Firewall Categories

A firewall system can be composed of many different devices and components. One of those components is the filtering of traffic, which is what most people commonly call a firewall. Filtering firewalls come in many different flavors, including the following:

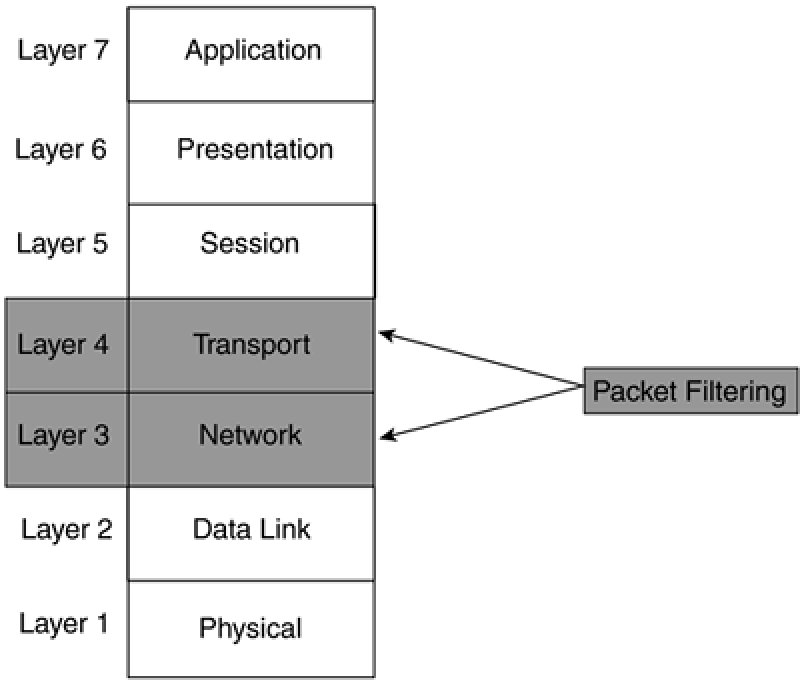
- Packet-filtering firewalls
- Stateful firewalls
- Application gateway firewalls
- Address-translation firewalls
- Host-based (server and personal) firewalls
- Hybrid firewalls

The following sections cover these different firewall categories, explaining how they function and describing the advantages and disadvantages each of them have.

Packet-Filtering Firewalls

The simplest form of a firewall is a packet-filtering firewall. A packet-filtering firewall is typically a router that has the capability to filter on some of the contents of packets. The information that the packet-filtering firewall can examine includes Layer 3 and sometimes Layer 4 information, as shown in Figure 2-5. For example, Cisco routers with standard ACLs can filter information at Layer 3, and Cisco routers with extended ACLs can filter information at both Layers 3 and 4.

Figure 2-5. Packet Filtering Firewalls and the OSI Reference Model



My First Firewall

The very first firewall product that I worked with, back in late 1993, was called the TIS Firewall Toolkit. From my recollection, it was the first commercial firewall product available. At that time, TIS was developing a UNIX-based version of the product, and the beta versions were available for free download to encourage use of the product and to help find and remove any bugs. This firewall could perform only simple packet filtering. It is a freely available source product, although TIS holds the rights to sell it as a commercial product.

Because TCP/IP is the de facto standard of communications protocols in today's networks, most packet-filtering firewalls support at least this protocol. However, packet-filtering firewalls can support other protocols as well, including IPX, AppleTalk, DECnet, and Layer 2 MAC address and bridging information. Cisco routers with standard and extended ACLs support many protocols, including the ones mentioned.

Filtering Actions

When implementing packet filtering, packet-filtering rules are defined on the firewall. These rules are used to match on packet contents to determine which traffic is allowed and which is denied. When denying traffic, two actions can be taken: notify the sender of traffic that its data was dropped or discard the data without any notification. This can be important when implementing a firewall solution. With the first option, the user knows that traffic was filtered by a firewall. If this is an internal user trying to access an internal resource, the user can call the administrator and discuss the problem. The administrator then can change the filtering rules to allow the user access if the user has the appropriate privileges. If the packet-filtering firewall did not send back a message, the user would have no idea why the connection could not be set up and would have to spend more time troubleshooting the problem.

On the other hand, if a hacker on the Internet is trying to access internal resources in your network, and the hacker gets back a message that the information is being filtered, this gives the hacker information about your network that might tell him how you are protecting it. In this situation, you might want to have your packet-filtering firewall silently drop the filtered traffic. As an example, you might have an internal web server running on port 80 of a device. You have a filtering rule that blocks port 80 traffic for most outside users except for your remote office locations. When a hacker tries to reach port 80 while doing a port scan, if your packet-filtering firewall sends back a message that port 80 is being filtered, the hacker now knows that you are using a filtering device to protect internal resources, and he will spend more time investigating the reaction to different kinds of packets that he tries to slip through the firewall.

Filtering Information

A packet-filtering firewall can filter on the following types of information:

- Source and destination Layer 3 address
- Layer 3 protocol information
- Layer 4 protocol information
- Interface of sent or received traffic

For example, a Cisco router can be used to filter on specific ICMP messages (Layer 3), or source and destination IP addresses (Layer 3) and TCP port numbers (Layer 4). Table 2-2 displays some of the things that a TCP/IP packet-filtering firewall can filter on.

Table 2-2. TCP/IP Packet Filtering Information

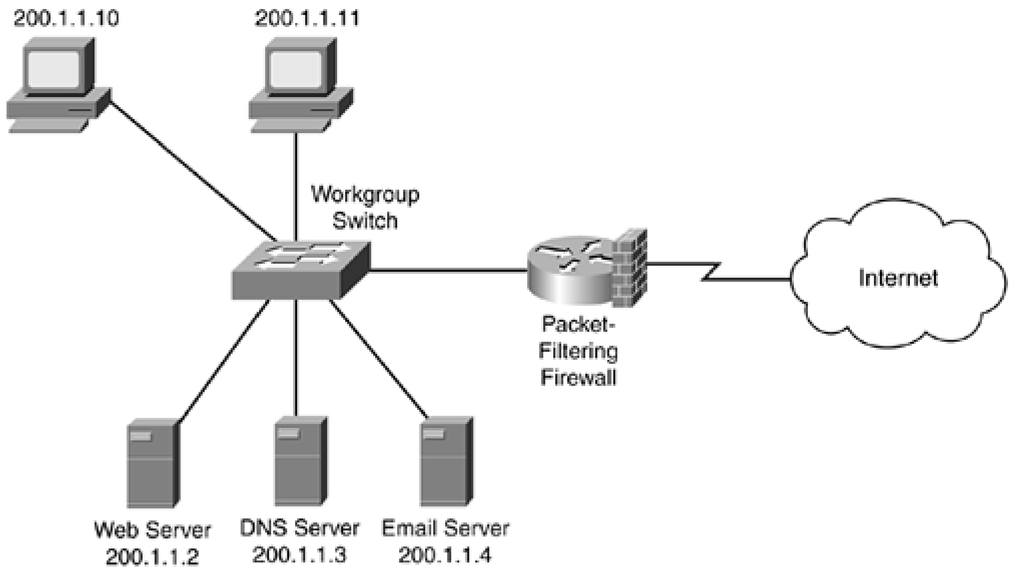
Layer	Filtered Information
3	IP addresses
3	TCP/IP protocols, such as IP, ICMP, OSPF, TCP, UDP, and others
3	IP precedence (type of service [ToS]) information
4	TCP and UDP port numbers
4	TCP control flags, such as SYN, ACK, FIN, PSH, RST, and others

As you can see, you can use a lot of information when making filtering decisions on your packet-filtering firewall. For example, examine the filtering table that the packet-filtering firewall (a router, in this example) is using in Figure 2-6. Table 2-3 shows the router's packet-filtering rules. Assume that these rules are activated on the WAN interface connected to the Internet as traffic comes into the interface.

Table 2-3. Packet-Filtering Table

Rule	Source Address	Destination Address	IP Protocol	IP Protocol Information	Action
1	Any	200.1.1.2	TCP	Port 80	Allow
2	Any	200.1.1.3	UDP	Port 53	Allow
3	Any	200.1.1.4	TCP	Port 25	Allow
4	Any	Any other address	Any	Any	Drop

Figure 2-6. Packet-Filtering Firewall Example



In this example, rule 1 states that if traffic from any device on the Internet is sent to TCP port 80 of 200.1.1.2, the packet-filtering firewall should allow it. Likewise, if any traffic is sent to UDP port 53 of 200.1.1.3 or TCP port 25 of 200.1.1.4, the traffic should be allowed. Any other type of traffic should be dropped.

