F5 White Paper

Load Balancing 101: Firewall Sandwiches

There are many advantages to deploying firewalls, in particular, behind Application Delivery Controllers. This white paper will show how you can implement ADCs in a "firewall sandwich" to improve availability, scalability, and manageability across the IT infrastructure.

By KJ (Ken) Salchow, Jr.
Manager, Technical Marketing

Contents

Introduction

Historically, the most common use of the network-based hardware load balancer and its modern incarnation, the Application Delivery Controller (ADC), has been to provide high availability, scalability, and manageability for the back-end application, particularly at the presentation tier. Coming in a very close second is providing those same benefits for other network infrastructure components like routers and firewalls. There are many advantages to deploying firewalls, in particular, behind ADCs (see the white paper Load Balancing 101: The Evolution to Application Delivery Controllers for other uses). This white paper will show how you can implement ADCs in a "firewall sandwich" to improve availability, scalability, and manageability across the IT infrastructure.

Availability

A primary reason for load balancing firewalls is to ensure high availability for all the services behind the firewall. Providing some form of redundancy helps ensure that your organization's web-based services are always available. While this can easily be achieved with a simple active/passive firewall cluster solution, one benefit of using hardware-based ADCs is the fact that the same devices can provide better visibility into service availability and failover services on a service-by-service basis.

For example, on a proxy-based firewall, suppose that only the SMTP process becomes nonresponsive. An active/passive firewall cluster solution typically would not be able to identify this type of failure. If it did, its only recourse would be to fail all connections over to the standby firewall, which would potentially inhibit connections that were working perfectly on other services. An ADC not only provides monitoring of the individual services, but it is also capable of failing over an individual service, such as SMTP, to a secondary firewall without affecting all other connections.

Scalability

Another drawback of active/passive firewall cluster solutions is that they do not scale well. Once the capacity of a single firewall is exceeded, the only option is to replace the firewall with a larger one that can handle the increased load. An ADC, on the other hand, makes it possible to simply add additional firewalls to handle increased load as it becomes necessary.

The intelligence of the ADC also enables a company to scale capacity based on service needs. For example, suppose that, like many organizations, the traffic for your public website is growing significantly and is causing strain on your firewall. Instead of upgrading all your firewalls, you can use an ADC to move all of your website traffic to dedicated firewalls without having to change any other aspect of your site configuration. This enables you to independently scale the security infrastructure of your public website separately from the security needs of the rest of your organization's network environment and keep web traffic from affecting other business-critical applications.

Manageability

Lastly, ADCs provide significant flexibility in managing firewalls. Because they can quickly and intelligently direct traffic at the service level in a variety of ways, they can be used to conduct maintenance (for example, direct traffic away from a firewall that needs to be worked on offline); perform upgrades to software, hardware, and policies (for example, switch users to the new systems, and if a problem occurs, send traffic back to the old ones until the problem is resolved); and to segment traffic across different firewalls for various business groups, initiatives, or other needs with little impact to existing traffic. ADCs can be used to perform SSL decryption, which enables the firewalls to properly enforce policy on encrypted communications (like HTTPS). Decrypting traffic helps when an administrator chooses to route traffic through an intrusion detection system (IDS) or intrusion prevention system (IPS) prior to or after traversing the firewall itself.

The Basic Design of a Firewall Sandwich

The name firewall sandwich reflects the basic design used for most load balanced firewall implementations (see Figure 1). Since the firewall itself is rarely the intended destination of client connections, traffic must be transparently directed through the firewalls in both directions, inbound and outbound. Therefore, ADC devices are needed on both sides of the firewalls, like bread surrounding filling in a sandwich.

There are several interesting points to note about this design. First, notice that, with the exception of the external interface of the Internet router, the entire infrastructure is built on non-routable IP address space. This makes it very difficult for attacks to the network to occur from the Internet.

