A system and method for filtering dynamic application information. A packet is received at a firewall and referred to a dynamic access control proxy. The proxy determines at least one connection parameter set by a dynamic application for data transfer. A rule is selected based upon a detected connection parameter and implemented at the firewall. At the end of the data transfer session, the rule is deleted.
Abstract of the Disclosure

A system and method for filtering dynamic application information. A packet is received at a firewall and referred to a dynamic access control proxy. The proxy determines at least one connection parameter set by a dynamic application for data transfer. A rule is selected based upon a detected connection parameter and implemented at the firewall. At the end of the data transfer session, the rule is deleted.
Access Control for Applications with Dynamic Parameters

Field of the Invention

The field of the invention is information systems access control, and in particular access control for applications with dynamic parameters.

5 Background of the Invention

A firewall regulates the flow of packetized information. A packet includes a header and a payload. The header includes header information (header parameters), such as a source and destination address for the packet, as well as source and destination port numbers and a protocol number. Other examples of header information include various flags (such as indicate the priority with which the packet is to be handled, security features for the packet, error correction and detection features, etc.), a time-to-live parameter, etc. The payload includes the data meant to be conveyed by the packet from its source to its intended destination.

A known firewall is placed between the packet’s source and intended destination, where it intercepts the packet. The known firewall filters a packet based upon the packet’s header parameters and a rule loaded into the firewall. The rule correlates a pattern in the header of a packet with a prescribed action, either PASS or DROP. The filter identifies the rule that applies to the packet based upon information in the packet’s header, and then implements the rule’s prescribed action. When a DROP
action is performed, the packet is not passed through the firewall to its intended
destination. For example, it is deleted. When a PASS action is performed, the packet
is passed on toward its intended destination. The set of rules loaded into a firewall
reflect a security policy, which prescribes what type of information is permissible to
pass through the firewall, e.g., from which source, to which destination, for which
applications, etc.

The set of rules loaded into a known firewall is typically static. The rules must
typically be loaded with the intervention of a system administrator, and any changes to
the rule set (additions, deletions, modifications) must also be implemented by the
administrator. Static rules are used by known firewalls to regulate the flow of packets
to and from applications that operate with essentially fixed, predetermined header
information ("connection parameters.") As used herein, a connection parameter is
defined to be any parameter that is used in establishing communications between
processes executing on two or more computers. Examples of a connection parameter
include any element or combination of elements of header information, such as a port
number, an address, a protocol number, a flag, etc. For example, if a user application
is known to communicate with a destination application through port 45 on any
computer on which the user application is executing, and the destination application is
known to communicate through port 50 on any computer on which the destination
computer is executing, then a static firewall rule (e.g., PASS all packets with a source
port number of 45 and a destination port number of 50) can effectively regulate the flow
of information between the user and destination applications based upon the port
information. Many known applications communicate through the same standardized
port numbers on any machine on which the applications execute.

Not all applications use static, predetermined connection parameters. For
example, the File Transfer Protocol (FTP) application determines through which ports
a communication with a user application will flow on a session-by-session basis. As
used herein, a session is an active communications connection, measured from
beginning to end, between computers or applications over a network. See Newton's
Telecom Dictionary, 15th Ed., by Harry Newton, 1999, p. 706. A session can include the 
exchange of information over a control channel regarding connection parameters to be 
used in transferring a file and the transfer of the file over a data channel.

An FTP application uses this two step process to transfer data (a file). A server 
and a client establish a control channel, over which the client sends a request to the 
server to send a particular file. The control channel is used to negotiate and set the port 
numbers that will be used by both server and client for transferring the file. A data 
connection is then established through the agreed-upon port numbers at both ends, and 
the file is transferred over the connection. Thus, the port numbers through which a file 
is transferred can change for the same server and client from session to session.

This is problematic for known firewalls because the firewall’s rules are static 
and not easily changed, e.g., in the above examples, the port numbers specified by a rule 
cannot be easily changed. In other words, a known firewall rule operates by specifying 
a PASS or DROP action for a fixed set of connection parameters. For a service such as 
FTP, which changes the connection parameters used to transfer a file on a session-by-
session basis, known firewall rules that are based upon a fixed set of connection 
parameters cannot "keep up" with the changing connection parameters employed by the 
FTP service. In essence, the changing connection parameters used by FTP represent a 
"moving target" with which certain known firewall rules cannot keep up.