It is important to point out that if you omit rule 4, you might have issues with a packet-filtering firewall. A packet-filtering firewall will make one of two assumptions:

- If there is no match in the rule set, allow the traffic.
- If there is no match in the rule set, drop the traffic.

For example, assume that you have a packet-filtering firewall that used the first process. In this example, if you omitted rule 4 in Table 2-3, if there were no matches in rules 1 through 3, all other traffic would be permitted.

If your packet-filtering firewall uses the second process, if you omitted rule 4 in Table 2-3, any traffic that did not match the first three rules would be dropped (Cisco uses this process with its ACLs).

CAUTION

Understanding the rule sets that your packet-filtering firewall uses, as well as understanding how the rules are processed, is extremely important. You need to be very careful when creating or changing your rules: An inadvertent configuration mistake can create a huge security hole in your packet-filtering firewall.

Another important item to point out is how packet filters are activated on the firewall. Typically, they are activated on an interface, with a direction specified for traffic flow. For example, a packet-filtering firewall might enable you to filter traffic as it comes into an interface, leaves an interface, or both. If it enables you to filter in both directions, this gives you more flexibility in creating your rule sets: You can restrict traffic coming into your network as well as user traffic trying to leave it. Typically, inbound rules for external users filter on both Layers 3 and 4 information. When setting up policies about Internet access, in many situations, a simple Layer 3 packet filter can be used to restrict which of your internal users have the right to access the Internet. Of course, you always can apply Layer 4 filtering to your users' traffic, to be more granular about what sites and applications they are allowed to access.

Advantages of Packet-Filtering Firewalls

Packet-filtering firewalls have two main advantages:

- They can process packets at very fast speeds.
- They easily can match on most fields in Layer 3 packets and Layer 4 segment headers, providing a lot of flexibility in implementing security policies.

Because packet-filtering firewalls examine only Layer 3 and/or Layer 4 information, many routing products support this type of filtering; this includes Cisco routers with the use of standard and extended ACLs. Depending on the router model that you purchase, you can scale your filtering to very high speeds.

Because routers are typically at the perimeter of your network, providing WAN and MAN access, you can use packet filtering to provide an additional layer of security. These routers commonly are called perimeter or boundary routers, and they were shown previously in Figure 2-6. Even with simple Layers 3 and 4 filtering, packet-filtering firewalls can provide protection against many types of attacks, including certain types of denial-of-service (DoS) attacks, and can filter out unnecessary, unwanted, and undesirable traffic. This allows an internal firewall to deal with other types of threats and attacks that the packet-filtering firewall cannot detect or deal with.

Limitations of Packet-Filtering Firewalls

Despite their advantages, packet-filtering firewalls have these disadvantages:

- They can be complex to configure.
- They cannot prevent application-layer attacks.
- They are susceptible to certain types of TCP/IP protocol attacks.
- They do not support user authentication of connections.
- They have limited logging capabilities.

As mentioned earlier, one of the disadvantages of packet-filtering firewalls involves the creation and maintenance of their rule sets. For TCP/IP, you must be familiar with the operation of various TCP/IP protocols and the fields in their headers, including IP, TCP, UDP, and ICMP, to name a few. Without this knowledge, you inadvertently could block traffic that you are supposed to allow, or allow traffic that you are supposed to block. In either situation, if you are not careful with your configuration, you could be creating a lot of problems for yourself. Plus, any changes that you do make must be tested thoroughly to ensure proper configuration.

Packet-filtering firewalls cannot prevent all types of attacks. For example, you might be allowing traffic to port 80 to a specific web server in your network. By doing this, the packet-filtering firewall is examining the destination address in the Layer 3 packet and the destination port number in the segment. If there is a match, the packet-filtering firewall allows the traffic. One problem with this approach is that the packet-filtering firewall does not examine the actual contents of the HTTP connection. One of the most popular methods of hacking a network is to take advantage of vulnerabilities found in web servers. A packet-filtering firewall cannot detect these attacks because they occur over TCP connections that have been permitted.

Also, packet-filtering firewalls cannot detect and prevent certain kinds of TCP/IP protocol attacks, such as TCP SYN floods and IP spoofing. If a packet-filtering firewall allows traffic to an internal web server, it does not care what kind of traffic it is. A hacker can take advantage of this and flood the web server with TCP SYNs to port 80, pretending to want resources on the server, but tying up resources on it instead. As another example, a packet-filtering firewall cannot detect all kinds of IP spoofing attacks. If you allow traffic from an external network, such as 201.1.1.0/24, your packet-filtering firewall can only examine the source IP address in the packets; it cannot determine whether this is the real source (or destination) of the packet. A hacker can take advantage of this to implement a DoS attack against your internal network by flooding it with allowed traffic from an allowed source.

These two problems, IP spoofing and DoS attacks, typically can be dealt with by causing an individual first to authenticate traffic before allowing it through the firewall. Unfortunately, a packet-filtering firewall examines only Layers 3 and 4 information. Performing authentication requires a firewall that processes authentication information, which is a Layer 7 (application layer) process.

Finally, packet-filtering firewalls typically support logging functions. However, their logging functions are limited to just Layers 3 and 4 information. If someone was executing a specific type of web server attack on port 80 and you were denying port 80 traffic, your packet-filtering firewall could log the deny action, but, unfortunately, the firewall wouldn't log the application-layer data encapsulated in the HTTP transport segment. Therefore, you, as an administrator, would know that someone was attempting to access port 80 on your server, but you would not know what that person was trying to do.

Uses for Packet-Filtering Firewalls

Because of these limitations, packet-filtering firewalls typically are used in the following areas:

- As a first line of defense (perimeter router)
- When security policies can be implemented completely in a packet filter and authentication is not an issue
- In SOHO networks that require minimal security and are concerned about cost

Many companies use packet-filtering firewalls as a first line of defense, with some other type of fully functioning firewall behind it providing additional security. An example of this is a Cisco PIX.

Likewise, packet-filtering firewalls can be used for internal access control between different subnets and departments where authentication is not necessary. In this example, you are concerned about controlling access to specific internal resources from your users; you are not as concerned about sophisticated hacking attacks from your users.

Many SOHO networks employ packet-filtering firewalls because of their simplicity and cost when compared to other types of firewalls. SOHOs are interested in basic security at a reasonable cost. You must realize, of course, that packet-filtering firewalls do not provide complete protection for the SOHO, but they do provide at least a minimal level of protection to keep out many types of network threats and attacks.

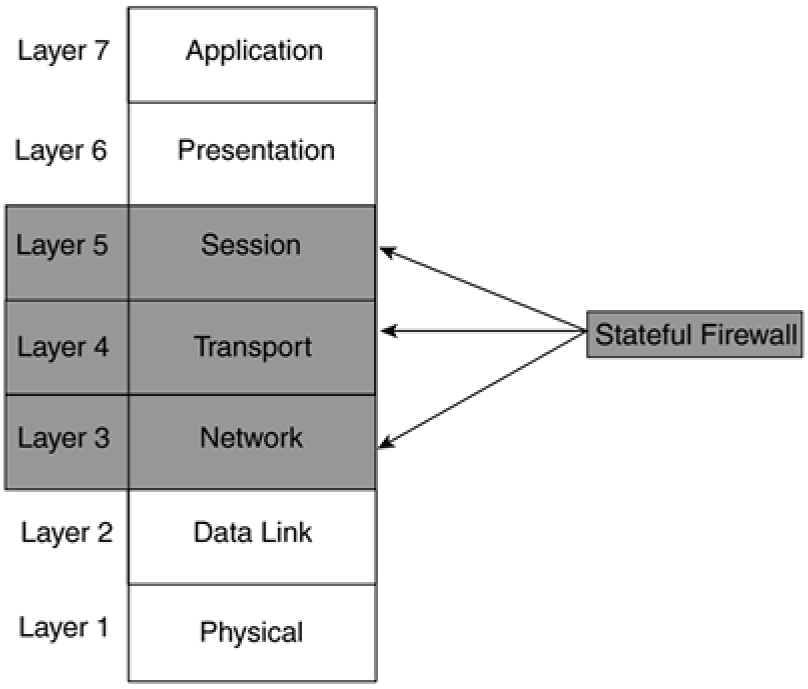
Stateful Firewalls

Unlike packet-filtering firewalls, stateful firewalls keep track of the state of a connection: whether the connection is in an initiation, data transfer, or termination state. This is useful when you want to deny the initiation of connections from external devices, but allow your users to establish connections to these devices and permit the responses to come back through the stateful firewall.

Many security people disagree on what layer of the OSI reference model stateful firewalls function at: Layers 3 and 4 (transport), or Layers 3, 4, and 5 (session). From a transport layer perspective, the stateful firewall examines information in the headers of Layer 3 packets and Layer 4 segments. For example, it looks at the TCP header for SYN, RST, ACK, FIN, and other control codes to determine the state of the connection.

