Buffer overflow in usb device class

High **ceolin** published **GHSA-fm6v-8625-99jf** on Feb 7

Package	
zephyr (west)	
Affected versions	Patched versions
v2.6.0	None

Description

Impact

I believe that the Bluetooth usb device class provided with Zephyr is vulnerable to a buffer overflow.

When a control transfer is handled by bluetooth_class_handler (subsys/usb/class/bluetooth.c) a bluetooth tx buffer is first acquired, populated with contents of data buffer and when this operation is successful the buffer is pushed to rx_queue.

```
static int bluetooth_class_handler(struct usb_setup_packet *setup,

                                 int32_t *len, uint8_t **data)

{

        struct net_buf *buf;


        LOG_DBG("len %u", *len);


        buf = bt_buf_get_tx(BT_BUF_CMD, K_NO_WAIT, *data, *len);

        if (!buf) {

                LOG_ERR("Cannot get free buffer\n");

                return -ENOMEM;

        }


        net_buf_put(&rx_queue, buf);


        return 0;

}
```

Following the execution flow of bt_buf_get_tx (subsys/bluetooth/host/hci_raw.c) one may observe few significant steps – buffer allocation, reservation, type set and net_buf_add_mem.

```
        buf = net_buf_alloc(pool, timeout);          //SH: allocate BT_HCI_CMD_HDR_SIZE + CONFIG_BT_BUF_CMD_TX_SIZE bytes

        if (!buf) {

               return buf;

        }

        net_buf_reserve(buf, BT_BUF_RESERVE);

        bt_buf_set_type(buf, type);

        if (data && size) {

               net_buf_add_mem(buf, data, size);

        }


        return buf;
```

With pool set to hci_cmd_pool and successful allocation, data and size provided in the control transfer request we finally reach `net_buf_add_mem` and `net_buf_simple_add_mem` .

```
static inline void *net_buf_add_mem(struct net_buf *buf, const void *mem,
                                  size_t len)

{

        return net_buf_simple_add_mem(&buf->b, mem, len);

}

void *net_buf_simple_add_mem(struct net_buf_simple *buf, const void *mem,
                                                               size_t len)

{
```

```
                    NET_BUF_SIMPLE_DBG("buf %p len %zu", buf, len);

                    return memcpy(net_buf_simple_add(buf, len), mem, len);     //SH: if buffer is not large enough copy len bytes from mem to buf violating boundaries

}

void *net_buf_simple_add(struct net_buf_simple *buf, size_t len)

{

        uint8_t *tail = net_buf_simple_tail(buf);

        NET_BUF_SIMPLE_DBG("buf %p len %zu", buf, len);

        __ASSERT_NO_MSG(net_buf_simple_tailroom(buf) >= len);             //SH: shouldn't this case be properly handled?

        buf->len += len;            //SH: does this buffer really have enough space?

        return tail;

}
```

As far as I can understand in case the net_buf_simple buffer does not have enough free space the buf->len will still be incremented by len "stretching" the buffer past actual boundaries.

Len bytes of data provided in the control transfer request will be copied to the buffer possibly overwriting memory past the end of buffer.

Depending on memory contents this might lead to malicious modifications of important data structures or execution of arbitrary code. With a "write to device" usb control transfer one may write up to CONFIG_USB_REQUEST_BUFFER_SIZE bytes of data (i.e. 2048 as in overlay-usb-nrf-br.conf) writing past a CONFIG_BT_BUF_CMD_TX_SIZE (i.e. 65 bytes for nrf51_blenano.conf) + BT_HCI_CMD_HDR_SIZE (3 bytes) buffer.

What kind of vulnerability is it? Who is impacted?

Patches

This has been fixed in:

- main #42093
- v2.7: #42167

References

Are there any links users can visit to find out more?

For more information

If you have any questions or comments about this advisory:

- Open an issue in zephyr
- Email us at Zephyr-vulnerabilities

embargo: 2021-10-02

Severity

High 8.2 / 10

CVSS base metrics	
Attack vector	Adjacent
Attack complexity	High
Privileges required	None
User interaction	None
Scope	Changed
Confidentiality	Low
Integrity	High
Availability	High

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

CVE ID

CVE-2021-3835

Weaknesses

CWE-122

Credits

szymonh