AMSS Security Bulletin – October 2013

80-N3172-31 A

November 1, 2013

Submit technical questions at:
https://support.cdmatech.com/

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Contents

Tables

Revision history

Revision	Date	Description
A	Nov 2013	Initial release

1 Introduction

1.1 Purpose

This is a security bulletin from Qualcomm Technologies, Inc. (QTI) intended to help our customers incorporate security updates in launched or upcoming devices. This document provides a description of security vulnerabilities that have been addressed, and provides a detailed account of the associated code changes.

These are software changes that are expected to have no impact to hardware chipsets and shipments.

QTI strongly recommends that customers implement these changes immediately.

1.2 Scope

This document describes security vulnerabilities addressed after publication of [Q32].

This document is intended for customers using QTI's AMSS software releases prior to QTI incorporation of the code changes described herein. These security vulnerabilities are applicable to any release prior to October 2013 that supports the technologies as indicated in Table 2-2.

For software released in October 2013 onward, all affected product lines will contain the software changes for security vulnerabilities with security rating of Critical. Incorporation of software changes for security vulnerabilities with ratings other than Critical into software releases varies. Ratings are indicated in the Chapter 2 tables. For a detailed description of the ratings, see [Q2].

1.3 Conventions

Function declarations, function names, type declarations, and code samples appear in a different font, e.g., `#include`.

If you are viewing this document using a color monitor, or if you print this document to a color printer, **red boldface** indicates code that is to be **added**, and ~~blue strikethrough~~ indicates code that is to be replaced or removed.

1.4 References

Reference documents are listed in Table 1-1. Reference documents that are no longer applicable are deleted from this table; therefore, reference numbers may not be sequential.

Table 1-1 Reference documents and standards

Ref.	Document	
Qualcomm Technologies		
Q1	*Application Note: Software Glossary for Customers*	CL93-V3077-1
Q2	*Software Security Vulnerability Assessment Definitions*	80-N3781-1
Q3	*AMSS Security Bulletin – September 2010*	80-N3172-1
Q4	*AMSS Security Bulletin – November 2010*	80-N3172-2
Q5	*AMSS Security Bulletin – December 2010*	80-N3172-3
Q6	*AMSS Security Bulletin – January 2011*	80-N3172-4
Q7	*AMSS Security Bulletin – February 2011*	80-N3172-5
Q8	*AMSS Security Bulletin – March 2011*	80-N3172-6
Q9	*AMSS Security Bulletin – September 2011*	80-N3172-7
Q10	*AMSS Security Bulletin – October 2011*	80-N3172-8
Q11	*AMSS Security Bulletin – November 2011*	80-N3172-9
Q12	*AMSS Security Bulletin – December 2011*	80-N3172-10
Q13	*AMSS Security Bulletin – February 2012*	80-N3172-11
Q14	*AMSS Security Bulletin – March 2012*	80-N3172-12
Q15	*AMSS Security Bulletin – April 2012*	80-N3172-13
Q16	*AMSS Security Bulletin – May 2012*	80-N3172-14
Q17	*AMSS Security Bulletin – June 2012*	80-N3172-15
Q18	*AMSS Security Bulletin – July 2012*	80-N3172-16
Q19	*AMSS Security Bulletin – August 2012*	80-N3172-17
Q20	*AMSS Security Bulletin – September 2012*	80-N3172-18
Q21	*AMSS Security Bulletin – October 2012*	80-N3172-19
Q22	*AMSS Security Bulletin – November 2012*	80-N3172-20
Q23	*AMSS Security Bulletin – December 2012*	80-N3172-21
Q24	*AMSS Security Bulletin – January 2013*	80-N3172-22
Q25	*AMSS Security Bulletin – February 2013*	80-N3172-23
Q26	*AMSS Security Bulletin – March 2013*	80-N3172-24
Q27	*AMSS Security Bulletin – April 2013*	80-N3172-25
Q28	*AMSS Security Bulletin – May 2013*	80-N3172-26
Q29	*AMSS Security Bulletin – June 2013*	80-N3172-27
Q30	*AMSS Security Bulletin – July 2013*	80-N3172-28
Q31	*AMSS Security Bulletin – August 2013*	80-N3172-29
Q32	*AMSS Security Bulletin – September 2013*	80-N3172-30

1.5 Technical assistance

For assistance or clarification on information in this document, submit a case to QTI at https://support.cdmatech.com/.

If you do not have access to the CDMATech Support Service website, register for access or send email to support.cdmatech@qti.qualcomm.com.

1.6 Acronyms

For definitions of terms and abbreviations, see [Q1].

2 Security Vulnerabilities

Security vulnerabilities that have been addressed after publication of [Q32] are summarized in Table 2-2. The table contains columns for technology, security assessment, the associated CR number, and the description of the security vulnerability. The table also provides links to the associated code changes.

The security assessment classifies each vulnerability into one of four ratings: Critical, High, Medium, and Low (Table 2-1). For a detailed description of the ratings and the other dimensions of this assessment, see [Q2].

These security vulnerabilities are applicable to any release that supports the indicated technology prior to October 2013.

For software released in October 2013 onward, all affected product lines will contain the software changes for security vulnerabilities with security rating of Critical. Incorporation of software changes for security vulnerabilities with ratings other than Critical into software releases varies. Security vulnerabilities with Medium or Low ratings will typically not be published in a Security Bulletin.

Table 2-1 Security rating descriptions

Rating	Example consequences of a successful attack
Critical	Remote code execution, remote permanent denial of service (inoperability)
High	Access to confidential device information or other confidential user information maintained on device, temporary denial-of-service attacks
Medium	Access to device configuration information
Low	Access to general information about the device

QTI strongly recommends that customers implement these changes immediately.

1 **Table 2-2 Security vulnerabilities – High**

	Technology	CR	Security rating	Access vector	CWE	Description	Code changes
1.	1X	523550	High	Local	CWE-120 Buffer Copy without Checking Size of Input (Classic Buffer Overflow)	NULL pointer access and buffer overflow in 1X call processing default settings processing **Problem description** At power-up, the 1X protocol initializes several default settings. There is improper handling of a NULL pointer, and if the default settings table contains maximum length strings, a buffer overflow occurs. **Change description** Null Check was added for the filename pointer returned by mc_efs_item_file in onex_nv_efs_init(). Buffer overflow is addressed by increasing the size of destination buffer (df_str) to the maximum possible length, and replacing strlcpy with snprintf in mccap_df_create_discovery_string(), which guards against overflowing maximum size. **Files affected** modem_proc/1x/cp/inc/mccapdf_v.h modem_proc/1x/cp/inc/mclog_v.h modem_proc/1x/cp/src/mccapdf.c modem_proc/1x/cp/src/mclog.c modem_proc/1x/diag/src/onex_nv_rd_wt.c **No side effect from this change is anticipated.**	Section 4.1

	Technology	CR	Security rating	Access vector	CWE	Description	Code changes
2.	Data Modem	526889	High	Local	CWE-476 NULL Pointer Dereference	NULL pointer access vulnerability while processing a circuit-switched video telephony (CSVT) QMI service request message **Problem description** During processing of a QMI CSVT service request message with the command type field set to zero, a NULL pointer dereference occurs. There is a QMI CSVT command handler structure array with a MAX_SIZE, and the array is populated only with (MAX_SIZE - 1) valid entries, resulting in one entry in the array being all zeros (invalid entry). **Change description** Changes were made to change the command table MAX_SIZE so that the array will be intact with the valid and the required data. **Files affected** modem_proc\datamodem\interface\qmidata\src\ds_qmi_csvt.c **No side effect from this change is anticipated.**	Section 4.2
3.	LTE	513139	High	Adjacent Network	CWE-617 Reachable Assertion	Reachable assert during PSHO when RRC Reconfiguration message contains measObjectToRemoveList **Problem description** When measObjectToRemoveList is provided in a RRCConnectionReconfiguration message during IRAT PSHO, RRC tries to get serving cell information, even before camping. and hits a validation assert. **Change description** Changes were made to not validate for serving cell information during measObjId removal, when the measObjId was not previously configured. **Files affected** lte_rrc.lib **No side effect from this change is anticipated.**	Section 4.3