However, because the session layer establishes and tears down the connection?the transport layer handles the actual mechanics of the connection?some say that stateful firewalls operate at Layer 5, as shown in Figure 2-7. In either case, remember that stateful firewalls know about a connection and its state.

Figure 2-7. Stateful Firewalls and the OSI Reference Model

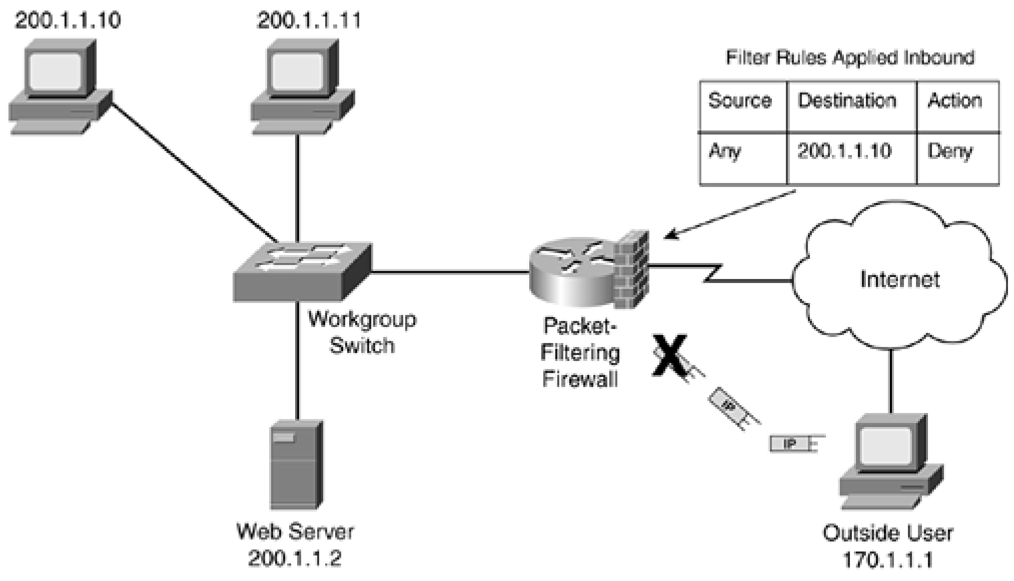


Problems with Packet-Filtering Firewalls

This section and the next one examine one of the issues that packet-filtering firewalls have with traffic and how stateful firewalls can deal with it.

Refer to Figure 2-8 for the example. In the figure, the packet-filtering firewall has a rule placed on its inbound interface from the Internet stating that any external traffic sent to 200.1.1.10 (a user's PC) is denied. As shown in Figure 2-8, when 170.1.1.1 tries to access 200.1.1.10, the packet-filtering firewall drops the traffic, as it is supposed to do.

Figure 2-8. Packet-Filtering Firewall Example?Initiating Connections

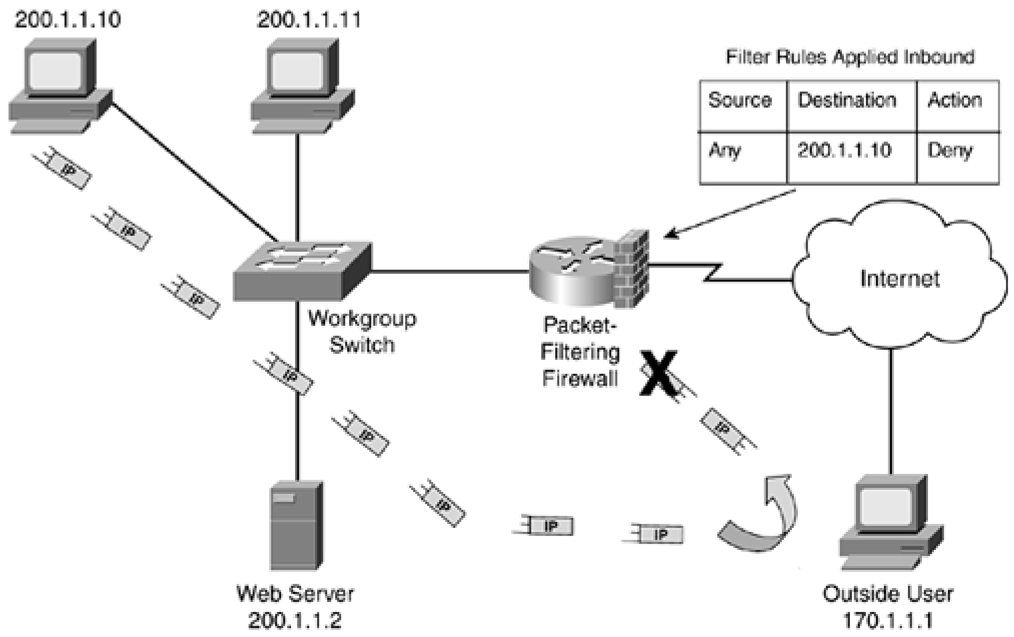


However, what happens if someone inside the network, such as 200.1.1.10, tries to access this external device (170.1.1.1)? Assume that this is an HTTP request to 170.1.1.1, which has a web server running on it. HTTP uses TCP, and TCP goes through a three-way handshake to establish a connection before data is transferred: SYN, SYN/ACK, and ACK. Initially, 200.1.1.10 sends a SYN to establish a connection. With TCP (and UDP), a source port number is chosen that is greater than 1,023, which represents this specific connection. The destination is port 80, telling 170.1.1.1 that this is an HTTP request for web services.

As the packet-filtering firewall receives the traffic on its internal interface, it checks to see if the traffic for 200.1.1.10 is allowed to leave the network. In this case, no filtering rules prevent this, so traffic for 200.1.1.10 traffic is sent to the 170.1.1.1.

170.1.1.1 now responds back to the TCP SYN message of 200.1.1.10 with a SYN/ACK (the second step in the three-way handshake), as shown in Figure 2-9. However, when the packet-filtering firewall examines the packet, it determines that because the destination is 200.1.1.10, the packet should be dropped, according to its packet-filtering rules. Therefore, the connection cannot be set up to the external web server, denying the internal user's web access.

Figure 2-9. Packet-Filtering Firewall Example?Handling Responses



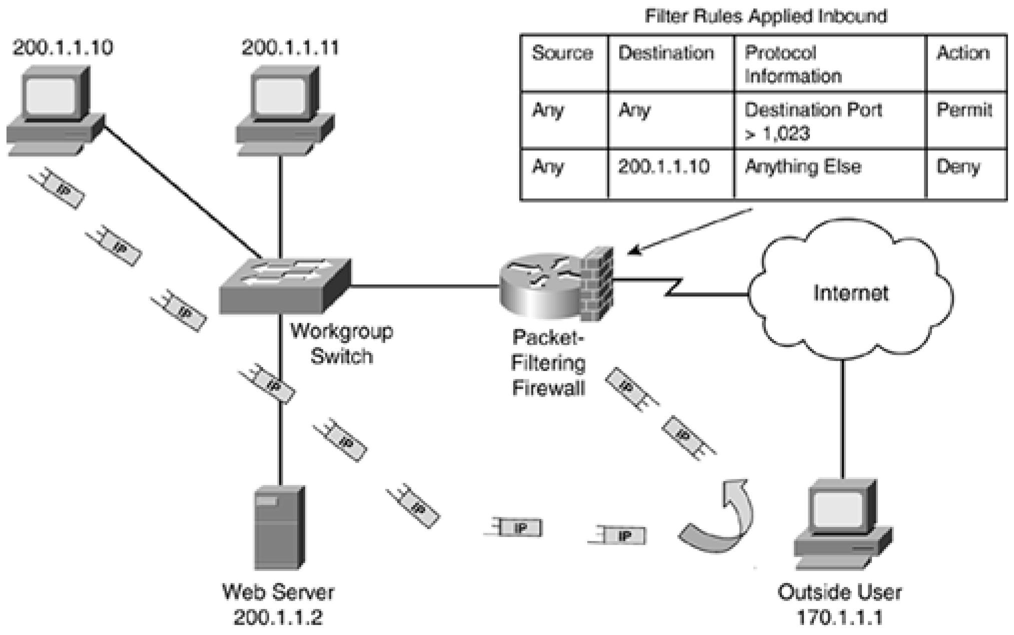
Opening Ports

You can solve this problem with packet-filtering firewalls in two ways:

- Open destination ports greater than 1023 as traffic comes back to the source
- Examine the TCP control bits to determine whether this is returning traffic

Take a look at the first solution. In this situation, the source originally opened a source port greater than 1023, such as 10,000, and used a destination port of 80 for HTTP. Therefore, to allow the traffic to return from 170.1.1.1, the packet-filtering firewall needs a rule that will allow port 10,000. Of course, the problem with this is that the source can use any source port number greater than 1023: Whichever one is free and is chosen by the operating system is the one assigned. Therefore, you would have to allow all ports greater than 1023 to allow the returning traffic to 200.1.1.10, as shown in Figure 2-10.

