Cellular Security

Florida Institute for Cybersecurity Research

Image created with DALLE-3

We discover **119 vulnerabilities** in LTE/5G core infrastructure, each of which can result in **persistent denial of cell service to an entire metropolitan area or city** and some of which can be used to **remotely compromise and access the cellular core**. Our research covers seven LTE implementations (Open5GS, Magma, OpenAirInterface, Athonet, SD-Core, NextEPC, srsRAN) and three 5G implementations (Open5GS, Magma, OpenAirInterface); we find **vulnerabilities in every single LTE/5G implementation tested**.

Our research finds these vulnerabilities are present in both well-maintained open-source LTE/5G cores and in proprietary software, both of which have active deployments in commercial settings. To learn more about how we were able to discover these vulnerabilities, take a look at our paper.

Impact

Cellular networks are considered critical infrastructure both for day-to-day communication and emergency services, to the extend that their availability and reliability is often highly regulated by government agencies. As such, denying service to regular and emergency cellular services is a

Service Distruption

Every one of the >100 vulnerabilities discussed below can be used to persistently distrupt all cellular communications (phone calls, messaging and data) at a city-wide level. An attacker can continuously crash the Mobility Management Entity (MME) or Access and Mobility Management Function (AMF) in an LTE/5G network, respectively, simply by sending a single small data packet over the network as an unauthenticated user (no SIM card required). This distruption could persist for as long as it would take network operators to identify and patch the vulnerability selected.

Remote Access

A number of vulnerabilities discovered cause buffer overflows or similar memory corruption errors. These could be used by an adversary to gain a foothold into the cellular core network. From there, attackers could:

- Monitor cellphone location and connection information for all subscribers at a city-wide level
- Perform targeted attacks on specific subscribers
- Pivot to attacking within the core network (such as targeting the Home Subscriber Service (HSS) or Unified Data Management (UDM) components in LTE/5G, respectively, to carry out nation-wide cell service disruption)

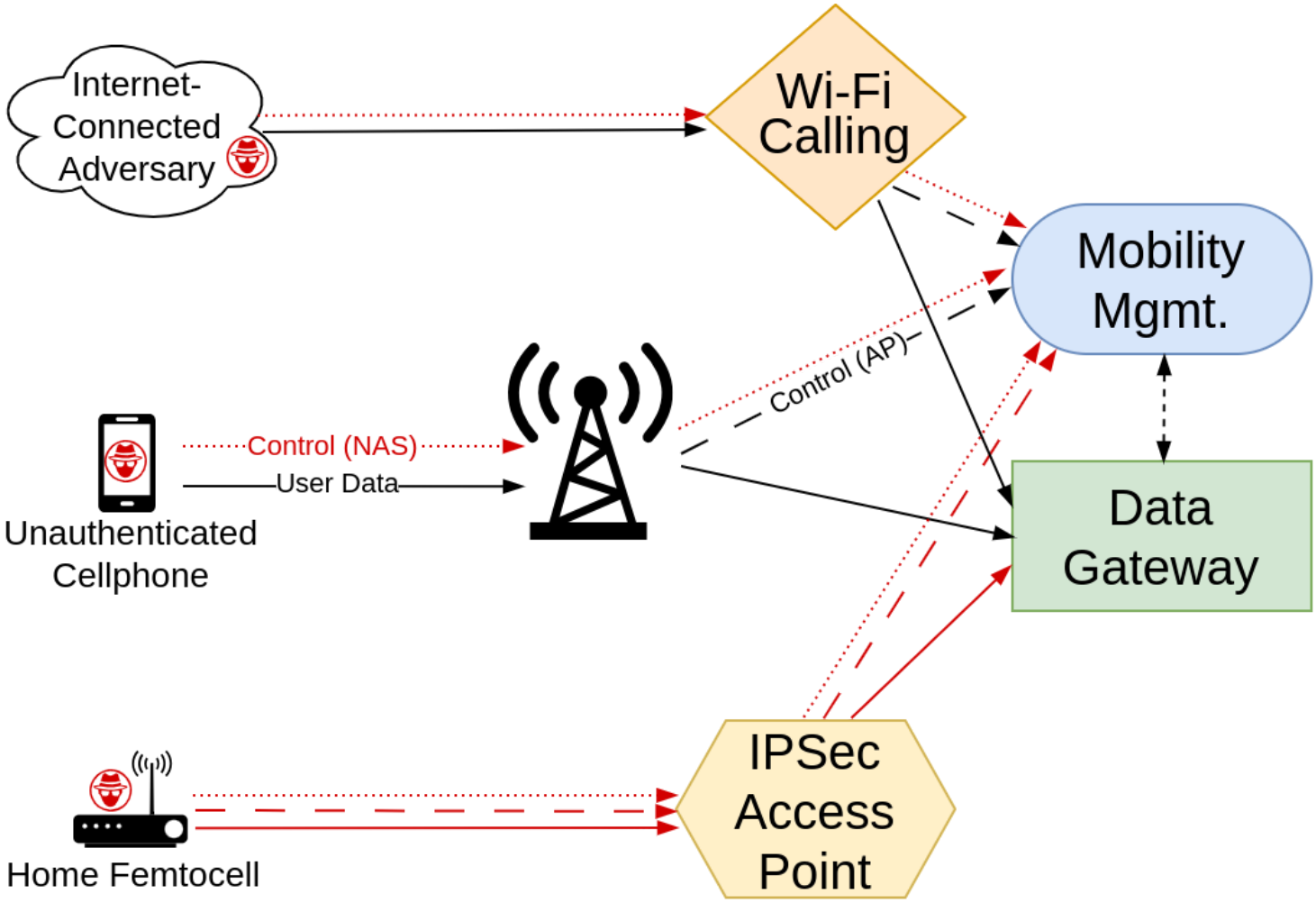
The exploitability of memory corruption vulnerabilities widely varies; in some cases, there may be no practical way of obtaining remote access despite the presence of a memory corruption error because of various limitations. We develop a **proof-of-concept RCE exploit** for one of the vulnerabilities in SD-Core as a way of practically demonstrating the severity of these findings.

