Commit e6a6e935 2 years ago by (mailto:kroot@google.com) **Kenny Root (mailto:kroot@google.com)**

Committed by **Neil Fuller (mailto:nfuller@google.com)** 2 years ago

Add support for TLS_FALLBACK_SCSV

```
Bug: 17750026
Change-Id: I8dec89ae59a6f745f63120b11b4f6dbe9b21a139
```

parent a912bd88 (/SaberMod/android-libcore/commit/a912bd88ce8001c65d367d06cde1680bd344b9ce)
sm-lp1 (/SaberMod/android-libcore/commits/sm-lp1) ...

Showing **5 changed files** with **118 additions** and **4 deletions**

luni/src/main/java/javax/net/ssl/SSLEngine.java

```
 ...  ...  @@ -542,6 +542,11 @@ import java.nio.ByteBuffer;
542  542   *             <td>20+</td>
543  543   *         </tr>
544  544   *         <tr>
     545  +  *             <td>TLS_FALLBACK_SCSV</td>
     546  +  *             <td>21+</td>
     547  +  *             <td></td>
     548  +  *         </tr>
     549  +  *         <tr>
545  550   *             <td>TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA</td>
546  551   *             <td>21+</td>
547  552   *             <td>21+</td>
 ...  ...
```

luni/src/main/java/javax/net/ssl/SSLSocket.java

```
 ...  ...  @@ -521,6 +521,11 @@ import java.net.UnknownHostException;
521  521   *             <td>11+</td>
522  522   *         </tr>
523  523   *         <tr>
     524  +  *             <td>TLS_FALLBACK_SCSV</td>
     525  +  *             <td>21+</td>
     526  +  *             <td></td>
```

```
      527  +  *         </tr>
      528  +  *         <tr>
524   529     *             <td>TLS_PSK_WITH_3DES_EDE_CBC_SHA</td>
525   530     *             <td>21+</td>
526   531     *             <td></td>
...   ...
```

luni/src/test/java/libcore/javax/net/ssl/SSLEngineTest.java

```
...   ...   @@ -154,11 +154,11 @@ public class SSLEngineTest extends TestCase {
154   154                     continue;
155   155                 }
156   156                 /*
157         -               * TLS_EMPTY_RENEGOTIATION_INFO_SCSV cannot be used on
158         -               * its own, but instead in conjunction with other
159         -               * cipher suites.
      157   +               * Signaling Cipher Suite Values (SCSV) cannot be used on their
            own, but instead in
      158   +               * conjunction with other cipher suites.
160   159                 */
161         -               if
            (cipherSuite.equals(StandardNames.CIPHER_SUITE_SECURE_RENEGOTIATION)) {
      160   +               if
            (cipherSuite.equals(StandardNames.CIPHER_SUITE_SECURE_RENEGOTIATION)
      161   +                       ||
            cipherSuite.equals(StandardNames.CIPHER_SUITE_FALLBACK)) {
162   162                     continue;
163   163                 }
164   164                 /*
...   ...
```