Figure 2-10. Packet-Filtering Firewall Example?Opening Ports



CAUTION

Opening ports greater than 1023 is not a recommended practice to allow returning traffic from an originating connection: You are creating a huge security hole in your firewall that will open your internal devices to all kinds of attacks.

Examining TCP Control Bits

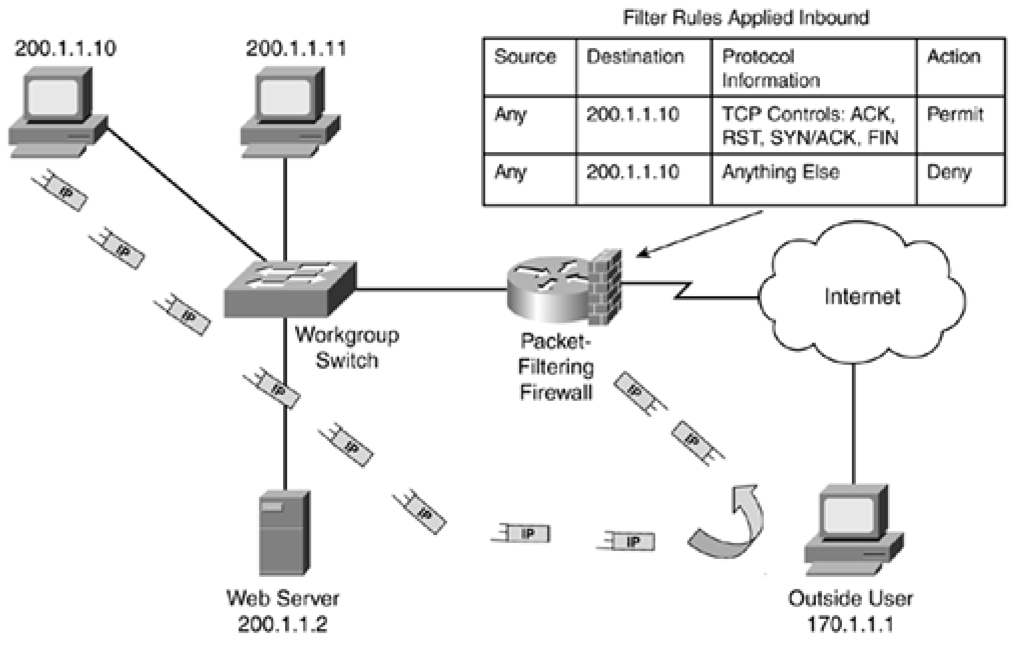
The second approach is to examine transport layer information about the connection to determine whether it is part of an existing connection and, if so, allow the returning traffic back to 200.1.1.1. With TCP, this can be done by examining the control flags in the TCP segment header. These are shown in Table 2-4 and are defined in RFC 793. Note that multiple codes, commonly called flags, can be sent in the same segment header, such as SYN and ACK (SYN/ACK), or FIN and ACK (FIN/ACK).

Table 2-4. TCP Control Information

TCP Message	Explanation
ACK	Acknowledges receipt of data
FIN	Terminates a connection
PSH	Acts as the push function
RST	Resets the connection
SYN	Initiates a connection and synchronizes sequence numbers
URG	Points to urgent data in the segment payload

In this situation, the packet-filtering firewall examines not only the source and destination addresses and port numbers, but, for TCP connections, it also examines the code bits to determine whether this is traffic being initiated from a device or traffic being sent in response to a request. For example, when the internal user (200.1.1.10) sends a TCP SYN, you know that the 170.1.1.1 will respond with a SYN and ACK in the TCP segment header. Therefore, if you know what kind of response control flags TCP uses, you could configure your packet-filtering firewall to allow this traffic, as shown in Figure 2-11.

Figure 2-11. Packet-Filtering Firewall Example?Examining Transport Control Codes



Filter Rules Applied Inbound

Source	Destination	Protocol Information	Action
Any	200.1.1.10	TCP Controls: ACK, RST, SYN/ACK, FIN	Permit
Any	200.1.1.10	Anything Else	Deny

Two problems exist with examining control codes at the transport layer:

- Not all transport layer protocols support control codes.
- Control codes can be manipulated manually to allow a hacker to slip packets through a packet-filtering firewall.

One of the biggest problems of having the packet-filtering firewall examine the control codes is that, in the TCP/IP protocol suite, TCP has control codes, but UDP doesn't. Therefore, for a UDP connection, you do not know whether this is the beginning, middle, or end of a connection unless you examine the data encapsulated in the UDP segment.

The other problem is that, even for transport-layer protocols that support control codes, such as TCP, these control codes can be manipulated manually. For example, the packet-filtering router in Figure 2-11 allows traffic to 200.1.1.10 if certain control codes, such as SYN and ACK, are set in the TCP header. The assumption here?and it is a big assumption?is that the data is a response to information that 200.1.1.10 requested from an external device, such as 170.1.1.10. However, the packet-filtering firewall cannot distinguish between a valid response and a fake response. With a fake response, a hacker generates TCP segments with certain code flags set, trying to gain access through your firewall. A packet-filtering firewall, cannot distinguish between the two types of traffic.

CAUTION

The established Cisco parameter for TCP extended ACLs examines the code flags and allows returning traffic into the network. However, a hacker can manipulate this returning traffic, opening you to reconnaissance, DoS, and other types of attacks.

State Table

Unlike packet-filtering firewalls, stateful firewalls use a mechanism to keep track of the state of a connection. See Figure 2-12 and Figure 2-13 for an illustration of this. Figure 2-12 uses the same network discussed in the last section with a packet-filtering firewall. In this example, the packet-filtering firewall has been replaced by a stateful firewall, but the filtering rule is unchanged: Any traffic sent to 200.1.1.10 is dropped.

Figure 2-12. Stateful Firewall Filtering Example?Part 1

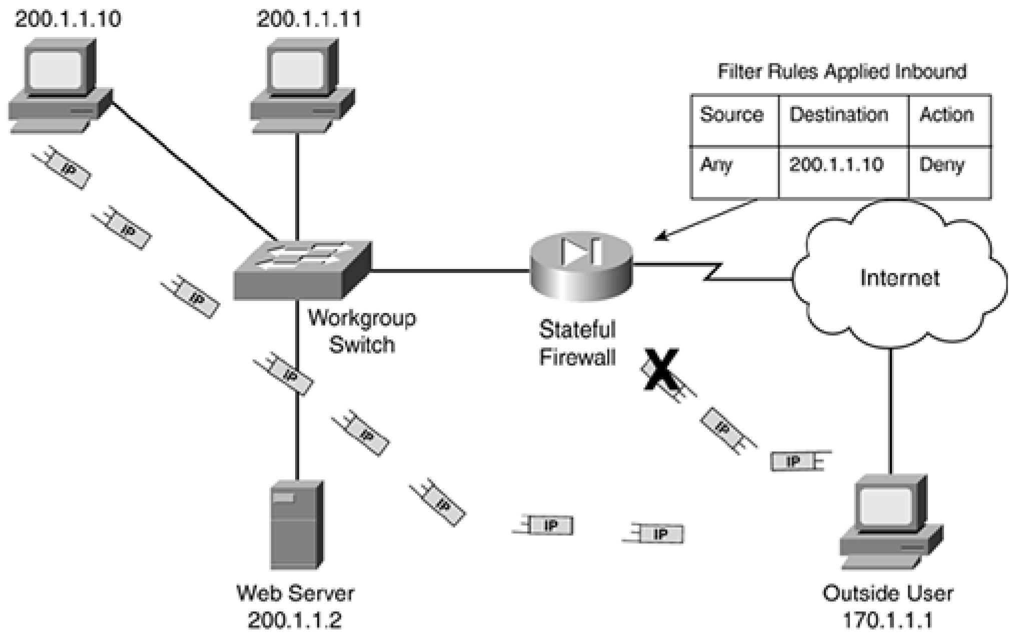
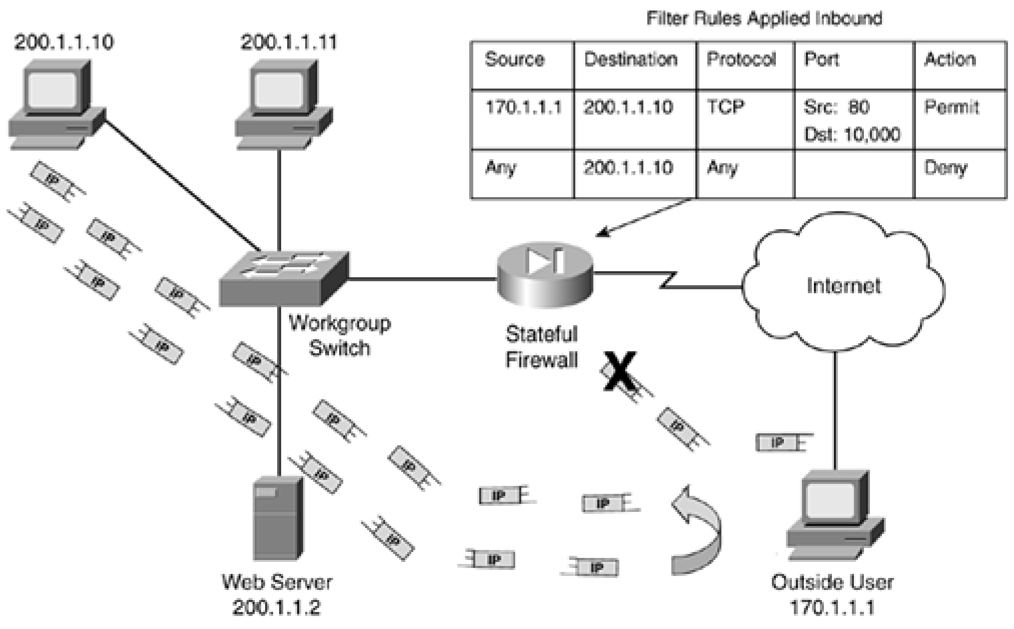


