nasa / fprime Public

Code Issues 402 Pull requests 22 Discussions Actions Projects

Commit cacdd55

LeStarch authored 3 weeks ago · 39 / 39 · Verified

```
Omega (#4919)

* Fix assert on invalid type

* Fixing assert when filenames overflow

* Fix assert with too-short space packet

* Fix overflows in F Prime frame detector

* Fix invalid size handling in deframers

* Fix overflow in FileUplink

* Formatting

* Minor fixes

* Fixing erroneous test

* Fix compiler warning
```

devel (#4919) · v4.2.1 v4.2.0

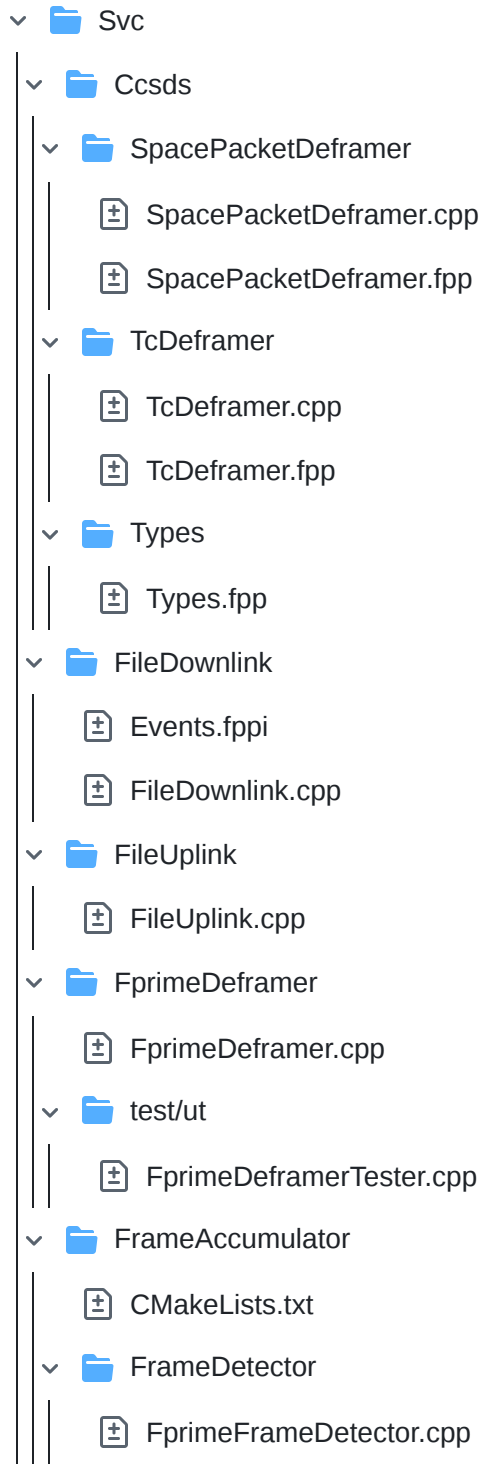
1 parent cd3d8a0 commit cacdd55

14 files changed +119 -42 lines changed

Top

Filter files…

- Fw
  - FilePacket
    - FilePacket.cpp
  - Types
    - Serializable.hpp



14 files changed +119 -42 lines changed

Search within code

Fw/FilePacket/FilePacket.cpp

```
@@ -132,7 +132,7 @@ SerializeStatus FilePacket
::fromSerialBuffer(SerialBuffer& serialBuffer) {
132  132              status = FW_DESERIALIZE_TYPE_MISMATCH;
133  133              break;
134  134          default:
```

```
135     -               FW_ASSERT(0, status);
        135 +           status = FW_DESERIALIZE_INVALID_DATA;
136     136             break;
137     137         }
138     138         return status;
```

Fw/Types/Serializable.hpp

```
        @@ -20,6 +20,7 @@ typedef enum {
20      20          FW_DESERIALIZE_SIZE_MISMATCH,     //!< Data was left in the buffer, but not enough to deserialize
21      21          FW_DESERIALIZE_TYPE_MISMATCH,     //!< Deserialized type ID didn't match
22      22          FW_DESERIALIZE_IMMUTABLE,         //!< Attempted to deserialize into an immutable buffer
        23  +       FW_DESERIALIZE_INVALID_DATA,      //!< Data failed validation
23      24          FW_SERIALIZE_DISCARDED_EXISTING,  //!< Serialization succeeded, but deleted old data
24      25      } SerializeStatus;
25      26
```

