Talos Vulnerability Report

TALOS-2022-1467

MZ Automation GmbH libiec61850 parseNormalModeParameters denial of service vulnerability

FEBRUARY 28, 2022

CVE NUMBER

CVE-2022-21159

Summary

A denial of service vulnerability exists in the parseNormalModeParameters functionality of MZ Automation GmbH libiec61850 1.5.0. A specially-crafted series of network requests can lead to denial of service. An attacker can send a sequence of malformed iec61850 messages to trigger this vulnerability.

Tested Versions

MZ Automation GmbH libiec61850 1.5.0

Product URLs

libiec61850 - https://github.com/mz-automation/libiec61850

CVSSv3 Score

7.5 - CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CWE

CWE-835 - Loop with Unreachable Exit Condition (‘Infinite Loop’)

Details

libiec61850 is a C implementation of the various protocols defined within the IEC61850 specification. Example client/server applications are provided for common use cases. This library can also be used as a building block to integrate IEC61850 communications into a custom client/server application.

When an IEC61850 message containing a Presentation layer with Normal Mode parameters is processed, execution enters the `parseNormalModeParameters` function in `iso_presentation.c`. This function loops over the message buffer starting from a provided `bufPos` position until the `endPos` position is reached or a failure case is encountered. At the start of each iteration, the BER tag for the current parameter is extracted from the buffer, as shown below.

```
static int
parseNormalModeParameters(IsoPresentation* self, uint8_t* buffer, int totalLength,
int bufPos)
{
    int endPos = bufPos + totalLength;

    self->calledPresentationSelector.size = 0;
    self->callingPresentationSelector.size = 0;

    bool hasUserData = false;

    while (bufPos < endPos) {
        uint8_t tag = buffer[bufPos++];
        int len;

        if (bufPos == endPos) {
            if (DEBUG_PRES)
                printf("PRES: invalid message\n");
            return -1;
        }

        bufPos = BerDecoder_decodeLength(buffer, &len, bufPos, endPos);
```

The value of this tag determines how the bytes following should be processed. Three cases in particular are notable: 0xA4, 0x00 and the default.

When the tag 0x00 is encountered, execution just continues on to the next loop iteration and starts processing the next tag. In this case the `bufPos` variable will be incremented by one when the next tag is read.

```
case 0x00: /* indefinite length end tag -> ignore */
break;
```

When an unknown tag is encountered, the buffer position is incremented to skip past it, and execution continues on to the next loop iteration and starts processing the next tag. This allows for longer position jumps within the buffer.

```
default:
    if (DEBUG_PRES)
        printf("PRES: unknown tag in normal-mode\n");

    bufPos += len;
    break;
```

When the `presentation-context-definition list` parameter (0xA4) is encountered, the function `parsePresentationContextDefinitionList` is used to perform additional parsing. Notably, the result of `parsePresentationContextDefinitionList` gets placed into the `bufPos` variable and is not checked before the next loop iteration.

```
case 0xa4: /* presentation-context-definition list */
    if (DEBUG_PRES)
        printf("PRES: pcd list\n");

    bufPos = parsePresentationContextDefinitionList(self, buffer, len, bufPos);
    break;
```

`parsePresentationContextDefinitionList` starts off by calculating the end of the presentation-context-definition list before entering a processing loop where the next tag is extracted and its length decoded.

```
static int
parsePresentationContextDefinitionList(IsoPresentation* self, uint8_t* buffer, int
totalLength, int bufPos)
{
    int endPos = bufPos + totalLength;

    while (bufPos < endPos) {
        uint8_t tag = buffer[bufPos++];
        int len;

        bufPos = BerDecoder_decodeLength(buffer, &len, bufPos, endPos);
        if (bufPos < 0)
            return -1;
```

`BerDecoder_decodeLength` starts recursively processing the length, returning the result.

```
int
BerDecoder_decodeLength(uint8_t* buffer, int* length, int bufPos, int maxBufPos)
{
    return BerDecoder_decodeLengthRecursive(buffer, length, bufPos, maxBufPos, 0,
50);
}
```

