ColdFusion, SSL, SNI, SAN and Wildcards - Stuff You Need to Know

Posted At : May 29, 2015 1:35 PM | Posted By : Mark Kruger
Related Categories: ColdFusion, Coldfusion Troubleshooting

The Muse welcomes back his friend and colleague (and super genius guru) Wil Genovese with an timely post on SSL and Certificate types. If you have had your head in the ground (or perhaps you have been guest staring on "Naked and Afraid" or "Survivor") you may have missed the hubbub surrounding TLS, SSL and changes and support. There is a lot going on and it is more important than ever that you get your hands around the issue to keep your users safe. Wil has done Yeomen's work identifying the types of certs, the versions of ColdFusion and Java that support them, and work arounds and caveats for those of you who need them. You will likely want to bookmark this one. Take it away Wil.

Here at **CF Webtools** we are getting a lot of companies coming to us with various CFHTTP issues. Lately this has been happening even more as SSL has been in the news more and certain SSL protocols and encryption levels have gone away or are on the way out. Most recently Wildcard and Subject Alternative Name certificates have been a concern for those running older ColdFusion servers with older versions of Java.

Wildcard vs. Subject Alternative Name (SAN) certificates

A wildcard SSL certificate allows for unlimited subdomains to be protected with a single certificate. For example if you owned "example.com" a wildcard would allow you to secure www.example.com, mail.example.com, or admin.example.com. Such a certificate would be issued to *.example.com and it could secure any subdomain of example.com for the device on which it was installed.

A SAN cert allows for multiple domain names to be protected with a single certificate. It could, for example be issued to multiple fully qualified domains such as www.domain.com, www.domain2.com, www.domain3.com and secure all of them. This allows for the SAN SSL to be used for multiple sites on the same server all bound to the same IP Address. This is similar to "SNI" - see my previous post on **"What You Need to Know About CFHTTP, SSL and SNI**. In addition, this article by **Thawte** is a good primer. Let's take them each in turn.

Case Study 1 - Wildcard SSL

We had a client's ColdFusion 9 server that was minding its own business and happily using CFHTTP to access a remote API via SSL. Even though the API provider claims they did not change anything our client was suddenly left without the ability to access the API. Debugging this gave the famous connection failure error.

When you see this error - especially with "peer not authenticated" - you can bet the problem is due to SSL somewhere along the line. It might mean there's a privately issued SSL or an obscure certificate authority provider leaving the Java keystone without a valid CAROOT certificates. The problem is normally resolved by using the keytool application in Java to import the SSL certificate in to the default Java keystore (/jre/lib/security/cacerts). See the Muse post for **Trusted Keystore** instructions. So in this case the first thing we tried was importing the SSL certificate, which did not work.

Taking a closer look at the certificate we noticed that it was a Wild Card cert.

Failure to handle wild card certs is a known issue in ColdFusion 9 and even made the bug list (https://bugbase.adobe.com/index.cfm?event=bug&id=3566218). It has since been fixed in ColdFusion 10 and 11, but with the client unwilling to upgrade we set about looking for a solution or work around.

Even after upgrading ColdFusion 9 to Java 1.7 we still had the same issue. I have a test bed system here with ColdFusion versions 8, 9, 10 and 11 and I'm setup for testing CFHTTP and SSL issues. It's been a hot topic in the past 6 to 9 months due to all the current and coming changes. I was able to test the same code snippet easily on each ColdFusion version with each version of Java from 1.6 to 1.8. Even on CF 11 using Java 1.8 we had the same problem. We did get a different error message however.

That message *I/O Exception: handshake alert: unrecognized_name* gave us something more that just *I/O Exception: peer not authenticated*. Our crack research team of ColdFusion Guru's here at CF Webtools (including the Muse) went to work Googling and reading. They quickly found a couple things I hadn't uncovered in my research. One of them was this blog post about a Java setting seen below.

```
-Djsse.enableSNIExtension=false"
```

This setting is added to the arguments in your jvm.config file. Do yourself a favor and learn to manually edit this file. The CF Admin has a finicky relationship with non-typical JVM arguments. In any case, this particular argument was added to the possible argument list in Java version 1.7 and it is one I had not seen or used before. I immediately tested the setting and it worked! Obviously more research was warranted but it seemed a possible solution might be around the corner.

The additional testing and research was to see if this Java setting worked on its own or if we still needed to import the SSL certificate. In this case we did need to import the SSL certificate *and* use the Java setting **"-Djsse.enableSNIExtension=false"**. It's a solution that works for ColdFusion 9, ColdFusion 10 and ColdFusion 11 all **running on Java 1.7 or newer**. This additional Java config setting turns off the Server Name Indication (SNI) capabilities in the JVM. Note the trunkful link posted above for more information on SNI.

