed to use particular services to access resources through the firewall. When users wish to establish a connection, they must first identify themselves so that the firewall can associate them (via their identifiers) with the connection. This identifier is then presented by the application gateway to the ACL system, which then decides if the connection request will be allowed or denied.

Although users logging directly into a computer system or an internal network run certain risks of having their passwords discovered by another person, most organizations are happy to use what are termed reusable passwords to protect access to computer resources within a private network. However, when a user wishes to gain access from an external network through the firewall to an internal network, the use of reusable passwords presents an unacceptable risk due to the availability of IP packet sniffers. A packet sniffer is a tool that allows an attacker to read the data in an IP packet and identify a user's identifier and password without the user becoming aware that this has happened. It is clear that a stronger form of user authentication is required for use with a firewall that controls access between two networks.

Several authentication mechanisms have been devised that are based on the concept of a one-time password. In the one-time password scheme, a particular user has a mechanism to generate a new password each time he or she is requested to respond with a password, and the authentication system has a similar mechanism to validate the one-time password as correct for that user. Most one-time password systems are based on a shared secret that is used in conjunction with some software or a hardware device (commonly known as a handheld authenticator [HHA]). Typically, the system challenges a user, often with a value. The user then uses an HHA to generate a response. The authentication system receives the response, and determines if it is the one expected from the user. Further attempts to authenticate involve a different challenge and a correspondingly different response.

The AltaVista Firewall supports a wide range of user authentication mechanisms and can be easily modified to support additional mechanisms if necessary. The authentication mechanism used to authenticate a user can be different depending on whether the access requested is inbound or outbound. This allows an organization to apply less stringent controls to accesses from the internal network. Users are registered with the authentication system through the GUI, and their details are stored in a database. This database includes the inbound and outbound authentication mechanisms that apply to the user and an expiration date for the record.

The authentication service is implemented using an authentication service daemon (authd) and a set of authentication server daemons. Authd runs continuously on the firewall and accepts connections concurrently from application gateways requesting authentication of users. The authentication server daemons implement support for each authentication mechanism. Each of the authentication servers implements a challenge-response authentication service and is started by inetd when a connection is made to the port on which the service is provided.

When an application gateway on the firewall requires a user to supply an identifier, a sequence of actions occurs as follows:

1. The gateway connects to authd and specifies the identifier for the user.
2. Authd accesses the user database and, after checking that the user is registered and that the user record has not expired, determines what mechanism to use to authenticate the user.
3. Authd contacts the relevant authentication server and passes the user identifier to it.
4. The authentication server generates a challenge for the user. Authd passes this challenge to the user.
5. The user generates a response, which the gateway provides to the authentication server, and validation proceeds.
6. If validation fails, the gateway is informed and access is denied. If validation succeeds, the user may proceed.

In the event a request is received to validate a user who is not registered or whose user record has expired, authd will fake a user authentication sequence. This additional security measure ensures that a potential attacker cannot determine if the specified user identifier is valid.

Adding a new authentication mechanism is simple, because configuration files specify the port to contact for each type of authentication mechanism, and a common authentication protocol governs the interaction between authd and the authentication servers.

The authentication protocol used between authd and the authentication servers is modeled on the FTP protocol but is much simplified. All user authentication sequences involve an initial step in which the client (authd) specifies the user's identifier to the authentication server, followed by one or more challenge-response cycles in which the server requests a reply from the client (by challenging the server for some information), and the client responds. The server processes each response and will either issue a further challenge to the client or issue a result indicating if the user is authenticated or not.

The authentication service supports the following authentication mechanisms:

- SecureNet Key
- CRYPTOCard
- WatchWord key
- SecurID card
- S/Key
- Reusable passwords (for outbound connections only)

The Name Service

Unlike the traditional model of the Internet in which all DNS[11] information is available to all hosts, certain organizations may decide to prevent external hosts from viewing information about internal hosts. Restricting this type of information allows an organization to prevent a would-be attacker from gaining a foothold to launch an attempt to subvert the security of a private network. For example, an attacker may support a claim to be an employee of an organization by showing knowledge of the network structure, or a recruitment agency may identify personnel engaged in development activities.

Traditionally, hiding information about internal hosts required providing two separate name servers. An internal name server ran on a server behind the firewall and handled internal DNS queries. The firewall ran the second name server and handled DNS queries from external hosts. In this way, the complete name service database was available to internal hosts, and the restricted database (which usually contains records for the firewall itself and any external WWW or FTP servers) was available to external hosts. The drawback of this approach is that a second host was required to run the internal name service.

