Commit 810fd66

Browse files

jlucas9 authored yesterday · 5 / 5 · Verified

```
Merge commit from fork

TM Apply/Process unit tests to show vuln no longer works
```

dev 2 parents 4bce885 + 7304bcc commit 810fd66

Filter files...

- include
  - crypto_config.h
  - crypto_error.h
- src/core
  - crypto_error.c
  - crypto_tc.c
  - crypto_tm.c
- test/unit
  - ut_tc_process.c
  - ut_tm_apply.c
  - ut_tm_process.c

8 files changed +349 -41 lines changed

Search within code

include/crypto_config.h +1

```
@@ -206,6 +206,7 @@
206  206  #define TM_FRAME_DATA_SIZE 1786 /* bytes */
207  207  #define TM_FILL_SIZE       1145 /* bytes */
208  208  #define TM_PAD_SIZE        2    /* bytes */
     209  + #define TM_SECONDARY_HDR_MAX_VALUE 63
209  210
210  211  // AOS Defines
211  212  #define AOS_FRAME_DATA_SIZE 1786 /* bytes */
```

include/crypto_error.h +3 -1

```
@@ -149,8 +149,10 @@
149  149   #define CRYPTO_LIB_ERR_AOS_FL_LT_MAX_FRAME_SIZE                    (-76)
150  150   #define CRYPTO_LIB_ERR_TM_FL_LT_MAX_FRAME_SIZE                     (-77)
151  151   #define CRYPTO_LIB_ERR_INVALID_FHECF                               (-78)
       152 + #define CRYPTO_LIB_ERR_TM_SECONDARY_HDR_SIZE                     (-79)
       153 + #define CRYPTO_LIB_ERR_TM_SECONDARY_HDR_VN                       (-80)
152  154
153        - #define CRYPTO_CORE_ERROR_CODES_MAX -78
       155 + #define CRYPTO_CORE_ERROR_CODES_MAX -80
154  156
155  157   // Define codes for returning MDB Strings, and determining error based on strings
156  158   #define CAM_ERROR_CODES     600
```

src/core/crypto_error.c +3 -1

```
@@ -99,7 +99,9 @@ char *crypto_enum_errlist_core[] = {(char *)"CRYPTO_LIB_SUCCESS",
 99   99                                    (char *)"CRYPTO_LIB_ERR_TM_FRAME_LENGTH_UNDERFLOW",
100  100                                    (char *)"CRYPTO_LIB_ERR_AOS_FL_LT_MAX_FRAME_SIZE",
101  101                                    (char *)"CRYPTO_LIB_ERR_TM_FL_LT_MAX_FRAME_SIZE",
102        -                                (char *)"CRYPTO_LIB_ERR_INVALID_FHECF"};
     102 +                                (char *)"CRYPTO_LIB_ERR_INVALID_FHECF",
     103 +                                (char *)"CRYPTO_LIB_ERR_TM_SECONDARY_HDR_SIZE",
     104 +                                (char *)"CRYPTO_LIB_ERR_TM_SECONDARY_HDR_VN"};
103  105
104  106   char *crypto_enum_errlist_config[] = {
105  107       (char *)"CRYPTO_CONFIGURATION_NOT_COMPLETE",
```

src/core/crypto_tc.c +1 -1

```
@@ -1805,7 +1805,7 @@ int32_t Crypto_TC_ProcessSecurity_Cam(uint8_t *ingest, int
*len_ingest, TC_t *tc
1805  1805            return status;
1806  1806        } // Unable to get necessary Managed Parameters for TC TF -- return with error.
1807  1807
1808        -     if (*len_ingest < current_managed_parameters_struct.max_frame_size ||
(tc_sdls_processed_frame->tc_header.fl + 1) < *len_ingest)
      1808  +     if ((tc_sdls_processed_frame->tc_header.fl + 1) < *len_ingest)
1809  1809        {
1810  1810            status = CRYPTO_LIB_ERR_TC_FRAME_LENGTH_UNDERFLOW;
```

```
1811  1811            mc_if->mc_log(status);
```