We were able to test with another site API (https://api.escapia.com/) that is using a wildcard SSL, finding that all that was needed was to import the SSL Certificate. This tends to confirm that the first SSL Certificate in our test case was actually using the SNI feature on their web server which is why the argument "-Djsse.enableSNIExtension=false" was needed. As far as I know there is no way to remotely detect if SNI is enabled on a host server.

NOTE: If you do not know how to import an SSL certificate into the Java keystore then read this Muse post on SSL and the Trusted Keystore in Java. Remember to that the keystore is located in the Java *root* directory - as in {java root}/jre/lib/security/cacerts. If you have upgraded Java that's the location of the new SDK. It takes a bit of trial and error to get those command line switches just right with regard to the proper location of hte cacert file. Just remember to make sure you get the correct Java location as defined in your JVM.config file.

Case Study 2 - SAN SSL Certificates

Case Study 2 - The SAN SSL

You might have noticed that the ubiquitous payment gateway, **PayPal**, has announced that they are dropping SHA-1 (160bit) SSL certificates and they will be going to SHA-2 (256bit) SSL certificates. Some quick initial tests on ColdFusion 8.0.1 failed to work with the following code.

```
<cfhttp url="https://api-s.sandbox.paypal.com" port="443" METHOD="get"
USERAGENT="Mozilla/5.0 (Windows; U; MSIE 9.0; WIndows NT 9.0; en-US))">
</cfhttp>
```

> ColdFusion 8.0.1 returns "I/O Exception: Name in certificate http://api-s.paypal.com does not match host name http://api-s.sandbox.paypal.com"

Yes, we have customers using ColdFusion 8 (one customer is even still on ColdFusion 5) - don't be judgy. Anyway, this got a few of us at **CF Webtools** researching to see what the issue could be and what can be done to resolve the issue. What we found is that using a SAN SSL certificate *won't work at all with Java 1.6.0_45*. Since ColdFusion 8 and older is only capable of running on Java 1.6 (or less) lower this leaves those servers using CFHTTP to access PayPal out in the cold[1]. Of course it's really time to start thinking about upgrading those server and this gives CFWT one more arrow in our quiver to convince our customers. Yes, there is a third party **CFX tag** you can use on Windows servers, but I'd rather encourage customers to get on a newer server. Such old, outdated and no doubt insecure servers are not wise in production use!

Update and Revision

This blog post has been a few weeks in the making due to all the various test cases and research. Currently the PayPal PRODUCTION URL *does work* with the new, upgraded SHA-2 (256Bit) SSL SAN certificate on *ColdFusion 8.0.1* with *Java 1.6.0_45*. I know I know - it was a shock to us as well. There is some bad news however. The sandbox URL still fails. For we are going to hang our hat on the following statement - ***ColdFusion 8.0.1 on Java 1.6 working with a SAN type SSL can be problematic and if you need to verify against one of the SAN names then most likely it will fail***. Some examples might help.

```
<!--- THIS WORKS --->
<cfhttp url="https://api-s.paypal.com" port="443" METHOD="get" USERAGENT="Mozilla/5.0
(Windows; U; MSIE 9.0; WIndows NT 9.0; en-US))">
</cfhttp>

<!--- THIS FAILS --->
<cfhttp url="https://api-s.sandbox.paypal.com" port="443" METHOD="get"
USERAGENT="Mozilla/5.0 (Windows; U; MSIE 9.0; Windows NT 9.0; en-US))">
</cfhttp>
```

Here's my theory and until someone proves it one way or the other it's what I'm sticking to. I believe that the engine for CFHTTP SSL in the Java layer ignores the SAN attribute

in the SSL certificate, but in this case the PayPal SSL certificate Common Name actually *matches the production URL* so that's why it works. It is sort of being "fooled" into treating it like a standard certificate. This gives those running on outdated systems a little more time to get up to date.

More Good News

Further testing on ColdFusion 9.0.2 with Java 1.6.0_45 did not go as I expected. The same sandbox URL that *failed* on Java 1.6 using CF 8, *succeeded* on Java 1.6 using CF 9. It does appear that ColdFusion 9.0.2 really does accept SAN SSL certificates. Future testing on addition SAN type SSL certificates will be needed to complete verify this. And the same is true for ColdFusion 9.0.2 on Java 1.7. Both test cases also worked for the production URL too. Testing on ColdFusion 10 and 11 thankfully passed for all case scenarios.

Aftermath

Thanks Wil for a great effort. It's tough staying ahead of the curve but you make us all look good. As always readers, the Muse welcome comments - especially ones that add to our universe of knowledge. Just keep it positive folks.