Java Client - Bug #7058

cannot successfully connect to SAEON node that only uses TLSv1.2

2015-04-21 22:10 - Rob Nahf

Status:	Closed	**Start date:**	2015-05-26
Priority:	Normal	**Due date:**	
Assignee:	Rob Nahf	**% Done:**	100%
Category:	d1_libclient_java	**Estimated time:**	0.00 hour
Target version:			
Story Points:			

Description

The web tester can't connect when using java7. Jing reports that running a client under java8 fixes the problem.

There seems to be differences in protocol negotiation between Java 6,7, and 8.
HttpClient respects the Java7 implementation with regards to which are enabled by default, however, there was a buggy implementation. see:

https://issues.apache.org/jira/browse/HTTPCLIENT-1595

for a possible solution.

SAEON only supports TLSv1.2,

use
openssl s_client -connect metacat.saeon.ca.za:443 -tls1_2

to verify which protocols it supports (-tls1_1, -tls1 options to see that they don't work)

Subtasks:

Task # 7131: implement a TLS protocol preference property in auth.properties	**Closed**
Task # 7132: implement protocol preference in d1_libclient_java v1.4 branch	**Closed**
Task # 7133: implement protocol preference in d1_libclient_java v2.0 (trunk)	**Closed**

Related issues:

Related to Java Client - Story #7060: refactor CertificateManager to specify ...	**Rejected**	**2015-04-23**

History

#1 - 2015-04-21 22:26 - Rob Nahf

background on SSLContext: http://download.java.net/jdk9/docs/technotes/guides/security/jsse/JSSERefGuide.html#SSLContext

#2 - 2015-04-22 00:08 - Rob Nahf

tried using HttpClient v4.4.1 vs. https://metacat.saeon.ac.za/metacat/d1/mn/ locally through Eclipse.

but still getting a connection exception. Maybe there is more config needed.

java.lang.AssertionError: ServiceFailure: 0 Client_Error:: class org.dataone.client.exception.ClientSideException: /Connect to metacat.saeon.ac.za:443 [metacat.saeon.ac.za/196.21.191.73] failed: Connection refused [for host GET https://metacat.saeon.ac.za/metacat/d1/mn/v1/object?formatId=http://www.openarchives.org/ore/terms]
at org.junit.Assert.fail(Assert.java:91)
at org.dataone.integration.ContextAwareTestCaseDataone$4.call(ContextAwareTestCaseDataone.java:1401)
at org.junit.rules.ErrorCollector.checkSucceeds(ErrorCollector.java:70)
at org.dataone.integration.ContextAwareTestCaseDataone.handleFail(ContextAwareTestCaseDataone.java:1396)
at org.dataone.integration.it.ContextAwareAdapter.handleFail(ContextAwareAdapter.java:100)
at org.dataone.integration.it.testImplementations.ContentIntegrityTestImplementations.testResourceMap_Parsing(ContentIntegrityTestImplementations.java:80)
at org.dataone.integration.it.testImplementations.ContentIntegrityTestImplementations.testResourceMap_Parsing(ContentIntegrityTestImplementations.java:50)
at org.dataone.integration.it.apiTests.MNContentIntegrityV1V2IT.testResourceMap_Parsing(MNContentIntegrityV1V2IT.java:35)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:606)
at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java:44)
at org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java:15)
at org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java:41)
at org.junit.internal.runners.statements.InvokeMethod.evaluate(InvokeMethod.java:20)
at org.junit.rules.Verifier$1.evaluate(Verifier.java:34)
at org.junit.internal.runners.statements.RunBefores.evaluate(RunBefores.java:28)
at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:76)
at org.junit.runners.BlockJUnit4ClassRunner.runChild(BlockJUnit4ClassRunner.java:50)
at org.junit.runners.ParentRunner$3.run(ParentRunner.java:193)
at org.junit.runners.ParentRunner$1.schedule(ParentRunner.java:52)
at org.junit.runners.ParentRunner.runChildren(ParentRunner.java:191)
at org.junit.runners.ParentRunner.access$000(ParentRunner.java:42)
at org.junit.runners.ParentRunner$2.evaluate(ParentRunner.java:184)
at org.junit.runners.ParentRunner.run(ParentRunner.java:236)
at org.eclipse.jdt.internal.junit4.runner.JUnit4TestReference.run(JUnit4TestReference.java:50)
at org.eclipse.jdt.internal.junit.runner.TestExecution.run(TestExecution.java:38)
at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests(RemoteTestRunner.java:467)
at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.runTests(RemoteTestRunner.java:683)
at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.run(RemoteTestRunner.java:390)
at org.eclipse.jdt.internal.junit.runner.RemoteTestRunner.main(RemoteTestRunner.java:197)