Finally, `BerDecoder_decodeLengthRecursive` is called, and a check is made to determine if the current position is greater than or equal to the maximum allowed buffer. When it is determined to be so, `-1` is returned.

```
static int
BerDecoder_decodeLengthRecursive(uint8_t* buffer, int* length, int bufPos, int
maxBufPos, int depth, int maxDepth)
{
    if (bufPos >= maxBufPos){
        return -1;
    }
```

When this occurs, `-1` gets returned up the chain to `parsePresentationContextDefinitionList`.

```
static int
parsePresentationContextDefinitionList(IsoPresentation* self, uint8_t* buffer, int
totalLength, int bufPos)
{
    int endPos = bufPos + totalLength;

    while (bufPos < endPos) {
        uint8_t tag = buffer[bufPos++];
        int len;

        bufPos = BerDecoder_decodeLength(buffer, &len, bufPos, endPos);
        if (bufPos < 0)
            return -1;
```

Here `bufPos` is determined to be less than zero and `-1` is again returned, this time back to the `presentation-context-definition list` case within `parseNormalModeParameters`.

```
case 0xa4: /* presentation-context-definition list */
    if (DEBUG_PRES)
        printf("PRES: pcd list\n");

    bufPos = parsePresentationContextDefinitionList(self, buffer, len, bufPos);
    break;
```

This leaves `bufPos` set to `-1` at the start of the next loop iteration, allowing the while condition to pass and begin again.

```
static int
parseNormalModeParameters(IsoPresentation* self, uint8_t* buffer, int totalLength,
int bufPos)
{
    int endPos = bufPos + totalLength;

    self->calledPresentationSelector.size = 0;
    self->callingPresentationSelector.size = 0;

    bool hasUserData = false;

    while (bufPos < endPos) {
        uint8_t tag = buffer[bufPos++];
        int len;
```

If a message is properly constructed such that `len` and `bufPos` add together to equal `endPos`, the failure case in `BerDecoder_decodeLengthRecursive` will be hit and `bufPos` will be set to `-1`. Then as long as nothing within the buffer causes `bufPos` to be greater than `endPos`, and the `presentation-context-definition list` continues to be executed. An infinite loop condition will be reached.

Since each connection is handled in its own thread, one of these messages will not be enough to cause a denial of service. However, by sending the message multiple times it is possible to tie up all of the available connections and prevent legitimate communications from getting through to the server. The maximum number of client connections is set to a default value of 100 in `stack_config.h`, as shown below:

```
/* number of concurrent MMS client connections the server accepts, -1 for no limit
*/
#define CONFIG_MAXIMUM_TCP_CLIENT_CONNECTIONS 100
```

If at least `CONFIG_MAXIMUM_TCP_CLIENT_CONNECTIONS` malformed messages are sent, all available connections will be stuck in infinite loops, preventing any new connections from being processed