If an organization is not concerned that its internal name service is available to the outside world, the firewall can act as a name server for both internal and external queries. This approach raises some questions regarding how information is delivered. For example, the mail exchange (MX) records specified for internal hosts will be incorrect for hosts on the external network. If the proper hierarchy is shown for mail delivery (that is, the internal host has the lowest MX priority, followed by the local mail hub, then the main corporate mail hub, and then the firewall itself), external hosts will attempt to connect unsuccessfully to the three internal hosts before eventually sending the mail to the firewall. Although this does not consume considerable bandwidth, it does cause a noticeable delay in mail delivery to hosts on the private network.

The AltaVista Firewall name service is implemented using a name service daemon (dnsd). Dnsd acts as a gateway accepting DNS queries from both the private and external networks. It also accepts DNS queries directly from application gateways and supports daemons running on the firewall. Two name servers based on the standard Berkeley name daemon (bind) also run on the firewall in a protected environment. One acts as an internal name server containing information concerning the full database of internal hosts; the other acts as an external name server containing only selected information. Both name servers are configured to accept requests from dnsd only.

The name service daemon dnsd considers both the origin and the type of each DNS query it receives. DNS queries that originate on the private network are treated differently from queries that originate on the external network. Requests received by the firewall are forwarded to one or the other of the name daemons, based on the form and origin of the question.

Table 1 shows the relationship between queries originating on the external and internal networks, as well as queries by gateways and other daemons running on the firewall. An authoritative query is a query that concerns a host within the domain of the firewall. Queries from external sites, for domains other than those for which the firewall is authoritative, are rejected by default, though this behavior can be disabled, allowing such queries to be forwarded to the external name server.

The main advantage of this dual-DNS system is that it removes the requirement for a second host to run the internal name service. It also allows precise control over the dissemination of name service information to hosts on the external network. The AltaVista Firewall GUI allows an administrator to enter a host name and specify whether or not that particular host name is visible to the external network as well as to the internal networks. Previously, it was possible to specify only whether or not all internal hosts were visible, but the AltaVista Firewall now allows this decision to be made for each host. The MX records are correctly generated so that external hosts do not see MX records for hosts or mail hubs on the private network. Similarly, if the installation includes an external mail hub, this hub will not hold an MX record for internal machines when viewed from the internal network.

The firewall does not permit zone transfers to external secondary name servers unless they appear in a firewall list of secondaries. A zone transfer involves transferring the full information for a domain from the primary domain server to one or more secondary servers. Any external secondary name server that is authorized to receive zone transfers can still only see the domain information as it exists from the perspective of the external network. This means that hosts on the private network that cannot be resolved from the external network will not be included in any zone transfer initiated from the external network. Zone transfers from secondary name servers on the internal network do not need to be authorized.

Access Control System

Access control is a core function of a firewall. The access control system in the AltaVista Firewall provides a powerful, flexible, and secure means for administrators to define who can use the application gateways on their firewall. The ACL system is implemented using a rule definition language, a library of functions used by all the application gateways to load and interpret the rules, and a comprehensive user interface to allow firewall administrators to exploit most of the power of the language.

The architecture of the ACL system is very simple. It is implemented as a static library. This static library is linked to an application gateway or any server that controls access. The API to the library is minimal; it consists of a function to load a rule set from an ACL file and a function that takes the details of a user's request as parameters and returns a decision to deny or allow the action based on the rule set. The AltaVista Firewall has been designed so that each application gateway or server that uses the ACL system has a separate rule file. However, the architecture of the ACL system does not require this, and ACL file sharing is possible.

In the rest of this section, we describe the requirements for the ACL system and how they were addressed.

- The access control system must be reliable. Since access control is a core function of a firewall, the design team considered the reliability of the ACL system as primary. The ACL system was designed as a separate library that is statically linked to the application gateways. Static linking is used to ensure that this component cannot be easily replaced by an agent attempting to subvert a gateway. Implementation as a separate component means that all gateways and other servers that require access control use a common service, and that this component can be tested thoroughly and independently. This approach also had the advantage of allowing more project resources to be allocated to the design, implementation, and testing of the ACL system.
- It was clear that a powerful language was required to define a wide range of security policies. A firewall administrator must be able to grant and deny access to individual users, to hosts, and to groups of users and hosts. The administrator must also be able to specify times at which access is allowed.
- Because the access control language was extremely flexible, it was considered that policies must be fail-safe and tend to deny rather than allow access. Several features of the language implement this requirement:
  - An implicit default rule states "If there is no rule granting access to a request, then that request is denied." This default rule cannot be altered.
  - The order of rules in the ACL file is not important. If there is a conflict between rules, a deny rule takes precedence.

