aligungr / **UERANSIM** Public

Code | Issues 237 | Pull requests 6 | Discussions | Actions | Projects

Commit `ca1a66f`

aligungr committed 2 weeks ago

```
Security hardening for Radio Link Simulation (RLS) layer.
```

`master` · v3.2.8

1 parent `b4157fa` commit `ca1a66f`

9 files changed +111 -27

Top

Filter files…

- README.md
- src
  - gnb/rls
    - ctl_task.cpp
    - udp_task.cpp
  - lib/rls
    - rls_pdu.cpp
  - ue/rls
    - ctl_task.cpp
    - udp_task.cpp
  - utils
    - constants.hpp
    - octet_view.cpp
    - octet_view.hpp

Search within code

README.md

```
@@ -2,7 +2,7 @@
  <a href="https://github.com/aligungr/UERANSIM"><img src="/.github/logo.png" width="75" title="UERANSIM"></a>
</p>
<p align="center">
- <img src="https://img.shields.io/badge/UERANSIM-v3.2.7-blue" />
+ <img src="https://img.shields.io/badge/UERANSIM-v3.2.8-blue" />
<img src="https://img.shields.io/badge/3GPP-R15-orange" />
<img src="https://img.shields.io/badge/License-AGPL--3.0-green"/>
</p>
@@ -13,9 +13,6 @@ testing 5G Core Network and studying 5G System.

UERANSIM introduces the world's first open source 5G-SA UE and gNodeB implementation.

- > [!IMPORTANT]
- > UERANSIM is no longer actively developed by the maintainer. However community contributions are welcomed.
-
## Current Status

Basic functionalities of UE and gNodeB are fully functional and ready to use. However some of the features are not complete.
@@ -33,8 +30,7 @@ On the other hand, UERANSIM does not fully provide physical layer. 5G-NR radio i

You can find the documentation on [UERANSIM Wiki](https://github.com/aligungr/UERANSIM/wiki).

- And, since the project is rapidly developing, please make sure that you have always
- the [latest](https://github.com/aligungr/UERANSIM/releases) UERANSIM.
+ And, please make sure that you have always the [latest](https://github.com/aligungr/UERANSIM/releases) UERANSIM.

## Contributing
```

```
@@ -50,7 +46,7 @@ You can support UERANSIM by:

## License

- Copyright (c) 2025 ALİ GÜNGÖR.
+ Copyright (c) 2026 ALİ GÜNGÖR.

All source code and related files including documentation and wiki pages are
dual licensed with [AGPL-3.0](https://www.gnu.org/licenses/agpl-3.0.en.html) and
a commercial license.
```

src/gnb/rls/ctl_task.cpp

```
@@ -123,7 +123,7 @@ void RlsControlTask::handleRlsMessage(int ueId,
rls::RlsMessage &msg)
        else if (msg.msgType == rls::EMessageType::PDU_TRANSMISSION)
        {
            auto &m = (rls::RlsPduTransmission &)msg;
-           if (m.pduId != 0)
+           if (m.pduId != 0 && m_pendingAck[ueId].size() < MAX_PDU_COUNT)
                m_pendingAck[ueId].push_back(m.pduId);

            if (m.pduType == rls::EPduType::DATA)
@@ -136,6 +136,12 @@ void RlsControlTask::handleRlsMessage(int ueId,
rls::RlsMessage &msg)
            }
            else if (m.pduType == rls::EPduType::RRC)
            {
+               if (m.payload > static_cast<uint32_t>(rrc::RrcChannel::UL_DCCH))
+               {
+                   m_logger->err("Invalid RRC channel value: %d", m.payload);
+                   return;
+               }
+
                auto w = std::make_unique<NmGnbRlsToRls>
(NmGnbRlsToRls::UPLINK_RRC);
                w->ueId = ueId;
                w->rrcChannel = static_cast<rrc::RrcChannel>(m.payload);
```

```
@@ -240,12 +246,12 @@ void RlsControlTask::onAckControlTimerExpired()
240 246
241 247   void RlsControlTask::onAckSendTimerExpired()
242 248   {
243     -     auto copy = m_pendingAck;
    249 +     auto copy = std::move(m_pendingAck);
244 250       m_pendingAck.clear();
245 251
246 252       for (auto &item : copy)
247 253       {
248     -         if (!item.second.empty())
    254 +         if (item.second.empty())
249 255               continue;
250 256
251 257           rls::RlsPduTransmissionAck msg{m_sti};
```

