Network Device Interpretation # 201643c

TLS Server Tests - Issue 3: Verification of application of encryption

Status: ☒ *Active* ☐ *Inactive*

Date: *4-Oct-2017*

Type of Document: ☒ *Technical Decision* ☐ *Technical Recommendation*

Approved by: ☒ *Network iTC Interpretations Team* ☒ *Network iTC*

Affected Document(s): *ND SD V1.0*

Affected Section(s): *FCS_TLSS_EXT.1, FCS_TLSS_EXT.2*

Superseded Interpretation(s): *None*

Issue:

TD0040 also identifies changes for [MDM] FCS_TLSS_EXT.1.1 Test 4 bullet 5. This corresponds to [ND] FCS_TLSS_EXT.1.1 Test 4e and [ND] FCS_TLSS_EXT.2.1 Test 4e. This change should not be carried forward, since bullet 5/test 4e does not duplicate bullet 4/test 4d. Bullet 4/Test 4d requires specifies sending a Finished message before the ChangeCipherSpec (and not sending a ChangeCipherSpec message. Bullet 5/Test 4e specifies sending a ChangeCipherSpec message, but sending a garbled message instead of a finished message.

„Test 4: The evaluator shall perform the following modifications to the traffic:

...
e) Send a garbled message from the client after the client has issued the ChangeCipherSpec message and verify that the Server denies the connection."

Resolution:

The NIT acknowledges the issue described in the 'Issue' section above. FCS_TLSS_EXT.1.1 Test 4e and FCS_TLSS_EXT.2.1 Test 4e shall therefore be modified as follows:

"Test Intent: The intent of this test is to ensure that the server's TLS implementation immediately makes use of the key exchange and authentication algorithms to:

a) Correctly encrypt TLS Finished message
b) Encrypt every TLS message after session keys are negotiated

Test 4 e): The evaluator shall use one of the claimed ciphersuites to complete a successful handshake and observe transmission of properly encrypted application data. The evaluator shall verify that no Alert with alert level Fatal (2) messages were sent.

The evaluator shall verify that the Finished message (handshake type hexadecimal 16) is sent immediately after the server's ChangeCipherSpec (handshake type hexadecimal 14) message. The evaluator shall examine the Finished message (encrypted example in hexadecimal, 16 03 03 00 40 xx xx xx xx xx... where xx represents ciphertext) and confirm that it does not contain unencrypted data (unencrypted example in hexadecimal, 16 03 03 00 40 14 00 00 0c yy yy yy where yy represents cleartext), where '14' is the hexidecimal message type code in the verify_data header and '00 00 0c' is the verify_data field length. According to RFC 5246, chap. 7.4.9, the standard length for the verify_data is 12 which is represented by the verify_data field length of '00 00 0c'. If a cipher suite is chosen that explicitly specifies this length, the corresponding value shall be used for verification instead of '00 00 0c'.

encrypted example: 16 03 03 00 40 xx xx xx xx xx

unencrypted example: 16 03 03 00 40 <u>14 00 00 0c</u> yy yy yy

Note: With the fixed value '14' for the message type code and the known verify_data field length, this test can be regarded as 'known value' test which is independent from the input data."

This resolution has been developed with support of the Network iTC's TLS Working Group.

Rationale:

Background Information:

```
Client                                                Server

      ClientHello                  -------->
                                                       ServerHello
                                                      Certificate*
                                                ServerKeyExchange*
                                               CertificateRequest*
                                   <--------      ServerHelloDone
      Certificate*
      ClientKeyExchange
      CertificateVerify*
```

From RFC TLS1.2 5246 on Finished:

When this message will be sent:

A Finished message is always sent immediately after a change cipher spec message to verify that the key exchange and authentication processes were successful. It is essential that a change cipher spec message be received between the other handshake messages and the Finished message.

Meaning of this message:

The Finished message is the first one protected with the just negotiated algorithms, keys, and secrets. Recipients of Finished messages MUST verify that the contents are correct. Once a side has sent its Finished message and received and validated the Finished message from its peer, it may begin to send and receive application data over the connection.

The ChangeCipherSpec message signals that the next record will be encrypted with the agreed upon algorithms (RFC 5246 section 7.1). It is its own record type (message type 20, instead of 22 for handshaking messages). This intentional record numbering is designed to force an implementation to construct independent records for the message immediately preceding the ChangeCipherSpec, an independent record for the ChangeCipherSpec itself, and then an independent record start for the messages immediately after the ChangeCipherSpec.

With that understanding, the test sets up the ChangeCipherSpec message which should signal the implementation to change to the new negotiated cipher suite. The next record (whatever it is -- and it is usually contains a Finished message in the initial handshake -- but need not be since it might be, say, an Alert), had better use the appropriate ciphersuite and be verifiable. Thus, if the evaluator sends a garbage message immediately after a ChangeCipherSpec, the immediate purpose of the next message will be that it is encrypted with the agreed upon ciphers. If this is not the case (e.g. a decrypt error occurs), then the channel must be torn down.

The Alert message description might be "bad_record_mac", "decrypt_error" or "record_overflow" depending on the implementation.

TLS Alert Meanings (in TLS v1.2):

bad_record_mac: This alert is returned if a record is received with an incorrect MAC. This alert also MUST be returned if an alert is sent because a TLSCiphertext decrypted in an invalid way...

record_overflow: ...or a record decrypted to a TLSCompressed record with more than 2^14+1024 bytes. This message is always fatal and should never be observed in communication between proper implementations (except when messages were corrupted in the network).

decrypt_error: A handshake cryptographic operation failed, including being unable to correctly verify a signature or validate a Finished message.

If it **is** the intent of the test to prove the implementation adheres to the ChangeCipherSpec signalling mechanism, then in the updated test we would want to try to remove as many other variables as possible. Because the Finished message can introduce different types of failures (e.g. handshake_failure, mac_error, decrypt_error, etc.), the only way to try to force a singular known reaction -- a decryption error -- is to try to force another message into the stream (e.g. an Alert). However, without fully comprehending the effect of Alerts after ChangeCipherSpec against numerous industry implementations, it is risky to suggest specific alternative test at this point.

There are numerous ways in which specific implementations could test that encryption keys are being used immediately after the ChangeCipherSpec message has been transmitted by the peer. The challenge is in trying to find a detailed mechanism that is both RFC compliant and also widely supported by a variety of software and hardware-based implementations. At the same time, because error conditions related to cryptographic states are often more opaque than we would like (to prevent certain classes of attacks), it is difficult to find a test method that isolates the appropriate feedback that isn't also seen under other (non-test) conditions.

A prime consideration of the updated test case is that it is not concerned with the quality of the encryption keys in use. In fact, it is assumed that the implementation is encrypting and decrypting things properly, but the main concern is the timing of the encryption and decryption. Therefore, if it is possible to trace that a supposedly encrypted message cannot be decoded into expected values (eg. message type 0x14 is a 'Finished' message with an expected length, etc.) and that the known verify_data is not found in the (supposedly) encrypted packet data, then this will confirm that encryption is occurring at a certain point of the handshake -- which is the point of the test.

To address concerns that Finished message result might be ignored by the server, then a secondary test is just to twiddle a bit in the Finished message and ensure that the result is an error must be performed. Of course, this test is already being performed in FCS_TLSS_EXT.1.1 Test 4c so there is no need to duplicate this behavior again for the benefits of this test case.

Further Action:

None

Action by Network iTC:

None