Crash Information

When a known good MMS Initiate Request message is sent to the server, an Initiate Response message is expected to be returned.

```
"No.","Time","Source","Destination","Protocol","Length","Info"
"1","0.000000000","127.0.0.1","127.0.0.1","TCP","74","46602  >  102 [SYN] Seq=0
Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=3775924408 TSecr=0 WS=128"
"2","0.000010190","127.0.0.1","127.0.0.1","TCP","74","102  >  46602 [SYN, ACK] Seq=0
Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=3775924408 TSecr=3775924408
WS=128"
"3","0.000019233","127.0.0.1","127.0.0.1","TCP","66","46602  >  102 [ACK] Seq=1
Ack=1 Win=65536 Len=0 TSval=3775924408 TSecr=3775924408"
"4","0.000144676","127.0.0.1","127.0.0.1","MMS","263","initiate-RequestPDU "
"5","0.000149145","127.0.0.1","127.0.0.1","TCP","66","102  >  46602 [ACK] Seq=1
Ack=198 Win=65408 Len=0 TSval=3775924408 TSecr=3775924408"
"6","0.000207725","127.0.0.1","127.0.0.1","MMS","208","initiate-ResponsePDU "
"7","0.000228681","127.0.0.1","127.0.0.1","TCP","66","46602  >  102 [ACK] Seq=198
Ack=143 Win=65408 Len=0 TSval=3775924408 TSecr=3775924408"
"8","0.000280524","127.0.0.1","127.0.0.1","TCP","66","46602  >  102 [FIN, ACK]
Seq=198 Ack=143 Win=65536 Len=0 TSval=3775924408 TSecr=3775924408"
"9","0.042521649","127.0.0.1","127.0.0.1","TCP","66","102  >  46602 [ACK] Seq=143
Ack=199 Win=65536 Len=0 TSval=3775924450 TSecr=3775924408"
"10","0.042576068","127.0.0.1","127.0.0.1","TCP","66","102  >  46602 [FIN, ACK]
Seq=143 Ack=199 Win=65536 Len=0 TSval=3775924450 TSecr=3775924408"
"11","0.042668070","127.0.0.1","127.0.0.1","TCP","66","46602  >  102 [ACK] Seq=199
Ack=144 Win=65536 Len=0 TSval=3775924450 TSecr=3775924450"
```

After the poc has been run and all available client connections are stuck in a loop, a TCP Reset is received instead of an Initiate Response message.

```
"No.","Time","Source","Destination","Protocol","Length","Info"
"1408","35.098853688","127.0.0.1","127.0.0.1","TCP","74","46810  >  102 [SYN] Seq=0
Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=3776076062 TSecr=0 WS=128"
"1409","35.098869281","127.0.0.1","127.0.0.1","TCP","74","102  >  46810 [SYN, ACK]
Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=3776076062 TSecr=3776076062
WS=128"
"1410","35.098881077","127.0.0.1","127.0.0.1","TCP","66","46810  >  102 [ACK] Seq=1
Ack=1 Win=65536 Len=0 TSval=3776076062 TSecr=3776076062"
"1411","35.098929987","127.0.0.1","127.0.0.1","MMS","263","initiate-RequestPDU "
"1412","35.098934721","127.0.0.1","127.0.0.1","TCP","66","102  >  46810 [ACK] Seq=1
Ack=198 Win=65408 Len=0 TSval=3776076062 TSecr=3776076062"
"1413","35.099298315","127.0.0.1","127.0.0.1","TCP","66","102  >  46810 [FIN, ACK]
Seq=1 Ack=198 Win=65408 Len=0 TSval=3776076063 TSecr=3776076062"
"1414","35.099328069","127.0.0.1","127.0.0.1","TCP","66","102  >  46810 [RST, ACK]
Seq=2 Ack=198 Win=65536 Len=0 TSval=3776076063 TSecr=3776076062"
```

Mitigation

The `presentation-context-definition list` case could check for failure similar to how the `user data` parameter (0x61) checks the `bufPos` value before allowing execution to continue. A diff including this suggested update is shown below:

```
user@machine:~$ git diff
diff --git a/src/mms/iso_presentation/iso_presentation.c
b/src/mms/iso_presentation/iso_presentation.c
index 96fd8a3..38cb9c1 100644
--- a/src/mms/iso_presentation/iso_presentation.c
+++ b/src/mms/iso_presentation/iso_presentation.c
@@ -469,6 +469,11 @@ parseNormalModeParameters(IsoPresentation* self, uint8_t*
buffer, int totalLengt
             if (DEBUG_PRES)
                 printf("PRES: pcd list\n");
             bufPos = parsePresentationContextDefinitionList(self, buffer, len,
bufPos);
+
+            if (bufPos < 0){
+                return -1;
+            }
+
             break;

         case 0xa5: /* context-definition-result-list */
user@machine:~$
```

Timeline

2022-02-28 - Public Release

CREDIT

Discovered by Jared Rittle of Cisco Talos.