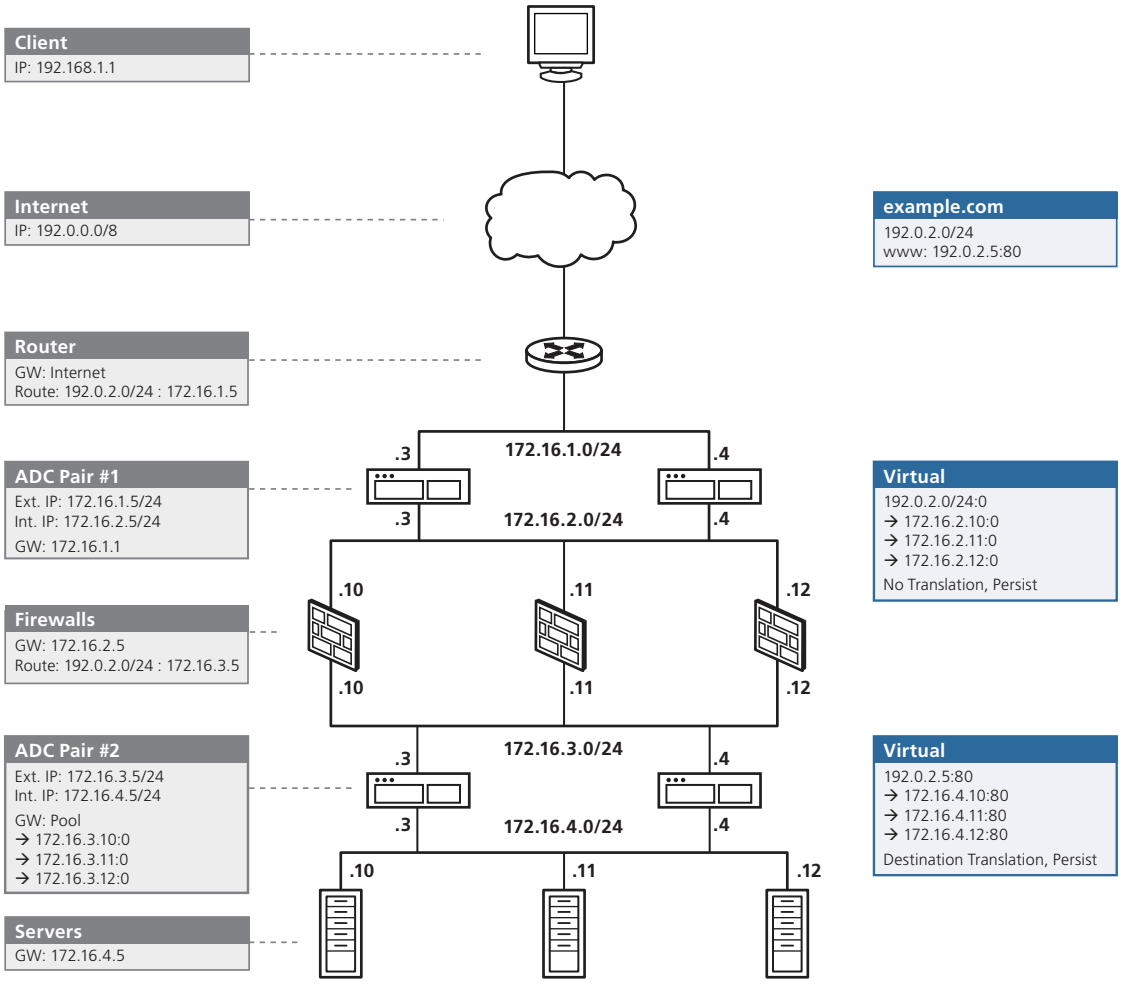

Figure 1. A basic firewall sandwich design

Second, note that the virtual server on ADC pair #1 is at port 0. Port 0 is shorthand for "all ports" (much like 0.0.0.0 is shorthand for all networks). While you can certainly configure multiple virtual servers at each of the ports where you want to receive traffic and simply block all other traffic at the ADC, using port 0 reduces the amount of configuration necessary if and when you add new services. Also, by blocking traffic before the firewall, the firewall logs (or the IDS associated with it) do not have a complete picture of the traffic being sent to your site. ADC pair #2 has a specific virtual server for port 80—web traffic—because it is distributing traffic to web servers running port 80 on the back end. You could just as easily use port 0, but the more restrictive use is common practice, unless you have a significant number of duplicate services running on all of the back-end servers.

