DFU UPLOAD buffer overflow revised

High liydu published GHSA-m9p8-xrp7-vvqp 22 days ago

Package

USBX (Azure RTOS)

Affected versions	Patched versions
< 6.1.12	6.1.12

Description

Impact

The USB DFU UPLOAD functionality may be utilized to introduce a buffer overflow resulting in overwrite of memory contents. In particular cases this may allow an attacker to bypass security features or execute arbitrary code.

The implementation of `ux_device_class_dfu_control_request` function prevents buffer overflow during handling of DFU UPLOAD command when current state is `UX_SYSTEM_DFU_STATE_DFU_IDLE`. Validation for this condition is implemented as in the following snippet.

```
case UX_SLAVE_CLASS_DFU_COMMAND_UPLOAD:

/* bitCanUpload != 1, or length = 0, or length > wTransferSize (we can support max of control bu
if (!(_ux_system_slave -> ux_system_slave_device_dfu_capabilities &amp; UX_SLAVE_CLASS_DFU_CAPAB
(request_length == 0) ||
(request_length > UX_SLAVE_REQUEST_CONTROL_MAX_LENGTH))
{
_ux_device_stack_endpoint_stall(&amp;device -> ux_slave_device_control_endpoint);

/* In the system, state the DFU state machine to DFU ERROR. */
_ux_system_slave -> ux_system_slave_device_dfu_state_machine = UX_SYSTEM_DFU_STATE_DFU_ERROR;

break;
}
```

Once a correct DFU UPLOAD control request transfer is handled the DFU state machine transitions to `UX_SYSTEM_DFU_STATE_DFU_UPLOAD_IDLE` state which lacks the required validation.

```c
case UX_SLAVE_CLASS_DFU_STATUS_STATE_DFU_UPLOAD_IDLE:

/* Here we process only the request we can accept in the DFU mode UPLOAD IDLE state. */
switch (request)
{

case UX_SLAVE_CLASS_DFU_COMMAND_UPLOAD:

/* Length 0 case undefined, just keep state. */
if (request_length == 0)
break;

/* We received a UPLOAD command with length > 0. */

/* Read the next block from the firmware. */
status = dfu -> ux_slave_class_dfu_read(dfu, request_value,
transfer_request -> ux_slave_transfer_request_data_pointer,
request_length,
&amp;actual_length);

/* Application can actively reject and set error state. */
if (status != UX_SUCCESS)
{
_ux_device_stack_endpoint_stall(&amp;device -> ux_slave_device_control_endpoint);
_ux_system_slave -> ux_system_slave_device_dfu_state_machine = UX_SYSTEM_DFU_STATE_DFU_ERROR;
break;
}

/* If it&#39;s short frame, switch to dfu IDLE. */
if (actual_length &lt; request_length)
{

/* Send a notification to the application. */
dfu -> ux_slave_class_dfu_notify(dfu, UX_SLAVE_CLASS_DFU_NOTIFICATION_END_UPLOAD);

/* In the system, state the DFU state machine to dfu IDLE. */
_ux_system_slave -> ux_system_slave_device_dfu_state_machine = UX_SYSTEM_DFU_STATE_DFU_IDLE;
}

/* We have a request to upload DFU firmware block. */
_ux_device_stack_transfer_request(transfer_request, actual_length, request_length);

break;
```

With state machine in `UX_SLAVE_CLASS_DFU_STATUS_STATE_DFU_UPLOAD_IDLE` processing a control transfer request for `UX_SLAVE_CLASS_DFU_COMMAND_UPLOAD` with `wLenght` greater than `UX_SLAVE_REQUEST_CONTROL_MAX_LENGTH` will result in a buffer overflow (`dfu->ux_slave_class_dfu_read` will overflow the `UX_SLAVE_REQUEST_CONTROL_MAX_LENGTH` buffer) which in turn may result in a RCE with attacker code injected to flash either by utilizing DFU DOWNLOAD command or manipulating contents of an external flash chip (when used).

Patches

We analyzed this bug and determined that we needed to fix it. This fix has been included in USBX release 6.1.12

Workarounds

Add the `UPLOAD_LENGTH` check in all possible states

For more information

If you have any questions or comments about this advisory:

Open an issue in azure-rtos/usbx
Post question on Microsoft Q&A

Severity

High

CVE ID

CVE-2022-39344

Weaknesses

No CWEs