luni/src/test/java/libcore/javax/net/ssl/SSLSocketTest.java

```
...   ...   @@ -126,6 +126,14 @@ public class SSLSocketTest extends TestCase {
126   126                         continue;
127   127                     }
128   128                     /*
      129   +                   * Similarly with the TLS_FALLBACK_SCSV suite, it is not
      130   +                   * a selectable suite, but is used in conjunction with
      131   +                   * other cipher suites.
      132   +                   */
      133   +                  if
            (cipherSuite.equals(StandardNames.CIPHER_SUITE_FALLBACK)) {
      134   +                      continue;
      135   +                  }
      136   +                  /*
129   137                       * Kerberos cipher suites require external setup. See
            "Kerberos Requirements" in
130   138                       *
            https://java.sun.com/j2se/1.5.0/docs/guide/security/jsse/JSSERefGuide.html
131   139                       * #KRBRequire
...   ...   @@ -1562,6 +1570,90 @@ public class SSLSocketTest extends TestCase {
1562  1570            }
1563  1571        }
1564  1572
```

```
1573 +    public void test_SSLSocket_sendsTlsFallbackScsv_Fallback_Success()
     throws Exception {
1574 +        TestSSLContext context = TestSSLContext.create();
1575 +
1576 +        final SSLSocket client = (SSLSocket)
1577 +
     context.clientContext.getSocketFactory().createSocket(context.host,
     context.port);
1578 +        final SSLSocket server = (SSLSocket) context.serverSocket.accept();
1579 +
1580 +        final String[] serverCipherSuites =
     server.getEnabledCipherSuites();
1581 +        final String[] clientCipherSuites = new
     String[serverCipherSuites.length + 1];
1582 +        System.arraycopy(serverCipherSuites, 0, clientCipherSuites, 0,
     serverCipherSuites.length);
1583 +        clientCipherSuites[serverCipherSuites.length] =
     StandardNames.CIPHER_SUITE_FALLBACK;
1584 +
1585 +        ExecutorService executor = Executors.newFixedThreadPool(2);
1586 +        Future<Void> s = executor.submit(new Callable<Void>() {
1587 +                public Void call() throws Exception {
1588 +                    server.setEnabledProtocols(new String[] { "TLSv1.2" });
1589 +                    server.setEnabledCipherSuites(serverCipherSuites);
1590 +                    server.startHandshake();
1591 +                    return null;
1592 +                }
1593 +            });
1594 +        Future<Void> c = executor.submit(new Callable<Void>() {
1595 +                public Void call() throws Exception {
1596 +                    client.setEnabledProtocols(new String[] { "TLSv1.2" });
1597 +                    client.setEnabledCipherSuites(clientCipherSuites);
1598 +                    client.startHandshake();
1599 +                    return null;
1600 +                }
1601 +            });
1602 +        executor.shutdown();
1603 +
1604 +        s.get();
1605 +        c.get();
1606 +        client.close();
1607 +        server.close();
1608 +        context.close();
1609 +    }
1610 +
1611 +    public void
     test_SSLSocket_sendsTlsFallbackScsv_InappropriateFallback_Failure() throws
     Exception {
1612 +        TestSSLContext context = TestSSLContext.create();
1613 +
1614 +        final SSLSocket client = (SSLSocket)
1615 +
     context.clientContext.getSocketFactory().createSocket(context.host,
     context.port);
1616 +        final SSLSocket server = (SSLSocket) context.serverSocket.accept();
```

```
1617 +
1618 +        final String[] serverCipherSuites = server.getEnabledCipherSuites();
1619 +        final String[] clientCipherSuites = new String[serverCipherSuites.length + 1];
1620 +        System.arraycopy(serverCipherSuites, 0, clientCipherSuites, 0, serverCipherSuites.length);
1621 +        clientCipherSuites[serverCipherSuites.length] = StandardNames.CIPHER_SUITE_FALLBACK;
1622 +
1623 +        ExecutorService executor = Executors.newFixedThreadPool(2);
1624 +        Future<Void> s = executor.submit(new Callable<Void>() {
1625 +                public Void call() throws Exception {
1626 +                    server.setEnabledProtocols(new String[] { "TLSv1", "SSLv3" });
1627 +                    server.setEnabledCipherSuites(serverCipherSuites);
1628 +                    try {
1629 +                        server.startHandshake();
1630 +                        fail("Should result in inappropriate fallback");
1631 +                    } catch (SSLHandshakeException expected) {
1632 +                    }
1633 +                    return null;
1634 +                }
1635 +            });
1636 +        Future<Void> c = executor.submit(new Callable<Void>() {
1637 +                public Void call() throws Exception {
1638 +                    client.setEnabledProtocols(new String[] { "SSLv3" });
1639 +                    client.setEnabledCipherSuites(clientCipherSuites);
1640 +                    try {
1641 +                        client.startHandshake();
1642 +                        fail("Should receive TLS alert inappropriate fallback");
1643 +                    } catch (SSLHandshakeException expected) {
1644 +                    }
1645 +                    return null;
1646 +                }
1647 +            });
1648 +        executor.shutdown();
1649 +
1650 +        s.get();
1651 +        c.get();
1652 +        client.close();
1653 +        server.close();
1654 +        context.close();
1655 +    }
1656 +
1565 1657      /**
1566 1658       * Not run by default by JUnit, but can be run by Vogar by
1567 1659       * specifying it explicitly (or with main method below)
 ... ...
```

support/src/test/java/libcore/java/security/StandardNames.java

```
 ... ...  @@ -82,6 +82,14 @@ public final class StandardNames extends Assert {
  82  82              = "TLS_EMPTY_RENEGOTIATION_INFO_SCSV";
```

```
 83   83
 84   84        /**
      85 +       * From https://tools.ietf.org/html/draft-ietf-tls-downgrade-scsv-00 it is a
      86 +       * signaling cipher suite value (SCSV) to indicate that this request is a
      87 +       * protocol fallback (e.g., TLS 1.0 -> SSL 3.0) because the server didn't respond
      88 +       * to the first request.
      89 +       */
      90 +      public static final String CIPHER_SUITE_FALLBACK = "TLS_FALLBACK_SCSV";
      91 +
      92 +      /**
 85   93         * A map from algorithm type (e.g. Cipher) to a set of algorithms (e.g. AES, DES, ...)
 86   94         */
 87   95        public static final Map<String,Set<String>> PROVIDER_ALGORITHMS
...  ...    @@ -723,6 +731,10 @@ public final class StandardNames extends Assert {
723  731            // RFC 5746's Signaling Cipher Suite Value to indicate a request for secure renegotiation
724  732            addBoth(CIPHER_SUITE_SECURE_RENEGOTIATION);
725  733
      734 +          // From https://tools.ietf.org/html/draft-ietf-tls-downgrade-scsv-00 to indicate
      735 +          // TLS fallback request
      736 +          addOpenSsl(CIPHER_SUITE_FALLBACK);
      737 +
726  738            // non-defaultCipherSuites
727  739            addBoth(   "TLS_ECDH_anon_WITH_AES_256_CBC_SHA");
728  740            addBoth(   "TLS_DH_anon_WITH_AES_256_CBC_SHA");
...  ...
```