Coldfusion, SSL 3.0 and Authorize.net

Posted At : February 24, 2009 4:38 PM | Posted By : Mark Kruger
Related Categories: ColdFusion, Hosting and Networking

I've been batting this around for a few days now. Recently, Mary Jo Sminkey of **CF Webstore** fame posted a note to an email list about the recent requirement by **Authorize.net** that incoming requests to their API use SSL 3.0. I confess to being unaware of the differences between SSL 2.0 and 3.0. So I set out to discover for myself. To start with SSL 2.0 uses weaker handshaking. A requesting client can, it seems, edit the list of preferences leaving the server no choice but to hand shake with the "lowest common denominator" cipher. There are some other issues as well dealing with how the packets are constructed etc. So the consensus is that SSL 2.0 is the weak sister and should be deprecated. For its part SSL 3.0 has been around for a decade or so and is widely supported.

The question is, will my CFHTTP calls from ColdFusion 6 or Coldfusion 7 still work when Authorize.net disables SSL 2.0? To answer this question I got some great help from Scott Krebs over at **Edge Web**. He dug out three or four URLs that were really helpful. I've included them at the bottom of this post. I also got some guidance from the Stephen Hawking of cryptography, Mr. Dean H. Saxe (the H is for Holy Cow he knows a lot). The answer is a qualified yes. Anyway, here's what I did to test while I wait for Authorize.net to get their act together and set up a test bed.

Testing SSL 3.0

First I had to prep a server so that it *only* supports SSL 3.0 and above (we'll talk about the "above" in a moment). I logged into the desktop of one of my ColdFusion 7 windows servers and used these **instructions** to disable SSL2.0. I also disabled PCT for good measure - although I did not have the time to figure out exactly what it is. I restarted the server.

Next I obtained a public cert for a URL and installed it on the web server. Now, theoretically, connecting to that site could only be done via SSL 3.0 or above. So far so good. Note, the "public" cert is important too. I used a Thawte cert so I could guarantee that the keystore would trust it. I didn't want to mistake an issue with the keystore for a problem handshaking.

Finally, I followed the instructions found **Here** to add a debug switch to my jvm.config file. I cleared out the coldfusion-out log and restarted ColdFusion on the server. At this point, Java was logging any cfhttp calls into the coldfusion-out log, including the handshake stuff. I ran the following code:

```
<cfhttp url="https://www.exampleSSL.com/index.html"></cfhttp>
```

...obviously replacing www.examplessl.com with the FQDN of the cert I had installed. The code ran correctly and I was able to retrieve the content of the page. That meant, at a minimum, my ColdFusion 7 server was able to connect successfully using SSL 3.0 (or above). I ran off to examine the coldfusion-out log (<%Cfusionroot%>/runtime/logs/coldfusion-out.log) to see what I could see. One note, the coldfusion-out.log on a CF Standard server traps the "standard output" from the JVM when you are set up with CF as a service. If you are doing something else - running enterprise over Websphere or multi-server/JRUN or perhaps running ColdFusion from the command line - your standard output might be piped to a different file or even to

the console.

The Findings

First off, there is a *great deal* of stuff going on behind the scenes when you make an SSL connection. Running the above request only one time generated over 30k of log data. About 95% of it is the crypto stuff - registers of tabbed hex data. But looking carefully I found the following lines:

```
jrpp-0, WRITE: SSLv2 client hello message, length = 98

jrpp-0, READ: TLSv1 Handshake, length = 914

*** ServerHello, TLSv1

RandomCookie:  GMT: 1218720998

....

A bunch more cypher stuff... Followed by:
....
jrpp-0, WRITE: TLSv1 Handshake, length = 32

jrpp-0, READ: TLSv1 Change Cipher Spec, length = 1

JsseJCE: Using JSSE internal implementation for cipher RC4

jrpp-0, READ: TLSv1 Handshake, length = 32
```

There were more references to TLSv1 before the connection was closed. What to make of it? Well, for one thing no mention of SSLv3 is made anywhere in the log. What gives? I thought that I had disabled SSL 2.0, and clearly I'm making a successful SSL connection, but looking at the log it seems I'm making a "TLS 1.0" connection instead of SSL 3.0.

The Fine Print

It turns out that both Coldfusion 6 and Coldfusion 7 (really the 1.4.2 JVM) support SSL 3.0 by way of the TLS 1.0 protocol. This protocol is sometimes called "SSL 3.1" and it represents a slightly *better* implementation. Connections will actually attempt this protocol first and use it if successful. I would go so far as to say if you are using ColdFusion 6 or 7 and making HTTPS calls from within your code you are likely *already* using TLS and therefore compliant. MSDN puts it this way:

> TLS is a standard closely related to SSL 3.0, and is sometimes referred to as "SSL 3.1". TLS supersedes SSL 2.0 and should be used in new development. Applications that require a high level of interoperability should support SSL 3.0 and TLS. Because of the similarities between these two protocols, SSL details are not included in this documentation, except where they differ from TLS. (**article link**)

Of course one thing to bear in mind would be if a provider *only* supports SSL 3.0 and chooses to *disable* TLS for some reason then CF 6 and 7 would be out of luck. Still, I'm

reassured that TLS is actually one step "above" SSL 3.0 and represents the higher level of encryption rather than a lateral or lower level. It is enabled by default and I have seen no guidance anywhere suggesting that it be disabled for any reason.

One Gotcha

Note that this does not address the specific use of client certificates to handshake with SSL. That's a different scenario. Check out this **post** from Steven Erat that gives a rundown on that issue. If you need to use a client cert to connect then the main problem is that there is no way in the cfhttp or cfinvoke or cfldap tag to specify the location of the cert. In that case you will either need a custom solution (a Java library or something) or you will need to upgrade to ColdFusion 8 which happily supports a *clientcert* and *clientcertpassword* attribute.

Resources Summary

The following links were helpful to me. Some of them are mentioned in this article. Again, my thanks to Scott Krebs and the folks from circle of Gurus who always come through with great information.

- **Disabling SSL 2.0**
- **Steve Erat on SSL 3.0 and ColdFusion Tags**
- **How to Enable SSL Debug Information**
- **Understanding SSL Debug Info** (from Sun)
- **Differences Between TLS and SSL 3.0**
- **Analysis of SSL 3.0** (David Wagner - Berkeley & Bruce Schneier - Counterpan sys.)

Dean Saxe also pointed me to an excellent **Foundstone** tool called **SSL Digger**. It allows you to test SSL connections and examine the handshake protocols to see what is happening under the hood. Thanks Dean.

As always, this article is intended as a starting point to pull together resources. I'm interested in the findings of anyone who is working on this issue or related issues. I'm always grateful to my vigilant readers for expanding my knowledge.