src/core/crypto_tm.c +77 -3

```
@@ -827,8 +827,42 @@ int32_t Crypto_TM_ApplySecurity(uint8_t *pTfBuffer, uint16_t len_ingest)
827  827        // Note: Secondary headers are static only for a mission phase, not guaranteed static
828  828        // over the life of a mission Per CCSDS 132.0-B.3 Section 4.1.2.7.2.3
829  829        // Secondary Header flag is 1st bit of 5th byte (index 4)
     830  +     uint8_t secondary_hdr_start = 6; // starts at 6th byte
     831  +     Crypto_TM_Check_For_Secondary_Header(pTfBuffer, &idx); // Sets idx to 6 + secondary_hdr_len + 1
830  832
831       -     Crypto_TM_Check_For_Secondary_Header(pTfBuffer, &idx);
     833  +     uint16_t secondary_hdr_len = idx - secondary_hdr_start;
     834  +     // Determine Secondary Header Version Number, should always be 0b00
     835  +     uint8_t shvn = (pTfBuffer[secondary_hdr_start] & 0xC0) >> 6;
     836  + #ifdef TM_DEBUG
     837  +     printf("Secondary Header Version Number: %d\n", shvn);
     838  +     printf("len_ingest: %d \n", len_ingest);
     839  +     printf("byte_idx: %d\n", idx);
     840  +     printf("Actual secondary header length: %d\n", secondary_hdr_len);
     841  + #endif
     842  +     if (shvn > 0 && idx > secondary_hdr_start) // idx will be > 6 if secondary header is present
     843  +     {
     844  +         status = CRYPTO_LIB_ERR_TM_SECONDARY_HDR_VN;
     845  +         mc_if->mc_log(status);
     846  +         return status;
     847  +     }
     848  +
     849  +     if (secondary_hdr_len > TM_SECONDARY_HDR_MAX_VALUE)
     850  +     {
     851  +         status = CRYPTO_LIB_ERR_TM_SECONDARY_HDR_SIZE;
     852  +         mc_if->mc_log(status);
     853  +         return status;
     854  +     }
     855  +
     856  +     // Protects from overruns on very short max frame sizes
     857  +     // Smallest frame here is Header | Secondary Header | 1 byte data
     858  +     if (len_ingest < ( TM_FRAME_PRIMARYHEADER_SIZE + secondary_hdr_len + 1))
     859  +     {
```

```
        860   + #ifdef TM_DEBUG
        861   + #endif
        862   +         status = CRYPTO_LIB_ERR_TM_SECONDARY_HDR_SIZE;
        863   +         mc_if->mc_log(status);
        864   +         return status;
        865   +     }
 832    866
 833    867         /**
 834    868          * Begin Security Header Fields
      @@ -1004,7 +1038,6 @@ int32_t Crypto_TM_Process_Setup(uint16_t len_ingest, uint16_t
      *byte_idx, uint8_t
1004   1038         // Check if secondary header is present within frame
1005   1039         // Note: Secondary headers are static only for a mission phase, not guaranteed
      static
1006   1040         // over the life of a mission Per CCSDS 132.0-B.3 Section 4.1.2.7.2.3
1007          -
1008   1041         if (status == CRYPTO_LIB_SUCCESS)
1009   1042         {
1010   1043             // Secondary Header flag is 1st bit of 5th byte (index 4)
      @@ -1016,13 +1049,50 @@ int32_t Crypto_TM_Process_Setup(uint16_t len_ingest, uint16_t
      *byte_idx, uint8_t
1016   1049     #endif
1017   1050                 // Secondary header is present
1018   1051                 *byte_idx = 6;
        1052  +
        1053  +             // Determine Secondary Header Version Number, should always be 0b00
        1054  +             uint8_t shvn = (p_ingest[*byte_idx] & 0xC0) >> 6;
        1055  + #ifdef TM_DEBUG
        1056  +             printf("Secondary Header Version Number: %d\n", shvn);
        1057  + #endif
        1058  +             if (shvn > 0)
        1059  +             {
        1060  +                 status = CRYPTO_LIB_ERR_TM_SECONDARY_HDR_VN;
        1061  +                 mc_if->mc_log(status);
        1062  +                 return status;
        1063  +             }
1019   1064                 // Determine length of secondary header
1020   1065                 // Length coded as total length of secondary header - 1
1021   1066                 // Reference CCSDS 132.0-B-2 4.1.3.2.3
1022   1067                 *secondary_hdr_len = (p_ingest[*byte_idx] & 0x3F) + 1;
1023   1068     #ifdef TM_DEBUG
1024          -             printf(KYEL "Secondary Header Length is decoded as: %d\n",
      *secondary_hdr_len);
```

```
1069  +             printf(KYEL "Secondary Header Length is decoded as: %d\n",
      *secondary_hdr_len - 1);
1070  +             printf("len_ingest: %d \n", len_ingest);
1071  +             printf("byte_idx: %d\n", *byte_idx);
1072  +             printf("Actual secondary header length: %d\n", *secondary_hdr_len);
1025 1073    #endif
1074  +             // We have a secondary header length now, is it sane?
1075  +             // Does it violate spec maximum?
1076  +             // Reference CCSDS 1320b3 4.1.3.1.3
1077  +             if (*secondary_hdr_len > TM_SECONDARY_HDR_MAX_VALUE + 1)
1078  +             {
1079  +                 status = CRYPTO_LIB_ERR_TM_SECONDARY_HDR_SIZE;
1080  +                 mc_if->mc_log(status);
1081  +                 return status;
1082  +             }
1083  +
1084  +             // Does it 'fit' in the overall frame correctly?
1085  +             // We can't validate it down to the byte yet,
1086  +             // we don't know the variable lengths from the SA yet
1087  +             // Protects from overruns on very short max frame sizes
1088  +             // Smallest frame here is Header | Secondary Header | 1 byte data
1089  +             if (len_ingest < ( TM_FRAME_PRIMARYHEADER_SIZE + *secondary_hdr_len + 1))
1090  +             {
1091  +                 status = CRYPTO_LIB_ERR_TM_SECONDARY_HDR_SIZE;
1092  +                 mc_if->mc_log(status);
1093  +                 return status;
1094  +             }
1095  +
1026 1096              // Increment from current byte (1st byte of secondary header),
1027 1097              // to where the SPI would start
1028 1098              *byte_idx += *secondary_hdr_len;
      @@ -1032,6 +1102,7 @@ int32_t Crypto_TM_Process_Setup(uint16_t len_ingest, uint16_t
      *byte_idx, uint8_t
1032 1102              // No Secondary header, carry on as usual and increment to SPI start
1033 1103              *byte_idx = 6;
1034 1104          }
1105  +
1035 1106      }
1036 1107
1037 1108      return status;
      @@ -1488,6 +1559,7 @@ int32_t Crypto_TM_ProcessSecurity(uint8_t *p_ingest, uint16_t
      len_ingest, uint8_
1488 1559      tm_frame_pri_hdr.vcid = ((uint8_t)p_ingest[1] & 0x0E) >> 1;
1489 1560
```

```
1490  1561          status = Crypto_TM_Process_Setup(len_ingest, &byte_idx, p_ingest,
                    &secondary_hdr_len);
      1562    +
1491  1563          if (status == CRYPTO_LIB_SUCCESS)
1492  1564          {
1493  1565              /**
      @@ -1537,13 +1609,15 @@ int32_t Crypto_TM_ProcessSecurity(uint8_t *p_ingest, uint16_t
      len_ingest, uint8_
1537  1609          }
1538  1610      #endif
1539  1611
      1612    +
1540  1613          if (current_managed_parameters_struct.max_frame_size <= byte_idx - sa_ptr-
                    >stmacf_len)
1541  1614          {
1542  1615              status = CRYPTO_LIB_ERR_TM_FRAME_LENGTH_UNDERFLOW;
1543  1616              mc_if->mc_log(status);
1544  1617              return status;
1545  1618          }
1546  1619
      1620    +     // Received the wrong amount of bytes from mandated frame size
1547  1621          if (len_ingest < current_managed_parameters_struct.max_frame_size)
1548  1622          {
1549  1623              status = CRYPTO_LIB_ERR_TM_FRAME_LENGTH_UNDERFLOW;
```