Threat Models

All discovered vulnerabilities fall under two threat models:

1. *Vulnerabilities that can be exploited by **any unauthenticated mobile device***. The mobile device doesn't need a valid SIM, it just needs to be capable of sending the right malformed packet sequence at the beginning of a cellular connection (e.g. using an SDR). Traditionally, these attacks were limited in scope to devices that are within radio distance of the LTE/5G core being attacked. However, with the widespread deployment of Wi-Fi Calling services, **these same attacks can be exploited by any entity on the Internet just by sending a few packets**–no SIM card or SDR equipment required.

2. *Vulnerabilities that can be exploited by an adversary who has base-station access to the cellular core*. This includes attackers that have either a) compromised a base station/femtocell, or b) gained access to the IPsec network used by base stations to communicate with the cellular core via a misconfiguration or key leak. While this threat model has more preconditions than the first, it is by no means unrealistic–most cellular providers offer home or office "femtocell" cell signal boosters, which ultimately operate as base stations under the hood. An adversary could easily be able to obtain persistent physical access to one of these devices and dump RAM/flash or carry out attacks specific to the device to gain access to its IPsec keys. The proliferation of smaller 5G base stations in easier-to-reach locations (*not* 100 feet in the air on a tower) also makes compromise of a regular base station more practical.



Disclosure Process

We reached out to the maintainers of each affected cellular core and followed best practice of allowing at least 90 days for internal patching prior to disclosure. For projects where maintainers did not respond (NextEPC, SD-Core), we attempted to reach out via other communication channels. When that was unsuccessful, we opted to disclose and provide patches directly on their Github repositories in coordination with public disclosure.

Vulnerability Analysis

The remaining sections contain more detailed analysis of the causes of each vulnerability. Vulnerabilities are broken down by implementation, cellular generation and protocol the vulnerability was found in. Where possible, a code listing of the precise location and conditions that lead to the vulnerability is provided.

OpenAirInterface (5G)

Affected versions: oai-cn5g-amf <= 2.0.0

NGAP Protocol Vulnerabilities

CVE-2024-24445 (**VULN-B01**):

OpenAirInterface contains a null dereference in its handling of unsupported NGAP protocol messages. When a procedure code/presence field tuple is received that is unsupported, OAI indexes into a null function pointer and subsequently dereferences it.