	Technology	**CR**	**Security rating**	**Access vector**	**CWE**	**Description**	**Code changes**
4.	LTE	525584	High	Adjacent Network	CWE-476 NULL Pointer Dereference	NULL pointer dereference when UE Information request RRC OTA message is received with spare value only **Problem description** During processing of a UE information request message with the spare value only, a NULL pointer access occurs. **Change description** Changes were made to ignore a UE information request message that contains only the spare value. **Files affected** lte_rrc.lib **No side effect from this change is anticipated.**	Section 4.4
5.	MMCP	536036	High	Local	CWE-253 Incorrect Check of Function Return Value	Crash when attempting to get EMBMS signal information **Problem description** The device crashes while attempting to retrieve signal strength information when back-to-back requests are sent. **Change description** In QMI_NAS_GET_EMBMS_SIG, the return value check in case of error was backwards. It has been corrected so that the proper actions in case of error are taken. **Files affected** qmi_nas.c **No side effect from this change is anticipated.**	Section 4.5

	Technology	CR	Security rating	Access vector	CWE	Description	Code changes
6.	RIL	502975	High	Local	CWE-457 Use of Uninitialized Variable	Use of uninitialized variable in RILD when OEM hook APIs are passed with random inputs **Problem description** When OEM HOOK API (QCRIL_EVT_HOOK_CSG_PERFORM_NW_SCAN) is passed invalid tag input (not 0x11 or 0x12), an uninitialized variable is used, resulting in a crash. **Change description** Changes were made to: ▪ Not send multiple response for OEM hook requests ▪ Initialize pointer to avoid QCRIL crash ▪ Restrict memcpy size so that stack does not become corrupted **Files affected** vendor/qcom/proprietary/qcril/qcril_qmi/qcril_qmi_nas.c **No side effect from this change is anticipated.**	Section 4.6
7.	UIM	523657	High	Local	CWE-129 Improper Validation of Array Index	Out-of-bounds array access in MMGSDI SimLock algorithm **Problem description** When verifying the control key during SimLock activation algorithm, if the control key comparison fails, we go ahead and update the control key counter. During the process of updating this counter, if the counters_length from the SimLock data in the FileSystem is less than or equal to 5, there is a buffer overread. There are similar issues at four other places. **Change description** Changes were made so that during the process of updating the control key counter, if the counters_length from the SimLock data in the FileSystem is less than or equal to 5, the counter is not updated and an error is returned from the function. The SimLock activation procedure may then fail. **Files affected** mmgsdi_perso_sec.c **No side effect from this change is anticipated.**	Section 4.7

	Technology	CR	Security rating	Access vector	CWE	Description	Code changes
8.	UIM	528497	High	Local	CWE-190 Integer Overflow or Wraparound	Integer overflow vulnerability in PERSO_SET_DATA **Problem description** If a PERSO_SET_DATA command is sent with the data length field set to more than half the maximum value of an unsigned integer, an integer overflow occurs on a variable that is used in later buffer accesses. **Change description** Checks for integer overflow have been added. If an overflow occurs, an error is returned. **Files affected** mmgsdi_perso_engine.c mmgsdi_perso_engine_ruim.c **No side effect from this change is anticipated.**	Section 4.8
9.	UIM	529366	High	Local	CWE-129 Improper Validation of Array Index	Improper validation of array index when processing personalization command to lock down the keys **Problem description** The personalization file contains a lock_down tag-length value. Under the current implementation, the length of the lock_down will be 2. If the length contains a value of 0 or 1, then while processing the personalization command to lock down the keys an out-of-bounds array access occurs. **Change description** Changes have been made to avoid attempts to allocate an array of 0 length, and to check that sufficient memory has been allocated before using an array. If not, an error is returned and the personalization command to lock down the keys will fail. **Files affected** mmgsdi_perso_sec.c **No side effect from this change is anticipated.**	Section 4.9

	Technology	CR	Security rating	Access vector	CWE	Description	Code changes
10.	Video	519773	High	Network	CWE-670 Always-Incorrect Control Flow Implementation	Playing a clip with H.264 video having bit-stream errors in slices causes phone to reboot **Problem description** A reboot occurs when an H.264 clip with H.264 video is played that has all of the following properties: ▪ The clip has multiple slices, with one or more slices having bit-stream corruption except the last slice. ▪ The slice with bit-stream corruption contains multiple rows and is followed by an I-slice. The I-slice starts in the same row in which the above mentioned corrupted slice ends and at least the first macro block in this I-slice is H.264-compliant. **Change description** ▪ Added control code when an error is detected while parsing slice data. ▪ Added code to ensure that the starting address of each row of frame is correctly filled, even for corrupted slices spanning multiple rows. **Files affected** adsp.b00,adsp.b01,adsp.b02,adsp.b03,adsp.b04, adsp.b05,adsp.b06,adsp.b07.adsp.mdt **No side effect from this change is anticipated.**	Section 4.10
11.	WLANHS	358100	High	Adjacent Network	CWE-126 Buffer Over-read	Receiving probe response without mandatory SSID IE causes buffer overread **Problem description** When the phone receives a probe response without the mandatory SSID IE, a buffer overread occurs. **Change description** Changes were made to check specifically for presence of desired data struct (SSID) in the OTA Probe Response frame before processing that data. If this mandatory data (Information Element) is not present, the frame is dropped and exit occurs. **Files affected** qcwlan8960.sys **No side effect from this change is anticipated.**	Section 4.11

3 Test Information

Table 3-1 provides information about the test impact of the security vulnerabilities described in Table 2-2. *Test case* indicates the test scenario used to verify the changes. *Certification test implications* indicates any standard test cases that exercise the changed code, e.g., any certification test cases that make use of a message involved in a buffer overflow vulnerability.

Table 3-1 Test information

	Technology	CR	Title	Test case	Certification
1.	1X	523550	NULL pointer access and buffer overflow in 1X call processing default settings processing	Manipulate 1X call processing source code to return a NULL filename pointer from mc_efs_item_file. Fill the default settings table with maximum length strings.	None
2.	Data Modem	526889	NULL pointer access vulnerability while processing a circuit-switched video telephony (CSVT) QMI service request message	Send a CSVT QMI Service Request message with the command type field set to 0.	None
3.	LTE	513139	Reachable assert during PSHO when RRC Reconfiguration message contains measObjectToRemoveList	During WCMA→LTE PSHO, send a RRCConnectionReconfiguration message with measObjectToRemoveList present.	None
4.	LTE	525584	NULL pointer dereference when UE Information request RRC OTA message is received with spare value only	With the UE in Connected state, send an OTA information request to the UE with only a spare value (e.g., 48 40).	None
5.	MMCP	536036	Crash when attempting to get EMBMS signal information	Request back-to-back valid EMBMS signal strength updates, with the second request coming before the first has returned.	None
6.	RIL	502975	Use of uninitialized variable in RILD when OEM hook APIs are passed with random inputs	Pass QCRIL_EVT_HOOK_CSG_PERFORM_NW_SCAN invalid tag input (not 0x11 or 0x12).	None
7.	UIM	523657	Out-of-bounds array access in MMGSDI SimLock algorithm	Create SimLock data with an invalid control key having counter length less than 5, then call the MMGSDI API to activate SimLock personalization.	None

	Technology	CR	Title	Test case	Certification
8.	UIM	528497	Integer overflow vulnerability in PERSO_SET_DATA	Send a PERSO_SET_DATA command with the data length field set to more than half the maximum value of an unsigned integer.	None
9.	UIM	529366	Improper validation of array index when processing personalization command to lock down the keys	Create a personalization file with the length of the lock_down set to 0 or 1 and then send a command to lock down the control keys to the phone.	None
10.	Video	519773	Playing a clip with H.264 video having bit-stream errors in slices causes phone to reboot	Play a clip with H.264 video that has all of the following properties: ▪ The clip should have multiple slices, with one or more slices having bit-stream corruption except the last slice. ▪ The slice with bit-stream corruption should contain multiple rows and should be followed by an I-slice. The I-slice should start in the same row in which the above mentioned corrupted slice ends and at least the first macro block in this I-slice should be H.264-compliant.	None
11.	WLANHS	538100	Receiving probe response without mandatory SSID IE causes buffer overread	Send a probe response to the phone Wi-Fi that does not contain the mandatory SSID IE.	None

4 Code Changes

This chapter provides a detailed account of the code changes involved in addressing each of the security vulnerabilities described in Table 2-2.

