New issue

Jump to bottom

Stack overflow in SNMP bulk request processing #1353

Open **mjurczak** opened this issue on Aug 17, 2020 · 1 comment

Labels bug/vulnerability

mjurczak commented on Aug 17, 2020 Contributor

Description of defect

References:

https://github.com/contiki-ng/contiki-ng/tree/release/v4.5
https://github.com/contiki-ng/contiki-ng/tree/release/v4.4

File:

snmp-engine.c
snmp-message.c

Analysis:

Memory access out of buffer boundaries may occur if an SNMP bulk get request with number of OIDs larger than supported by the engine is received and processed.

The OIDs listed in a request are processed by snmp_message_decode() function without verification of the varbinds buffer capacity.
The varbinds memory buffer is written with the values provided in SNMP request:

contiki-ng/os/net/app-layer/snmp/snmp-message.c
Line 245 in 23db957

```
245      buf = snmp_oid_decode_oid(buf, &buf_len, varbinds[i].oid, &oid_len);
```

The buffer capacity is determined by:

contiki-ng/os/net/app-layer/snmp/snmp-conf.h
Lines 81 to 87 in 23db957

```
81      #define SNMP_MAX_NR_VALUES SNMP_CONF_MAX_NR_VALUES
82      #else
83      /**
84       * \brief Default maximum number of OIDs in one response
85       */
86      #define SNMP_MAX_NR_VALUES 2
87      #endif
```

SNMP get bulk requests are processed by snmp_engine_get_bulk() function that allocates a local stack buffer for buffering OIDs of the requested variables.

contiki-ng/os/net/app-layer/snmp/snmp-engine.c
Lines 116 to 121 in 23db957

```
116      snmp_engine_get_bulk(snmp_header_t *header, snmp_varbind_t *varbinds, uint32_t *varbinds_length)
117      {
118        snmp_mib_resource_t *resource;
119        uint32_t i, j, original_varbinds_length;
120        uint32_t oid[SNMP_MAX_NR_VALUES][SNMP_MSG_OID_MAX_LEN];
121        uint8_t repeater;
```

The stack buffer in snmp_engine_get_bulk() is populated with OIDs as a first step before any further processing of the data.

contiki-ng/os/net/app-layer/snmp/snmp-engine.c
Lines 123 to 130 in 23db957

```
123        /*
124         * A local copy of the requested oids must be kept since
125         *  the varbinds are modified on the fly
126         */
127        original_varbinds_length = *varbinds_length;
128        for(i = 0; i < original_varbinds_length; i++) {
129          snmp_oid_copy(oid[i], varbinds[i].oid);
130        }
```

The varbinds_length variable value is not verified against the capacity of the temporary oid stack buffer. If the number of requested OIDs exceeds the buffer capacity a stack buffer overflow condition occurs and stack memory beyond the allocated oid buffer is overwritten with OIDs received in SNMP get bulk request.

As the OIDs are supplied in the request content it may be possible to alter the return address from the snmp_engine_get_bulk() function. If the target architecture uses common addressing space for program and data memory (which is common in IoT devices) it may also be possible to supply code in the SNMP request payload and redirect the execution path to the injected code by modification of the return address.

Type:

- Out-of-bounds memory write
- Stack memory overwrite
- Return address alteration
- Remote altering of code execution path
- Remote executable code injection
- Remote code execution

Result:

- Memory corruption
- Remote code execution

Target(s) affected by this defect ?

- contiki-ng v4.5
- contiki-ng v4.4

Fix

Rudimentary fix to address the most critical aspect of the issue:
https://github.com/mjurczak/contiki-ng/tree/bugfix/snmp-engine

How is this defect reproduced ?

An example SNMP request causing stack overwrite:

```
306102010104067075626C6963A554020431D065A702010402010A3046300C06082B060102010102000500300C06082B06010201010201050300C06082B0601
0201010202050300C06082B060102010102030500300C06082B0601020101020405 00
```

mjurczak mentioned this issue on Aug 17, 2020

Bugfix/snmp engine #1355

Merged

Yagoor mentioned this issue on Sep 8, 2020

SNMP Engine - New Unit Tests #1376

Closed

g-oikonomou commented on Nov 25, 2020 Member

@Yagoor @mjurczak: Am I right to assume that this has been fixed in #1355 and/or #1397? Can we close?

g-oikonomou added the bug/vulnerability label on Nov 25, 2020

Assignees

No one assigned

Labels

bug/vulnerability

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

2 participants