As another example, suppose an application called XYZ is used to send and 
receive information pertaining to the status and configuration of two computers A and 
B coupled by a network. For example, this information could include the amount of 
memory being used at each computer, the amount of processor resources available at 
each computer, the bandwidth available at each endpoint for transferring information 
to and from each computer, etc. In this example, a control channel is established to set 
up communications between the two computers to exchange information using the XYZ 
service application. All communications between the two computers pass through a 
firewall that stores rules. One of these rules states that for any communication 
originating from port 65 on computer A to port 65 computer B that has a priority
parameter of URGENT or higher may PASS. Another rule states that any communication from port 65 computer B to any port on computer A that has a security flag set to MEDIUM and an error detection flag set to OFF must DROP. In this example, the port number (65) for the XYZ service never change, but the security flag and the priority parameter can change on a session-by-session basis, as appropriate to the conditions present for the session. In known systems, the system administrator would have to manually reload a new rule to accommodate changes in these connection parameters (the security flag and the priority parameter) as conditions warrant, or else risk improperly filtering XYZ communications between computers A and B. In other words, certain packets that should be PASSED will be DROPPED, and vice versa.

Thus, known firewalls cannot efficiently regulate the flow of information to and/or from dynamic applications, i.e., applications that use different connection parameters to provide service on a session-by-session basis, such as FTP. In particular, as used herein, the term "dynamic application" is any application that can assigns at least one connection parameter on a session-by-session basis. An example of a dynamic application is an FTP application because it assigns port numbers dynamically.

**Summary of the Invention**

In accordance with an embodiment of the present invention, a packet sent by a dynamic application is received at a filter. The filter includes a processor that is adapted to execute dynamic access instructions, which include firewall instructions (the "firewall" when executing) and dynamic access proxy instructions (the "dynamic access proxy" when executing). The firewall implements a rule with respect to the packet that refers the packet to a dynamic access control proxy. The proxy detects at least one connection parameter set by the dynamic application for data transfer, e.g., by monitoring the packets flowing over a control channel being used by the dynamic application. The proxy selects a dynamic access rule based upon the detected connection parameter. In one embodiment of the present invention, the dynamic access rule is selected from a set of predetermined dynamic access rules stored at the same site.
as the proxy. In another embodiment, the dynamic access rule is selected from a set of rules at a site external to the proxy, e.g., at an external database. In yet another embodiment, the dynamic access rule is dynamically constructed by the proxy, e.g., generated by the proxy based on a detected connection parameter. As used herein, the term "selected" is meant to include such a dynamic construction of a rule. The dynamic access rule is implemented at the firewall. At the end of the data transfer, e.g., when the data transfer connection is terminated, the rule is deleted.

An embodiment of the present invention advantageously regulates the flow of information to and from a dynamic application efficiently, which cannot be done by known firewalls.

**Brief Description of the Drawings**

FIG. 1 shows an apparatus in accordance with an embodiment of the present invention.

FIG. 2 shows a system in accordance with an embodiment of the present invention.

FIG. 3 is a flow chart illustrating the method in accordance with one embodiment of the present invention.

**Detailed Description**

An apparatus in accordance with an embodiment of the present invention is shown in FIG 1. Peer A 201 (the sender) sends a packet of information addressed to destination Peer B 202 (the destination) through filtering device 203. Filtering device 203 comprises a processor 204, a memory 205 that stores firewall rules 206 and dynamic access instructions 207 adapted to be executed by processor 204 to perform steps of the method in accordance with an embodiment of the present invention, e.g., receive a packet, implement a rule with respect to the packet and refer the packet to a dynamic access proxy, detect a connection parameter at the dynamic access proxy, select a dynamic access rule based on the detected connection parameter, and implement the dynamic access rule.
The filtering device 203 also includes a first port 208 through which the packet is received from Peer A 201, and a second port 209 through which the packet will pass to Peer B 202 through network 210 if the pertinent rule prescribes a PASS action with respect to the packet. Ports 209 and 210, memory 205 and processor 204 are coupled. The term "coupled" is intended to encompass and be broader than the term "directly connected." If A is directly connected to B, and B is directly connected to C, then A is said to be "coupled" to C. In other words, the term coupled includes the term "indirectly connected."

Peers 201 and 202 are each a computer with a permanent or temporary network address. Network 210 is any information systems network across which the information in the packet can be sent. Examples of network 210 include the Internet, an intranet, a virtual private network, etc.

In one embodiment, processor 204 is a general purpose microprocessor, such as the Pentium II microprocessor manufactured by the Intel Corporation of Santa Clara, California. In another embodiment, processor 204 is an Application Specific Integrated Circuit (ASIC), which has been specifically designed to perform at least some of the steps of the method in accordance with an embodiment of the present invention. ASICs are well-known in the art for applications such as digital signal processing. In an embodiment of the present invention that includes an ASIC, at least part of the rule instructions 207 can be implemented in the design of the ASIC.