Figure 2-13. Stateful Firewall Filtering Example?Part 2



Filter Rules Applied Inbound

Source	Destination	Protocol	Port	Action
170.1.1.1	200.1.1.10	TCP	Src: 80 Dst: 10,000	Permit
Any	200.1.1.10	Any		Deny

Assume that 170.1.1.1 sends traffic to 200.1.1.10. As shown in Figure 2-12, this traffic is dropped. Now assume that 200.1.1.10 opens a web connection to 170.1.1.1, as shown in the bottom part of Figure 2-12. When 200.1.1.10 does this, it uses a TCP segment with a source port of 10,000 and a destination port of 80. It uses a SYN flag in the control field.

When the stateful firewall receives this traffic, it first checks to see whether the 200.1.1.10 connection is allowed out. In this case, no filtering rules prevent this. Unlike a packet-filtering firewall, which just forwards the packet to 170.1.1.1, a stateful firewall adds a filtering rule to its configuration. This information either is added to the top of the existing filtering rule set or is placed into a state table. This table is used to keep track of the state of connections. The former process is shown in Figure 2-13.

After 170.1.1.1 receives the connection request, it responds to 200.1.1.1 with a SYN/ACK. When this segment reaches the stateful firewall, the firewall looks in its state table first (if the second method discussed previously is used) to see if the connection exists. Then it processes the filtering rules on the interface. In this example, only one table was used, but the connection entry was placed at the top. Because the connection information was added when 200.1.1.1 initiated the connection, the stateful firewall knows that the response from 170.1.1.1 (TCP port 80) to 200.1.1.1 (TCP port 10,000) is part of an existing connection and, therefore, that should allow the traffic, as shown in Figure 2-13.

One advantage of the stateful process is that when the connection terminates, the source or destination device tears down the connection and the stateful firewall notices this by examining the TCP header control flags and dynamically removes the connection from the state table (or filtering rules table).

Therefore, when comparing packet-filtering firewalls and stateful firewalls, stateful firewalls are more intelligent because they understand the state of a connection: initiating a connection, transferring data, or terminating a connection. Basically, a stateful firewall contains a superset of packet-filtering functions.

Advantages of Stateful Firewalls

As you learned in the previous explanation, stateful firewalls have advantages over packet-filtering firewalls:

- Stateful firewalls are aware of the state of a connection.
- Stateful firewalls do not have to open up a large range of ports to allow communication.
- Stateful firewalls prevent more kinds of DoS attacks than packet-filtering firewalls and have more robust logging.

First, stateful firewalls are aware of a connection's state: Stateful firewalls typically build a state table and use this table to allow only returning traffic from connections currently listed in the state table. After a connection is removed from the state table, no traffic from the external device of this connection is permitted. Therefore, these types of connections are more difficult to spoof. For instance, with HTTP, connections are very short lived, so if a hacker noticed the connection being torn down and tried to sneak in some data by spoofing the TCP port numbers and IP addresses, the data would be stopped because the connection entry already would have been removed.

However, if this was a Telnet connection, which might last many minutes or hours, the hacker could attempt to spoof the connection. To do this, the hacker would not only have to spoof the port numbers in the transport layer segment, but he also would have to spoof the IP addressing information, which is a difficult process if the hacker wants this information returned to his desktop. Even so, when the real source or destination terminated the connection, the stateful firewall would remove the entry.

Second, stateful firewalls do not require you to open a large range of port numbers to allow returning traffic back into your network: The state table is used to determine whether this is returning traffic; otherwise, the filtering table is used to filter the traffic.

Third, by using a state table, the stateful firewall can prevent more kinds of DoS attacks than a packet-filtering firewall. Plus, the stateful firewall can log more information than a packet-filtering firewall, such as when a connection was set up, how long it was up, and when it was torn down.

Limitations of Stateful Firewalls

You would think that with these advantages, stateful firewalls are a great tool, and they are. But stateful firewalls have these limitations:

- They can be complex to configure.
- They cannot prevent application-layer attacks.
- They do not support user authentication of connections.
- Not all protocols contain state information.
- Some applications open multiple connections, some of which use dynamic port numbers for the additional connections.
- Additional overhead is involved in maintaining a state table.

The first three items I already discussed in the section on packet-filtering firewalls. As with packet-filtering firewalls, most of the filtering is done by specifying rule sets with Layers 3 and 4 information, which can be complex. Plus, because stateful firewalls really only process Layers 3, 4, and 5 (depending on whom you speak with) information, they can't detect application-layer attacks or perform any type of user authentication to allow the setup of a connection.

Stateful Firewall Problem: Nonstateful Protocols

In addition to these problems, stateful firewalls have issues with nonstateful protocols. Protocols that go through a defined process to establish, maintain, and tear down a connection are called stateful; mechanics are defined as to how these processes occur. TCP is an example of a stateful protocol.

However, not all protocols are stateful: UDP and ICMP are not. For example, UDP has no defined process for how to set up, maintain, and tear down a connection; this is defined on an application-by-application basis. A simple example is a DNS resolution: A source generates a DNS request to resolve a name, and a DNS server responds with a reply with the IP address for the name. In this example, keeping track of the state of the connection is simple: The stateful firewall looks for a DNS request, adds an entry to the state table to allow the reply, and then removes the entry when the DNS server sends the reply.

Not all UDP applications are this simple, however. In most of these applications, many packets are sent between the source and destination, typically at a constant rate. Most stateful firewall solutions treat UDP traffic as stateful by assigning an idle timer to these connections in the state table. As an example, a stateful firewall might use an idle timer of 30 seconds; if after 30 seconds no UDP traffic is seen for a UDP entry in the state table, the stateful firewall removes it. The main problem with this approach is that if a hacker sends spoofed packets into your network, this would keep the entry in the table indefinitely. Of course, a hacker must be quick about this because most UDP connections are temporary.

Like UDP, ICMP also presents problems. A simple example is a ping test. Depending on the ping program, it might generate 1, 3, 5, or even 10 echo test requests. Given this discrepancy, how should a stateful firewall handle this nonstateful protocol? As with UDP, one solution is to use an idle timer, but this entry in the state table can be spoofed. As with UDP, ICMP connections are very brief, so spoofing this connection is unlikely. Another solution would be to put a separate entry in the state table for each echo request. The problem with this is that if you are doing a congestion test by generating thousands of echo requests, your stateful firewall can become overburdened by trying to manage all of these short, temporary connections.