Finally, if you look at the gateway (GW) of ADC pair #2, you will notice that it is actually a pool, or "reverse virtual server," that distributes traffic across all three outbound firewalls. This provides a highly available outbound route in the case of a firewall failure after the first connection; you may have issues, however, with asymmetric routing (see the Asymmetric Routing section later in this document).

Typical Traffic Flow: Inbound

In a typical web server load balancing solution, the ADC has a virtual server that is the destination for client traffic, terminates requests, and then distributes them directly to the servers that host the application (see the white paper Load Balancing 101: Nuts and Bolts). In the previous diagram, the valid Internet IP address associated with the website (www.example.com) of 192.0.2.5 does not technically exist on any interface of any device. So how is the user going to connect to the web server? The solution requires "network" or "wildcard" virtual servers, regular virtual servers, and a touch of routing magic. To follow the traffic:

1. User requests www.example.com and gets resolved to 192.0.2.5:80.
2. Client makes first request to website.

   SYN: 192.168.1.1:5000 → 192.0.2.5:80
3. Since example.com owns the 192.0.2.0 network, the client's packet is routed through the Internet to arrive at www.example.com's Internet router.
4. Router 1 has a routing entry that says that the next hop route to reach the 192.0.2.0 network is 172.16.1.5, the shared external interface of ADC pair #1.

   SYN: 192.168.1.1:5000 → 192.0.2.5:80 next hop router 172.16.1.5
5. ADC pair #1 has a network virtual server that listens for traffic destined to the entire 192.0.2.0/24 network and is configured to not do address translation. That virtual server distributes traffic to the three firewalls, load balancing as needed and routing the request to the selected firewall.

   SYN: 192.168.1.1:5000 → 192.0.2.5:80 next hop router 172.16.2.10
6. The firewalls are configured with a route that says that the next hop to reach the 192.0.2.0 network is 172.16.3.5, the shared external interface of ADC pair #2.

   SYN: 192.168.1.1:5000 → 192.0.2.5:80 next hop router 172.16.3.5

7. ADC pair #2 has a regular virtual server listening for traffic destined to 192.0.2.5:80, and when the connection is received, it load balances the connection across the back-end application servers.

   SYN: 192.168.1.1:5000 → 192.0.2.5:80 becomes

   SYN: 192.168.1.1:5000 → 172.16.4.11:80

8. Since ADC pair #2 has an interface on the 172.16.4.0/24 network, it simply sends the packet to the server that responds.

Typical Traffic Flow: Outbound

Outbound traffic flow is not very different from a typical deployment with the exception of selecting an appropriate firewall, ideally using the same firewall that the original inbound connection used (see Potential Issues with Implementation, later in this document). To follow default routes all the way out to the Internet:

1. The server responds to the request and sends the packet to its normal default gateway.

   SYN/ACK: 172.16.4.11:80 → 192.168.1.1:5000 default GW 172.16.4.5

2. ADC pair #2 recognizes this as a load balanced connection return and rewrites the addresses.

   SYN/ACK: 172.16.4.11:80 → 192.168.1.1:5000 becomes

   SYN/ACK: 192.0.2.5:80 → 192.168.1.1:5000

3. ADC pair #2 now has a choice of which firewall to use to return the packet. The decision factor here can be affected by many things, but for the purposes of this example, it will route the packet back to the firewall from which the original SYN came.

   SYN/ACK: 192.0.2.5:80 → 192.168.1.1:5000 → route via 172.16.3.10

4. Firewall 1 receives the response and simply forwards the packet via its default route.

   SYN/ACK: 192.0.2.5:80 → 192.168.1.1:5000 → route via default GW 172.16.2.5

5. Router 1 receives the packet on its internal interface and forwards the packet via its default route onto the Internet.

6. Eventually the client receives the response.
   SYN/ACK: 192.0.2.5:80 → 192.168.1.1:5000

Potential Issues with Implementation

There are several potential issues in this configuration, including handling encrypted traffic, avoiding asymmetric routing, and managing complexity as the number of firewall interfaces increases.