src/gnb/rls/udp_task.cpp

```
@@ -26,13 +26,16 @@ static constexpr const int HEARTBEAT_THRESHOLD = 2000;
// (LOOP_PERIOD + RECEIVE
26 26
27 27   static constexpr const int MIN_ALLOWED_DBM = -120;
28 28
    29 + static constexpr const size_t MAX_UE_COUNT = 1024;
    30 +
29 31   static int EstimateSimulatedDbm(const Vector3 &myPos, const Vector3 &uePos)
30 32   {
31     -     int deltaX = myPos.x - uePos.x;
32     -     int deltaY = myPos.y - uePos.y;
33     -     int deltaZ = myPos.z - uePos.z;
    33 +     double deltaX = static_cast<double>(myPos.x) - static_cast<double>(uePos.x);
    34 +     double deltaY = static_cast<double>(myPos.y) - static_cast<double>(uePos.y);
    35 +     double deltaZ = static_cast<double>(myPos.z) - static_cast<double>(uePos.z);
34 36
35     -     int distance = static_cast<int>(std::sqrt(deltaX * deltaX + deltaY * deltaY + deltaZ * deltaZ));
    37 +     double distSq = deltaX * deltaX + deltaY * deltaY + deltaZ * deltaZ;
```

```
      38 +     int distance = static_cast<int>(std::sqrt(distSq));
36    39       if (distance == 0)
37    40           return -1; // 0 may be confusing for people
38    41       return -distance;
         @@ -109,9 +112,16 @@ void RlsUdpTask::receiveRlsPdu(const InetAddress &addr,
         std::unique_ptr<rls::Rls
109   112          }
110   113          else
111   114          {
      115 +            if (m_ueMap.size() >= MAX_UE_COUNT)
      116 +            {
      117 +                m_logger->warn("Max UE count reached, rejecting new UE");
      118 +                return;
      119 +            }
      120 +
112   121              int ueId = ++m_newIdCounter;
113   122
114   123              m_stiToUe[msg->sti] = ueId;
      124 +            m_ueMap[ueId].sti = msg->sti;
115   125              m_ueMap[ueId].address = addr;
116   126              m_ueMap[ueId].lastSeen = utils::CurrentTimeMillis();
117   127
```

src/lib/rls/rls_pdu.cpp

```
         @@ -54,6 +54,10 @@ void EncodeRlsMessage(const RlsMessage &msg, OctetString
         &stream)
54    54
55    55   std::unique_ptr<RlsMessage> DecodeRlsMessage(const OctetView &stream)
56    56   {
      57 +     // Minimum header: 1 (compat) + 3 (version) + 1 (msgType) + 8 (sti) = 13
         bytes
      58 +     if (stream.remainingSize() < 13)
      59 +         return nullptr;
      60 +
57    61       auto first = stream.readI(); // (Just for old RLS compatibility)
58    62       if (first != 3)
59    63           return nullptr;
         @@ -66,10 +70,19 @@ std::unique_ptr<RlsMessage> DecodeRlsMessage(const
         OctetView &stream)
```

```
66  70              return nullptr;
67  71
68  72          auto msgType = static_cast<EMessageType>(stream.readI());
    73  +
    74  +       // Validate message type before reading further
    75  +       if (msgType != EMessageType::HEARTBEAT && msgType !=
                EMessageType::HEARTBEAT_ACK &&
    76  +           msgType != EMessageType::PDU_TRANSMISSION && msgType !=
                EMessageType::PDU_TRANSMISSION_ACK)
    77  +           return nullptr;
    78  +
69  79          uint64_t sti = stream.read8UL();
70  80
71  81          if (msgType == EMessageType::HEARTBEAT)
72  82          {
    83  +           // Need 12 bytes (3 x int32)
    84  +           if (stream.remainingSize() < 12)
    85  +               return nullptr;
73  86              auto res = std::make_unique<RlsHeartBeat>(sti);
74  87              res->simPos.x = stream.read4I();
75  88              res->simPos.y = stream.read4I();
    @@ -78,28 +91,54 @@ std::unique_ptr<RlsMessage> DecodeRlsMessage(const
    OctetView &stream)
78  91          }
79  92          else if (msgType == EMessageType::HEARTBEAT_ACK)
80  93          {
    94  +           // Need 4 bytes (int32)
    95  +           if (stream.remainingSize() < 4)
    96  +               return nullptr;
81  97              auto res = std::make_unique<RlsHeartBeatAck>(sti);
82  98              res->dbm = stream.read4I();
83  99              return res;
84 100          }
85 101          else if (msgType == EMessageType::PDU_TRANSMISSION)
86 102          {
   103  +           // Need 1 (pduType) + 4 (pduId) + 4 (payload) + 4 (pduLength) = 13
                bytes minimum
   104  +           if (stream.remainingSize() < 13)
   105  +               return nullptr;
   106  +
```

```
 87   107            auto res = std::make_unique<RlsPduTransmission>(sti);
 88        -         res->pduType = static_cast<EPduType>((uint8_t)stream.read());
      108   +         auto rawPduType = static_cast<uint8_t>(stream.read());
      109   +         if (rawPduType != static_cast<uint8_t>(EPduType::RRC) &&
      110   +             rawPduType != static_cast<uint8_t>(EPduType::DATA))
      111   +             return nullptr;
      112   +         res->pduType = static_cast<EPduType>(rawPduType);
      113   +
 89   114            res->pduId = stream.read4UI();
 90   115            res->payload = stream.read4UI();
 91   116
 92        -         int pduLength = stream.read4I();
      117   +         uint32_t pduLength = stream.read4UI();
 93   118            if (pduLength > 16384)
 94   119                return nullptr;
 95        -
 96        -         res->pdu = stream.readOctetString(pduLength);
      120   +         if (stream.remainingSize() < pduLength)
      121   +             return nullptr;
      122   +
      123   +         res->pdu = stream.readOctetString(static_cast<int>(pduLength));
 97   124            return res;
 98   125        }
 99   126        else if (msgType == EMessageType::PDU_TRANSMISSION_ACK)
100   127        {
      128   +         static constexpr uint32_t MAX_ACK_COUNT = 4096;
      129   +
      130   +         // Need 4 bytes for count
      131   +         if (stream.remainingSize() < 4)
      132   +             return nullptr;
      133   +
101   134            auto res = std::make_unique<RlsPduTransmissionAck>(sti);
102   135            auto count = stream.read4UI();
      136   +         if (count > MAX_ACK_COUNT)
      137   +             return nullptr;
      138   +         // Need count * 4 bytes for pduIds
      139   +         if (stream.remainingSize() < static_cast<size_t>(count) * 4)
      140   +             return nullptr;
      141   +
103   142            res->pduIds.reserve(count);
```

```
104  143            for (uint32_t i = 0; i < count; i++)
105  144                res->pduIds.push_back(stream.read4UI());
```