test/unit/ut_tc_process.c +15 -15

```
      @@ -42,7 +42,7 @@ UTEST(TC_PROCESS, EXERCISE_IV)
42    42        // Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003, 0, TC_HAS_FECF,
                TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 1024,
43    43        // AOS_FHEC_NA, AOS_IZ_NA, 0);
44    44        GvcidManagedParameters_t TC_UT_Managed_Parameters = {
45          -       0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 38,
                TC_OCF_NA, 1};
      45    +       0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS,
                1024, TC_OCF_NA, 1};
46    46        Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
47    47        TC_UT_Managed_Parameters.vcid = 1;
48    48        Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
      @@ -173,7 +173,7 @@ UTEST(TC_PROCESS, EXERCISE_ARSN)
173   173       // AOS_FHEC_NA, AOS_IZ_NA, 0); Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003,
                1, TC_HAS_FECF,
```

```
174  174          // TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 1024, AOS_FHEC_NA, AOS_IZ_NA, 0);
175  175          GvcidManagedParameters_t TC_UT_Managed_Parameters = {
176       -          0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 44,
                 TC_OCF_NA, 1};
     176  +          0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS,
                 1024, TC_OCF_NA, 1};
177  177          Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
178  178          TC_UT_Managed_Parameters.vcid = 1;
179  179          Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
          @@ -303,7 +303,7 @@ UTEST(TC_PROCESS, HAPPY_PATH_PROCESS_STATIC_IV_ROLLOVER)
303  303          // AOS_FHEC_NA, AOS_IZ_NA, 0); Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003,
                 1, TC_HAS_FECF,
304  304          // TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 1024, AOS_FHEC_NA, AOS_IZ_NA, 0);
305  305          GvcidManagedParameters_t TC_UT_Managed_Parameters = {
306       -          0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 46,
                 TC_OCF_NA, 1};
     306  +          0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS,
                 1024, TC_OCF_NA, 1};
307  307          Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
308  308          TC_UT_Managed_Parameters.vcid = 1;
309  309          Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
          @@ -400,7 +400,7 @@ UTEST(TC_PROCESS,
          HAPPY_PATH_PROCESS_NONTRANSMITTED_INCREMENTING_IV_ROLLOVER)
400  400          // AOS_FHEC_NA, AOS_IZ_NA, 0); Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003,
                 1, TC_HAS_FECF,
401  401          // TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 1024, AOS_FHEC_NA, AOS_IZ_NA, 0);
402  402          GvcidManagedParameters_t TC_UT_Managed_Parameters = {
403       -          0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 46,
                 TC_OCF_NA, 1};
     403  +          0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS,
                 1024, TC_OCF_NA, 1};
404  404          Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
405  405          TC_UT_Managed_Parameters.vcid = 1;
406  406          Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
          @@ -508,7 +508,7 @@ UTEST(TC_PROCESS,
          HAPPY_PATH_PROCESS_NONTRANSMITTED_INCREMENTING_ARSN_ROLLOVER)
508  508          // AOS_FHEC_NA, AOS_IZ_NA, 0); Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003,
                 1, TC_HAS_FECF,
509  509          // TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 1024, AOS_FHEC_NA, AOS_IZ_NA, 0);
510  510          GvcidManagedParameters_t TC_UT_Managed_Parameters = {
511       -          0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 42,
                 TC_OCF_NA, 1};
```

```
511     +         0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS,
          1024, TC_OCF_NA, 1};
512 512         Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
513 513         TC_UT_Managed_Parameters.vcid = 1;
514 514         Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
        @@ -601,7 +601,7 @@ UTEST(TC_PROCESS, ERROR_TC_INPUT_FRAME_TOO_SHORT_FOR_SPEC)
601 601         // Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003, 0, TC_HAS_FECF,
          TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 4,
602 602         // AOS_FHEC_NA, AOS_IZ_NA, 0);
603 603         GvcidManagedParameters_t TC_UT_Managed_Parameters = {
604       -         0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 4,
          TC_OCF_NA, 1};
604     +         0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS,
          1024, TC_OCF_NA, 1};
605 605         Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
606 606         status = Crypto_Init();
607 607         ASSERT_EQ(CRYPTO_LIB_SUCCESS, status);
        @@ -642,7 +642,7 @@ UTEST(TC_PROCESS,
        ERROR_TC_INPUT_FRAME_TOO_SHORT_FOR_SPECIFIED_FRAME_LENGTH_HEAD
642 642         // Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003, 0, TC_HAS_FECF,
          TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 4,
643 643         // AOS_FHEC_NA, AOS_IZ_NA, 0);
644 644         GvcidManagedParameters_t TC_UT_Managed_Parameters = {
645       -         0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 38,
          TC_OCF_NA, 1};
645     +         0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS,
          1024, TC_OCF_NA, 1};
646 646         Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
647 647         status = Crypto_Init();
648 648         ASSERT_EQ(CRYPTO_LIB_SUCCESS, status);
