Java加密套件強度限制引起的SSL handshake_failure

codertw.com/程式語言/496111

HttpURLConnection的呼叫非常簡單。

```
HttpURLConnection connection =
(HttpURLConnection)m_url.openConnection();
connection.setRequestMethod("GET");
connection.setAllowUserInteraction(false);
connection.setDefaultUseCaches(false);
connection.setDoInput(true);
connection.setDoOutput(false);
connection.setInstanceFollowRedirects(true);
connection.setUseCaches(false);
connection.connect(); <----- handshake error
```

錯誤也很抽象。

```
javax.net.ssl.SSLHandshakeException: Received fatal alert: handshake_failure
at sun.security.ssl.Alerts.getSSLException(Unknown Source)
at sun.security.ssl.Alerts.getSSLException(Unknown Source)
at sun.security.ssl.SSLSocketImpl.recvAlert(Unknown Source)
at sun.security.ssl.SSLSocketImpl.readRecord(Unknown Source)
at sun.security.ssl.SSLSocketImpl.performInitialHandshake(Unknown Source)
at sun.security.ssl.SSLSocketImpl.startHandshake(Unknown Source)
at sun.security.ssl.SSLSocketImpl.startHandshake(Unknown Source)
at sun.net.www.protocol.https.HttpsClient.afterConnect(Unknown Source)
at sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.connect(Unknown
Source)
at sun.net.www.protocol.https.HttpsURLConnectionImpl.connect(Unknown Source)
at com.agile.common.HttpReader.getInputStream(HttpReader.java:76)
```

初步懷疑本地的JRE證書信任檔案中沒有包含對方伺服器的根證書。

```
keytool -list -v -keystore "C:\Java\jdk1.8.0_152\jre\lib\security\cacerts"
>store.txt
```

檢查發現，根證書和中間證書都存在，信任鏈沒有問題。

```
## 中間證書
Alias name: comodorsaca [jdk]
Creation date: 25 Aug, 2016
Entry type: trustedCertEntry
Owner: CN=COMODO RSA Certification Authority, O=COMODO CA Limited, L=Salford,
ST=Greater Manchester, C=GB
Issuer: CN=COMODO RSA Certification Authority, O=COMODO CA Limited, L=Salford,
ST=Greater Manchester, C=GB
Serial number: 4caaf9cadb636fe01ff74ed85b03869d
Valid from: Tue Jan 19 05:30:00 IST 2010 until: Tue Jan 19 05:29:59 IST 2038
Certificate fingerprints:
MD5:  1B:31:B0:71:40:36:CC:14:36:91:AD:C4:3E:FD:EC:18
SHA1: AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
SHA256:
52:F0:E1:C4:E5:8E:C6:29:29:1B:60:31:7F:07:46:71:B8:5D:7E:A8:0D:5B:07:27:34:63:53:4B

Signature algorithm name: SHA384withRSA
Version: 3
## 根證書
Alias name: addtrustqualifiedca [jdk]
Creation date: 25 Aug, 2016
Entry type: trustedCertEntry
Owner: CN=AddTrust Qualified CA Root, OU=AddTrust TTP Network, O=AddTrust AB, C=SE
Issuer: CN=AddTrust Qualified CA Root, OU=AddTrust TTP Network, O=AddTrust AB,
C=SE
Serial number: 1
Valid from: Tue May 30 16:14:50 IST 2000 until: Sat May 30 16:14:50 IST 2020
Certificate fingerprints:
MD5:  27:EC:39:47:CD:DA:5A:AF:E2:9A:01:65:21:A9:4C:BB
SHA1: 4D:23:78:EC:91:95:39:B5:00:7F:75:8F:03:3B:21:1E:C5:4D:8B:CF
SHA256:
80:95:21:08:05:DB:4B:BC:35:5E:44:28:D8:FD:6E:C2:CD:E3:AB:5F:B9:7A:99:42:98:8E:B8:F4

Signature algorithm name: SHA1withRSA
Version: 3
```

那就去抓SSL包吧。

請求包：

```
Secure Sockets Layer
TLSv1.2 Record Layer: Handshake Protocol: Client Hello
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 229
Handshake Protocol: Client Hello
Handshake Type: Client Hello (1)
Length: 225
Version: TLS 1.2 (0x0303)
Random: 5af01897734438e606e3342398727fe8a539522a2ef0dfa6...
GMT Unix Time: May  7, 2018 17:12:55.000000000 中國標準時間
Random Bytes: 734438e606e3342398727fe8a539522a2ef0dfa6b698a1eb...
Session ID Length: 0
Cipher Suites Length: 58
Cipher Suites (29 suites)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)
Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA256 (0x003c)
Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256 (0xc025)
Cipher Suite: TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256 (0xc029)
Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x0067)
Cipher Suite: TLS_DHE_DSS_WITH_AES_128_CBC_SHA256 (0x0040)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA (0xc004)
Cipher Suite: TLS_ECDH_RSA_WITH_AES_128_CBC_SHA (0xc00e)
Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
Cipher Suite: TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x0032)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02d)
Cipher Suite: TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256 (0xc031)
Cipher Suite: TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x009e)
Cipher Suite: TLS_DHE_DSS_WITH_AES_128_GCM_SHA256 (0x00a2)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA (0xc008)
Cipher Suite: TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (0xc012)
Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
Cipher Suite: TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA (0xc003)
Cipher Suite: TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA (0xc00d)
Cipher Suite: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x0016)
Cipher Suite: TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA (0x0013)
Cipher Suite: TLS_EMPTY_RENEGOTIATION_INFO_SCSV (0x00ff)
```