`src/ngap/ngap_app/ngap_app.cpp`:

```
void ngap_app::handle_receive(
        bstring payload, sctp_assoc_id_t assoc_id, sctp_stream_id_t stream,
        sctp_stream_id_t instreams, sctp_stream_id_t outstreams) {
    // ...

    // Handle the message
    (*messages_callback[ngap_msg_pdu->choice.initiatingMessage->procedureCode]
                       [ngap_msg_pdu->present - 1])(
        assoc_id, stream, ngap_msg_pdu);
    // ^ function pointer dereference of potentially null value

    // ...
}
```

CVE-2024-24450 (**VULN-B02**):

Stack-based memcpy buffer overflow in the ngap_handle_pdu_session_resource_setup_response routine in OpenAirInterface CN5G AMF <= 2.0.0 allows a remote attacker with access to the N2 interface to carry out denial of service against the AMF and potentially execute code by sending a PDU Session Resource Setup Response with a suffciently large FailedToSetupList IE

`src/ngap/ngap_app/ngap_message_callback.hpp`:

```
int ngap_amf_handle_pdu_session_resource_setup_response(
        const sctp_assoc_id_t assoc_id, const sctp_stream_id_t stream,
        struct Ngap_NGAP_PDU* message_p) {
    // ...

    std::vector<PDUSessionResourceFailedToSetupItem_t> list_fail;
    if (!pdu_session_resource_setup_resp->getPduSessionResourceFailedToSetupList(
            list_fail)) {
        Logger::ngap().error(
            "decoding PduSessionResourceSetupResponseMsg "
            "getPduSessionResourceFailedToSetupList IE  error");
    } else {
        PduSessionResourceSetupUnSuccessfulTransferIE* UnSuccessfultransfer =
            new PduSessionResourceSetupUnSuccessfulTransferIE();
        uint8_t buffer[BUFFER_SIZE_512];
        // ^ static buffer of 512 bytes allocated
        memcpy(
            buffer, list_fail[0].pduSessionResourceSetupUnsuccessfulTransfer.buf,
            list_fail[0].pduSessionResourceSetupUnsuccessfulTransfer.size);
        // ^ static buffer copied in data from buffer that could have more than 512 bytes

        // ...
    }

    // ...
}
```

VULN-B03:

The OAI AMF is susceptible to uninitialized memory access when handling `UE Radio Capability Indication` NGAP messages. Specifically, a crafted message containing no Radio Capability field will cause the `ue_radio_cap` octet string to remain uninitialized, leading to an out-of-bounds read when `blk2bstr` is called.

`src/ngap/ngapIEs/UERadioCapability.hpp`:

```
class UERadioCapability {
 public:
  UERadioCapability();
  // UERadioCapability(const OCTET_STRING_t& capability);
  // UERadioCapability(const bstring& capability);
  virtual ~UERadioCapability();

  bool encode(Ngap_UERadioCapability_t& ueRadioCapability);
  bool decode(Ngap_UERadioCapability_t& ueRadioCapability);

  bool set(const OCTET_STRING_t& capability);
  bool get(OCTET_STRING_t& capability);

  bool set(const bstring& capability);
  bool get(bstring& capability);
```

```
 private:
  bstring ue_radio_capability_;
  // ^ This must be initialized explicitly
};
```

src/ngap/ngapIEs/UERadioCapability.cpp:

```
// ...

UERadioCapability::UERadioCapability() {}
// ^ default constructor doesn't initialize `ue_radio_capability_`...

// ...

void UeRadioCapabilityInfoIndicationMsg::setUERadioCapability(
    const OCTET_STRING_t& capability) {
  // ^ When a message is decoded, this function is called

    ueRadioCapability.set(capability);

    Ngap_UERadioCapabilityInfoIndicationIEs_t* ie =
        (Ngap_UERadioCapabilityInfoIndicationIEs_t*) calloc(
            1, sizeof(Ngap_PDUSessionResourceSetupRequestIEs_t));
    ie->id          = Ngap_ProtocolIE_ID_id_UERadioCapability;
    ie->criticality = Ngap_Criticality_ignore;
    ie->value.present =
        Ngap_UERadioCapabilityInfoIndicationIEs__value_PR_UERadioCapability;

    if (!ueRadioCapability.encode(ie->value.choice.UERadioCapability)) {
        Logger::ngap().error("Encode NGAP UERadioCapability IE error");
        free_wrapper((void**) &ie);
        return;
    }
    // ^ A malformed message could cause this to trigger, leaving the default UERadioCapability

    int ret = ASN_SEQUENCE_ADD(
        &ueRadioCapabilityInfoIndicationIEs->protocolIEs.list, ie);
    if (ret != 0) Logger::ngap().error("Encode NGAP UERadioCapability IE error");
    }
```