        @@ -684,7 +684,7 @@ UTEST(TC_PROCESS, HAPPY_PATH_DECRYPT_CBC)
684 684         // Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003, 0, TC_HAS_FECF,
          TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 1024,
685 685         // AOS_FHEC_NA, AOS_IZ_NA, 0);
686 686         GvcidManagedParameters_t TC_UT_Managed_Parameters = {
687       -         0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 43,
          TC_OCF_NA, 1};
687     +         0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS,
          1024, TC_OCF_NA, 1};
688 688         Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
689 689         status = Crypto_Init();
690 690         ASSERT_EQ(CRYPTO_LIB_SUCCESS, status);
```

```
@@ -759,7 +759,7 @@ UTEST(TC_PROCESS, DECRYPT_CBC_1B)
759  759          // AOS_FHEC_NA, AOS_IZ_NA, 0);
760  760
761  761          GvcidManagedParameters_t TC_UT_Managed_Parameters = {
762       -          0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 43, TC_OCF_NA, 1};
     762  +          0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 1024, TC_OCF_NA, 1};
763  763          Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
764  764          TC_UT_Managed_Parameters.vcid = 1;
765  765          Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
@@ -840,7 +840,7 @@ UTEST(TC_PROCESS, DECRYPT_CBC_16B)
840  840          // AOS_FHEC_NA, AOS_IZ_NA, 0);
841  841
842  842          GvcidManagedParameters_t TC_UT_Managed_Parameters = {
843       -          0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 59, TC_OCF_NA, 1};
     843  +          0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 1024, TC_OCF_NA, 1};
844  844          Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
845  845          TC_UT_Managed_Parameters.vcid = 1;
846  846          Crypto_Config_Add_Gvcid_Managed_Parameters(TC_UT_Managed_Parameters);
@@ -1153,7 +1153,7 @@ UTEST(TC_PROCESS, TC_SA_SEGFAULT_TEST)
1153 1153          // Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003, 0, TC_HAS_FECF, TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 1024,
1154 1154          // AOS_FHEC_NA, AOS_IZ_NA, 0);
1155 1155          GvcidManagedParameters_t AOS_Managed_Parameters = {
1156      -          0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 43, TC_OCF_NA, 1};
     1156 +          0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 1024, TC_OCF_NA, 1};
1157 1157          Crypto_Config_Add_Gvcid_Managed_Parameters(AOS_Managed_Parameters);
1158 1158
1159 1159          status = Crypto_Init();
@@ -1193,7 +1193,7 @@ UTEST(TC_PROCESS, TC_SA_NOT_OPERATIONAL)
1193 1193          // Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003, 0, TC_HAS_FECF, TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 1024,
1194 1194          // AOS_FHEC_NA, AOS_IZ_NA, 0);
1195 1195          GvcidManagedParameters_t AOS_Managed_Parameters = {
```

```
1196          -         0, 0x0003, 0, TC_NO_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_NO_SEGMENT_HDRS, 13,
                TC_OCF_NA, 1};
      1196    +         0, 0x0003, 0, TC_NO_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_NO_SEGMENT_HDRS, 1024,
                TC_OCF_NA, 1};
1197  1197          Crypto_Config_Add_Gvcid_Managed_Parameters(AOS_Managed_Parameters);
1198  1198
1199  1199          status = Crypto_Init();
                @@ -1233,7 +1233,7 @@ UTEST(TC_PROCESS, TC_KEY_STATE_TEST)
1233  1233          // Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003, 0, TC_HAS_FECF,
                TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 1024,
1234  1234          // AOS_FHEC_NA, AOS_IZ_NA, 0);
1235  1235          GvcidManagedParameters_t AOS_Managed_Parameters = {
1236          -         0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 43,
                TC_OCF_NA, 1};
      1236    +         0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS,
                1024, TC_OCF_NA, 1};
1237  1237          Crypto_Config_Add_Gvcid_Managed_Parameters(AOS_Managed_Parameters);
1238  1238
1239  1239          status = Crypto_Init();
                @@ -1288,7 +1288,7 @@ UTEST(TC_PROCESS, TC_HEAP_BUFFER_OVERFLOW_TEST)
1288  1288          // Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003, 0, TC_HAS_FECF,
                TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 1024,
1289  1289          // AOS_FHEC_NA, AOS_IZ_NA, 0);
1290  1290          GvcidManagedParameters_t AOS_Managed_Parameters = {
1291          -         0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 109,
                TC_OCF_NA, 1};
      1291    +         0, 0x0003, 0, TC_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS,
                1024, TC_OCF_NA, 1};
1292  1292          Crypto_Config_Add_Gvcid_Managed_Parameters(AOS_Managed_Parameters);
1293  1293
1294  1294          status = Crypto_Init();
                @@ -1341,7 +1341,7 @@ UTEST(TC_PROCESS, TC_PROCESS_PREP_AAD_UNDERFLOW_TEST)
1341  1341          // Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003, 0, TC_HAS_FECF,
                TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 1024,
1342  1342          // AOS_FHEC_NA, AOS_IZ_NA, 0);
1343  1343          GvcidManagedParameters_t AOS_Managed_Parameters = {
1344          -         0, 0x0003, 0, TC_NO_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 11,
                TC_OCF_NA, 1};
      1344    +         0, 0x0003, 0, TC_NO_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TC_HAS_SEGMENT_HDRS, 1024,
                TC_OCF_NA, 1};
1345  1345          Crypto_Config_Add_Gvcid_Managed_Parameters(AOS_Managed_Parameters);
```

```
1346  1346
1347  1347          status = Crypto_Init();
```

