Apache
Host multiple SSL sites on a single network card with IP aliasing

A budget-conscious enterprise solution with Apache Web server

http://www.ibm.com/developerworks/web/library/wa-multissl/index.html - Host multiple SSL sites on a single network card with IP aliasing A budget-conscious enterprise solution with Apache Web server

Level: Intermediate

John Yao-An Liao, Senior Technical Architect, Capital Group Companies

Jim Miles, UNIX System Administrator, Capital Group Companies

19 Dec 2006

The interest in using SSL and name-based virtual hosts together is on the increase. Some people will tell you that such a thing is impossible, but you can implement virtual hosts in Apache through IP-based virtual hosts. In this article, John Liao and Jim Miles show you how.

In an earlier developerWorks article entitled "Secure remote data access for DominoÂ®," we talked about leveraging the Apache Web server to solve enterprise problems in a budget-conscious way. (See Resources for a link.) In this article, we will continue with that theme and explain how you can use the Apache Web server to provide multiple Secure Sockets Layer (SSL) Web sites on a single server connected to the network with a single physical network card.

Why would anyone want to put multiple SSL sites on a single server? Is there ever a business need to host multiple SSL sites on a single server in the enterprise? We will attempt to address these questions with a real-life scenario. Innovative users will no doubt find additional creative uses for this idea.

Case study: Two applications, one server

On an earlier project in our company, the human resources (HR) department wanted to provide external Internet access to a Web-based benefits application. The Web application in question was to be accessed mostly from inside the corporate network, with users occasionally accessing it from the external Internet. To meet security requirements, we decided to put the application on a server inside the corporate network and build a reverse proxy server using Apache's HTTP server. The reverse proxy server would terminate the SSL connection and reopen another SSL connection to the Web application server that hosted the HR application. By adding the mod_security module to the Apache Web server, you can change the reverse proxy server to an application gateway and provide even more security to the Web application. The HR department carefully chose a fully qualified domain name (FQDN) that was user-friendly and easy to remember. Then we went ahead and obtained the SSL certificates and thought that

was the end of the story.

Fast-forward to a year later. Another corporate Web application appeared that had requirements similar to those for the HR application. It also needed to provide access to external users. The number of external users was very small. Most of the heavy-duty work would be done within the corporate network. We immediately thought of using a reverse proxy server to provide external access to this new Web application.

However, a few wrinkles occurred with the new application. First of all, we were concerned with the physical space in the data center and looked hard at server consolidation in the deployment of any applications. Second, we had to justify the costs of purchasing additional reverse proxy servers. The combination of these two concerns encouraged us to look closely at how to use of the existing reverse proxy server to fit the needs of the new Web application. The only problem was that this application required an FQDN that was different from the one established for the existing HR application.

We looked at several ideas to make use of the existing reverse proxy servers for this second Web application. The first idea was to change the domain name for both the new and the old applications to something generic, such as rp.company.com, and separate the applications by their contextual paths. However, the original users of the reverse proxy server objected vehemently to any potential name change. They would have to communicate the name change to the entire company and change all their printed material to reflect this updated URL. The cost of the change would be enormous, even ignoring the possible impact to the help desk, which inevitably would receive an large number of incoming calls from users with problems accessing the site. In addition, both application groups preferred to keep their FQDNs, saying that the selected FQDNs were more memorable than the proposed generic URL and that they were also used as a branding effort for these Web applications.

Another idea was floated: Why not register a DNS entry that pointed the new domain name to the existing server? That particular suggestion was quickly put to rest. In an SSL application, an SSL certificate must match the requested URL by the user or a warning message will pop up saying that the request URL does not match the domain name of the SSL certificate. Due to the rise of pop-up ads and malware, everyone in the company had been trained to cancel out of a Web interaction that produced a pop-up warning box. Under corporate architectural standards requirements, production Web applications were forbidden to generate pop-up warning messages.

Another suggestion put forth was to host the second SSL site on another port on the same server running the first site. However, it was felt that this would cause too much confusion for the user, and that it would be too hard for the user to remember the port number along with the site URL. If the user just put in the URL without the port number designation, they'd be redirected to the HR application instead. That would simply cause too many problems.