Memory 205 can be Random Access Memory (RAM), a hard disk, a floppy disk, an optical digital storage medium, or any combination thereof. Memory 205 is meant to encompass any means for storing digital information, although at least part of the memory 205 should be writable. The present invention encompasses memory 205 structures that are distributed, i.e., the rules and instructions stored in memory 205 may be stored in separate structures that are accessible to the processor 204, for example, through a network. For example, in one embodiment, rules 206 are stored on a hard disk on a server coupled through a network to the processor 204, while the rule instructions 207 are stored in RAM coupled to the processor through a bus, the RAM, processor 204 and bus being co-located as parts of the same computer.
In one embodiment of the present invention, dynamic access instructions 207 are implemented in two parts: firewall instructions ("firewall" when executing on a processor) and dynamic access proxy instructions ("proxy" when executing on a processor). In one embodiment, these two sets of instructions are implemented as two separate processes executing on the same processor. In another embodiment of the present invention, two processors on two separate nodes are employed. A first processor executes the firewall instructions, and the second processor executes the dynamic proxy instructions. The first processor is coupled to the second processor, and both first and second processors are coupled to a memory. The firewall performs firewall functions, including receiving a packet, referring the packet to the proxy, and implementing a dynamic access rule. The proxy performs the proxy functions, including detecting connection parameters and selecting a rule based on the detected parameters.

The processors and memory are coupled to ports through which a packet can be received and/or sent. In one embodiment, the firewall process (the executing firewall instructions) is executed as a part of the kernel, i.e., at a relatively low level at which operating system processes are executed. Thus implemented, the firewall process can take advantage of the kernel's protected memory, rendering the firewall process robust and less vulnerable in the event of a system failure. In the kernel's protected memory, the firewall process is protected from user applications that are being executed. The dynamic proxy instructions are executed at the application level, i.e., the level at which software applications (e.g., a word processor, a spreadsheet, etc.) are executed.

Ports 208 and 209 shown in FIG 1 only illustrate one embodiment of the present invention. In the embodiment shown in FIG 1, port 208 is dedicated to communication with peer A 201, while port 209 is dedicated to communication with peer B 202 through network 210. In one embodiment, there are a plurality of ports to and from numerous destinations. The port configuration is expected to vary to suit the particular connectivity required of a filtering device 203 in a given situation, i.e., in a given context or architecture in which parties communicate through filtering device 203. An embodiment of the present invention is advantageously scalable, in part because the rule for a data transfer connection is only selected, stored and implemented for the session
for which the rule is needed. In one embodiment, the rule is deleted when the session is terminated.

In various embodiments, the functions of the present invention are performed on separate nodes. In one embodiment shown in FIG 2, a packet is received from a sender 301 coupled to one 302 of a plurality of receiving nodes 302, 307, 308. Node 302 then determines if the packet pertains to a dynamic application. If it does, node 302 sends a query through network 304 to another separate node 305 that can advantageously function as a central library that stores a large number of general rules 306 that can be used in combination with the connection parameters detected at node 302 in selecting a session rule for the packets pertaining to the dynamic application. The library node 305 identifies the pertinent rule from its collection of rules 306, and then sends it to the requesting receiving node 302, which then selects and implements the dynamic access rule. This further illustrates the advantageous scalability of the present invention. Only relatively few library sites are needed in relation to the number of receiving nodes, which is more efficient because all of the general rules need to be stored only at a few locations rather than at each of the receiving nodes.

A flow chart showing the method in accordance with an embodiment of the present invention is shown in FIG 3. A packet is received by a firewall, step 101. A rule is applied by the firewall to the packet that refers the packet to a dynamic access proxy, step 102. The dynamic access proxy identifies at least one connection parameter set by the dynamic application for data transfer, step 103. A connection parameter is used in combination with a general rule to select a dynamic application rule, step 104. This dynamic application rule is then implemented by the firewall for data transfer packets that are received at the filter, step 105. In one embodiment, when the session is terminated, the rule is deleted, thereby advantageously saving memory resources. Thus, it is determined if the session is terminated, step 106. If it is terminated, then the rule is deleted, step 107.

The connection parameters are used in combination with a general rule (e.g., a rule that has undeclared variables for the negotiated connection parameters) to select a session rule to be applied to data transfer packets. In one embodiment, this general rule
is stored locally at the firewall. In another embodiment, it is fetched from an external database, e.g., stored as a part of a user profile.

For example, a general rule indexed to a particular user (the destination) prescribes that IP voice packets between a given source and a given destination address be PASSed. However, since the port numbers through which the packets will flow are not determined until IP voice dynamic applications at the source and destination negotiate them, the port numbers in the general rule are undeclared variables. Packets are received that indicate the source is setting up a connection to communicate with the destination. These are detected, and a control channel over which the port numbers are negotiated is monitored, and the negotiated port numbers are identified. These port numbers are substituted for the undeclared variables in the general rule to obtain a fully validated (completely specified) session rule. The session rule is loaded at the firewall, and then implemented for subsequent IP voice packets that have the source and destination addresses and negotiated source and destination port numbers.