...SpacePacketDeframer/SpacePacketDeframer.cpp

```
        @@ -40,10 +40,19 @@ void SpacePacketDeframer ::dataIn_handler(FwIndexType portNum, Fw::Buffer& data,
40      40          // 16b - n/a - Packet Data Length
41      41          // ################################
42      42
43      -       FW_ASSERT(data.getSize() > SpacePacketHeader::SERIALIZED_SIZE, static_cast<FwAssertArgType>(data.getSize()));
        43  +       // Check size and drop packets that are too-small
        44  +       if (data.getSize() <= SpacePacketHeader::SERIALIZED_SIZE) {
        45  +           this->log_WARNING_HI_InvalidPacket();
        46  +           if (this->isConnected_errorNotify_OutputPort(0)) {
        47  +               this->errorNotify_out(0, Svc::Ccsds::FrameError::SP_INVALID_PACKET);
        48  +           }
        49  +           this->dataReturnOut_out(0, data, context);  // Drop the packet
        50  +           return;
        51  +       }
44      52
```

```
45  53        SpacePacketHeader header;
46  54        Fw::SerializeStatus status = data.getDeserializer().deserializeTo(header);
    55  +     // Length is valid, so deserialization here should succeed
47  56        FW_ASSERT(status == Fw::FW_SERIALIZE_OK, status);
48  57
49  58        // Space Packet protocol defines the Data Length as number of bytes minus 1
```

...SpacePacketDeframer/SpacePacketDeframer.fpp

```
    @@ -11,6 +11,11 @@ module Ccsds {
11  11            @ Port to notify of a deframing error
12  12            output port errorNotify: Ccsds.ErrorNotify
13  13
    14  +         @ Deframing received a malformed packet
    15  +         event InvalidPacket() \
    16  +             severity warning high \
    17  +             format "Malformed packet received refusing to deframe"
    18  +
14  19            @ Deframing received an invalid frame length
15  20            event InvalidLength(transmitted: U16, actual: FwSizeType) \
16  21                severity warning high \
```

Svc/Ccsds/TcDeframer/TcDeframer.cpp

```
    @@ -50,9 +50,12 @@ void TcDeframer ::dataIn_handler(FwIndexType portNum,
    Fw::Buffer& data, const Co
50  50
51  51        // CCSDS TC Trailer:
52  52        // 16b - Frame Error Control Field (FECF): CRC16
53      -
54      -     FW_ASSERT(data.getSize() > TCHeader::SERIALIZED_SIZE +
    TCTrailer::SERIALIZED_SIZE,
55      -               static_cast<FwAssertArgType>(data.getSize()));
    53  +     if (data.getSize() <= TCHeader::SERIALIZED_SIZE +
    TCTrailer::SERIALIZED_SIZE) {
    54  +         // Incoming buffer is not long enough to contain a valid frame
    (header+trailer)
    55  +         this->log_WARNING_LO_InvalidPacket();
    56  +         this->dataReturnOut_out(0, data, context);
```

```
        57  +         return;
        58  +     }
56      59
57      60          TCHeader header;
58      61          Fw::SerializeStatus status = data.getDeserializer().deserializeTo(header);
```

Svc/Ccsds/TcDeframer/TcDeframer.fpp

```
            @@ -8,6 +8,12 @@ module Ccsds {
8       8           @ Port to notify of a deframing error
9       9           output port errorNotify: Ccsds.ErrorNotify
10      10
        11  +
        12  +       @ Invalid packet received that will be dropped
        13  +       event InvalidPacket() \
        14  +           severity warning low \
        15  +           format "Invalid packet received refusing to deframe"
        16  +
11      17          @ Deframing received an invalid SCID
12      18          event InvalidSpacecraftId(transmitted: U16, configured: U16) \
13      19              severity warning low \
```

Svc/Ccsds/Types/Types.fpp

```
            @@ -3,11 +3,12 @@ module Ccsds {
3       3
4       4       @ Enum representing an error during framing/deframing in the CCSDS protocols
5       5       enum FrameError: U8 {
6           -       SP_INVALID_LENGTH = 0
7           -       TC_INVALID_SCID = 1
8           -       TC_INVALID_LENGTH = 2
9           -       TC_INVALID_VCID = 3
10          -       TC_INVALID_CRC = 4
        6   +       SP_INVALID_PACKET = 0
        7   +       SP_INVALID_LENGTH = 1
        8   +       TC_INVALID_SCID = 2
        9   +       TC_INVALID_LENGTH = 3
        10  +       TC_INVALID_VCID = 4
        11  +       TC_INVALID_CRC = 5
```

```
11  12        }
12  13
13  14        # ----------------------------------------------
```