src/ue/rls/ctl_task.cpp

```
@@ -108,7 +108,7 @@ void RlsControlTask::handleRlsMessage(int cellId,
rls::RlsMessage &msg)
108  108        else if (msg.msgType == rls::EMessageType::PDU_TRANSMISSION)
109  109        {
110  110            auto &m = (rls::RlsPduTransmission &)msg;
111       -          if (m.pduId != 0)
     111  +          if (m.pduId != 0 && m_pendingAck[cellId].size() < MAX_PDU_COUNT)
112  112                m_pendingAck[cellId].push_back(m.pduId);
113  113
114  114            if (m.pduType == rls::EPduType::DATA)
@@ -127,6 +127,12 @@ void RlsControlTask::handleRlsMessage(int cellId,
rls::RlsMessage &msg)
127  127            }
128  128            else if (m.pduType == rls::EPduType::RRC)
129  129            {
     130  +             if (m.payload > static_cast<uint32_t>(rrc::RrcChannel::UL_DCCH))
     131  +             {
     132  +                 m_logger->err("Invalid RRC channel value: %d", m.payload);
     133  +                 return;
     134  +             }
     135  +
130  136              auto w = std::make_unique<NmUeRlsToRls>
(NmUeRlsToRls::DOWNLINK_RRC);
131  137              w->cellId = cellId;
132  138              w->rrcChannel = static_cast<rrc::RrcChannel>(m.payload);
@@ -233,12 +239,12 @@ void RlsControlTask::onAckControlTimerExpired()
233  239
234  240  void RlsControlTask::onAckSendTimerExpired()
235  241  {
236       -    auto copy = m_pendingAck;
     242  +    auto copy = std::move(m_pendingAck);
237  243      m_pendingAck.clear();
238  244
239  245      for (auto &item : copy)
```

```
240  246         {
241       -          if (!item.second.empty())
     247  +          if (item.second.empty())
242  248               continue;
243  249
244  250           rls::RlsPduTransmissionAck msg{m_shCtx->sti};
```

src/ue/rls/udp_task.cpp

```
     @@ -21,6 +21,8 @@ static constexpr const int LOOP_PERIOD = 1000;
21   21    static constexpr const int RECEIVE_TIMEOUT = 200;
22   22    static constexpr const int HEARTBEAT_THRESHOLD = 2000; // (LOOP_PERIOD +
           RECEIVE_TIMEOUT)'dan büyük olmalı
23   23
     24  + static constexpr const size_t MAX_CELL_COUNT = 256;
     25  +
24   26    namespace nr::ue
25   27    {
26   28
     @@ -93,6 +95,12 @@ void RlsUdpTask::receiveRlsPdu(const InetAddress &addr,
           std::unique_ptr<rls::Rls
93   95        {
94   96            if (!m_cells.count(msg->sti))
95   97            {
     98  +             if (m_cells.size() >= MAX_CELL_COUNT)
     99  +             {
     100 +                 m_logger->warn("Max cell count reached, ignoring new cell");
     101 +                 return;
     102 +             }
     103 +
96   104               m_cells[msg->sti].cellId = ++m_cellIdCounter;
97   105               m_cellIdToSti[m_cells[msg->sti].cellId] = msg->sti;
98   106           }
```