src/amf-app/amf_n2.cpp:

```
void amf_n2::handle_itti_message(
        itti_ue_radio_capability_indication& itti_msg) {
    // ...

    std::shared_ptr<gnb_context> gc = {};
    if (!assoc_id_2_gnb_context(itti_msg.assoc_id, gc)) {
        Logger::amf_n2().error(
            "No existed gNB context with assoc_id (%d)", itti_msg.assoc_id);
        return;
    }

    unsigned long amf_ue_ngap_id = {0};
    amf_ue_ngap_id               = itti_msg.ueRadioCap->getAmfUeNgapId();
    uint32_t ran_ue_ngap_id      = {0};
    ran_ue_ngap_id               = itti_msg.ueRadioCap->getRanUeNgapId();
    OCTET_STRING_t ue_radio_cap;
    itti_msg.ueRadioCap->getUERadioCapability(ue_radio_cap);
    gc->ue_radio_cap_ind = blk2bstr(ue_radio_cap.buf, ue_radio_cap.size);

    // ...
}
```

CVE-2024-24447 (VULN-B04):

Stack-based memcpy buffer overflow in the ngap_handle_pdu_session_resource_setup_response routine in OpenAirInterface CN5G AMF <= 2.0.0 allows a remote attacker with access to the N2 interface to carry out denial of service against the AMF and potentially execute code by sending a PDU Session Resource Setup Response with a `ResourceFailedToSetupList` containing zero elements.

src/ngap/ngap_app/ngap_message_callback.hpp:

```
int ngap_amf_handle_pdu_session_resource_setup_response(
        const sctp_assoc_id_t assoc_id, const sctp_stream_id_t stream,
        struct Ngap_NGAP_PDU* message_p) {
    // ...

    std::vector<PDUSessionResourceFailedToSetupItem_t> list_fail;
    if (!pdu_session_resource_setup_resp->getPduSessionResourceFailedToSetupList(
            list_fail)) {
        Logger::ngap().error(
            "decoding PduSessionResourceSetupResponseMsg "
            "getPduSessionResourceFailedToSetupList IE  error");
    } else {
        PduSessionResourceSetupUnSuccessfulTransferIE* UnSuccessfultransfer =
            new PduSessionResourceSetupUnSuccessfulTransferIE();
        uint8_t buffer[BUFFER_SIZE_512];
        memcpy(
            buffer, list_fail[0].pduSessionResourceSetupUnsuccessfulTransfer.buf,
            list_fail[0].pduSessionResourceSetupUnsuccessfulTransfer.size);
        // ^ The "FailedToSetupList" may have 0 elements; this indexes into the first
```

```
        // ...
    }

    // ...
}
```

CVE-2024-24451 (VULN-B05):

Missing fd_set bounds checks in the `sctp_receiver_thread` function of oai-cn5g-amf can cause a buffer overflow when more than 1024 descriptors are open. An attacker may repeatedly establish connections to the server to trigger this.