If you are viewing this document using a color monitor, or if you print this document to a color printer, **red boldface** indicates code that is to be **added**, and ~~blue strikethrough~~ indicates code that is to be replaced or removed.

4.1 [523550] NULL pointer access and buffer overflow in 1X call processing default settings processing

4.1.1 Changes when file onex_nv_rd_wt.c is present

NOTE: The diffs in this section should be used only if file 1x/diag/src/onex_nv_rd_wt.c is present in the build. If this file is not present, then the code changes described under the file onex_nv_rd_wt.c will be present in file mc.c in function mc_efs_init (see Section 4.4.2).

4.1.1.1 Change 1

File: onex_nv_rd_wt.c

Function: onex_nv_efs_init

```
/* Check to see if the file already exists and has valid information */
file_stat_result = efs_stat( conf_file, &conf_file_det );


/* Calculate the length of the NV item path that needs to be written
into onex_mux.conf file */
for ( item = 0; item < file_max; item++ )
{
  /* # of bytes in the NV item path + '\n' */
  ~~bytes_to_write += strlen( efs_item_files[item].path ) + 1;~~
    if (efs_item_files[item].path  != NULL)
    {
      bytes_to_write += strlen( efs_item_files[item].path ) + 1;
    }
    else
    {
      MSG_ERROR("File name returned as NULL for item no:%d",item,0,0);
    }
    } /*for ( item = 0; item < (int) MUX_EFS_ITEM_FILE_MAX; item++ )*/

    /* Open a conf file and write only if
    1. The Conf file doesn't exist or
    2. The # of bytes in the file doesn't match with the
    # of bytes we need to write in the file.
    (Since conf files are incremental, the bytes can't ever match the
    size of what we want to write but contain wrong information) */
    if ( (file_stat_result == 0) && (bytes_to_write == conf_file_det.st_size) )
    {

. . .

          error = TRUE;
          MSG_ERROR( "Skipping efs_write, error allocating memory", 0, 0, 0 );
        } /* if( file_write_buffer == NULL )*/
        else
        {
          /* Paths for all the looped files will be added in .conf */
          for ( item = 0; item < file_max; item++ )
          {
            /* Append the NV item path to the write buffer */
            file_name = efs_item_files[item].path;
            if (file_name != NULL)
            {
                file_name_size = (uint32)strlen(file_name); /* Skip '\0' */
                if( (bytes_written + file_name_size + 1) <= buffer_size )
                {
                  copy_size = memscpy( (file_write_buffer + bytes_written),
                                       (buffer_size - bytes_written),
                                       file_name, file_name_size );
                  if( copy_size != file_name_size )
                  {
                    ERR("Truncated memscpy,
bytes_copied:%d,src_size:%d,dest_size:%d",
                        copy_size, file_name_size,
                        (buffer_size - bytes_written));
                  }

                  /* Write newline character at the end of every file */
                  file_write_buffer[bytes_written + file_name_size] = '\n';
                  bytes_written += file_name_size + 1;
                } /*if( (bytes_written + file_name_size + 1) <= buffer_size )*/
                else
                {
                  MSG_ERROR("bytes_to_write:%d > buffer_size:%d",
                            (bytes_written + file_name_size + 1), buffer_size, 0);
                  error = TRUE;
                  break;
                }/*if else ( (bytes_written + file_name_size + 1) <= buffer_size )*/
            }
            else
            {
               MSG_ERROR("File name returned as NULL for item no:%d",item,0,0);
            }
          }/* for (item = 0 ; item < (int)MC_OWNED_EFS_ITEM_FILE_MAX; item++)*/

          /* Write the efs item path to the config file */
          bytes_written = efs_write( file_handle,
                                     file_write_buffer,
                                     bytes_to_write );

          if ( bytes_written < bytes_to_write )
          {
            MSG_ERROR( "Truncated EFS write error: %d < %u bytes",
```

4.1.1.2 Change 2

File: mccadpf.c

Function: mccap_df_query

```
~~boolean mccap_df_query (PACKED char *df_str)~~
boolean mccap_df_query (PACKED char *df_str, uint32 max_length)
 {
   boolean df_read_ok = TRUE;

   if (df_str)
   {

     /* Update the P_REV */
     DF_VALUE(MCCAP_DF_P_REV) = cdma.mob_cai_rev; /*lint !e713*/

     /* Set the default DF values. These values are compared with the values read

. . .

     if (!mccap_df_unpack(MC_DYNAMIC_FEATURE_ITEM_BYTE, (uint8 *)mcc_nv_data.df))
     {
       MSG_FATAL("Unable to unpack the DF items", 0, 0, 0);
```

```
    return FALSE;
   }

   /* Verify the NV items */
   (void )mccap_df_discovery_verify();

   /* Create the DF item string */
   ~~mccap_df_create_discovery_string(df_str);~~
   mccap_df_create_discovery_string(df_str, max_length);
  }
  else
  {
   MSG_FATAL("NULL DF String", 0, 0, 0);
   return FALSE;
  }

  return df_read_ok;

 } /* mccap_df_query */
```

4.1.1.3 Change 3

File: mccadpf.c

Function: mccap_df_create_discovery_string

```
~~void mccap_df_create_discovery_string(PACKED char *df_str)~~
void mccap_df_create_discovery_string(PACKED char *df_str, uint32 max_length)
 {
  int i, j=0;

  /* Create the DF String */

  /* First item in the string is the number of DF items */
  df_str[j++] = MCCAP_DF_MAX;  /*lint !e641 */

  for (i=0; i<MCCAP_DF_MAX; i++)  /*lint !e641 */
  {
   /* Copy the DF item number */
   df_str[j++] = (char)i;
   /* Copy the DF String */
   ~~(void)strlcpy((char *)&df_str[j],~~
            ~~mccap_df_tbl[i].df_str_name,~~
            ~~strlen(mccap_df_tbl[i].df_str_name) + 1);~~
   (void)snprintf((char *)&df_str[j],
           max_length-j, "%s",
           mccap_df_tbl[i].df_str_name);
```

```
            j = j + (int)strlen(mccap_df_tbl[i].df_str_name) + 1;

            /* Copy the Status */
            df_str[j++] = DF_STATUS(i);

            /* Copy the range */
            df_str[j++] = mccap_df_tbl[i].df_range.df_min;
            df_str[j++] = mccap_df_tbl[i].df_range.df_max;
```

4.1.1.4 Change 4

File: mclog.c

Function: mc_diag_df_items_query_handler

```
                        diagpkt_subsys_alloc (DIAG_SUBSYS_CALLP1X, /*lint !e641 */
                                              MC_DIAG_SUBSYS_DF_ITEMS_QUERY,
                                              rsp_len);

          if (dfq_rsp_ptr == NULL) {
            return NULL;
          }


          /* Discover the DF items supported by the mobile */
          ~~dfq_rsp_ptr->status = mccap_df_query(dfq_rsp_ptr->str);~~
          dfq_rsp_ptr->status = mccap_df_query(dfq_rsp_ptr->str,sizeof(dfq_rsp_ptr->str));
```

