NameBasedSSLVHostsWithSNI

SSL with Virtual Hosts Using SNI

Summary

Using name-based virtual hosts with SSL adds another layer of complication. Without the SNI extension, it's not generally possible (though a subset of virtual host might work). With SNI, it's necessary to consider the configuration carefully to ensure security is maintained.

(Note: this page is just about support that comes with the Apache web server. Alternatives such as mod_gnutls are another topic.)

The Problem

The problem with using named virtual hosts over SSL is that named virtual hosts rely on knowing what hostname is being requested, and the request can't be read until the SSL connection is established. The ordinary behavior, then, is that the SSL connection is set up using the configuration in the default virtual host for the address where the connection was received.

While Apache can renegotiate the SSL connection later after seeing the hostname in the request (and does), that's too late to pick the right server certificate to use to match the request hostname during the initial handshake, resulting in browser warnings/errors about certificates having the wrong hostname in them.

And while it's possible to put multiple hostnames in a modern certificate and just use that one certificate in the default vhost, there are many hosting providers who are hosting far too many sites on a single address for that to be practical for them.

Server Name Indication

The solution is an extension to the SSL protocol called
Server Name Indication (RFC 4366), which
allows the client to include the requested hostname in the first message of its SSL handshake (connection setup). This allows the server to determine the correct named virtual host for the request and set the connection up accordingly from the start.

With SNI, you can have many virtual hosts sharing the same IP address and port, and each one can have its own unique certificate (and the rest of the configuration).

Prerequisites to use SNI

- Use OpenSSL 0.9.8f or later
- Build OpenSSL with the TLS Extensions option enabled (option enable-tlsext; OpenSSL 0.9.8k and later has this enabled by default).
- Apache must have been built with that OpenSSL (./configure --with-ssl=/path/to/your/openssl). In that case, mod_ssl will automatically detect the availability of the TLS extensions and support SNI.
- Apache must use that OpenSSL at run-time, which might require setting LD_LIBRARY_PATH or equivalent to point to that OpenSSL, maybe in bin/envvars. (You'll get unresolved symbol errors at Apache startup if Apache was built with SNI but isn't finding the right openssl libraries at run-time.)

How can you tell if your Apache build supports SNI? If you configure multiple name-based virtual hosts for an address where SSL is configured, and SNI isn't built into your Apache, then upon Apache startup a message like "You should not use name-based virtual hosts in conjunction with SSL!!" will occur in the error log. If SNI is built in, then the error log will show "[warn] Init:
Name-based SSL virtual hosts only work for clients with TLS server name indication support (RFC 4366)".

The client browser must also support SNI. Here are some browsers that do:

- Mozilla Firefox 2.0 or later
- Opera 8.0 or later (with TLS 1.1 enabled)
- Internet Explorer 7.0 or later (on Vista, not XP)
- Google Chrome
- Safari 3.2.1 on Mac OS X 10.5.6

(per Wikipedia)

Changes in configuration to use SNI

There is one new directive related to using SNI with name-based virtual hosts, SSLStrictSNIVHostCheck, which controls whether to allow non SNI clients to access a name-based virtual host.

The first (default) vhost for SSL name-based virtual hosts **must** include at least one TLSv1.0-or-later permitted protocol, otherwise Apache will not accept the SNI information from the client and it will be as if the client did not support SNI at all.

Since the first (default) vhost will be used for any request where the provided server name doesn't match another vhost, it is important that the first vhost have the most restrictive access control, otherwise clients can access restricted resources by sending a request for any unknown hostname. (This isn't actually any different from using virtual hosts without SSL.)

Environment variables

When Apache supports SNI and the client provided the hostname using SNI, the new environment variable SSL_TLS_SNI will be set to the hostname that the client provided.

Scenarios

SNI/Request hostname mismatch, or SNI provides hostname and request doesn't.

This is a browser bug. Apache will reject the request with a 400-type error.

Client doesn't support SNI.

If Apache has SNI support, and a request without the SNI hostname is received for a name-based virtual host over SSL, and SSLStrict{{`SNIVHost}} `Check is **on**, it will be rejected (403) and this message logged:

```
[error] No hostname was provided via SNI for a name based virtual host
```

If SSLStrict{{`SNIVHost}}`Check is **off**, then the request will be handled as if the server did not have SNI support; see above.

Examples

Server configuration

```
# Ensure that Apache listens on port 443
Listen 443

# Listen for virtual host requests on all IP addresses
NameVirtualHost *:443

# Go ahead and accept connections for these vhosts
# from non-SNI clients
SSLStrictSNIVHostCheck off

<VirtualHost *:443>
  # Because this virtual host is defined first, it will
  # be used as the default if the hostname is not received
  # in the SSL handshake, e.g. if the browser doesn't support
  # SNI.
  DocumentRoot /www/example1
  ServerName www.example.com

  # Other directives here

</VirtualHost>

<VirtualHost *:443>
  DocumentRoot /www/example2
  ServerName www.example2.org

  # Other directives here

</VirtualHost>
```

Detailed Processing

Before there is even an SSL handshake, Apache finds the best match for the IP address and TCP port the connection is established on (IP-based virtual hosting)

If there is a NameVirtualHost directive that has the same literal arguments as this best-matching VirtualHost, Apache will instead consider ALL VirtualHost entries with identical arguments to the matched VirtualHost. Otherwise, SNI processing has no selection to perform.

If the client sends a hostname along with its TLS handshake request, Apache will compare this TLS hostname to the ServerName ServerAlias of the candidate VirtualHost set determined in the preceding steps.

Whichever VirtualHost is selected on the preceding basis will have its SSL configuration used to continue the handshake. Notably, the contents of the certificates are not used in any comparison.

This process mimics the normal (albet misundersood) consecutive application of IP-based, then name-based, vhost matching algorithms used with HTTP, except that the input is the TLS data and not an HTTP header.