Table 1
Handling DNS Queries

	Queries for which the firewall is authoritative	Queries for which the firewall is not authoritative
Queries from internal network	Internal name server	External name server
Queries from external network	External name server	Reject the query

  - A blacklist rule has been defined. This is a simple statement that takes a list of host names or addresses. If a host appears in a blacklist statement, then no user can access that host using the firewall, and no user can access the firewall from the blacklisted host.
  - Time-based rules can be specified to deny or allow access for particular periods during a day or week.
  - A GUI interface to the ACL system provides an interface to most features of the system without requiring the firewall administrator to generate the ACL files manually.
- The interface to the ACL system is very simple. The API consists of just two functions that an application gateway must call. The first function loads the rule set that the gateway will use, and the second function takes the details of a request a user is making as parameters and returns a deny or allow decision.
- The ACL system is also well integrated with the other systems in the firewall. For example, the ACL system performs its own logging of each user request and the resulting decision to deny or allow access.

The ACL grammar is a simple rule-based language that supports the definition of a list of deny and allow rules for application gateways. Each application reads an ACL file to load the rule set it uses to make access control decisions.

The core of the ACL system is the algorithm used to allow or deny access through the gateway. Each time the user makes a request to an application gateway, the gateway builds a data structure describing the user and the requested action and makes a call to the ACL system. The data passed to the ACL system includes the following information:

- The canonical form of the FQDN of the host from which the user is making the call.
- The IP address of the host from which the user is connecting.
- The user identifier, if the user has been authenticated.
- The action the user is requesting.
- The server that the user wishes to access. Note that this could be the firewall itself.

The ACL system uses the following algorithm to decide whether to grant or deny access to a user request.

- Search through all the time-based rules. If any time rule denies access to the requested operation, log that a request was denied because of a time rule and set a flag to indicate that the request was denied.
- Search through all the blacklist rules. If the source or target host is blacklisted, log that a request was denied because of a blacklist rule and set a flag to indicate that the request was denied.
- Search through all the deny rules. If access is denied because of an explicit deny rule, log that a request was denied because of a deny rule and set a flag to indicate that the request was denied.
- If the flag to indicate that the request has been denied is set, return 'DENY' to the gateway.
- Search through all the allow rules. If access is granted because of an explicit allow rule, log that a request was granted and return 'ALLOW' to the caller.
- Log that access was denied because no rule matched the request and return 'DENY' to the caller.

Note that denying one request can generate up to three log entries: access can be denied because of a time rule, a blacklist rule, and a deny rule. If all three rules are triggered, they are all logged. The decision to implement logging in this manner was made to ensure that logs were complete and to ensure that all applicable alarms were triggered. Note also that if any deny rule is triggered, the ACL system does not even look at the allow rules. This means that the deny rules take precedence over the allow rules. Also note that if no rule triggers that the request will be denied, the request is denied. This implements the requirement that any request that is not explicitly allowed is denied.

Reporting and Logging

It is a primary function of a firewall to keep an audit trail of all network traffic for both allowed and denied connections. Most authors on the subject of firewalls stress the importance of maintaining comprehensive logs so that break-in attempts can be identified quickly, and the necessary actions can be taken against the attacker. These actions can include contacting the firewall administrator of the host the attacker is using, blacklisting the host the attacker is using, or temporarily shutting down some of the services that the attacker is trying to exploit.

Logging in the AltaVista Firewall is implemented using a common library. This library is statically linked to all the firewall components and provides a uniform logging interface and uniform entries in the firewall's log files. Static linking is used to ensure that this component cannot be easily replaced by an agent attempting to subvert a gateway. Each firewall component initializes the logging function on start-up, passing it an acronym that the library uses to tag all subsequent entries in log files. When a firewall component calls the logging library to write an entry to the log files, it specifies a flag indicating the type of the log message and the log information. Log messages are of the following types:

- FW_LOG. The message is informational.
- FW_WARN. The message is a warning; there may be some security implications. This level usually

indicates that some configuration information expected by the component is not present or cannot be accessed.

- FW_EVENT. This message is a security event. Event messages are detected by the alarm system as described above.