test/unit/ut_tm_apply.c +141

```
@@ -2331,4 +2331,145 @@ UTEST(TM_APPLY_SECURITY, TM_APPLY_HEAP_UNDERFLOW_TEST)
2331  2331      Crypto_Shutdown();
2332  2332  }
2333  2333
      2334  + UTEST(TM_APPLY, TM_APPLY_Secondary_Hdr_OVERFLOW_TEST)
      2335  + {
      2336  +     // Local Variables
      2337  +     int32_t  status              = CRYPTO_LIB_SUCCESS;
      2338  +
      2339  +     // Configure Parameters
      2340  +     Crypto_Config_CryptoLib(KEY_TYPE_INTERNAL, MC_TYPE_INTERNAL, SA_TYPE_INMEMORY,
              CRYPTOGRAPHY_TYPE_LIBGCRYPT,
      2341  +                             IV_INTERNAL, CRYPTO_TC_CREATE_FECF_TRUE,
              TC_PROCESS_SDLS_PDUS_TRUE, TC_HAS_PUS_HDR,
      2342  +                             TC_IGNORE_SA_STATE_FALSE, TC_IGNORE_ANTI_REPLAY_TRUE,
              TC_UNIQUE_SA_PER_MAP_ID_FALSE,
      2343  +                             TC_CHECK_FECF_TRUE, 0x3F,
              SA_INCREMENT_NONTRANSMITTED_IV_TRUE);
      2344  +     // TM Tests
      2345  +     // Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003, 0, TC_HAS_FECF,
              TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 1024,
      2346  +     // AOS_FHEC_NA, AOS_IZ_NA, 0);
      2347  +     GvcidManagedParameters_t TM_UT_Managed_Parameters = {
      2348  +         0, 0x002c, 1, TM_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TM_SEGMENT_HDRS_NA, 7,
              TM_NO_OCF, 1};
      2349  +     Crypto_Config_Add_Gvcid_Managed_Parameters(TM_UT_Managed_Parameters);
      2350  +
      2351  +     status = Crypto_Init();
      2352  +
      2353  +     TC_t *tc_sdls_processed_frame;
      2354  +     tc_sdls_processed_frame = malloc(sizeof(uint8_t) * TC_SIZE);
      2355  +     memset(tc_sdls_processed_frame, 0, (sizeof(uint8_t) * TC_SIZE));
      2356  +
      2357  +     char *framed_tm_h   = "02C200009800FF";
      2358  +     char *framed_tm_b   = NULL;
      2359  +     int   framed_tm_len = 0;
      2360  +     hex_conversion(framed_tm_h, &framed_tm_b, &framed_tm_len);
      2361  +
```

```
2362  +     SecurityAssociation_t *sa_ptr;
2363  +     sa_if->sa_get_from_spi(5, &sa_ptr);
2364  +     sa_ptr->sa_state        = SA_OPERATIONAL;
2365  +     sa_ptr->shivf_len       = 0;
2366  +     sa_ptr->iv_len          = 0;
2367  +     sa_ptr->shsnf_len       = 0;
2368  +     sa_ptr->arsnw           = 0;
2369  +     sa_ptr->arsnw_len       = 0;
2370  +     sa_ptr->arsn_len        = 0;
2371  +     sa_ptr->gvcid_blk.scid  = 0x002c;
2372  +     sa_ptr->gvcid_blk.vcid  = 1;
2373  +     sa_ptr->gvcid_blk.mapid = TYPE_TM;
2374  +
2375  +     status = Crypto_TM_ApplySecurity((uint8_t *)framed_tm_b, framed_tm_len);
2376  +
2377  +     ASSERT_EQ(CRYPTO_LIB_ERR_TM_SECONDARY_HDR_VN, status);
2378  +     free(framed_tm_b);
2379  +     free(tc_sdls_processed_frame);
2380  +     Crypto_Shutdown();
2381  + }
2382  +
2383  + UTEST(TM_APPLY, TM_APPLY_Secondary_Hdr_Spec_Violation)
2384  + {
2385  +     // Local Variables
2386  +     int32_t  status             = CRYPTO_LIB_SUCCESS;
2387  +
2388  +     // Configure Parameters
2389  +     Crypto_Config_CryptoLib(KEY_TYPE_INTERNAL, MC_TYPE_INTERNAL, SA_TYPE_INMEMORY,
        CRYPTOGRAPHY_TYPE_LIBGCRYPT,
2390  +                             IV_INTERNAL, CRYPTO_TC_CREATE_FECF_TRUE,
        TC_PROCESS_SDLS_PDUS_TRUE, TC_HAS_PUS_HDR,
2391  +                             TC_IGNORE_SA_STATE_FALSE, TC_IGNORE_ANTI_REPLAY_TRUE,
        TC_UNIQUE_SA_PER_MAP_ID_FALSE,
2392  +                             TC_CHECK_FECF_TRUE, 0x3F,
        SA_INCREMENT_NONTRANSMITTED_IV_TRUE);
2393  +     // TM Tests
2394  +     GvcidManagedParameters_t TM_UT_Managed_Parameters = {
2395  +         0, 0x002c, 1, TM_NO_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TM_SEGMENT_HDRS_NA, 8,