Svc/FileDownlink/Events.fppi

```
@@ -81,4 +81,16 @@ event DownlinkZeroSizeFile(
81  81                          ) \
82  82      severity warning high \
83  83      id 0x09 \
84      -   format "Downlink of file {} stopped due to zero-size."
        ⊖
      84  +   format "Downlink of file {} stopped due to zero-size."
      85  +
      86  + @ Supplied filename has overflowed. This intentionally discards the filename to
            avoid cascading overflows.
      87  + event FilenameSourceOverflow() \
      88  +   severity warning high \
      89  +   id 0x10 \
      90  +   format "Commanded source filename too long"
      91  +
      92  + @ Supplied filename has overflowed
      93  + event FilenameDestinationOverflow() \
      94  +   severity warning high \
      95  +   id 0x11 \
      96  +   format "Commanded destination filename too long"
        ⊖
```

Svc/FileDownlink/FileDownlink.cpp

```
@@ -114,18 +114,26 @@ Svc::SendFileResponse FileDownlink ::SendFile_handler(
114 114         entry.cmdSeq = 0;
115 115         entry.context = m_cntxId++;
116 116
117     -       FW_ASSERT(sourceFilename.length() < entry.srcFilename.getCapacity());
118     -       FW_ASSERT(destFilename.length() < entry.destFilename.getCapacity());
119     -       entry.srcFilename = sourceFilename;
120     -       entry.destFilename = destFilename;
      117 +       // Guard against filename overflow
      118 +       if (sourceFilename.length() >= entry.srcFilename.getCapacity()) {
```

```
119 +         this->log_WARNING_HI_FilenameSourceOverflow();
120 +         return SendFileResponse(SendFileStatus::STATUS_ERROR,
    std::numeric_limits<U32>::max());
121 +     } else if (destFilename.length() >= entry.destFilename.getCapacity()) {
122 +         this->log_WARNING_HI_FilenameDestinationOverflow();
123 +         return SendFileResponse(SendFileStatus::STATUS_ERROR,
    std::numeric_limits<U32>::max());
124 +     } else {
125 +         entry.srcFilename = sourceFilename;
126 +         entry.destFilename = destFilename;
121 127
122     -     Os::Queue::Status status = m_fileQueue.send(reinterpret_cast<U8*>(&entry),
    static_cast<FwSizeType>(sizeof(entry)),
123     -                                                 0,
    Os::Queue::BlockingType::NONBLOCKING);
    128 +         Os::Queue::Status status =
    129 +             m_fileQueue.send(reinterpret_cast<U8*>(&entry),
    static_cast<FwSizeType>(sizeof(entry)), 0,
    130 +                              Os::Queue::BlockingType::NONBLOCKING);
124 131
125     -     if (status != Os::Queue::Status::OP_OK) {
126     -         return SendFileResponse(SendFileStatus::STATUS_ERROR,
    std::numeric_limits<U32>::max());
    132 +         if (status != Os::Queue::Status::OP_OK) {
    133 +             return SendFileResponse(SendFileStatus::STATUS_ERROR,
    std::numeric_limits<U32>::max());
    134 +         }
    135 +         return SendFileResponse(SendFileStatus::STATUS_OK, entry.context);
127 136         }
128     -     return SendFileResponse(SendFileStatus::STATUS_OK, entry.context);
129 137     }
130 138
131 139     void FileDownlink ::pingIn_handler(const FwIndexType portNum, U32 key) {
        @@ -170,16 +178,24 @@ void FileDownlink ::SendFile_cmdHandler(const
        FwOpcodeType opCode,
170 178         entry.cmdSeq = cmdSeq;
171 179         entry.context = std::numeric_limits<U32>::max();
172 180
173     -     FW_ASSERT(sourceFilename.length() < entry.srcFilename.getCapacity());
174     -     FW_ASSERT(destFilename.length() < entry.destFilename.getCapacity());
```

```
175     -     entry.srcFilename = sourceFilename;
176     -     entry.destFilename = destFilename;
        181 +     // Guard against filename overflow
        182 +     if (sourceFilename.length() >= entry.srcFilename.getCapacity()) {
        183 +         this->log_WARNING_HI_FilenameSourceOverflow();
        184 +         this->cmdResponse_out(opCode, cmdSeq,
                    Fw::CmdResponse::VALIDATION_ERROR);
        185 +     } else if (destFilename.length() >= entry.destFilename.getCapacity()) {