Encrypted Traffic

If the client connection was HTTPS instead of HTTP, the traffic would still be encrypted as it crosses the firewall, inhibiting the firewall from doing any kind of payload inspection. The easiest solution to this problem is to implement an application firewall within ADC pair #2 or, after it decrypts the content, have the ADC forward it to an IDS/IPS/WAF before sending it to the actual application. This enables you to use the exact same configuration and handle all of your security inside the firewall perimeter.

A second solution involves configuring ADC #1 to handle decryption of packets prior to their passing to the firewall. There are several methods that can accomplish this, but since decryption takes places externally to the firewall perimeter, they are generally not recommended.

Asymmetric Routing

Another common issue has to do with asymmetric routing through the firewalls. Problems occur when an inbound packet goes through one firewall, but the return packet goes through a different one. Typically, stateful inspection firewalls have rules that allow the connection to initiate and then permit only packets that are part of that established connection to pass through. If the return packet goes to a different firewall, it will be rejected. There are several solutions to this.

One solution is to enable shared state between the firewalls, so that all firewalls will know about the state of connections and therefore allow the outbound packet through. This is not available on all firewalls, and while compelling, it often becomes a performance problem as the number of firewalls increases. Additionally, shared state can be a detriment to scalability because it restricts the entire cluster of firewalls to the state table capacity of a single device. On the upside, if a firewall

were to fail, all existing connections could easily be moved without having to recreate all state information, which is why many organizations share state even when they solve this problem using other means.

Another more common solution to asymmetric routing is the use of persistence. Normally, this would mean making sure that ADC pair #1 always sends connections to the same firewall it originally did, or that ADC pair #2 always sends connections to the same web server. In this case, it is important to also ensure that ADC #2 remembers which firewall (by IP or Media Access Control [MAC]) sent it the original connection so that it can send the return packet back to that same firewall. In essence, this amounts to ensuring that asymmetric routing does not occur by eliminating dynamic load balancing once the first connection is made.

This is typically done in one of two ways. The first, which works only for HTTP traffic, is to have ADC pair #1 insert an HTTP cookie that tells ADC pair #2 which firewall to return the traffic. This only works for HTTP (not even HTTPs, unless you break encryption at ADC pair #1) and is therefore not usually implemented except by ADCs that are capable of no other alternative.

Most ADC vendors support "last hop" or "reverse persistence" capability. With this capability, ADC #2 creates another connection state entry that tells it the MAC address of the device that sent the original request. Rather than routing in a traditional sense, ADC #2 simply forwards the traffic to the MAC address of the firewall from which it was received.