All logged messages include the following information in the log entry written to the log file:

- The current date and time
- The component generating the message
- The log message type
- The log message information

As well as ensuring that all firewall activity is logged, it is also critical that the firewall manages the logs files that are generated to ensure that the log information is retained for later analysis. The AltaVista Firewall automatically archives logs files on a monthly basis and includes features to allow for the retention and/or deletion of log archives after one year. The firewall also monitors log disk space usage and will automatically inform the firewall administrator if the amount of space available falls below a specified size. If the log disk is full, the firewall will disable itself so that no network traffic can occur that is not logged.

The AltaVista Firewall also provides comprehensive traffic-reporting facilities, including an automated reporting service and a GUI-based custom report generator. The firewall's reporting system allows a firewall administrator to choose from a number of report types, including summaries or detailed information, on individual gateways or on all firewall services. Reports can be generated for daily, weekly, and monthly periods. These reports can be mailed automatically to a specified distribution list.

Remote Management

Firewall management has rapidly become a key issue for organizations implementing connections to the Internet, or for organizations required to protect several parts of their internal private network. Typically, firewall products allowed operator access only at the firewall system's console to ensure the security of the installation. Most organizations today prefer to centralize their network control operations and expertise. Firewall products that require console-only access are often not considered. Also, many organizations place their firewalls at separate locations where console access for monitoring and control may be restricted for various reasons. The AltaVista Firewall's remote management provides a mechanism by which a central network management body can control every firewall within an organization.

The key requirements for remote management of a firewall are as follows:

- A secure channel between the firewall and the remote management client
- A consistent user interface for both local and remote management
- Support for multiple administrators with the ability to control the tasks each administrator can perform

Clearly, any firewall remote management capability must be completely secure, offering no possibility of compromise. If an attacker can break into a firewall's remote management channel, then the complete subversion of the firewall is likely.

For ease of management, it is also important that the user interface that is available remotely is the same as the one available locally, so that all firewall monitoring, control, and configuration facilities are available from the remote host.

Finally, the ability to support multiple administrators is required when a firewall is being managed remotely, so that the organization that operates the firewall can control who has access to the firewall and what operations they can perform. Typically, an organization restricts the administrators who can configure or reconfigure the firewall so that changes to the security policy the firewall implements are tightly controlled.

As described previously, the user interface for the AltaVista Firewall is implemented using an HTML-based approach, thus enabling any platform capable of supporting a Web browser to manage the firewall. The key requirement is to secure the connection between the firewall and the remote host the GUI traffic travels over, because all management actions are performed by the GUI Web server running on the firewall. The need to authenticate both the firewall and the remote host is critical to this. Although SSL-based solutions can easily deliver firewall authentication, it is less easy to deliver remote host authentication, without providing an appropriate key generation mechanism for potential clients within the firewall product. A solution is required that provides an encrypted channel for the GUI traffic and includes a mechanism that allows for authentication of both parties.

The AltaVista Tunnel product was selected to provide a secure channel for remote management for two reasons: (1) It delivers a mechanism by which traffic between the host running the AltaVista Firewall and a remote host is secured through encryption, and (2) It provides an easy-to-use mechanism for authentication of the two parties. For remote management, a tunnel is established between the remote host and the firewall. This tunnel provides the secure channel through which the firewall can be managed from a remote host. The firewall acts as a tunnel server (running the

AltaVista Workgroup Edition Tunnel software), while the remote host may be either a tunnel server or a tunnel client (running the AltaVista Personal Edition Tunnel software). In this way, an administrator can manage a firewall from any platform that supports the AltaVista Tunnel. A modified version of the AltaVista Workgroup Edition Tunnel software is shipped with the AltaVista Firewall, as well as AltaVista Personal Edition Tunnel software for Windows 95 and Windows NT platforms. The firewall tunnel software is modified to restrict the number of concurrent tunnels that can be started to one (for licensing reasons) and to operate the tunnel server on a nonstandard port. This avoids clashes with organizations that are relaying tunnels by means of the firewall.

A Web browser running on the firewall connects to the GUI Web server using the local host's address. In contrast, a Web browser running on the remote host connects to the GUI Web server using the IP address of the tunnel endpoint. The endpoint is the pseudo-IP address allocated to the tunnel on the firewall. The GUI Web server on the firewall is configured to allow only connections from the local host (the firewall) and from the tunnel endpoint (that is, the pseudo-IP address) of the remote host. The Web server automatically manages the GUI universal resource locators (URLs) generated for each user interface component, depending on whether the connection originates locally or through a remote management channel. Because the AltaVista Tunnel product adds a route on the host at one end of a tunnel to the pseudo-IP address of the tunnel endpoint on another host, traffic between the browser and GUI Web server is automatically routed through the tunnel and is secure from interception. The GUI Web server rejects any attempt to connect from a host that is not the local host or does not have a remote management channel configured for it.