響應包：

```
Secure Sockets Layer
TLSv1.2 Record Layer: Alert (Level: Fatal, Description: Handshake Failure)
Content Type: Alert (21)
Version: TLS 1.2 (0x0303)
Length: 2
Alert Message
Level: Fatal (2)
Description: Handshake Failure (40)
```

客戶端傳送了Hello，伺服器翻了個白眼直接拒絕了。這個響應包，沒有任何線索。但是能明確一點就是客戶端和伺服器都通過TLS 1.2協議協商。不用再懷疑協議版本問題了。

聯想到瀏覽器訪問對方https系統是成功的，再次抓取瀏覽器請求和響應包。發現了一點有用的線索。

瀏覽器的請求包同樣採用TLS 1.2協議，但是加密套件 (Cipher Suites)和前面的差異很大，提供了17個可選加密套件。前面的包提供了29個可選列表。

```
Secure Sockets Layer
TLSv1.2 Record Layer: Handshake Protocol: Client Hello
Content Type: Handshake (22)
Version: TLS 1.0 (0x0301)
Length: 512
Handshake Protocol: Client Hello
Handshake Type: Client Hello (1)
Length: 508
Version: TLS 1.2 (0x0303)
Random: 7179caea804a5281b411adca67c11883f616e13e13756d0d...
GMT Unix Time: May  1, 2030 03:20:10.000000000 中國標準時間
Random Bytes: 804a5281b411adca67c11883f616e13e13756d0d5764d936...
Session ID Length: 32
Session ID: 324885cac7ecf4dd09e38acdd3e45ceb2e0c5e248b31d267...
Cipher Suites Length: 34
Cipher Suites (17 suites)
Cipher Suite: Reserved (GREASE) (0xcaca)
Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
```

接著看瀏覽器接收到的tcp包Server Hello。伺服器通過TLS 1.2協議協商，告訴瀏覽器它要使用 TLS_RSA_WITH_AES_256_GCM_SHA384 加密套件作為後續資料的加密演算法。

```
Secure Sockets Layer
TLSv1.2 Record Layer: Handshake Protocol: Server Hello
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 81
Handshake Protocol: Server Hello
Handshake Type: Server Hello (2)
Length: 77
Version: TLS 1.2 (0x0303)
Random: f1649150aeb3366381e54392bfdb8f49ae8ead9f47dbcb1f...
GMT Unix Time: May  3, 2098 04:10:56.000000000 中國標準時間
Random Bytes: aeb3366381e54392bfdb8f49ae8ead9f47dbcb1fc13f95a4...
Session ID Length: 32
Session ID: 264c4906bbd0fd2b8f94f66ea2992433b82690fb7fb844d7...
Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
Compression Method: null (0)
Extensions Length: 5
Extension: renegotiation_info (len=1)
Type: renegotiation_info (65281)
Length: 1
Renegotiation Info extension
Renegotiation info extension length: 0
```

馬上注意到在前面的請求包中，29個加密套件裡都沒有TLS_RSA_WITH_AES_256_GCM_SHA384。似乎問題就在這裡，就是通過java程式碼訪問https伺服器時，候選的加密套件中沒有伺服器希望的TLS_RSA_WITH_AES_256_GCM_SHA384。

真的是這樣嗎？為了弄清伺服器和本地JRE是否存在加密套件不匹配，還得使用有足夠說服力的診斷方法來驗證。

首先搬上OPENSSL來除錯。

```
openssl s_client -connect server.mycompany.com:443
```

診斷髮現，伺服器確實需要AES256-GCM-SHA384，這是一個很長長度的強加密，一般128位長度加密很牛逼了。

```
New, TLSv1.2, Cipher is AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
Protocol  : TLSv1.2
Cipher    : AES256-GCM-SHA384
Session-ID: 5BFE44E0BE1248156266BE6947FA113C1035DDDC3A3BD1888940EF8257CAA18C
```

對Java程式碼也來一次除錯，確認它所支援的所有加密套件。

```
-Dssl.debug=true -Djavax.net.debug=all
```

返回結果確實如此，根本就不存在TLS_RSA_WITH_AES_256_GCM_SHA384，也不存在基於AES256-GCM-SHA384的加密方法。

```
Cipher Suites: [
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,
TLS_RSA_WITH_AES_128_CBC_SHA256,
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256,
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256,
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA,
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
TLS_DHE_DSS_WITH_AES_128_CBC_SHA,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
TLS_RSA_WITH_AES_128_GCM_SHA256,
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256,
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256,
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_RSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA,
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA,
TLS_EMPTY_RENEGOTIATION_INFO_SCSV
]
```

問題找到了。解決方法看來現在只能回到JRE本身的加密授權問題上來了。

Java支援所有的加密套件，但是對於發行的JDK版本，它預設做了很多加密長度限制的裁剪，就是隻出口強度低的加密，這是美國政府對於安全軟體的強制性規定。但Oracle允許下載強加密的未限制版本，其實就是幾個授權屬性檔案，因為原始碼都在發行的JDK中。

當前問題發生在JDK1.8中，所以可以去官網下載一個壓縮包叫做 “Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8”。對於JDK1.8版本但是低於1.8.0_151版本的JDK，將下載的包裡的兩個檔案直接覆蓋到本地 Java\jre\lib\security\

```
local_policy.jar
US_export_policy.jar
```

1.8.0_151和以後的版本，無需下載任何檔案，只要修改Java\jre\lib\security\java.security檔案，修改這一行註釋並啟用就可以了。

```
crypto.policy=unlimited
```