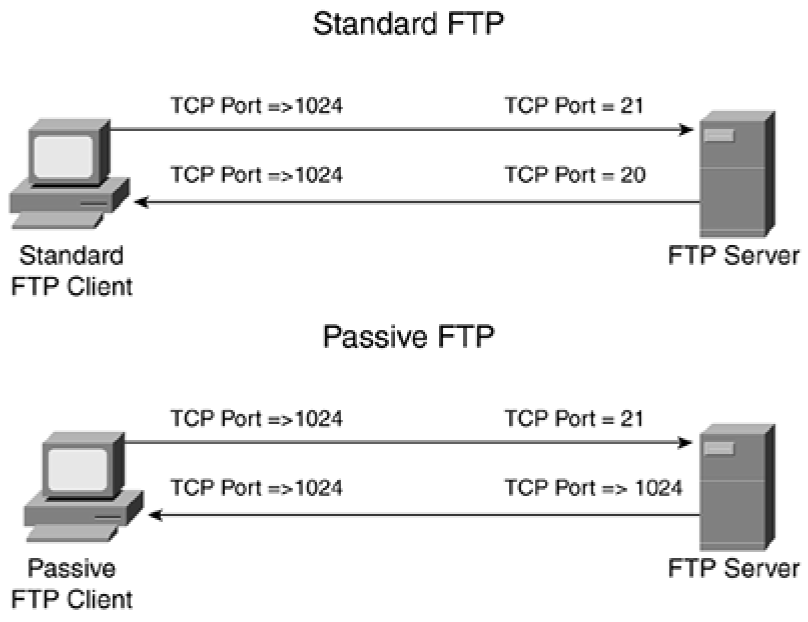
Stateful Firewall Problem: Multiple Application Connections

Another problem that stateful firewalls have involves dealing with applications that open additional connections to transmit information. These can include FTP, multimedia, NetBIOS, and many others. FTP is used as an example here. FTP supports two different modes:

- Standard (or active)
- Passive

Both modes set up two TCP connections. An example of these connections is shown in Figure 2-14. With passive-mode FTP, as long as the user is inside the network establishing connections going out, you have no problems: Both outbound connections are placed in the state table, and the returning traffic for these automatically is allowed. However, if the client device is outside the stateful firewall, you would need a specific filtering rule to allow the port 21 connection (called the control channel) and a very expansive filtering rule to allow the second connection (the data channel).

Figure 2-14. FTP Connections



NOTE

Packet-filtering firewalls have the same problem with external users and passive FTP.

With standard FTP, if the client is inside the network and the server is outside, both stateful and packet-filtering firewalls would have problems dealing with the data connection that the FTP server was establishing to the client: You would have to open a whole range of ports to allow this second connection.

Stateful Firewall Problem: Size of State Table

When it comes to the state table, it is a double-edged sword for stateful firewalls. Yes, it does provide a lot of advantages. But in large networks, the stateful firewall might be busy building and maintaining the state table, putting an extra burden on its processing capacity. The more connections your stateful firewall must monitor, the more horsepower your stateful firewall needs to maintain the table, thus increasing its cost. Unlike stateful firewalls, packet-filtering firewalls typically have small filtering tables, which has much less impact on its processing than a stateful firewall has with its state table.

Uses for Stateful Firewalls

Because of its increased intelligence over packet-filtering firewalls, stateful firewalls typically are used in the following areas:

- As a primary means of defense
- As an intelligent first line of defense (perimeter router with stateful capabilities)
- Where more stringent controls over security than packet filtering are needed, without adding too much cost

In most situations, a stateful firewall is used as a primary means of defense by filtering unwanted, unnecessary, or undesirable traffic. Its main advantage over packet-filtering firewalls is that it opens dynamic, temporary holes in the filter rule set to allow returning traffic from connections that originated inside your network.

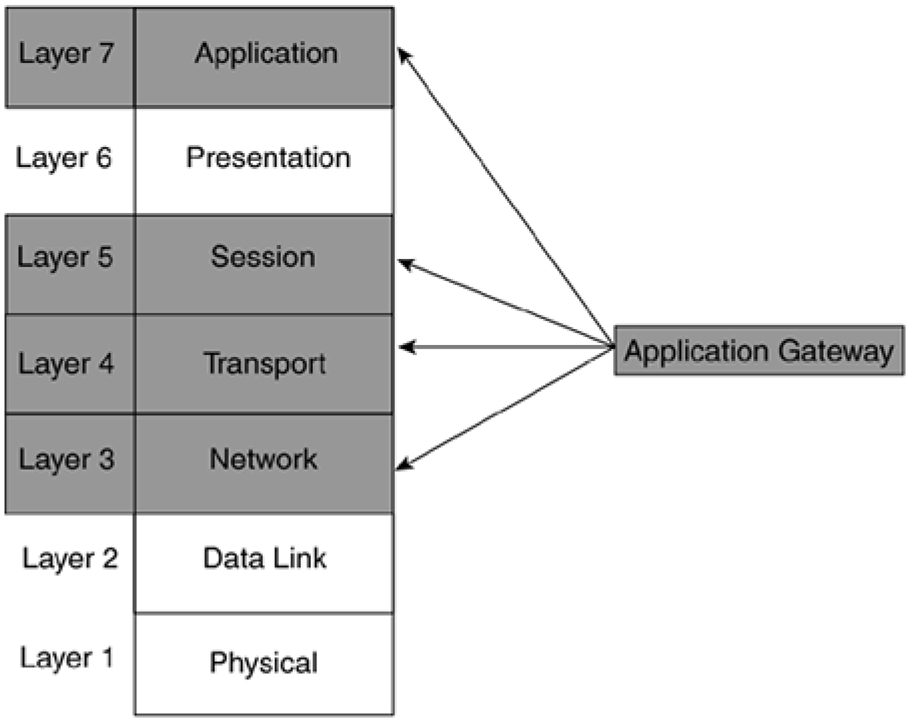
In some situations, certain routing products support a stateful function and can be used as either a primary line of defense or, more typically, as an additional security boost on your perimeter router. In a design with two firewalls?main and perimeter?a router typically is not used as the main stateful firewall because of performance issues; instead, a dedicated firewall appliance is used as the main firewall device.

Also, stateful firewalls provide more control over packet-filtering firewalls, typically at not too much additional cost. If you are more concerned about security, you might consider purchasing a stateful firewall or upgrading a router that supports stateful features to take advantage of more intelligent packet filtering.

Application Gateway Firewalls

Application gateway firewalls (AGFs), commonly called proxy firewalls, filter information at Layers 3, 4, 5, and 7 of the OSI reference model, as shown in Figure 2-15. Because AGFs process information at the application layer, most of the firewall control and filtering is done in software, which provides much more control over traffic than packet-filtering or stateful firewalls.

Figure 2-15. Application Gateway Firewalls and the OSI Reference Model



Sometimes AGFs support only a limited number of applications, or even just one application. Some of the more common applications that an AGF might support include e-mail, web services, DNS, Telnet, FTP, Usenet news, LDAP, and finger.

TIP

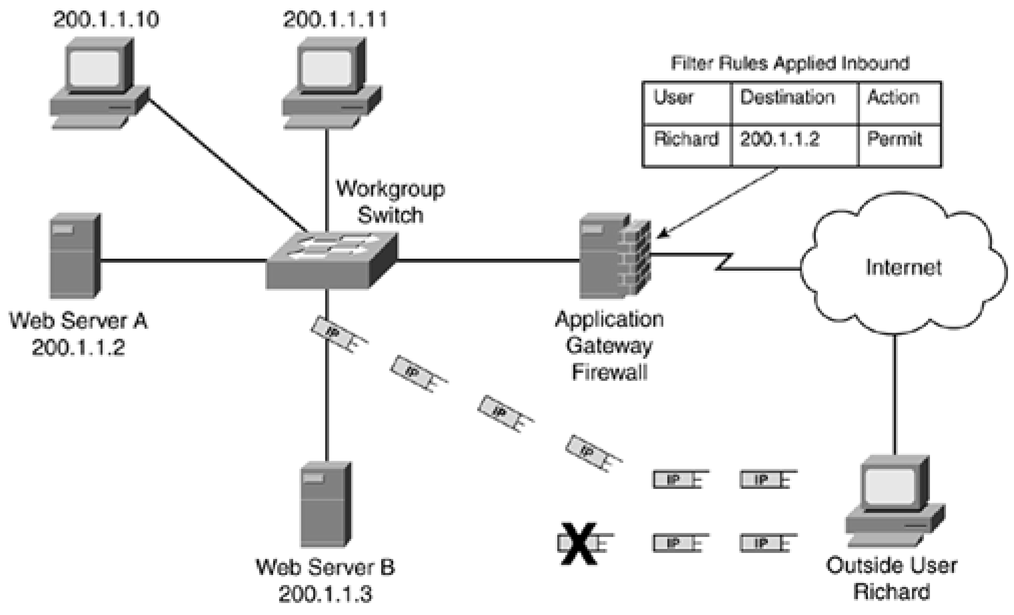
When choosing an AGF, one of the issues to evaluate is the applications supported and the applications needed to send through it.