The implementation of remote management support in the AltaVista Firewall includes GUI functionality that allows a firewall administrator to add, view, and delete remote management channels. When adding a new channel, the administrator specifies the pseudo-IP addresses for both tunnel endpoints, the name of the channel, and the type of tunnel to be used (Workgroup or Personal Edition). The GUI automatically configures the AltaVista Tunnel software, sets up the tunnel configuration required, and generates the necessary key and connection files for the remote host. The administrator is not required to perform any direct AltaVista Tunnel management activities on the firewall.

The implementation of multiple administrator support in the AltaVista Firewall includes GUI functionality that allows an existing administrator to add other administrators, change their GUI login passwords, specify what GUI tasks they can perform, and delete an administrator. Although the GUI restricts administrator logins from a particular source to one at a time, it is possible that separate administrators can log in locally and remotely. Administrators are granted privileges to monitor, control, and configure the firewall for each GUI subsystem. One administrator may be able only to monitor the firewall status, while another administrator may have the necessary privileges to configure the security policies for application gateways or to manage the user authentication system. In addition, GUI privileges are allocated separately for local and remote access, providing further flexibility for administrator privilege control.

Future Enhancements

Large organizations that have local private networks in several geographies currently link these networks using expensive private connections. The growing availability of inexpensive, high-speed Internet connections and the development of secure IP tunneling software products, such as the AltaVista Tunnel, is prompting many of these organizations to construct virtual private networks (VPNs). Since each Internet connection must be secure, the next logical step is to integrate the IP tunneling capability into the AltaVista Firewall.

Larger organizations are moving away from the idea of having one firewall as a single choke-point connection to the Internet. Instead, multiple firewalls may be dispersed at several locations throughout a private network, perhaps on different continents. It will therefore be necessary to ensure that the network security of each firewall remains synchronized with the others. The ability to provide secure enterprise management of several firewalls from a single location is a major challenge for the AltaVista Firewall.

IP multicast technology[12] is now becoming a core component of the Internet and private corporate networks. Multicasting is the ability to distribute data packets to a group of one or more hosts, as opposed to unicasting, which refers to normal point-to-point Internet communications, and broadcasting, which refers to one-to-all communication. IP multicast has enormous potential, most notably in low-cost, real-time conferencing (for example, video and audio), where each host must send data to all other conference participants and other Internet multimedia applications. Multicast datagrams, however, may pose security vulnerabilities to machines that receive them. The AltaVista Firewall must address the need to relay multicast packets while continuing to ensure the security of the private network.

The deployment of IP networks based on the next-generation Internet Protocol Suite, IP version 6 (IPv6),[13] will address many of the structural and security issues that currently exist with IPv4. IPv6 will provide many advantages, including

- Scalability. Rapid growth in the Internet has resulted in the available IP address space being con-

sumed at an alarming rate. IPv6 provides a much larger address space.

- Security. IPv6 addresses authentication, integrity, and confidentiality issues. Because IPv6 corrects many of the threats and vulnerabilities associated with IPv4, the architecture and security policy of an IPv6 firewall will significantly differ from those of an IPv4 firewall. The integration of IPv6 support into the AltaVista Firewall will require researching any outstanding security threats, as well as addressing the practical issues of performance slowed by cryptography.

Summary

The emergence of new technologies and the growth of electronic commerce on the Internet means that network security will continue to increase in importance. The AltaVista Firewall addresses customer requirements for securing their Internet connections by providing a powerful and flexible firewall product that includes application-level and packet-level network traffic control functions, traffic logging, and security monitoring capabilities, together with comprehensive firewall configuration and management support through a GUI.

The AltaVista Firewall 97 for the DIGITAL UNIX operating system is now in its third release since its introduction in September 1995 and has achieved market recognition for its high performance, its comprehensive firewall features, and its ease of use. At the same time, the product provides systems integrators with a comprehensive set of firewall features and functionality to enable them to provide customized network security.

Acknowledgments

We would like to acknowledge the following people for their contributions to the design and development of this product: Frank O'Dwyer, for his contributions to the access control system; Chris Evans and Eugene O'Connor for their extensive user interface and documentation work; Sarah Keating, whose testing contributed to the security and robustness of the product; Tim Shine, engineering manager, who ensured the timely delivery of the product; Brendan Noonan, product manager, whose market knowledge was invaluable; Jeff Mogul, for his advice on packet filtering; Dan Walsh and Ken Alden, for their advice and assistance with the AltaVista Tunnel product; Kevin Carey and Tony Pitt, for their help with beta testing; and the members of the AltaVista Internet Security Product Group.