        186 +         this->log_WARNING_HI_FilenameDestinationOverflow();
        187 +         this->cmdResponse_out(opCode, cmdSeq,
                    Fw::CmdResponse::VALIDATION_ERROR);
        188 +     } else {
        189 +         entry.srcFilename = sourceFilename;
        190 +         entry.destFilename = destFilename;
177     191
178     -     Os::Queue::Status status = m_fileQueue.send(reinterpret_cast<U8*>(&entry),
                static_cast<FwSizeType>(sizeof(entry)),
179     -                                                 0,
                Os::Queue::BlockingType::NONBLOCKING);
        192 +         Os::Queue::Status status =
        193 +             m_fileQueue.send(reinterpret_cast<U8*>(&entry),
                    static_cast<FwSizeType>(sizeof(entry)), 0,
        194 +                              Os::Queue::BlockingType::NONBLOCKING);
180     195
181     -     if (status != Os::Queue::Status::OP_OK) {
182     -         this->cmdResponse_out(opCode, cmdSeq,
                    Fw::CmdResponse::EXECUTION_ERROR);
        196 +         if (status != Os::Queue::Status::OP_OK) {
        197 +             this->cmdResponse_out(opCode, cmdSeq,
                    Fw::CmdResponse::EXECUTION_ERROR);
        198 +         }
183     199     }
184     200 }
185     201
            @@ -197,16 +213,24 @@ void FileDownlink
            ::SendPartial_cmdHandler(FwOpcodeType opCode,
197     213     entry.cmdSeq = cmdSeq;
198     214     entry.context = std::numeric_limits<U32>::max();
199     215
200     -     FW_ASSERT(sourceFilename.length() < entry.srcFilename.getCapacity());
```

```
201       -     FW_ASSERT(destFilename.length() < entry.destFilename.getCapacity());
202       -     entry.srcFilename = sourceFilename;
203       -     entry.destFilename = destFilename;
      216 +     // Guard against filename overflow
      217 +     if (sourceFilename.length() >= entry.srcFilename.getCapacity()) {
      218 +         this->log_WARNING_HI_FilenameSourceOverflow();
      219 +         this->cmdResponse_out(opCode, cmdSeq,
          Fw::CmdResponse::VALIDATION_ERROR);
      220 +     } else if (destFilename.length() >= entry.destFilename.getCapacity()) {
      221 +         this->log_WARNING_HI_FilenameDestinationOverflow();
      222 +         this->cmdResponse_out(opCode, cmdSeq,
          Fw::CmdResponse::VALIDATION_ERROR);
      223 +     } else {
      224 +         entry.srcFilename = sourceFilename;
      225 +         entry.destFilename = destFilename;
204   226
205       -     Os::Queue::Status status = m_fileQueue.send(reinterpret_cast<U8*>(&entry),
          static_cast<FwSizeType>(sizeof(entry)),
206       -                                                 0,
          Os::Queue::BlockingType::NONBLOCKING);
      227 +         Os::Queue::Status status =
      228 +             m_fileQueue.send(reinterpret_cast<U8*>(&entry),
          static_cast<FwSizeType>(sizeof(entry)), 0,
      229 +                              Os::Queue::BlockingType::NONBLOCKING);
207   230
208       -     if (status != Os::Queue::Status::OP_OK) {
209       -         this->cmdResponse_out(opCode, cmdSeq,
          Fw::CmdResponse::EXECUTION_ERROR);
      231 +         if (status != Os::Queue::Status::OP_OK) {
      232 +             this->cmdResponse_out(opCode, cmdSeq,
          Fw::CmdResponse::EXECUTION_ERROR);
      233 +         }
210   234       }
211   235   }
212   236
```

Svc/FileUplink/FileUplink.cpp

```
@@ -132,7 +132,7 @@ void FileUplink::handleDataPacket(const
Fw::FilePacket::DataPacket& dataPacket)
```

```
132 132         this->checkSequenceIndex(sequenceIndex);
133 133         const U32 byteOffset = dataPacket.getByteOffset();
134 134         const U32 dataSize = dataPacket.getDataSize();
135     -       if (byteOffset + dataSize > this->m_file.size) {
    135 +       if ((std::numeric_limits<U32>::max() - byteOffset < dataSize) ||
                (byteOffset + dataSize > this->m_file.size)) {
136 136             this->m_warnings.packetOutOfBounds(sequenceIndex, this->m_file.name);
137 137             return;
138 138         }
```