4.1.1.5 Change 5

File: mccapdf_v.h

Function: mccap_df_query

```
          None.

        RETURN VALUE
          TRUE if the DF items were successfully read.

        SIDE EFFECTS
          This function should be called only from the DIAG task context.

        ===========================================================================*/

      ~~boolean mccap_df_query (PACKED char *df_str);~~
      boolean mccap_df_query (PACKED char *df_str, uint32 max_length);
```

4.1.1.6 Change 6

File: mccapdf_v.h

Function: mccap_df_create_discovery_string

```
DEPENDENCIES
   None.

 RETURN VALUE



 SIDE EFFECTS
   None.

 ============================================================================*/
~~void mccap_df_create_discovery_string(PACKED char *df_str);~~
void mccap_df_create_discovery_string(PACKED char *df_str, uint32 max_length);
```

4.1.1.7 Change 7

File: mclog_v.h

```
   diagpkt_subsys_header_type    hdr;

 } mc_diag_df_items_query_req_type;

 typedef PACK (struct)
 {
   /* cmd_code, subsys_id, subsys_cmd_code: MC_DIAG_SUBSYS_DF_ITEMS_QUERY */
   diagpkt_subsys_header_type    hdr;
   /* Status indicates whether Dynamic Feature is supported */
   byte                          status;
   ~~/* DF String buffer */~~
   ~~char                          str[(uint32) MCCAP_DF_MAX * (MAX_DF_STR_LEN + 6)];~~
   /* DF String buffer. This string buffer stores MCCAP_DF_MAX no. of strings
    * with maximum size MAX_DF_STR_LEN along with some additional data. The
    * size of the buffer can be arrived from the way this string is built in
    * the function mccap_df_create_discovery_string*/
   char                          str[((uint32)MCCAP_DF_MAX * (MAX_DF_STR_LEN +
6))+2];
 } mc_diag_df_items_query_rsp_type;

 /*************** MC_DIAG_SUBSYS_DF_ITEM_VERIFY ***************/

 /* Verify a change made by the user */
```

```
typedef PACK (struct)
{
  /* cmd_code, subsys_id, subsys_cmd_code: MC_DIAG_SUBSYS_DF_ITEM_VERIFY */
  diagpkt_subsys_header_type      hdr;
  /* First byte is the item no. Second byte is the item value and the third
```

4.1.2 Changes when file onex_nv_rd_wt.c is not present

NOTE: The diffs in this section should be used only if file 1x/diag/src/onex_nv_rd_wt.c is *not* present in the build. If this file is present, then the code changes described under the file mc.c will be present in the file onex_nv_rd_wt.c in function onex_nv_efs_init (see Section 4.4.1).

4.1.2.1 Change 1

File: mc.c

Function: mc_efs_init

```
  if( write_buffer != NULL )
  {
    /* Instead of writing EFS Item file to config file every time,
     * collect the EFS file names in a buffer and write only once
     */
    memset(write_buffer, 0, buffer_size);

    for (item = 0 ; item < (int)MC_OWNED_EFS_ITEM_FILE_MAX; item++)
    {
      file_name = mc_efs_item_file[item].path;
      if (file_name != NULL)
      {
        file_name_size = (uint32)strlen(file_name); /* Skip '\0' */
        if( (bytesToWrite + file_name_size + 1) <= buffer_size )
        {
          copy_size = memscpy( (write_buffer + bytesToWrite),
                               (buffer_size - bytesToWrite),
                               file_name, file_name_size );
          if( copy_size != file_name_size )
          {
            ERR("Truncated memscpy, bytes_copied:%d, src_size:%d,dest_size:%d",
                 copy_size, file_name_size,
                 (buffer_size - bytesToWrite));
          }

          /* Write newline character at the end of every file */
          write_buffer[bytesToWrite + file_name_size] = '\n';
          bytesToWrite += file_name_size + 1;
```

```
              }
              else
              {
                MSG_ERROR("bytesToWrite:%d > buffer_size:%d",
                          (bytesToWrite + file_name_size + 1), buffer_size, 0);
                error = TRUE;
                break;
              }
            }
            else
            {
               ERR("File name returned as NULL for item no:%d",item,0,0);
            }
          }

          file_stat_result = efs_stat(MC_EFS_CONF_FILE, &conf_file_det);

          /* If the conf file does not exist or if the size of the existing conf file is
             not what we expect, create a new conf file */
          if ( (file_stat_result != 0) || (bytesToWrite != conf_file_det.st_size) )
          {
            file_handle = efs_open(MC_EFS_CONF_FILE, O_CREAT|O_WRONLY|O_TRUNC,
     ALLPERMS);
            if (file_handle == -1)
```

1 #### 4.1.2.2 Change 2

2 **File:** mccadpf.c

3 **Function:** mccap_df_query

```
~~boolean mccap_df_query (PACKED char *df_str)~~
boolean mccap_df_query (PACKED char *df_str, uint32 max_length)
 {
   boolean df_read_ok = TRUE;

   if (df_str)
   {

     /* Update the P_REV */
     DF_VALUE(MCCAP_DF_P_REV) = cdma.mob_cai_rev; /*lint !e713*/

     /* Set the default DF values. These values are compared with the values read

. . .
```

```
                if (!mccap_df_unpack(MC_DYNAMIC_FEATURE_ITEM_BYTE, (uint8 *)mcc_nv_data.df))
                {
                  MSG_FATAL("Unable to unpack the DF items", 0, 0, 0);
                  return FALSE;
                }

                /* Verify the NV items */
                (void )mccap_df_discovery_verify();

                /* Create the DF item string */
                ~~mccap_df_create_discovery_string(df_str);~~
                mccap_df_create_discovery_string(df_str, max_length);
              }
              else
              {
                MSG_FATAL("NULL DF String", 0, 0, 0);
                return FALSE;
              }

              return df_read_ok;

            } /* mccap_df_query */
```

4.1.2.3 Change 3

File: mccadpf.c

Function: mccap_df_create_discovery_string

```
~~void mccap_df_create_discovery_string(PACKED char *df_str)~~
void mccap_df_create_discovery_string(PACKED char *df_str, uint32 max_length)
 {
   int i, j=0;

   /* Create the DF String */

   /* First item in the string is the number of DF items */
   df_str[j++] = MCCAP_DF_MAX;  /*lint !e641 */

   for (i=0; i<MCCAP_DF_MAX; i++)  /*lint !e641 */
   {
     /* Copy the DF item number */
     df_str[j++] = (char)i;
     /* Copy the DF String */
     ~~(void)strlcpy((char *)&df_str[j],~~
                       ~~mccap_df_tbl[i].df_str_name,~~
                       ~~strlen(mccap_df_tbl[i].df_str_name) + 1);~~
```

```
            (void)snprintf((char *)&df_str[j],
                            max_length-j, "%s",
                            mccap_df_tbl[i].df_str_name);

            j = j + (int)strlen(mccap_df_tbl[i].df_str_name) + 1;

            /* Copy the Status */
            df_str[j++] = DF_STATUS(i);

            /* Copy the range */
            df_str[j++] = mccap_df_tbl[i].df_range.df_min;
            df_str[j++] = mccap_df_tbl[i].df_range.df_max;
```

4.1.2.4 Change 4

File: mclog.c

Function: mc_diag_df_items_query_handler

```
                    diagpkt_subsys_alloc (DIAG_SUBSYS_CALLP1X, /*lint !e641 */
                                          MC_DIAG_SUBSYS_DF_ITEMS_QUERY,
                                          rsp_len);


          if (dfq_rsp_ptr == NULL) {
            return NULL;
          }


          /* Discover the DF items supported by the mobile */
          ~~dfq_rsp_ptr->status = mccap_df_query(dfq_rsp_ptr->str);~~
          dfq_rsp_ptr->status = mccap_df_query(dfq_rsp_ptr->str,sizeof(dfq_rsp_ptr-
>str));
```

4.1.2.5 Change 5

File: mccapdf_v.h

Function: mccap_df_query

```
          None.

        RETURN VALUE
          TRUE if the DF items were successfully read.

        SIDE EFFECTS
          This function should be called only from the DIAG task context.
```

```
 ==========================================================================*/

~~boolean mccap_df_query (PACKED char *df_str);~~
boolean mccap_df_query (PACKED char *df_str, uint32 max_length);
```