Authentication Process

One of the features of AGFs is that they typically allow you to authenticate connection requests before allowing the traffic to an internal or external resource. This enables you to authenticate the user requesting the connection instead of the device. This is one disadvantage that packet-filtering and stateful firewalls have: They examine only Layers 3 and 4 information and, thus, can authenticate only the Layer 3 address of a device.

Figure 2-16 shows a simple example of an AGF using an authentication process. In this example, the user first must authenticate to the AGF. This can be done by having the user open a special connection?perhaps a web browser connection to the AGF, or the AGF can intercept the user's initial connection request and send the user a request for authentication information, like a web browser pop-up window. The AGF or an authentication server then authenticates the user's identity. The authentication process occurs in software at the application layer. In Figure 2-16, the authentication database is on the AGF and uses a username and password. In this database, the AGF allows Richard to access web server A upon successful authentication, but it will not allow Richard to access web server B. As you can see from this example, when Richard authenticates, he can access web server A; however, he cannot access any other resource. In Figure 2-16, when Richard tries to send information to web server B, the AGF drops that information.

Figure 2-16. AGF Authentication Process


200.1.1.10
200.1.1.11
Filter Rules Applied Inbound
User | Destination | Action
Richard | 200.1.1.2 | Permit
Workgroup Switch
Internet
Web Server A
200.1.1.2
Application Gateway Firewall
IP
Web Server B
200.1.1.3
Outside User Richard

NOTE

To make the authentication and connection process more efficient, many AGFs authenticate a user once and then use authorization information stored in the authentication database to determine what resources a person can access. The authorization then is used to limit the additional resources that the user is allowed to access, if any, instead of requiring the user to authenticate for each resource that he wants to access. Also note that the AGF can be used to authenticate both inbound and outbound connections.

Authentication Methods

An AGF can use many methods to authenticate a connection request, including username and passwords, token card information, Layer 3 source addresses, and biometric information. Typically, Layer 3 source addresses are not used for authentication, unless they are combined with one of the other methods. Authentication information can be stored locally or on a security server or directory service. An example of a security server is Cisco Secure ACS. Examples of directory services include Novell NDS, Microsoft Active Directory, and LDAP.

CAUTION

You should not rely on the use of just Layer 3 source addresses to perform authentication because Layer 3 addresses are susceptible to masquerading and spoofing attacks.

If you are using a username and password for authentication, the AGF prompts for the username and password. Then it does a lookup for the username and compares the password to what the person typed. If both are the same, the user is validated. One problem with this authentication method is that if the username and password are sent across the connection in clear text, this information is susceptible to eavesdropping; a hacker then can use this information to execute a masquerading attack. Therefore, this information should be encrypted. Typically, this is done through the Secure Socket Layer (SSL) protocol within a web browser connection. This means that the person must open an HTTP SSL (HTTPS) connection to the AGF to perform the authentication. Another method is to have the person use an encrypted Telnet application, such as Secure Shell (SSH). In either method, though, the person first must access the AGF directly to perform the authentication.

Biometric information is more secure than a username and password because usernames and passwords are guessed more easily. With biometrics, some unique physical characteristic of a person, such as a fingerprint or retinal scan, is examined. However, when sending this information to the AGF, it, like usernames and passwords, is susceptible to eavesdropping. Therefore, a secure connection should be used to transmit this authentication information.

Of all of the authentication methods, token cards are the most secure. Token cards create a one-time password, a password that can be used only once. After the password has been used, it is no longer valid. The advantage of token cards is that they are not susceptible to eavesdropping: If the hacker eavesdrops and sees the token card information, it is of no use to him because it will have expired by the time the hacker tries to use it. The downside of token cards is that they can be expensive to deploy on a large scale and are more difficult to troubleshoot and set up.

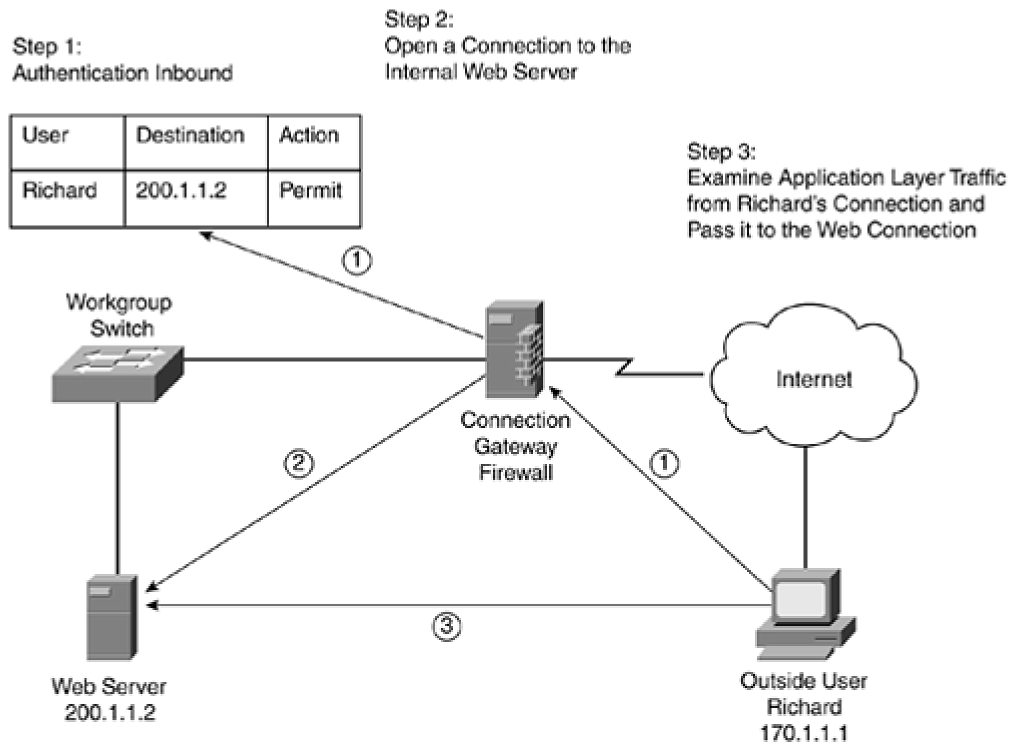
Application Gateway Firewall Types

AGFs fall under two categories: connection gateway firewalls (CGFs) and cut-through proxy (CTP) firewalls. The next two sections compare the two different types.

Connection Gateway Firewalls

CGFs offer more protection than CTP firewalls. Figure 2-17 shows the process that a person goes through when setting up a connection through a CGF. In Step 1, Richard attempts to set up a connection to the internal web server (200.1.1.2). The CGF intercepts the connection and authenticates it, if this has been configured. After authentication, the CGF opens a separate connection to the internal web server (Step 2). At this point, any web traffic sent by Richard to 200.1.1.2 first is processed by the CGF and then is redirected to the internal web server, as shown in Step 3. Any other traffic from Richard is dropped unless it has been authorized by the first authentication request or unless the CGF asks for authentication for any additional connections. If Richard does not authenticate successfully, the CGF terminates the connection.

Figure 2-17. Connection Gateway Firewall Process


Step 1:
Authentication Inbound
Step 2:
Open a Connection to the Internal Web Server
Step 3:
Examine Application Layer Traffic from Richard's Connection and Pass it to the Web Connection
User
Destination
Action
Richard
200.1.1.2
Permit
Workgroup Switch
Connection Gateway Firewall
Internet
Web Server 200.1.1.2
Outside User Richard 170.1.1.1
①
②
③

NOTE

Many CGFs (and CTPs) enable you to configure multiple authorization rules for a single user. Therefore, when the user successfully authenticates, all the authorization rules are put into effect without requiring the user to authenticate for each connection request.

One nice feature of a CGF is that it can examine all data that Richard sends to the web server, even specific URL requests. This allows the CGF to examine what pages Richard tries to access and whether Richard is trying to sneak malformed URLs or data that might try to crash the server or open the server because of a security weakness. Basically, any character that Richard types in for the connection is visible to the CGF. This type of examination is not possible by a packet-filtering or stateful firewall because these devices operate, for the most part, at Layers 3 and 4.