        TM_NO_OCF, 1};
2396  +     Crypto_Config_Add_Gvcid_Managed_Parameters(TM_UT_Managed_Parameters);
2397  +
2398  +     status = Crypto_Init();
2399  +
```

```
2400 +     // Secondary header length set to 0x40, overflows into secondary header version
       number
2401 +     char *framed_tm_h   = "02C20009800040BB";
2402 +     char *framed_tm_b   = NULL;
2403 +     int   framed_tm_len = 0;
2404 +     hex_conversion(framed_tm_h, &framed_tm_b, &framed_tm_len);
2405 +
2406 +     SecurityAssociation_t *sa_ptr;
2407 +     sa_if->sa_get_from_spi(5, &sa_ptr);
2408 +     sa_ptr->sa_state        = SA_OPERATIONAL;
2409 +     sa_ptr->shivf_len       = 0;
2410 +     sa_ptr->iv_len          = 0;
2411 +     sa_ptr->shsnf_len       = 0;
2412 +     sa_ptr->arsnw           = 0;
2413 +     sa_ptr->arsnw_len       = 0;
2414 +     sa_ptr->arsn_len        = 0;
2415 +     sa_ptr->gvcid_blk.scid  = 0x002c;
2416 +     sa_ptr->gvcid_blk.vcid  = 1;
2417 +     sa_ptr->gvcid_blk.mapid = TYPE_TM;
2418 +
2419 +     status = Crypto_TM_ApplySecurity((uint8_t *)framed_tm_b, framed_tm_len);
2420 +
2421 +     ASSERT_EQ(CRYPTO_LIB_ERR_TM_SECONDARY_HDR_VN, status);
2422 +     free(framed_tm_b);
2423 +     Crypto_Shutdown();
2424 + }
2425 +
2426 + UTEST(TM_APPLY, TM_APPLY_Secondary_Hdr_One_Too_Big)
2427 + {
2428 +     // Local Variables
2429 +     int32_t  status             = CRYPTO_LIB_SUCCESS;
2430 +
2431 +     // Configure Parameters
2432 +     Crypto_Config_CryptoLib(KEY_TYPE_INTERNAL, MC_TYPE_INTERNAL, SA_TYPE_INMEMORY,
       CRYPTOGRAPHY_TYPE_LIBGCRYPT,
2433 +                             IV_INTERNAL, CRYPTO_TC_CREATE_FECF_TRUE,
       TC_PROCESS_SDLS_PDUS_TRUE, TC_HAS_PUS_HDR,
2434 +                             TC_IGNORE_SA_STATE_FALSE, TC_IGNORE_ANTI_REPLAY_TRUE,
       TC_UNIQUE_SA_PER_MAP_ID_FALSE,
2435 +                             TC_CHECK_FECF_TRUE, 0x3F,
       SA_INCREMENT_NONTRANSMITTED_IV_TRUE);
2436 +     // TM Tests
2437 +     // Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003, 0, TC_HAS_FECF,
       TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 1024,
```

```
2438 +     // AOS_FHEC_NA, AOS_IZ_NA, 0);
2439 +     GvcidManagedParameters_t TM_UT_Managed_Parameters = {
2440 +         0, 0x002c, 1, TM_NO_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TM_SEGMENT_HDRS_NA, 8,
       TM_NO_OCF, 1};
2441 +     Crypto_Config_Add_Gvcid_Managed_Parameters(TM_UT_Managed_Parameters);
2442 +
2443 +     status = Crypto_Init();
2444 +
2445 +     TC_t *tc_sdls_processed_frame;
2446 +     tc_sdls_processed_frame = malloc(sizeof(uint8_t) * TC_SIZE);
2447 +     memset(tc_sdls_processed_frame, 0, (sizeof(uint8_t) * TC_SIZE));
2448 +
2449 +     char *framed_tm_h   = "02C20008800001BB";
2450 +     char *framed_tm_b   = NULL;
2451 +     int   framed_tm_len = 0;
2452 +     hex_conversion(framed_tm_h, &framed_tm_b, &framed_tm_len);
2453 +
2454 +     SecurityAssociation_t *sa_ptr;
2455 +     sa_if->sa_get_from_spi(5, &sa_ptr);
2456 +     sa_ptr->sa_state        = SA_OPERATIONAL;
2457 +     sa_ptr->shivf_len       = 0;
2458 +     sa_ptr->iv_len          = 0;
2459 +     sa_ptr->shsnf_len       = 0;
2460 +     sa_ptr->arsnw           = 0;
2461 +     sa_ptr->arsnw_len       = 0;
2462 +     sa_ptr->arsn_len        = 0;
2463 +     sa_ptr->gvcid_blk.scid  = 0x002c;
2464 +     sa_ptr->gvcid_blk.vcid  = 1;
2465 +     sa_ptr->gvcid_blk.mapid = TYPE_TM;
2466 +
2467 +     status = Crypto_TM_ApplySecurity((uint8_t *)framed_tm_b, framed_tm_len);
2468 +
2469 +     ASSERT_EQ(CRYPTO_LIB_ERR_TM_SECONDARY_HDR_SIZE, status);
2470 +     free(framed_tm_b);
2471 +     free(tc_sdls_processed_frame);
2472 +     Crypto_Shutdown();
2473 + }
2474 +
2334 2475   UTEST_MAIN();
```