4.1.2.6 Change 6

File: mccapdf_v.h

Function: mccap_df_create_discovery_string

```
DEPENDENCIES
   None.

 RETURN VALUE



 SIDE EFFECTS
   None.

 ==========================================================================*/
~~void mccap_df_create_discovery_string(PACKED char *df_str);~~
void mccap_df_create_discovery_string(PACKED char *df_str, uint32 max_length);
```

4.1.2.7 Change 7

File: mclog_v.h

```
   diagpkt_subsys_header_type    hdr;

 } mc_diag_df_items_query_req_type;

 typedef PACK (struct)
 {
   /* cmd_code, subsys_id, subsys_cmd_code: MC_DIAG_SUBSYS_DF_ITEMS_QUERY */
   diagpkt_subsys_header_type    hdr;
   /* Status indicates whether Dynamic Feature is supported */
   byte                          status;
   ~~/* DF String buffer */~~
   ~~char                          str[(uint32) MCCAP_DF_MAX * (MAX_DF_STR_LEN + 6)];~~
   /* DF String buffer. This string buffer stores MCCAP_DF_MAX no. of strings
    * with maximum size MAX_DF_STR_LEN along with some additional data. The
    * size of the buffer can be arrived from the way this string is built in
    * the function mccap_df_create_discovery_string*/
   char                          str[((uint32)MCCAP_DF_MAX * (MAX_DF_STR_LEN +
6))+2];
```

```
} mc_diag_df_items_query_rsp_type;

/*************** MC_DIAG_SUBSYS_DF_ITEM_VERIFY ***************/

/* Verify a change made by the user */
typedef PACK (struct)
{
  /* cmd_code, subsys_id, subsys_cmd_code: MC_DIAG_SUBSYS_DF_ITEM_VERIFY */
  diagpkt_subsys_header_type      hdr;
  /* First byte is the item no. Second byte is the item value and the third
```

4.2 [526889] NULL pointer access vulnerability while processing a circuit-switched video telephony (CSVT) QMI service request message

File: ds_qmi_csvt.c

Enum: qmi_csvti_cmd_e_type;

```
  typedef enum
  {
    CSVTI_CMD_MIN                   = 0,
    CSVTI_CMD_RESET                 = CSVTI_CMD_MIN,
    CSVTI_CMD_SET_EVENT_REPORT,
   ~~CSVTI_CMD_SET_EVENT_REPORT_IND,~~
    CSVTI_CMD_ORIGINATE_CALL,
    CSVTI_CMD_CONFIRM_CALL,
    CSVTI_CMD_ANSWER_CALL,
    CSVTI_CMD_END_CALL,
    CSVTI_CMD_MODIFY_CALL,
    CSVTI_CMD_ACK_CALL_MODIFY,
    CSVTI_CMD_GET_RLP_PARAMS,
    CSVTI_CMD_SET_RLP_PARAMS,
    CSVTI_CMD_GET_ACTIVE_CALL_LIST,
    CSVTI_CMD_GET_CALL_INFO,
    CSVTI_CMD_GET_CALL_STATS,
    CSVTI_CMD_MAX,
    CSVTI_CMD_WIDTH                 = 0xFFFF
  } qmi_csvti_cmd_e_type;
```

4.3 [513139] Reachable assert during PSHO when RRC Reconfiguration message contains measObjectToRemoveList

The changes for this vulnerability are contained within a library.

4.4 [525584] NULL pointer dereference when UE Information request RRC OTA message is received with spare value only

The changes for this vulnerability are contained within a library.

4.5 [536036] Crash when attempting to get EMBMS signal information

File: qmi_nas.c

Function: qmi_nasi_get_embms_sig

```
      response = (dsm_item_type*)QMI_SVC_RESPONSE_PENDING;
      MSG_HIGH ( "Sent response pending for QMI_NAS_GET_EMBMS_SIG to QMI
Framework",0,0,0);
      // Will respond after getting RSP message through MSGR
    }
    else
    {
      nas_006F_rsp.t02.result_code = QMI_RESULT_FAILURE;
      nas_006F_rsp.t02.error_code  = errval;
      nas_006F_rsp.t02_valid       = TRUE;

      ~~if ( qmi_nas_006F_rsp_write( &nas_006F_rsp, &response) == QMI_ERR_NONE )~~
      if ( qmi_nas_006F_rsp_write( &nas_006F_rsp, &response) != QMI_ERR_NONE )
      {
        response = NULL;
        MSG_ERROR( "qmi_nas_006F_rsp_write failed", 0, 0, 0 );
      }

      qmi_nasi_global.cmd_buf_p = NULL;
    }

    return response;
  }

  /*============================================================================
    FUNCTION QMI_NASI_LIMIT_SYS_INFO_IND_REPORTING ()

    DESCRIPTION
      Sets limitations on the reporting of QMI_NAS_SYS_INFO_IND.
```

4.6 [502975] Use of uninitialized variable in RILD when OEM hook APIs are passed with random inputs