Svc/FprimeDeframer/FprimeDeframer.cpp

```
@@ -52,7 +52,8 @@ void FprimeDeframer ::dataIn_handler(FwIndexType portNum,
Fw::Buffer& data, cons
52 52         // We expect the frame size to be size of header + body (of size specified
in header) + trailer
53 53         const FwSizeType expectedFrameSize =
FprimeProtocol::FrameHeader::SERIALIZED_SIZE + header.get_lengthField() +
54 54
FprimeProtocol::FrameTrailer::SERIALIZED_SIZE;
55     -       if (data.getSize() < expectedFrameSize) {
    55 +       // Reject packets whose data does not match the header
    56 +       if (data.getSize() != expectedFrameSize) {
56 57             this->log_WARNING_HI_InvalidLengthReceived();
57 58             this->dataReturnOut_out(0, data, context);  // drop the frame
58 59             return;
```

...meDeframer/test/ut/FprimeDeframerTester.cpp

```
@@ -94,8 +94,8 @@ void FprimeDeframerTester ::testIncorrectStartWord() {
94 94     }
95 95
96 96     void FprimeDeframerTester ::testIncorrectCrc() {
97     -       // Frame:     |   F´ start word      |      Length = 1       |Data
(2bytes)| INCORRECT Checksum  |
98     -       U8 data[14] = {0xDE, 0xAD, 0xBE, 0xEF, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00};
    97 +       // Frame:     |   F´ start word      |      Length = 2       |Data
(2bytes)| INCORRECT Checksum  |
```

```
      98 +      U8 data[14] = {0xDE, 0xAD, 0xBE, 0xEF, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00,
           0x00, 0x00, 0x00, 0x00};
 99   99          this->mockReceiveData(data, sizeof(data));
100  100          ASSERT_from_dataOut_SIZE(0);        // nothing emitted on dataOut
101  101          ASSERT_from_dataReturnOut_SIZE(1);  // invalid buffer was deallocated
```

Svc/FrameAccumulator/CMakeLists.txt

```
           @@ -19,6 +19,7 @@ register_fprime_library(
19   19          Svc_FprimeProtocol
20   20          Svc_Ccsds_Types
21   21      )
      22 + target_compile_options(${FPRIME_CURRENT_MODULE} PRIVATE -Wno-tautological-
           constant-out-of-range-compare -Wno-type-limits)
22   23
23   24      #### UTs ####
24   25      register_fprime_ut(
```

...lator/FrameDetector/FprimeFrameDetector.cpp

```
           @@ -49,11 +49,25 @@ FrameDetector::Status FprimeFrameDetector::detect(const
           Types::CircularBuffer& d
49   49          if (header.get_startWord() != default_value.get_startWord()) {
50   50              return Status::NO_FRAME_DETECTED;
51   51          }
      52 +     // Validate size before proceeding
      53 +     const FwSizeType max_payload_size = std::numeric_limits<FwSizeType>::max() -
      54 +
           FprimeProtocol::FrameHeader::SERIALIZED_SIZE -
      55 +
           FprimeProtocol::FrameTrailer::SERIALIZED_SIZE;
      56 +     // If the header length is larger than size can store, then frame is invalid
      57 +     if (max_payload_size < header.get_lengthField()) {
      58 +         // Size overflow - frame is invalid
      59 +         return Status::NO_FRAME_DETECTED;
      60 +     }
      61 +
52   62          // We expect the frame size to be size of header + body (of size specified
           in header) + trailer
```

```
53  63          const FwSizeType expected_frame_size =
                FprimeProtocol::FrameHeader::SERIALIZED_SIZE + header.get_lengthField() +
54  64
                FprimeProtocol::FrameTrailer::SERIALIZED_SIZE;
55      -       // If the current allocated size can't hold the expected_frame_size ->
                MORE_DATA_NEEDED
56      -       if (data.get_allocated_size() < expected_frame_size) {
        65  +       // If the frame will never fit, then report NO_FRAME_DETECTED to drop the
                    erroneous frame
        66  +       if (data.get_capacity() < expected_frame_size) {
        67  +           return Status::NO_FRAME_DETECTED;
        68  +       }
        69  +       // If the frame could fit but we haven't received enough data yet, report
                    MORE_DATA_NEEDED
        70  +       else if (data.get_allocated_size() < expected_frame_size) {
57  71              size_out = expected_frame_size;
58  72              return Status::MORE_DATA_NEEDED;
59  73          }
```

Comments 0

Please sign in to comment.