test/unit/ut_tm_process.c +108 -20

```
@@ -345,9 +345,6 @@ UTEST(TM_PROCESS_SECURITY, SECONDARY_HDR_PRESENT_PLAINTEXT)
```

```
345  345              0, 0x002c, 0, TM_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TM_SEGMENT_HDRS_NA, 1786,
                 TM_NO_OCF, 1};
346  346          Crypto_Config_Add_Gvcid_Managed_Parameters(TM_UT_Managed_Parameters);
347  347
348       -      // Crypto_Config_Add_Gvcid_Managed_Parameters(0, 0x002c, 0, TM_HAS_FECF,
                 TM_SEGMENT_HDRS_NA, TM_NO_OCF, 1786,
349       -      // AOS_FHEC_NA, AOS_IZ_NA, 0);
350       -
351  348          status = Crypto_Init();
352  349
353  350          // Test frame setup
                 @@ -2137,10 +2134,6 @@ UTEST(TM_PROCESS, TM_SA_NOT_OPERATIONAL)
2137 2134
2138 2135          status = Crypto_Init();
2139 2136
2140      -      TC_t *tc_sdls_processed_frame;
2141      -      tc_sdls_processed_frame = malloc(sizeof(uint8_t) * TC_SIZE);
2142      -      memset(tc_sdls_processed_frame, 0, (sizeof(uint8_t) * TC_SIZE));
2143      -
2144 2137          char *framed_tm_h   = "02C000001800002C414243444546";
2145 2138          char *framed_tm_b   = NULL;
2146 2139          int   framed_tm_len = 0;
                 @@ -2161,7 +2154,6 @@ UTEST(TM_PROCESS, TM_SA_NOT_OPERATIONAL)
2161 2154
2162 2155          ASSERT_EQ(CRYPTO_LIB_ERR_SA_NOT_OPERATIONAL, status);
2163 2156          free(framed_tm_b);
2164      -      free(tc_sdls_processed_frame);
2165 2157          Crypto_Shutdown();
2166 2158      }
2167 2159
                 @@ -2186,10 +2178,6 @@ UTEST(TM_PROCESS, TM_KEY_STATE_TEST)
2186 2178
2187 2179          status = Crypto_Init();
2188 2180
2189      -      TC_t *tc_sdls_processed_frame;
2190      -      tc_sdls_processed_frame = malloc(sizeof(uint8_t) * TC_SIZE);
2191      -      memset(tc_sdls_processed_frame, 0, (sizeof(uint8_t) * TC_SIZE));
2192      -
2193 2181          char *framed_tm_h   = "02C000018000008414243444546";
2194 2182          char *framed_tm_b   = NULL;
2195 2183          int   framed_tm_len = 0;
                 @@ -2215,10 +2203,12 @@ UTEST(TM_PROCESS, TM_KEY_STATE_TEST)
2215 2203
```

```
2216  2204          ASSERT_EQ(CRYPTO_LIB_ERR_KEY_STATE_INVALID, status);
2217  2205          free(framed_tm_b);
2218        -       free(tc_sdls_processed_frame);
2219  2206          Crypto_Shutdown();
2220  2207      }
2221  2208
      2209  + /*
      2210  + ** Test that we won't process a TM frame that is obviously too small
      2211  + */
2222  2212      UTEST(TM_PROCESS, TM_PROCESS_HEAP_UNDERFLOW_TEST)
2223  2213      {
2224  2214          // Local Variables
          @@ -2232,18 +2222,12 @@ UTEST(TM_PROCESS, TM_PROCESS_HEAP_UNDERFLOW_TEST)
2232  2222                                      TC_IGNORE_SA_STATE_FALSE, TC_IGNORE_ANTI_REPLAY_TRUE,
                TC_UNIQUE_SA_PER_MAP_ID_FALSE,
2233  2223                                      TC_CHECK_FECF_TRUE, 0x3F,
                SA_INCREMENT_NONTRANSMITTED_IV_TRUE);
2234  2224          // TM Tests
2235        -       // Crypto_Config_Add_Gvcid_Managed_Parameter(0, 0x0003, 0, TC_HAS_FECF,
                TC_HAS_SEGMENT_HDRS, TC_OCF_NA, 1024,
2236        -       // AOS_FHEC_NA, AOS_IZ_NA, 0);
2237  2225          GvcidManagedParameters_t TM_UT_Managed_Parameters = {
2238  2226              0, 0x002c, 0, TM_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TM_SEGMENT_HDRS_NA, 1786,
                TM_NO_OCF, 1};
2239  2227          Crypto_Config_Add_Gvcid_Managed_Parameters(TM_UT_Managed_Parameters);
2240  2228
2241  2229          status = Crypto_Init();
2242  2230
2243        -       TC_t *tc_sdls_processed_frame;
2244        -       tc_sdls_processed_frame = malloc(sizeof(uint8_t) * TC_SIZE);
2245        -       memset(tc_sdls_processed_frame, 0, (sizeof(uint8_t) * TC_SIZE));
2246        -
2247  2231          char *framed_tm_h   = "02C000001800000841424344454 6";
2248  2232          char *framed_tm_b   = NULL;
2249  2233          int   framed_tm_len = 0;
          @@ -2262,7 +2246,111 @@ UTEST(TM_PROCESS, TM_PROCESS_HEAP_UNDERFLOW_TEST)
2262  2246
2263  2247          ASSERT_EQ(CRYPTO_LIB_ERR_TM_FRAME_LENGTH_UNDERFLOW, status);
2264  2248          free(framed_tm_b);
2265        -       free(tc_sdls_processed_frame);
      2249  +       Crypto_Shutdown();
      2250  + }
      2251  +
      2252  + /*
```

```
2253 + ** Test that a Secondary Header that violates spec, or doesn't leave room for a single
       byte
2254 + ** of data, won't pass.
2255 + */
2256 + UTEST(TM_PROCESS, TM_PROCESS_Secondary_Hdr_OVERFLOW_TEST)
2257 + {
2258 +     // Local Variables
2259 +     int32_t  status              = CRYPTO_LIB_SUCCESS;
2260 +     uint8_t *ptr_processed_frame = NULL;
2261 +     uint16_t processed_tm_len;
2262 +
2263 +     // Configure Parameters
2264 +     Crypto_Config_CryptoLib(KEY_TYPE_INTERNAL, MC_TYPE_INTERNAL, SA_TYPE_INMEMORY,
       CRYPTOGRAPHY_TYPE_LIBGCRYPT,
2265 +                             IV_INTERNAL, CRYPTO_TC_CREATE_FECF_TRUE,