4.6.1 Change 1

File: qcril_qmi_nas.c

Function: qcril_qmi_nas_csg_handle_oem_hook_perform_network_scan

```
   QCRIL_LOG_DEBUG("Recieved Data length from oem hook: %d ",params_ptr-
>datalen);
   qcril_qmi_print_hex((unsigned char *)
csg_perform_nw_scan_req_ptr,params_ptr->datalen);

   ~~ril_req_res =
qcril_qmi_nas_parse_csg_scan_oem_req(csg_perform_nw_scan_req_ptr,&qmi_request,(
uint8)params_ptr->datalen);~~

   ~~if( ril_req_res == RIL_E_GENERIC_FAILURE )~~
   ~~{~~
       ~~qcril_default_request_resp_params( instance_id,~~
                                           ~~params_ptr->t,~~
                                           ~~params_ptr->event_id,~~
                                           ~~ril_req_res,~~
                                           ~~&resp );~~
       ~~qcril_send_request_response( &resp );~~
   ~~}~~
   ~~qcril_reqlist_default_entry( params_ptr->t,~~
                                ~~params_ptr->event_id,~~
                                ~~QCRIL_DEFAULT_MODEM_ID,~~
                                ~~QCRIL_REQ_AWAITING_MORE_AMSS_EVENTS,~~
                                ~~QCRIL_EVT_HOOK_CSG_PERFORM_NW_SCAN,~~
                                ~~NULL,~~
                                ~~&qcril_req_info_ptr );~~
   ~~if ( qcril_reqlist_new( instance_id, &qcril_req_info_ptr ) == E_SUCCESS )~~
   ~~{~~
       ~~qmi_response = qcril_malloc( sizeof( *qmi_response ) );~~
       ~~if ( qmi_response )~~
       ~~{~~
           ~~ril_rep_helper = qcril_malloc( sizeof( *ril_rep_helper ) );~~
           ~~if ( ril_rep_helper )~~
           ~~{~~
               ~~qmi_client_error =  qmi_client_send_msg_async(~~
~~qcril_qmi_client_get_user_handle( QCRIL_QMI_CLIENT_NAS ),~~

~~QMI_NAS_PERFORM_NETWORK_SCAN_REQ_MSG_V01,~~
```

```
                                                                    ~~&qmi_request,~~

      ~~sizeof(qmi_request),~~
                                                                    ~~(void*)~~
      ~~qmi_response,~~
                                                                    ~~sizeof(~~
      ~~*qmi_response ),~~

      ~~qcril_qmi_nas_csg_handle_oem_hook_perform_network_scan_command_cb,~~
                                                                    ~~(void~~
      ~~*)ril_rep_helper,~~
                                                                    ~~&txn_handle~~
                                                               ~~);~~
                    ~~ril_req_res = ( QMI_NO_ERR == qmi_client_error ) ? RIL_E_SUCCESS~~
      ~~: RIL_E_GENERIC_FAILURE;~~
                ~~}~~
            ~~}~~
            ~~if ( RIL_E_SUCCESS != ril_req_res )~~
            ~~{~~
                ~~qcril_default_request_resp_params( instance_id, params_ptr->t,~~
      ~~params_ptr->event_id, ril_req_res, &resp );~~
                ~~qcril_send_request_response( &resp );~~
                ~~if ( ril_rep_helper )~~
                ~~{~~
                    ~~qcril_free( ril_rep_helper );~~
                ~~}~~
                ~~if ( qmi_response )~~
                ~~{~~
                    ~~qcril_free( qmi_response );~~
                ~~}~~
            ~~}~~
        ~~}~~

        do
        {
          ril_req_res =
      qcril_qmi_nas_parse_csg_scan_oem_req(csg_perform_nw_scan_req_ptr,&qmi_request,(
      uint8)params_ptr->datalen);

          if( ril_req_res == RIL_E_GENERIC_FAILURE )
          {
              qcril_default_request_resp_params( instance_id,
                                                 params_ptr->t,
                                                 params_ptr->event_id,
                                                 ril_req_res,
                                                 &resp );
              qcril_send_request_response( &resp );
              break;
          }

          qcril_reqlist_default_entry( params_ptr->t,
                                       params_ptr->event_id,
                                       QCRIL_DEFAULT_MODEM_ID,
                                       QCRIL_REQ_AWAITING_MORE_AMSS_EVENTS,
                                       QCRIL_EVT_HOOK_CSG_PERFORM_NW_SCAN,
                                       NULL,
                                       &qcril_req_info_ptr );
          if ( qcril_reqlist_new( instance_id, &qcril_req_info_ptr ) == E_SUCCESS )
          {
              qmi_response = qcril_malloc( sizeof( *qmi_response ) );
              if ( qmi_response )
              {
                  ril_rep_helper = qcril_malloc( sizeof( *ril_rep_helper ) );

                  if ( ril_rep_helper )
                  {
                      qmi_client_error = qmi_client_send_msg_async(
     qcril_qmi_client_get_user_handle( QCRIL_QMI_CLIENT_NAS ),
                                       QMI_NAS_PERFORM_NETWORK_SCAN_REQ_MSG_V01,
                                       &qmi_request,
                                       sizeof(qmi_request),
                                       (void*) qmi_response,
                                       sizeof( *qmi_response ),
            qcril_qmi_nas_csg_handle_oem_hook_perform_network_scan_command_cb,
                                       (void *)ril_rep_helper,
                                       &txn_handle
                                      );
                      ril_req_res = ( QMI_NO_ERR == qmi_client_error ) ?
     RIL_E_SUCCESS : RIL_E_GENERIC_FAILURE;
                  }
              }

              if ( RIL_E_SUCCESS != ril_req_res )
              {
                  qcril_default_request_resp_params( instance_id, params_ptr->t,
     params_ptr->event_id, ril_req_res, &resp );
                  qcril_send_request_response( &resp );

                  if ( ril_rep_helper )
                  {
                      qcril_free( ril_rep_helper );
                  }
                  if ( qmi_response )
                  {
                      qcril_free( qmi_response );
                  }
              }
          }
        } while (0);

        QCRIL_LOG_INFO("completed with %d", ril_req_res );
        QCRIL_LOG_FUNC_RETURN();
      } /* qcril_qmi_nas_csg_handle_oem_hook_perform_network_scan */
```

4.6.2 Change 2

File: qcril_qmi_nas.c

Function: qcril_qmi_nas_csg_handle_oem_hook_perform_network_scan_command_cb

```
  unsigned int i=0;
  uint16 total_len=0,csg_len=0,csg_info_len_position=0;
  user_handle = user_handle;
  ~~uint8 *resp_byte_stream_ptr,*resp_byte_stream,*resp_print;~~
  uint8 *resp_byte_stream_ptr = NULL;
  uint8 *resp_byte_stream = NULL;
 uint8  *resp_print = NULL;
  msg_id = msg_id;
  resp_c_struct_len = resp_c_struct_len;
```

4.6.3 Change 3

File: qcril_qmi_nas.c

Function: qcril_qmi_nas_dsds_util_request_set_subscription_mode

```
     RIL_Errno res = RIL_E_GENERIC_FAILURE;

     nas_set_dual_standby_pref_req_msg_v01 qmi_req_msg;
     ~~nas_set_dual_standby_pref_resp_msg_v01 * qmi_resp_msg;~~
     nas_set_dual_standby_pref_resp_msg_v01 * qmi_resp_msg = NULL;

     qmi_resp_msg = qcril_malloc(
sizeof(nas_set_dual_standby_pref_resp_msg_v01) );

. . .

         QCRIL_LOG_INFO( ".. priority_subs %d", (int) qmi_req_msg.priority_subs );
         QCRIL_LOG_INFO( ".. default_data_subs %d", (int)
qmi_req_msg.default_data_subs );
         QCRIL_LOG_INFO( ".. default_voice_subs %d", (int)
qmi_req_msg.default_voice_subs );

         res = qcril_qmi_client_send_msg_async_ex(
                                       QCRIL_QMI_CLIENT_NAS,
                                       QMI_NAS_DUAL_STANDBY_PREF_REQ_MSG_V01,
```

4.6.4 Change 4

File: qcril_qmi_nas.c

Function: qcril_qmi_nas_dsds_request_set_tune_away

```
   do
   {
   ~~memcpy(&tune_away, params_ptr->data, params_ptr->datalen);~~
   memcpy(&tune_away, params_ptr->data, sizeof(tune_away));
   QCRIL_LOG_INFO( "tune_away: %d", (int) tune_away );

   if( (tune_away != TRUE) && (tune_away != FALSE) )

. . .

                                    ril_req_res,
                                    &resp );
       qcril_send_request_response( &resp );
       break;
   }

   NAS_CACHE_LOCK()
```

4.6.5 Change 5

File: qcril_qmi_nas.c

Function: qcril_qmi_nas_dsds_request_set_paging_priority

```
   do
   {
   ~~memcpy(&paging_priority, params_ptr->data, params_ptr->datalen);~~
   memcpy(&paging_priority, params_ptr->data, sizeof(paging_priority));
   QCRIL_LOG_INFO( "paging_priority: %d", (int) paging_priority );

   if( (paging_priority < RIL_SUBSCRIPTION_1) || (paging_priority >
RIL_SUBSCRIPTION_3) )
. . .
                                                ril_req_res,
                                                &resp );
        qcril_send_request_response( &resp );
        break;
    }

    NAS_CACHE_LOCK()
```

4.6.6 Change 6

File: qcril_qmi_nas.c

Function: qcril_qmi_nas_dsds_request_set_default_voice_sub

```
                                                ril_req_res,
                                                &resp );
        qcril_send_request_response( &resp );
        break;
    }
```

4.7 [523657] Out-of-bounds array access in MMGSDI SimLock algorithm

4.7.1 Change 1

File: mmgsdi_perso_sec.c

Function: gsdi_perso_security_update_counters

```
      Get the Control Key Counters
      -------------------------------------------------------------------------*/
    gsdi_status = gsdi_perso_security_decrypt_data(encrypted_ck_counters_ptr,
                                                   &decrypted_data,
                                                   encrypted_ck_counters_ptr->len /
2);

    if ( gsdi_status != GSDI_SUCCESS )
    {
      MMGSDIUTIL_TMC_MEM_FREE(decrypted_data.buffer_ptr);
      return gsdi_status;
    }

    /* If the offset is greater than the length of buffer decrypted_data.buffer_ptr,
       we will run into buffer overread when trying to access
```

```
   decrypted_data.buffer_ptr at index "offset". To avoid that, we have a
   check here. The assumption is that the function used to decrypt security data
   correctly populates the length of buffer into decrypted_data.len */
if ( decrypted_data.len <= offset )
{
  MMGSDIUTIL_TMC_MEM_FREE(decrypted_data.buffer_ptr);
  return GSDI_PERSO_INVALID_DATA;
}


/* Get number of retries */
retries = decrypted_data.buffer_ptr[offset];

/* ---------------------------------------------------------------------------
  If not currently blocked...ok to decrement
  -------------------------------------------------------------------------*/
if (retries != GSDI_PERSO_NO_MORE_CK_RETRIES)
{
```

