Commit

✕ Merge commit from fork

Browse files

```
Motivation:

We need to handle the situation correctly in all cases when there is not enough data yet to
unwrap the packet. Not doing so might cause undefined behaviour

Modifications:

- Correctly check for SslUtils.NOT_ENOUGH_DATA
- Add assert

Result:

Correctly handle incomplete packets while unwrap
```

4.1
netty-4.1.118.Final

normanmaurer authored yesterday Verified 1 parent d1fbda6 commit 87f4072

Showing **2 changed files** with **19 additions** and **6 deletions**.

Whitesp... Ignore whitespace Split Unifi...

Filter changed files

- handler/src/main/java/io...
  - ReferenceCounted...
  - SslUtils.java

2

handler/src/main/java/io/netty/handler/ssl/ReferenceCountedO…

```
1202 1202                     throw new NotSslRecordException("not an SSL/TLS record");
1203 1203                 }
1204 1204
     1205 +               assert packetLength >= 0;
     1206 +
1205 1207                 final int packetLengthDataOnly = packetLength - SSL_RECORD_HEADER_LENGTH;
1206 1208                 if (packetLengthDataOnly > capacity) {
1207 1209                     // Not enough space in the destination buffer so signal the caller that the buffer needs to be
```

23

handler/src/main/java/io/netty/handler/ssl/SslUtils.java

```
314  314        *                       the given {@link ByteBuf} is not encrypted at all.
315  315        */
316  316       static int getEncryptedPacketLength(ByteBuf buffer, int offset, boolean probeSSLv2) {
     317   +        assert offset >= buffer.readerIndex();
     318   +        int remaining = buffer.writerIndex() - offset;
     319   +        if (remaining < SSL_RECORD_HEADER_LENGTH) {
     320   +            return NOT_ENOUGH_DATA;
     321   +        }
317  322            int packetLength = 0;
318       -
319  323            // SSLv3 or TLS - Check ContentType
320  324            boolean tls;
321  325            switch (buffer.getUnsignedByte(offset)) {
346  350                    tls = false;
347  351                }
348  352            } else if (version == DTLS_1_0 || version == DTLS_1_2 || version == DTLS_1_3) {
349       -            if (buffer.readableBytes() < offset + DTLS_RECORD_HEADER_LENGTH) {
     353   +            if (remaining < DTLS_RECORD_HEADER_LENGTH) {
350  354                    return NOT_ENOUGH_DATA;
351  355                }
352  356                // length is the last 2 bytes in the 13 byte header.
367  371                packetLength = headerLength == 2 ?
368  372                        (shortBE(buffer, offset) & 0x7FFF) + 2 : (shortBE(buffer, offset) & 0x3FFF) + 3;
369  373                if (packetLength <= headerLength) {
370       -                return NOT_ENOUGH_DATA;
     374   +                // If there's no data then consider this package as not encrypted.
     375   +                return NOT_ENCRYPTED;
371  376                }
372  377            } else {
373  378                return NOT_ENCRYPTED;
420  425        }
421  426
422  427        // We need to copy 5 bytes into a temporary buffer so we can parse out the packet length easily.
423       -        ByteBuffer tmp = ByteBuffer.allocate(5);
```

```
+        ByteBuffer tmp = ByteBuffer.allocate(SSL_RECORD_HEADER_LENGTH);

         do {
             buffer = buffers[offset++].duplicate();
             if (buffer.remaining() > tmp.remaining()) {
                 buffer.limit(buffer.position() + tmp.remaining());
             }
             tmp.put(buffer);
-        } while (tmp.hasRemaining());
+        } while (tmp.hasRemaining() && offset < buffers.length);

         // Done, flip the buffer so we can read from it.
         tmp.flip();
         return getEncryptedPacketLength(tmp);
     }

     private static int getEncryptedPacketLength(ByteBuffer buffer) {
+        int remaining = buffer.remaining();
+        if (remaining < SSL_RECORD_HEADER_LENGTH) {
+            return NOT_ENOUGH_DATA;
+        }
         int packetLength = 0;
         int pos = buffer.position();
+
         // SSLv3 or TLS - Check ContentType
         boolean tls;
         switch (unsignedByte(buffer.get(pos))) {
                 packetLength = headerLength == 2 ?
                         (shortBE(buffer, pos) & 0x7FFF) + 2 : (shortBE(buffer, pos) & 0x3FFF) + 3;
                 if (packetLength <= headerLength) {
-                    return NOT_ENOUGH_DATA;
+                    // If there's no data then consider this package as not encrypted.
+                    return NOT_ENCRYPTED;
                 }
             } else {
                 return NOT_ENCRYPTED;
```

0 comments on commit 87f4072

Please sign in to comment.