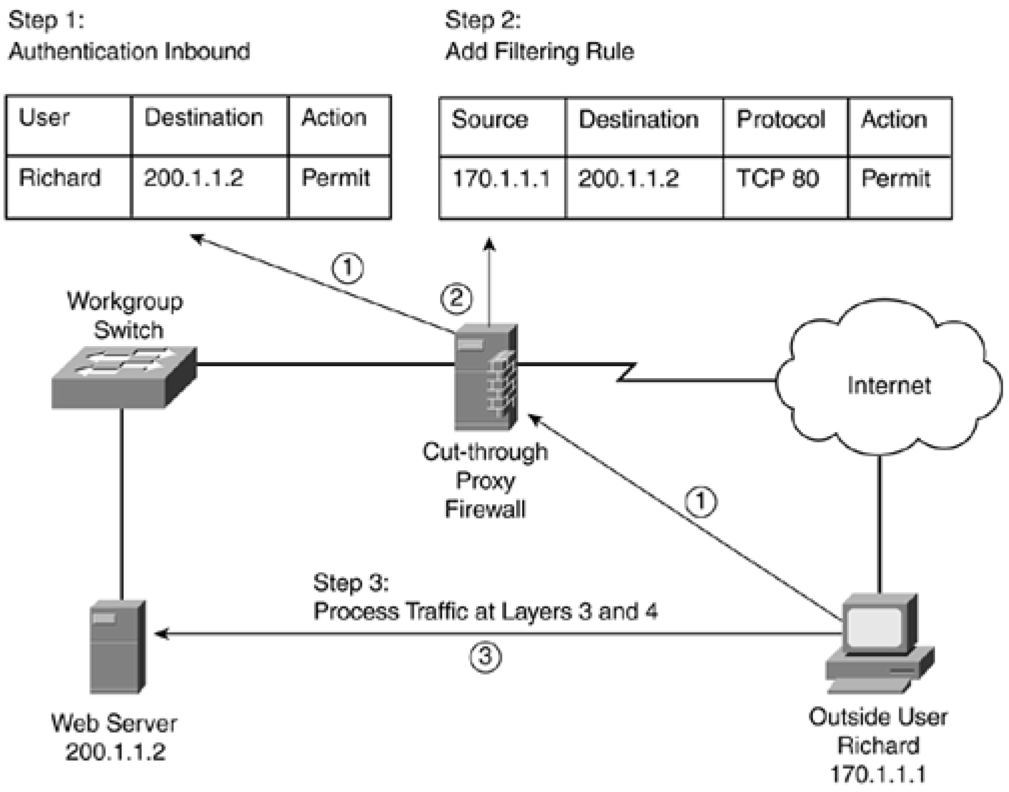
With a CGF, you can be creative in your filtering. For example, you can restrict what Java applets or ActiveX scripts a user has access to. With a Telnet connection, you can monitor and restrict what commands a person can execute on a networking device, such as a Cisco router. With an FTP server, you can restrict what directories a person can see and what files he can download. As you can see, you can create some very powerful filtering rules to secure your network with a CGF.

Cut-Through Proxy Firewalls

One of the main problems of a CGF is that, for the applications that it supports, all traffic is processed at the application layer; this is very process-intensive. In some cases, you might be interested only in performing authentication of a connection at the application layer; after that, you are not really interested in monitoring the activities on the connection?you want just the authentication features. Of course, you could perform this function with a CGF; however, a CGF always processes information at Layer 7, which can introduce a noticeable delay in individuals' connections, especially on an CGF that handles thousands of connections.

Cut-through proxy (CTP) firewalls are a modified version of CGF that deals with this inefficiency. Figure 2-18 shows a simple example of the process that a CTP uses to allow connections into a network. In this example, Richard tries to access the internal web server (200.1.1.2). The CTP intercepts the connection request and authenticates Richard, shown in Step 1. After authentication, this connection and any other authorized connections are added to the filtering rules table, shown in Step 2. From here, any traffic from Richard to the web server is handled by the filtering rules at Layers 3 and 4.

Figure 2-18. Cut-Through Proxy Firewall Process



User	Destination	Action
Richard	200.1.1.2	Permit

Source	Destination	Protocol	Action
170.1.1.1	200.1.1.2	TCP 80	Permit

As you can see from this example, the authentication process is handled at Layer 7; after being authenticated, however, all traffic is processed at Layers 3 and 4. Therefore, the advantage that CTP has over CGF is a huge boost in throughput. However, because CTP does not examine application-layer data, it cannot detect application-layer attacks.

Typically, the CTP supports Telnet, HTTP, and HTTPS for handling the initial authentication. This is done by having the individual set up an authentication to the CTP itself. Optionally, some CTPs can intercept certain connections and respond with authentication prompts. After authentication, the CTP allows the connection initiation request to the internal resource.

Advantages of Application Gateway Firewalls

AGFs have many advantages over packet-filtering and stateful firewalls, including the following:

- They authenticate individuals, not devices.
- Hackers have a harder time with spoofing and implementing DoS attacks.
- They can monitor and filter application data.
- They can provide detailed logging.

One of the advantages of an AGF (CGF and CTP) is that it enables you to authenticate the individual who is trying to access internal resources. This enables you to prevent most spoofing attacks. Plus, DoS attacks are limited to the AGF itself: The AGF can detect these, reducing the burden on your internal resources.

With a CGF, you can monitor all data on a connection, which enables you to detect application attacks such as malformed URLs, buffer overflow attempts, unauthorized access, and more. You even can control what commands or functions an individual is allowed to perform based on the authentication and authorization information.

Finally, with an AGF, you can generate very detailed logs. With a CTP, you can log only the authentication process and any filtering done at Layers 3 and 4; with a CGF, you can monitor the actual data that the individual is sending across a connection. This can be extremely useful if a hacker finds a new type of attack: You can monitor what he does and how he does it so that you can address the problem. Besides using logging for security purposes, you can use it for management purposes by keeping track of who is accessing what resources, how much bandwidth is used, and how often the resources are accessed.

Limitations of Application Gateway Firewalls

Even with their advantages, AGFs have the following limitations:

- They process packets in software.
- They support a small number of applications.
- They sometimes require special client software.

The main limitation of AGFs is that they are very process intensive. They require a lot of CPU cycles and memory to process every packet that they see, which sometimes creates throughput problems. Plus, the detailed logging can create disk space problems. To address these issues, you can use one of these two solutions:

- Use a CTP
- Have the AGF monitor only key applications

For the first solution, using a CTP enables you to do authentication and authorization only; you cannot monitor data on the connection. With the second solution, you limit the AGF to processing only certain application types, such as e-mail, Telnet, FTP, or web services?and then, perhaps, processing just connections to specific internal resources. The problem with this approach is that you are not monitoring all applications and connections, creating a security weakness.

AGFs also typically do not support all applications. They generally are limited to one or a handful of connection types, such as those listed at the beginning of this section. Therefore, you cannot monitor data on all connections.

Finally, AGFs sometimes require you to install vendor-specific software on the client, which is used to handle the authentication process and any possible connection redirection. This can create scalability and management issues if you need to support thousands of clients.

Other Types of Application Proxy Devices

Other types of application gateway devices exist besides AGFs. AGFs are used mainly for security purposes; however, other application gateways (commonly called proxies) can be used to help with throughput issues.

For example, a common type of proxy is an HTTP proxy. With an HTTP proxy, an individual configures the web browser to point to the proxy. Whenever the individual requests a web page, the request goes to the proxy first. The proxy examines its local cache to see if the information was retrieved previously. If it is in the cache, the proxy responds with the information; otherwise, the proxy generates a request to pull the information from the real web server, caches that information, and forwards it to the client. This is similar to a CGF, but without the security functions.

Sometimes these proxies are used to help reduce logging functions on the AGF itself. The AGF monitors connections and creates logging records for security events, but the application proxy handles the detailed logging of all connection data. Likewise, application proxies are useful for tracking your internal users' Internet and remote resource access. This is important if you have acceptable use and abuse policies and need to monitor resource requests so that you can enforce these policies.

Uses for Application Gateway Firewalls

Because of its increased intelligence over packet-filtering and stateful firewalls, AGFs typically are used in the following areas:

- A CGF commonly is used as a primary filtering function.
- A CTP commonly is used as a perimeter defense.
- An application proxy is used to reduce the logging overhead on the CGF, as well as to monitor and log other types of traffic.

In many situations, a CGF is used as a primary filtering defense. In this situation, a perimeter router might or might not be used as a first line of defense; however, the CGF functions as the main line of defense and protection. The CGF performs authentication, but this can be offloaded to a CTP perimeter device. The CGF also performs application filtering and monitoring, and is used to prevent many types of application layer attacks.

A CTP commonly is used as a perimeter defense tool. It performs initial authentication but processes filtering at Layers 3 and 4, thereby reducing its load and allowing a smaller-end device to handle perimeter connectivity.

If too much traffic needs to be monitored or logged, an application proxy is used to offload these functions from the CGF. This allows the CGF to handle important security screening tasks, but it still enables you to keep a detailed record of all connections.

Address-Translation Firewalls

Address translation was developed to address two issues with IP addressing:

- It expands the number of IP addresses at your disposal.
- It hides network addressing designs.

The main reason that address translation (RFC 1631) and private addresses (RFC 1918) were developed was to deal