4.7.2 Change 2

File: mmgsdi_perso_sec.c

Function: gsdi_perso_security_reset_counters

```
   Based on the Mask provided, calculate the starting offset of the
   control key counter
   ------------------------------------------------------------------------*/
offset = gsdi_perso_sec_get_offset(req_data_ptr->feature_mask);
if (offset == 0xFFFF)
{
  return GSDI_PERSO_INVALID_DATA;
}

/* ---------------------------------------------------------------------------
   Allocate memory to hold the uncrypted values
   -------------------------------------------------------------------------*/
decrypted_data.buffer_ptr = mmgsdi_malloc((encrypted_ck_counters_ptr->len/2));
MMGSDI_RETURN_IF_NULL(decrypted_data.buffer_ptr);

/* ---------------------------------------------------------------------------
  Get the Control Key Counters
  -------------------------------------------------------------------------*/
gsdi_status = gsdi_perso_security_decrypt_data(encrypted_ck_counters_ptr,
                                               &decrypted_data,
                                                  encrypted_ck_counters_ptr->len /
2);

    if ( gsdi_status != GSDI_SUCCESS )
    {
      MMGSDIUTIL_TMC_MEM_FREE(decrypted_data.buffer_ptr);
      return gsdi_status;
    }

    /* If the offset is greater than the length of buffer decrypted_data.buffer_ptr,
       we will run into buffer overread when trying to access
       decrypted_data.buffer_ptr at index "offset". To avoid that, we have a
       check here. The assumption is that the function used to decrypt security data
       correctly populates the length of buffer into decrypted_data.len */
    if ( decrypted_data.len <= offset )
    {
      MMGSDIUTIL_TMC_MEM_FREE(decrypted_data.buffer_ptr);
      return GSDI_PERSO_INVALID_DATA;
    }

    decrypted_data.buffer_ptr[offset] = num_retries;

    /* ---------------------------------------------------------------------------
      Write data back to the file
      ------------------------------------------------------------------------*/
    /* Encrypt the Data Accordingly */
    gsdi_status = gsdi_perso_security_encrypt_data(&decrypted_data,
                                                  encrypted_ck_counters_ptr);
```

4.7.3 Change 3

File: mmgsdi_perso_sec.c

Function: gsdi_perso_security_change_feature_inds

```
    /* --------------------------------------------------------------------------
       Decrypt the data provided in the ME Encrypted
       Data Structure and set the feature indicator
       to the new state.
       -----------------------------------------------------------------------*/
    gsdi_status = gsdi_perso_security_decrypt_data(&enc_me_data_ptr->feat_inds,
                                                  &decrypted_data,
                                                  len);

    ~~if ( gsdi_status != GSDI_SUCCESS )~~
    /* If the offset is greater than the length of buffer decrypted_data.buffer_ptr,
       we will run into buffer overread when trying to access
       decrypted_data.buffer_ptr at index "offset". To avoid that, we have a
       check here. The assumption is that the function used to decrypt security data
       correctly populates the length of buffer into decrypted_data.len */
    if ( gsdi_status != GSDI_SUCCESS ||
         decrypted_data.len <= offset )
    {
      MMGSDIUTIL_TMC_MEM_FREE(decrypted_data.buffer_ptr);
      return GSDI_PERSO_INVALID_DATA;
    }

    /*If the feature is to be activated but there are no SIM Lock Codes, go to the
 Auto Lock State.
      On Power Up, the Initialization function will create the sim lock codes.*/
    if(feature_state == GSDI_PERSO_SEC_ACTIVATE_FEATURE)
    {
      do
```

4.7.4 Change 4

File: mmgsdi_perso_sec.c

Function: gsdi_perso_sec_reset_num_retries

```
   gsdi_status = gsdi_perso_security_decrypt_data(&encrypted_data_ptr->dck_counts,
                                                  &decrypted_data,
                                                  encrypted_data_ptr-
>dck_counts.len / 2);

   if ( gsdi_status != GSDI_SUCCESS )
   {
     MMGSDIUTIL_TMC_MEM_FREE(decrypted_data.buffer_ptr);
     return gsdi_status;
   }

   /* If the offset is greater than the length of buffer decrypted_data.buffer_ptr,
      we will run into buffer overread when trying to access
      decrypted_data.buffer_ptr at index "offset". To avoid that, we have a
      check here. The assumption is that the function used to decrypt security data
      correctly populates the length of buffer into decrypted_data.len */
   if ( decrypted_data.len <= offset )
   {
     MMGSDIUTIL_TMC_MEM_FREE(decrypted_data.buffer_ptr);
     return GSDI_PERSO_INVALID_DATA;
   }
```

```
    decrypted_data.buffer_ptr[offset] =  GSDI_PERSO_MAX_NUMBER_OF_RETRIES;

    gsdi_status = gsdi_perso_security_encrypt_data(&decrypted_data,
                                                   &encrypted_data_ptr->dck_counts);


    if ( gsdi_status != GSDI_SUCCESS )
    {
      MMGSDIUTIL_TMC_MEM_FREE(decrypted_data.buffer_ptr);
      return GSDI_PERSO_INVALID_DATA;
    }
```

4.7.5 Change 5

File: mmgsdi_perso_sec.c

Function: gsdi_perso_sec_reset_num_retries

```
  gsdi_status = gsdi_perso_security_decrypt_data(&encrypted_data_ptr-
>unblock_ck_counts,
                                                 &decrypted_data,
                                                 encrypted_data_ptr->dck_counts.len
/ 2);

  if ( gsdi_status != GSDI_SUCCESS )
  {
    MMGSDIUTIL_TMC_MEM_FREE(decrypted_data.buffer_ptr);
    return gsdi_status;
  }

   /* If the offset is greater than the length of buffer decrypted_data.buffer_ptr,
      we will run into buffer overread when trying to access
      decrypted_data.buffer_ptr at index "offset". To avoid that, we have a
      check here. The assumption is that the function used to decrypt security data
      correctly populates the length of buffer into decrypted_data.len */
   if ( decrypted_data.len <= offset )
   {
     MMGSDIUTIL_TMC_MEM_FREE(decrypted_data.buffer_ptr);
     return GSDI_PERSO_INVALID_DATA;
   }

  decrypted_data.buffer_ptr[offset] =  GSDI_PERSO_MAX_UNBLOCK_NUMBER_OF_RETRIES;

  gsdi_status = gsdi_perso_security_encrypt_data(&decrypted_data,
                                                 &encrypted_data_ptr-
>unblock_ck_counts);
```

```
   MMGSDIUTIL_TMC_MEM_FREE(decrypted_data.buffer_ptr);
   return gsdi_status;
}/* gsdi_perso_sec_reset_num_retries */
```

4.8 [528497] Integer overflow vulnerability in PERSO_SET_DATA

4.8.1 Change 1

File: mmgsdi_perso_engine.c

Function: gsdi_perso_engine_get_encrypted_data

```
             MMGSDIUTIL_TMC_MEM_FREE(orig_file_buffer_ptr);
             return GSDI_INCORRECT_PARAMS;
           }
           /* Check for integer overflow */
           if(((gsdi_perso_set_feature_data_req_T *)req_ptr)->num_bytes >
MMGSDI_INT32_MAX / 2 ||
              mem_pointer_ptr->memory_curr > MMGSDI_UINT32_MAX -
((gsdi_perso_set_feature_data_req_T *)req_ptr)->num_bytes *2)
           {
             MMGSDIUTIL_TMC_MEM_FREE(orig_file_buffer_ptr);
             return GSDI_INCORRECT_PARAMS;
           }
           mem_pointer_ptr->memory_curr += ((gsdi_perso_set_feature_data_req_T
*)req_ptr)->num_bytes *2;
         }
         file_buffer_ptr += GSDI_PERSO_ENGINE_LEN_SIZE + sec_data_ptr-
>sim_lock_codes.len;
```