       TC_PROCESS_SDLS_PDUS_TRUE, TC_HAS_PUS_HDR,
2266 +                             TC_IGNORE_SA_STATE_FALSE, TC_IGNORE_ANTI_REPLAY_TRUE,
       TC_UNIQUE_SA_PER_MAP_ID_FALSE,
2267 +                             TC_CHECK_FECF_TRUE, 0x3F,
       SA_INCREMENT_NONTRANSMITTED_IV_TRUE);
2268 +     // TM Tests
2269 +     GvcidManagedParameters_t TM_UT_Managed_Parameters = {
2270 +         0, 0x002c, 0, TM_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TM_SEGMENT_HDRS_NA, 7,
       TM_NO_OCF, 1};
2271 +     Crypto_Config_Add_Gvcid_Managed_Parameters(TM_UT_Managed_Parameters);
2272 +
2273 +     status = Crypto_Init();
2274 +
2275 +     char *framed_tm_h   = "02C00009800FF";
2276 +     char *framed_tm_b   = NULL;
2277 +     int   framed_tm_len = 0;
2278 +     hex_conversion(framed_tm_h, &framed_tm_b, &framed_tm_len);
2279 +
2280 +     status = Crypto_TM_ProcessSecurity((uint8_t *)framed_tm_b, framed_tm_len,
       &ptr_processed_frame, &processed_tm_len);
2281 +
2282 +     ASSERT_EQ(CRYPTO_LIB_ERR_TM_SECONDARY_HDR_SIZE, status);
2283 +     free(framed_tm_b);
2284 +     Crypto_Shutdown();
2285 + }
2286 +
2287 + /*
2288 + ** Test a Secondary Header that violates spec catches the error
2289 + */
```

```
2290  + UTEST(TM_PROCESS, TM_PROCESS_Secondary_Hdr_Spec_Violation)
2291  + {
2292  +     // Local Variables
2293  +     int32_t  status              = CRYPTO_LIB_SUCCESS;
2294  +     uint8_t *ptr_processed_frame = NULL;
2295  +     uint16_t processed_tm_len;
2296  +
2297  +     // Configure Parameters
2298  +     Crypto_Config_CryptoLib(KEY_TYPE_INTERNAL, MC_TYPE_INTERNAL, SA_TYPE_INMEMORY,
        CRYPTOGRAPHY_TYPE_LIBGCRYPT,
2299  +                             IV_INTERNAL, CRYPTO_TC_CREATE_FECF_TRUE,
        TC_PROCESS_SDLS_PDUS_TRUE, TC_HAS_PUS_HDR,
2300  +                             TC_IGNORE_SA_STATE_FALSE, TC_IGNORE_ANTI_REPLAY_TRUE,
        TC_UNIQUE_SA_PER_MAP_ID_FALSE,
2301  +                             TC_CHECK_FECF_TRUE, 0x3F,
        SA_INCREMENT_NONTRANSMITTED_IV_TRUE);
2302  +     // TM Tests
2303  +     GvcidManagedParameters_t TM_UT_Managed_Parameters = {
2304  +         0, 0x002c, 0, TM_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TM_SEGMENT_HDRS_NA, 8,
        TM_NO_OCF, 1};
2305  +     Crypto_Config_Add_Gvcid_Managed_Parameters(TM_UT_Managed_Parameters);
2306  +
2307  +     status = Crypto_Init();
2308  +
2309  +     // Secondary header length set to 0x40, overflows into secondary header version
        number
2310  +     char *framed_tm_h   = "02C00009800040BB";
2311  +     char *framed_tm_b   = NULL;
2312  +     int   framed_tm_len = 0;
2313  +     hex_conversion(framed_tm_h, &framed_tm_b, &framed_tm_len);
2314  +
2315  +     status = Crypto_TM_ProcessSecurity((uint8_t *)framed_tm_b, framed_tm_len,
        &ptr_processed_frame, &processed_tm_len);
2316  +
2317  +     ASSERT_EQ(CRYPTO_LIB_ERR_TM_SECONDARY_HDR_VN, status);
2318  +     free(framed_tm_b);
2319  +     Crypto_Shutdown();
2320  + }
2321  +
2322  + /*
2323  + ** Test a Secondary Header that is too big relative to the bytes received
2324  + */
2325  + UTEST(TM_PROCESS, TM_PROCESS_Secondary_Hdr_One_Too_Big)
2326  + {
```

```
2327 +     // Local Variables
2328 +     int32_t  status             = CRYPTO_LIB_SUCCESS;
2329 +     uint8_t *ptr_processed_frame = NULL;
2330 +     uint16_t processed_tm_len;
2331 +
2332 +     // Configure Parameters
2333 +     Crypto_Config_CryptoLib(KEY_TYPE_INTERNAL, MC_TYPE_INTERNAL, SA_TYPE_INMEMORY,
       CRYPTOGRAPHY_TYPE_LIBGCRYPT,
2334 +                             IV_INTERNAL, CRYPTO_TC_CREATE_FECF_TRUE,
       TC_PROCESS_SDLS_PDUS_TRUE, TC_HAS_PUS_HDR,
2335 +                             TC_IGNORE_SA_STATE_FALSE, TC_IGNORE_ANTI_REPLAY_TRUE,
       TC_UNIQUE_SA_PER_MAP_ID_FALSE,
2336 +                             TC_CHECK_FECF_TRUE, 0x3F,
       SA_INCREMENT_NONTRANSMITTED_IV_TRUE);
2337 +     // TM Tests
2338 +     GvcidManagedParameters_t TM_UT_Managed_Parameters = {
2339 +         0, 0x002c, 0, TM_NO_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TM_SEGMENT_HDRS_NA, 8,
       TM_NO_OCF, 1};
2340 +     Crypto_Config_Add_Gvcid_Managed_Parameters(TM_UT_Managed_Parameters);
2341 +
2342 +     status = Crypto_Init();
2343 +     // 6 byte header + 2 byte secondary header
2344 +     // The Header length should be 0 (hdr len -1), with one byte of data after
2345 +     char *framed_tm_h   = "02C00008800001BB";
2346 +     char *framed_tm_b   = NULL;
2347 +     int   framed_tm_len = 0;
2348 +     hex_conversion(framed_tm_h, &framed_tm_b, &framed_tm_len);
2349 +
2350 +     status = Crypto_TM_ProcessSecurity((uint8_t *)framed_tm_b, framed_tm_len,
       &ptr_processed_frame, &processed_tm_len);
2351 +
2352 +     ASSERT_EQ(CRYPTO_LIB_ERR_TM_SECONDARY_HDR_SIZE, status);
2353 +     free(framed_tm_b);
2266 2354      Crypto_Shutdown();
2267 2355  }
2268 2356
        ...
```

Comments 0

Please sign in to comment.