References

1. W. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols* (Reading, Mass.: Addison Wesley, ISBN 0-201-63346-9, 1994).
2. S. Bellovin, "Security Problems in the TCP/IP Protocol Suite," *Computer Communication Review*, vol. 19, no. 2 (1989): 32–48.
3. R. Morris, *A Weakness in the 4.2 BSD Unix TCP/IP Software* (Murray Hill, N.J.: AT&T Bell Laboratories, February 1985).
4. *CERT Coordination Center 1996 Annual Report* (http://www.cert.org/cert.report.96.html).
5. J. Mogul, R. Rashid, and M. Accetta, "The Packet Filter: An Efficient Mechanism for User Level Network Code," *Proceedings of the 11th Symposium on Operating Systems Principles*, AGM SYSOPS, Austin, Texas (November 1987).
6. J. Mogul, "Simple and Flexible Datagram Access Controls for UNIX-based Gateways" (Digital Equipment Corporation, Western Research Laboratory Research Report, April 1984).
7. J. Mogul, "Using screend to Implement IP/TCP Security Policies" (Digital Equipment Corporation, Network Systems Laboratory Technical Note TN-2, July 1991).
8. CERT Advisory CA-96.21, "TCP SYN Flooding and IP Spoofing Attacks," September 1996.
9. Y. Rekhter, B. Moskowitz, D. Karrenberg, G.T. de Groot, and E. Lear, "Address Allocation for Private Internets," RFC 1918 (February 1996).
10. D. Newman, H. Holzbaur, and K. Bishop, "Lab Tests: Firewalls: Don't Get Burned," *Data Communications* (March 21, 1997) http:www.data.com/cgi-bin/dynamic/lab_tests/firewalls97-extras1.txt.
11. P. Albitz and C. Liu, *DNS and BIND*, 2nd Edition (Sebastopol, Calif.: O'Reilly & Associates, Inc., ISBN 1-56592-236-0, December 1996).
12. IP Multicast Initiative (http://www.ipmulticast.com).
13. S. Deering and R. Hinden, "Internet Protocol Version 6 (IPv6) Specification," RFC 1883 (December 1995).

General References

W. Cheswick and S. Bellovin, *Firewalls and Internet Security, Repelling the Wily Hacker* (Reading Mass.: Addison Wesley, ISBN 0-201-63357-4, 1994).

D. Chapman and E. Zwicky, *Building Internet Firewalls* (Sebastopol, Calif.: O'Reilly & Associates, Inc., ISBN 1-56592-124-0, 1995).

Biographies

Oliver J. Leahy
A principal engineer in the AltaVista Security Products Group, Oliver Leahy has worked in security engineering for several years. He led the POLYCENTER Security Compliance Manager (PSCM) for NetWare project and worked on the development of PSCM for OpenVMS. Prior to this, he worked in the Publishing Technology Group and in the DIGITAL Semiconductor Acquisition Group, developing quality control and semiconductor test software. He holds a B.Sc. (1982) from Trinity College Dublin and an M.Eng. (1988) from the University of Limerick.

Dermot M. Tynan
A principal engineer in the AltaVista Internet Security Product Group, Dermot Tynan is the architect of the AltaVista Firewall product. Prior to joining DIGITAL, he worked as a UNIX kernel engineer for Altos Computer Systems, Unisys, and ICL. He designed software and hardware components for a real-time FFT system for Ultrasystems Space and Defense, Inc. and developed interprocess communication systems for British Telecoms' Work Management System. Dermot is an active member of the Internet Engineering Task Force (IETF), the ISO Motion Picture Experts Group (MPEG), and the Internet Society (ISOC). He is currently working on three Internet draft standards and is coinventor on a firewall patent.

Sean G. Doherty
Sean Doherty joined DIGITAL in 1993 and is a senior software engineer in the AltaVista Internet Security Product Group where he has designed and developed components of the AltaVista Firewall since its inception. Previously, Sean contributed to the design and development of the POLYCENTER Security Compliance Manager for NetWare product. Prior to joining DIGITAL, Sean was responsible for the development of system management software for Third Wave Ltd. Sean received a B.Sc. in computer applications from Dublin City University, Ireland (1992).