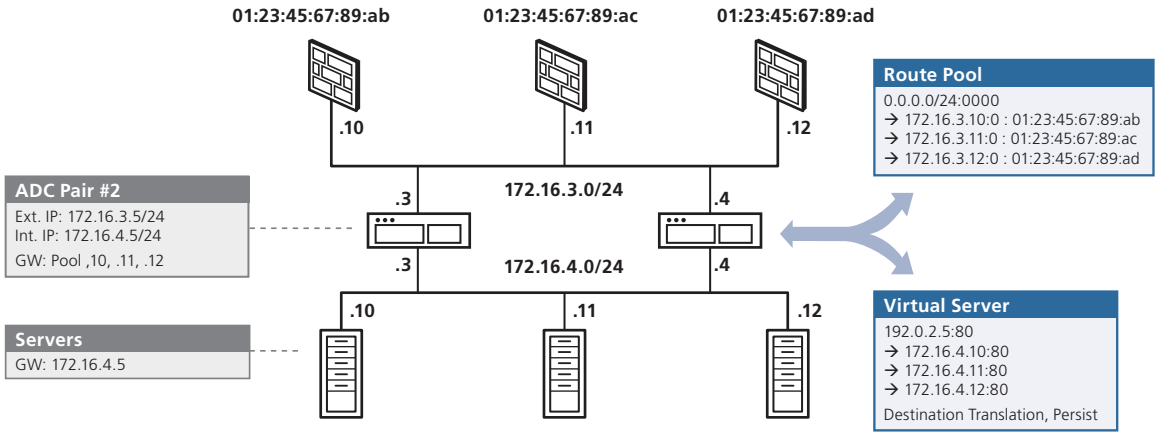

Figure 2. Last-hop return routing by MAC

Looking at the original traffic flow (see Figure 2), the inbound traffic going to the virtual server of 192.0.2.5:80 is terminated at ADC pair #2. When this happens, ADC pair #2 makes state table entries in order to maintain that connection. This usually includes the low-level TCP connection characteristics as well as (because of the persistence settings) the IP of the server selected to receive that traffic. With last-hop capability, the ADC will also record the MAC address of the firewall from which it received the packet, in this case: 01:23:45:67:89:ab.

When the selected server responds and sends the traffic back to the ADC, instead of doing a normal route lookup, the ADC checks the state table, sees the MAC entry, and forwards the packet (after changing the source destination back to the IP of the virtual server) to the MAC address specified with the destination IP of the client. In most ADC implementations, this is an automatic default configuration and requires no user interaction.

Complexity with Growth

The design in the previous section is not too complicated; however, the firewall has only two "legs," or interfaces—one external and one internal. Many firewall implementations have at least three, if not more. To fully use this design, each leg must have an ADC-type device on it to ensure that traffic is routed correctly in both directions. As the number of interfaces grows, this can become quite complex and difficult to troubleshoot. Although the design is basically the same, and firewall sandwiches of up to eight legs have been deployed successfully, the complexity can be a potential issue.

Recovering from Firewall Failure

What happens when a firewall fails? Obviously, if a firewall fails before an initial client connection, the ADC will simply redirect the client connection to a firewall that is working. Remember that with this design, this can be done on the service basis. If a firewall cannot handle SMTP traffic, but can still handle HTTP, the ADC will continue to send HTTP transactions to the firewall, but it will send SMTP traffic to the other firewalls. The two failures that cause potential issues are: 1) when a firewall fails after passing the inbound packet but before passing the return packet; and 2) when a firewall fails after a complete connection is established and being used.

In the former case, the problem is that the ADC pair #2 will want to send the traffic back to the originating firewall, which is now non-functional. Fortunately, in this scenario, the client connection will time out and the client will attempt to resend the connection request, which will now go through a functioning firewall on the inbound request, thus solving the problem with little to no user impact.

The latter case actually has much more impact on the client experience. The same issues are involved in making sure the connection goes to a functioning firewall. This time, however, regardless of directionality (inbound or outbound), the packet will end up on a firewall that has never seen the connection before. Unless shared state is implemented between the firewalls, the connection will fail and the client will be required to reconnect and start the session over. The positive side is that this really only affects long-lived TCP connections, and even in that case, once users reconnect, they will again be connected to the service; FTP would be affected, but HTTP v1.0 will most likely not be affected at all.

Interestingly, with an ADC that implements full-proxy connections (independent connections to the client and server) there are solutions that can mitigate even these failures, at least from the client perspective. Take, for example, a situation where the client connection is terminated at ADC pair #1, and that pair makes its own connection to the service. If a firewall fails and the connection must be reinitiated, it is the connection between ADC pair #1 and the server that needs to be restarted—not necessarily the connection from the client. Although this would require several nonstandard configurations and would depend on the individual protocol, this solution can easily mask the firewall failure from the client.

One of the real benefits of this overall solution is that once the offending firewall is placed back into service, it will immediately be available for new connections and there is no additional user impact.

Conclusion

While the configuration presented in this document is the simplest example of hundreds of potential deployment options, it is sufficient for demonstrating the benefits of deploying firewalls behind Application Delivery Controllers. Your deployment will ultimately depend on your organization's specific needs and existing infrastructure as well as acquisition, operations, and management cost considerations.

This basic concept of the firewall sandwich can be used to manage traffic across many transparent and semi-transparent devices, stateful or not, like SSL accelerators, IDS/IPS, and routers and proxy servers—any inline device that requires better availability, scalability, and manageability.

F5 Networks, Inc. 401 Elliott Avenue West, Seattle, WA 98119 888-882-4447 www.f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com

CS01-97983 0510