A medium that stores instructions adapted to be executed on a processor, like memory 205, is meant to encompass any medium capable of storing digital information. Examples of a medium that stores instructions include a hard disk, a floppy disk, a Compact Disk Read Only Memory (CD-ROM), magnetic tape, flash memory, etc.

The term “instructions adapted to be executed” is meant to encompass more than machine code. The term “instructions adapted to be executed” is meant to encompass source code, assembler, and any other expression of instructions that may require preprocessing in order to be executed by processor. For example, also included is code that has been compressed or encrypted, and must be uncompressed and/or unencrypted in order to be executed by a processor.

The present invention advantageously provides a more efficient and flexible system and method for implementing the rules of a security policy or policies at a filtering device, because a rule is only loaded at the filtering device when the rule is needed.
What is claimed is:

1. A method for filtering dynamic application information, including the steps of:
   a. receiving a packet at a firewall;
   b. implementing a rule with respect to the packet that refers the packet to a dynamic access proxy;
   c. detecting at the dynamic access proxy a dynamic application connection parameter;
   d. selecting a dynamic access rule based upon a detected connection parameter; and
   e. implementing the dynamic access rule.

2. The method of claim 1, wherein the received packet has a source address, source port, destination address, destination port and protocol number.

3. The method of claim 1, wherein the step of implementing a rule includes the step of sending a query to a database external to the firewall and receiving a rule from the database.

4. The method of claim 1, wherein the step of detecting connection parameter for a dynamic application data connection includes the step of monitoring a control channel established between dynamic applications.

5. The method of claim 1 wherein the step of performing the action prescribed by the dynamic access rule includes the step of performing a PASS action or a DROP action with respect to a data transfer packet.
6. The method of claim 1, further including the steps of determining if the session is terminated, and if the session is terminated, then deleting the dynamic access rule.

7. An apparatus for filtering dynamic application information, comprising:
   a. a processor;
   b. a memory that stores a rule and dynamic access instructions adapted to be executed by said processor to receive a packet, refer the packet to a dynamic access proxy, detect a dynamic application connection parameter at the dynamic proxy, select a rule based upon the detected connection, and implement the dynamic access rule, said memory coupled to said processor;
   c. a first port adapted to be coupled to the sender of the packet, said first port coupled to said processor; and
   d. a second port adapted to be coupled to the destination to which the packet is addressed, said second port coupled to said processor.

8. The apparatus of claim 7, wherein said dynamic access instructions are comprised of firewall instructions and dynamic access proxy instructions, wherein said firewall instructions are implemented in the kernel and said dynamic access proxy instructions are implemented at the application layer, and wherein said firewall instructions are adapted to be executed by said processor to receive a dynamic application packet, refer the packet to the dynamic access proxy instructions executing on said processor, and implement a dynamic access rule, and said dynamic access proxy instructions are adapted to be executed on said processor to detect a dynamic application connection parameter and select a dynamic access rule based upon the detected connection parameter.
9. The apparatus of claim 7, wherein said dynamic access instructions are further adapted to be executed by said processor to detect if a session is terminated and to delete a rule.

10. A medium that stores instructions adapted to be executed by a processor to perform steps including:
   a. receiving a packet at a firewall;
   b. implementing a rule with respect to the packet that refers the packet to a dynamic access proxy;
   c. detecting at the dynamic access proxy a dynamic application connection parameter;
   d. selecting a dynamic access rule based upon a detected connection parameter; and
   e. implementing the dynamic access rule.

11. The medium of claim 10, wherein said instructions are further adapted to be executed by a processor to perform the step of detecting a connection parameter by monitoring a control channel established between dynamic applications.

12. The medium of claim 10, wherein said instructions are further adapted to be executed by a processor to perform the step of determining that a session is terminated and deleting a rule.

13. A system for filtering a packet that is part of a session between applications that negotiate a connection parameter, including the steps of:
   a. means for receiving a packet at a firewall;
   b. means for implementing a rule with respect to the packet that refers the packet to a dynamic access proxy;
c. means for detecting at the dynamic access proxy a dynamic application connection parameter;

d. means for selecting a dynamic access rule based upon a detected connection parameter; and

e. means for implementing the dynamic access rule.

14. The system of claim 13, further comprising means for detecting if a session is terminated and means for deleting a rule.
RECEIVE PACKET

REFER TO DYNAMIC ACCESS PROXY

IDENTIFY CONNECTION PARAMETER

BUILD DYNAMIC SESSION APPLICATION RULE

IMPLEMENT RULE

SESSION TERMINATED?

DELETE RULE

FIG 3