Solution: IP aliasing

The solution that we eventually stumbled upon was IP aliasing. The trickiest part of searching for this solution was identifying the correct terminology. When we were first introduced to this concept, we heard terms such as virtual interface and virtual IP. We had a hard time finding information about either of these concepts; but when we finally realized that what we were

looking for a concept commonly known as IP aliasing capability, it helped us find more literature on the topic. Sometimes IP aliasing is also referred to as network interface aliasing or logical interfaces.

IP aliasing on Linux systems

Promiscuous mode: A warning

Some ethernet cards enter into what is known as promiscuous mode when configured with multiple IP addresses. In promiscuous mode, the network card will capture all traffic on the local network. This may cause the server to be susceptible to attacks that are directed at other hosts on the net. Most sniffers and network monitoring software put ethernet cards in promiscuous mode to capture all network packets.

The concept behind IP aliasing is simple: you can configure multiple IP addresses on a single network interface. This allows you to run multiple Web servers on the same server using a single interface. It is fairly simple to set up an IP alias. You merely have to configure the network interface on the system to listen for the additional IP address. On Linuxâ„¢ systems, you can add the IP aliases by using standard network configuration tools, such as the ifconfig and route commands, or through a graphical network administrative tool.

You would normally configure each ethernet card with a physical unit number. To add an additional IP alias to an ethernet card that's already configured, you configure an interface with the same physical unit number but qualify it with a logical unit number. For example, if you configured an existing IP address on the ethernet card with the physical unit number eth0, you can create an IP alias by adding the logical unit number :1 to it, as illustrated in Listing 1. You can add additional IP addresses by incrementing the logical unit number. (Note that you need to log in as the root user.)

Listing 1. Adding an additional IP address to an existing network interface

```
ifconfig eth0:1 192.168.0.2 netmask 255.255.255.0
```

The Linux kernel on the system you're configuring must support IP aliasing before you can use this technique. If the support is not there, you might need to rebuild the kernel. To find out if your kernel supports IP aliasing, look to see if /proc/net/alias* files are present.

Once you configure the new IP address, set up the routes for the new interface, as shown in Listing 2.

Listing 2. Adding routes for the new IP address

```
route add -host 192.168.0.2 dev eth0:1
```

After you create the new IP address, you also need to give this new address its own name in the /etc/hosts file, as shown in Listing 3.

Listing 3. Adding a name for the new IP address

```
192.168.0.1 primaryserver
192.168.0.2 secondaryserver
```

IP aliasing on Solaris systems

To set up an IP alias on Solarisâ„¢, you use a slightly different set of commands. Configure the network interface as shown in Listing 4. You need to log in as the root user.

Listing 4. Adding a virtual IP on Solaris

```
 ifconfig eth0:1 plumb
ifconfig eth0:1 192.168.0.2 netmask 255.255.255.0
ifconfig eth0:1 up
```

To make the virtual IP persist after a reboot, you can add the IP address or hostname from /etc/hosts in the /etc/hostname.eth0:1 file. On both Linux and Solaris, you can create multiple virtual interfaces to IP addresses on different subnets on one physical ethernet card. However, you typically want to avoid doing so because it will become a bottleneck between the two subnets, and the performance of all network devices on these two subnets will suffer to some degree because of it.

Alternative uses for IP aliasing

IP aliasing is also used on the client side to perform load and stress testing. Please see the article entitled "Testing and tuning load balancers and networks" for more information (see Resources for a link).

Configuring multiple SSL sites by IP address

Once you configure a second IP address, you can now add additional SSL sites by IP address to the Apache Web server configuration file, as shown in Listing 5.

That's it! You just built multiple SSL Web sites on the same server and on the same physical network card.

Listing 5. Configuration for two SSL Web sites

```
Listen 443

DocumentRoot "/Web site1/docs"
ServerName Web site1.company.com:443
SSLEngine on
SSLCertificateFile ssl/site1.crt
SSLCertificateKeyFile ssl/site1.key


DocumentRoot "/Web site2/docs"
ServerName Web site2.company.com:443
SSLEngine on
SSLCertificateFile ssl/site2.crt
```

Unique solution ID: #1037
Author: Rod
Last update: 2012-02-19 23:09