#3 - 2015-04-22 16:05 - Rob Nahf

- % Done changed from 0 to 30

- Subject changed from cannot successfully connect to SEON node that only uses TLSv1.2 to cannot successfully connect to SAEON node that only uses TLSv1.2

- Description updated

- Status changed from New to In Progress

#4 - 2015-04-22 16:40 - Rob Nahf

It could be that the saeon node is closing the TLS negotiation handshake prematurely, not allowing our client to switch to TLSv1.2. From the first answer in http://stackoverflow.com/questions/27573192/how-do-i-set-ssl-protocol-version-in-java-and-how-do-i-know-which-one-javax-ne :

"the client announces TLS1.2 in the ClientHello and the server can do only TLS1.0 it should announce this, i.e. reply with TLS1.0. The client can then close the connection if TLS1.0 is not good enough or continue with TLS1.0. But, in this case the server just told the client that it does not like this version and closed the connection. "

#5 - 2015-04-23 02:45 - Rob Nahf

A deep dive into the source code of SSLContext.getInstance("TLS") shows that the "TLS" alias is used to find an SSLEngine implementation from one of the Security Providers (that more or less come with the java you're running).

TLS is an alias to an actual implementation, which in the case of Java 7, points to the TLS1.0 implementation, evidenced by the following code:

```
Provider[] ps = Security.getProviders();
for (Provider p : ps) {
System.out.println(p.getName() + ": " + p.getClass().getCanonicalName());
for (Entry n : p.entrySet()) {
if (n.getKey().toString().contains("SSLContext"))
System.out.println(String.format("    %s : %s", n.getKey(),
n.getValue()));

}
}
```

```
SUN: sun.security.provider.Sun
SunRsaSign: sun.security.rsa.SunRsaSign
SunEC: sun.security.ec.SunEC
SunJSSE: com.sun.net.ssl.internal.ssl.Provider
SSLContext.TLSv1.2 : sun.security.ssl.SSLContextImpl$TLS12Context
SSLContext.TLSv1.1 : sun.security.ssl.SSLContextImpl$TLS11Context
Alg.Alias.SSLContext.SSLv3 : TLSv1
Alg.Alias.SSLContext.SSL : TLSv1
Alg.Alias.SSLContext.TLS : TLSv1
SSLContext.Default : sun.security.ssl.SSLContextImpl$DefaultSSLContext
SSLContext.TLSv1 : sun.security.ssl.SSLContextImpl$TLS10Context
SunJCE: com.sun.crypto.provider.SunJCE
SunJGSS: sun.security.jgss.SunProvider
SunSASL: com.sun.security.sasl.Provider
XMLDSig: org.jcp.xml.dsig.internal.dom.XMLDSigRI
SunPCSC: sun.security.smartcardio.SunPCSC
Apple: apple.security.AppleProvider
BC: org.bouncycastle.jce.provider.BouncyCastleProvider
```

see SSLContext source here: http://developer.classpath.org/doc/javax/net/ssl/SSLContext-source.html
and SSLEngine source here: http://www.docjar.org/html/api/gnu/java/security/Engine.java.html

Please note: HttpClient uses SSLSocket instead of SSLEngine when establishing connections. The pertinent difference is that SSLSocket does not enable all of the supported protocols. SSLSockets are designed to add a security layer over the actual transport, so this I guess is to be expected.

For the nitty-gritty of the handshake, see: SSLContextImpl: http://www.docjar.com/html/api/sun/security/ssl/SSLSocketImpl.java.html

It is presumably the actual implementation of the protocol that negotiates which TLS version to use. (for example: sun.security.ssl.SSLContextImpl$TLS10Context from above)

#6 - 2015-04-23 02:56 - Rob Nahf

- Tracker changed from Story to Bug

- Target version deleted (CLJ-2.0.0)

#7 - 2015-04-23 02:58 - Rob Nahf

- Related to Story #7060: refactor CertificateManager to specify TLSv1.2 in v1.x branch added

#8 - 2015-04-23 03:01 - Rob Nahf

for testing a server's connection setup:

https://www.ssllabs.com/ssltest/analyze.html?d=metacat.saeon.ac.za

also this bash script from Dave to determine what nodes in an environment are running

```
for HOST in $(d1nodes -b "https://cn.dataone.org/cn" -B);
do echo ${HOST};
curl -v ${HOST}  2>&1 > /dev/null | grep "* TLS";
done
```

#9 - 2015-05-20 23:13 - Rob Nahf

SAEON has decided to support TLSv1.0 for the short-term. We do not need to update production to explicitly choose TLSv1.2 in CCI-1.5.1.

Solutions will be focused on d1_libclient_java 1.4.x branch and 2.x

#10 - 2017-03-28 17:58 - Dave Vieglais

- Status changed from In Progress to Closed

- % Done changed from 30 to 100