4.8.2 Change 2

File: mmgsdi_perso_engine.c

Function: gsdi_perso_engine_calc_enc_data_len

```
   if(feature_ind == GSDI_PERSO_ENGINE_SET_DATA_FEATURE)
   {
     MMGSDI_RETURN_IF_NULL(req_ptr);
     /* Check for integer overflow */
     if(((gsdi_perso_set_feature_data_req_T *)req_ptr)->num_bytes >
MMGSDI_INT32_MAX / 2 ||
        *len_ptr > MMGSDI_INT32_MAX - ((gsdi_perso_set_feature_data_req_T
*)req_ptr)->num_bytes *2)
     {
       return GSDI_ERROR;
     }
     *len_ptr += ((gsdi_perso_set_feature_data_req_T *)req_ptr)->num_bytes *2;
   }
   /* Check to see if new SIM Lock Codes have to be generated */
```

4.8.3 Change 3

File: mmgsdi_perso_engine_ruim.c

Function: gsdi_perso_engine_calc_ruim_enc_data_len

```
   if(feature_ind == GSDI_PERSO_ENGINE_SET_DATA_FEATURE)
   {
     MMGSDI_RETURN_IF_NULL(req_ptr);
     /* Check for integer overflow */
     if(((gsdi_perso_set_feature_data_req_T *)req_ptr)->num_bytes >
MMGSDI_INT32_MAX / 2 ||
        *len_ptr > MMGSDI_INT32_MAX - ((gsdi_perso_set_feature_data_req_T
*)req_ptr)->num_bytes *2)
     {
       return GSDI_ERROR;
     }
     *len_ptr += ((gsdi_perso_set_feature_data_req_T *)req_ptr)->num_bytes *2;
   }
   /* Check to see if new SIM Lock Codes have to be generated */
```

4.8.4 Change 4

File: mmgsdi_perso_engine_ruim.c

Function: gsdi_perso_engine_calc_ruim_enc_data_len

```
         }
         else if(feature_type == GSDI_PERSO_ENGINE_SET_DATA_FEATURE)
         {
           ~~MMGSDI_RETURN_IF_NULL(req_ptr);~~
           if(req_ptr == NULL)
           {
             MMGSDIUTIL_TMC_MEM_FREE(orig_file_buffer_ptr);
             return GSDI_INCORRECT_PARAMS;
           }
           /* Check for integer overflow */
           if(((gsdi_perso_set_feature_data_req_T *)req_ptr)->num_bytes >
MMGSDI_INT32_MAX / 2 ||
              mem_pointer_ptr->memory_curr > MMGSDI_UINT32_MAX -
((gsdi_perso_set_feature_data_req_T *)req_ptr)->num_bytes *2)
           {
             MMGSDIUTIL_TMC_MEM_FREE(orig_file_buffer_ptr);
             return GSDI_INCORRECT_PARAMS;
           }
           mem_pointer_ptr->memory_curr += ((gsdi_perso_set_feature_data_req_T
*)req_ptr)->num_bytes *2;
         }
         file_buffer_ptr += GSDI_PERSO_ENGINE_LEN_SIZE + sec_data_ptr-
>sim_lock_codes.len;
```

4.9 [529366] Improper validation of array index when processing personalization command to lock down the keys

4.9.1 Change 1

File: mmgsdi_perso_sec.c

Function: gsdi_perso_sec_get_locked_mask

```
   gsdi_status = gsdi_perso_sec_calc_mem_required(
                              GSDI_PERSO_SEC_CALC_LOCK_DOWN,
                              &encrypted_data_ptr->lock_down,
                              &len,
                              0);

   if ( gsdi_status != GSDI_SUCCESS )
   {
     return gsdi_status;
   }
   if ( len == 0 )
   {
     MSG_ERROR("PERSO: Wrong length 0x%x",len,0,0);
     return GSDI_PERSO_INVALID_DATA;
   }

   decrypted_data.buffer_ptr = mmgsdi_malloc(len);
   MMGSDI_RETURN_IF_NULL(decrypted_data.buffer_ptr);

   /* ---------------------------------------------------------------------
      Next resolve the data provided in the feature indicators and
      determine the state of each one.
      -------------------------------------------------------------------*/
   gsdi_status = gsdi_perso_security_decrypt_data(&encrypted_data_ptr->lock_down,
                                                  &decrypted_data,
```

4.9.2 Change 2

File: mmgsdi_perso_sec.c

Function: gsdi_perso_sec_validate_data_against_fs

```
     return GSDI_PERSO_INVALID_DATA;
   }

   if(len == 0)
   {
     MSG_ERROR("PERSO: Wrong length 0x%x",len,0,0);
     return GSDI_PERSO_INVALID_DATA;
   }

   features.buffer_ptr = mmgsdi_malloc(len);

   if ( features.buffer_ptr == NULL )
   {
     MMGSDI_DEBUG_MSG_ERROR("PERSO: NULL Pointer in
gsdi_perso_sec_validate_data_against_fs",0,0,0);
     return GSDI_ERROR;
   }

   gsdi_status = gsdi_perso_sec_decrypt_tlv(mask,
                                            &encrypted_data_ptr->sim_lock_codes,
                                            &features,
                                            len);

   if ( gsdi_status != GSDI_SUCCESS )
   {
     MMGSDIUTIL_TMC_MEM_FREE(features.buffer_ptr);
     return GSDI_PERSO_INVALID_DATA;
```

4.9.3 Change 3

File: mmgsdi_perso_sec.c

Function: gsdi_perso_sec_check_sim_codes

```
   {
     return GSDI_PERSO_INVALID_DATA;
   }

   if(len ==0)
   {
      return gsdi_status;
```

```
    }

    features.buffer_ptr = mmgsdi_malloc(len);

    if ( features.buffer_ptr == NULL )
    {
      MMGSDI_DEBUG_MSG_ERROR("PERSO: NULL Pointer in
gsdi_perso_sec_check_sim_codes",0,0,0);
      return GSDI_ERROR;
    }

    gsdi_status = gsdi_perso_sec_decrypt_tlv(mask,
                                             &encrypted_data_ptr->sim_lock_codes,
                                             &features,
                                             len);

    if ( gsdi_status != GSDI_SUCCESS )
    {
      MMGSDIUTIL_TMC_MEM_FREE(features.buffer_ptr);
      return gsdi_status;
```

4.9.4 Change 4

File: mmgsdi_perso_sec.c

Function: gsdi_perso_security_lock_down

```
    /* ------------------------------------------------------------------------
       Determine how much memory is required to store the
       decrypted Featuer Indications.
       ----------------------------------------------------------------------*/
    gsdi_status = gsdi_perso_sec_calc_mem_required(
                               GSDI_PERSO_SEC_CALC_LOCK_DOWN,
                               &encrypted_data_ptr->lock_down,
                               &len,
                               req_data_ptr->feature_mask);

    ~~if ( gsdi_status != GSDI_SUCCESS )~~
    if ( gsdi_status != GSDI_SUCCESS ||
         len == 0 )
    {
      return GSDI_PERSO_INVALID_DATA;
    }

    decrypted_data.buffer_ptr = mmgsdi_malloc(len);
    MMGSDI_RETURN_IF_NULL(decrypted_data.buffer_ptr);
```

```
/* ---------------------------------------------------------------------------
   Next resolve the data provided in the feature indicators and
   determine the state of each one.
```

4.10 [519773] Playing a clip with H.264 video having bit-stream errors in slices causes phone to reboot

The changes for this vulnerability are delivered in DSP files.

4.11 [358100] Receiving probe response without mandatory SSID IE causes buffer overread

The changes for this vulnerability are contained within a binary file.