src/utils/constants.hpp

```
     @@ -15,10 +15,10 @@ struct cons
15   15        // Version information
16   16        static constexpr const uint8_t Major = 3;
```

```
17  17      static constexpr const uint8_t Minor = 2;
18          - static constexpr const uint8_t Patch = 7;
        18  + static constexpr const uint8_t Patch = 8;
19  19      static constexpr const char *Project = "UERANSIM";
20          - static constexpr const char *Tag = "v3.2.7";
21          - static constexpr const char *Name = "UERANSIM v3.2.7";
        20  + static constexpr const char *Tag = "v3.2.8";
        21  + static constexpr const char *Name = "UERANSIM v3.2.8";
22  22      static constexpr const char *Owner = "ALİ GÜNGÖR";
23  23
24  24      // Some port values
```

src/utils/octet_view.cpp

```
    @@ -21,10 +21,12 @@ OctetView::OctetView(const uint8_t *data, size_t size) :
    data(data), index(0), s
21  21
22  22  OctetString OctetView::readOctetString(int length) const
23  23  {
        24  +     if (length < 0)
        25  +         throw std::out_of_range("readOctetString: negative length");
24  26      if (length == 0)
25  27          return {};
26          - else if (index + length > size)
27          -     throw std::out_of_range("Invalid arguments for readOctetString");
        28  + if (index + static_cast<size_t>(length) > size)
        29  +     throw std::out_of_range("readOctetString: out of bounds");
28  30
29  31      std::vector<uint8_t> v{data + index, data + index + length};
30  32      index += length;
    @@ -33,6 +35,8 @@ OctetString OctetView::readOctetString(int length) const
33  35
34  36  OctetString OctetView::readOctetString(size_t length) const
35  37  {
        38  +     if (length > static_cast<size_t>(INT32_MAX))
        39  +         throw std::out_of_range("readOctetString: length exceeds INT32_MAX");
36  40      return readOctetString(static_cast<int>(length));
37  41  }
38  42
    @@ -43,12 +47,16 @@ OctetString OctetView::readOctetString() const
```

```
43  47
44  48    std::string OctetView::readUtf8String(int length) const
45  49    {
    50  +     if (length < 0 || index + static_cast<size_t>(length) > size)
    51  +         throw std::out_of_range("readUtf8String: out of bounds");
46  52        auto res = std::string(data + index, data + index + length);
47  53        index += length;
48  54        return res;
49  55    }
50  56
51  57    std::string OctetView::readUtf8String(size_t length) const
52  58    {
    59  +     if (length > static_cast<size_t>(INT32_MAX))
    60  +         throw std::out_of_range("readUtf8String: length exceeds INT32_MAX");
53  61        return readUtf8String(static_cast<int>(length));
54  62    }
```

src/utils/octet_view.hpp

```
        @@ -13,11 +13,11 @@
13  13
14  14    #include <cstdint>
15  15    #include <cstdlib>
    16  + #include <stdexcept>
16  17    #include <utility>
17  18
18  19    class OctetString;
19  20
20      - // TODO: add bound check
21  21    class OctetView
22  22    {
23  23        const uint8_t *data;
        @@ -30,11 +30,15 @@ class OctetView
30  30
31  31        inline octet peek() const
32  32        {
    33  +         if (index >= size)
    34  +             throw std::out_of_range("OctetView::peek out of bounds");
33  35            return data[index];
34  36        }
35  37
```

```
36  38          inline octet peek(int offset) const
37  39          {
    40  +           if (offset < 0 || index + static_cast<size_t>(offset) >= size)
    41  +               throw std::out_of_range("OctetView::peek out of bounds");
38  42              return data[index + offset];
39  43          }
40  44
    @@ -50,6 +54,8 @@ class OctetView
50  54
51  55          inline octet read() const
52  56          {
    57  +           if (index >= size)
    58  +               throw std::out_of_range("OctetView::read out of bounds");
53  59              return data[index++];
54  60          }
55  61
    @@ -123,6 +129,11 @@ class OctetView
123 129             return index < size;
124 130         }
125 131
    132 +     inline size_t remainingSize() const
    133 +     {
    134 +         return size - index;
    135 +     }
    136 +
126 137         OctetString readOctetString(int length) const;
127 138         OctetString readOctetString(size_t length) const;
128 139         OctetString readOctetString() const;
```

Comments 0

Please sign in to comment.