`src/sctp/sctp_server.cpp`:

```
void* sctp_server::sctp_receiver_thread(void* arg) {
  sctp_server* ptr = (sctp_server*) arg;
  Logger::sctp().info("Create pthread to receive SCTP message");
  int fdmax;
  int clientsock;
  fd_set master;
  fd_set read_fds;
  // ^ fd_set used (max 1024 fds; static buffer)

  if (arg == NULL) pthread_exit(NULL);
  FD_ZERO(&master);
  FD_ZERO(&read_fds);
  FD_SET(ptr->getSocket(), &master);
  fdmax = ptr->getSocket();

  while (true) {
    memcpy(&read_fds, &master, sizeof(master));
    if (select(fdmax + 1, &read_fds, NULL, NULL, NULL) == -1) {
      Logger::sctp().error(
          "[socket(%d)] Select() error: %s:%d", ptr->getSocket(),
          strerror(errno), errno);
      pthread_exit(NULL);
    }
    for (int i = 0; i <= fdmax; i++) {
      if (FD_ISSET(i, &read_fds)) {
        if (i == ptr->getSocket()) {
          if ((clientsock = accept(ptr->getSocket(), NULL, NULL)) < 0) {
            Logger::sctp().error(
                "[socket(%d)] Accept() error: %s:%d", ptr->getSocket(),
                strerror(errno), errno);
            pthread_exit(NULL);
          } else {
            FD_SET(clientsock, &master);
            // ^ if clientsock > 1024, then this causes a buffer overflow
            if (clientsock > fdmax) fdmax = clientsock;
          }
        } else {
          int ret = ptr->sctp_read_from_socket(i, ptr->app_->getPpid());
          if (ret == SCTP_RC_DISCONNECT) {
            FD_CLR(i, &master);
            if (i == fdmax) {
              while (FD_ISSET(fdmax, &master) == false) fdmax -= 1;
            }
          }
        }
      }
    }
  }
  return NULL;
}
```

uninitialized pointer dereference in the NasPdu::NasPdu component of OpenAirInterface CN5G AMF up to v2.0.0 allows attackers to cause a Denial of Service (DoS) via a crafted InitialUEMessage message sent to the AMF.

CVE-2024-24444 (VULN-B06):

Missing descriptor cleanup in the `sctp_receiver_thread` function of oai-cn5g-amf can cause resource exhaustion after a sufficient number of network connections have been made. An attacker may repeatedly establish and close connections to the server to trigger this vulnerability.

`src/sctp/sctp_server.cpp`:

```
void* sctp_server::sctp_receiver_thread(void* arg) {
    // ...

    int ret = ptr->sctp_read_from_socket(i, ptr->app_->getPpid());
    if (ret == SCTP_RC_DISCONNECT) {
        FD_CLR(i, &master);
        if (i == fdmax) {
            while (FD_ISSET(fdmax, &master) == false) fdmax -= 1;
        }
        // ^ missing close() on file descriptor
    }

    // ...
}
```

CVE-2024-24442 (VULN-B07):

```
void ngap_app::handle_receive(
    bstring payload, sctp_assoc_id_t assoc_id, sctp_stream_id_t stream,
    sctp_stream_id_t instreams, sctp_stream_id_t outstreams) {
    // ...

    Ngap_NGAP_PDU_t* ngap_msg_pdu =
            (Ngap_NGAP_PDU_t*) calloc(1, sizeof(Ngap_NGAP_PDU_t));
    asn_dec_rval_t rc = asn_decode(
            NULL, ATS_ALIGNED_CANONICAL_PER, &asn_DEF_Ngap_NGAP_PDU,
            (void**) &ngap_msg_pdu, bdata(payload), blength(payload));
    // ^ return value not checked for `asn_decode()`. If there's an error...

    Logger::ngap().debug(
            "Decoded NGAP message, procedure code %d, present %d",
            ngap_msg_pdu->choice.initiatingMessage->procedureCode,
            ngap_msg_pdu->present);
    // ^ ... then ngap_msg_pdu is all zeros; pointer accesses will cause null dereference.
    output_wrapper::print_asn_msg(&asn_DEF_Ngap_NGAP_PDU, ngap_msg_pdu);

    // ...
}
```

CVE-2024-24449 (VULN-B08):

An uninitialized pointer dereference in the NasPdu::NasPdu component of OpenAirInterface CN5G AMF up to v2.0.0 allows attackers to cause a Denial of Service (DoS) via a crafted InitialUEMessage message sent to the AMF.

`src/ngap/ngapIEs/NAS-PDU.hpp`:

```
class NAS_PDU {
 public:
  NAS_PDU();
  virtual ~NAS_PDU();

  bool encode(Ngap_NAS_PDU_t&);
  bool decode(Ngap_NAS_PDU_t&);
  // bool get(uint8_t*& buffer, size_t& size) const;
  void set(uint8_t* buffer, size_t size);

  bool get(OCTET_STRING_t& pdu) const;
  bool set(const OCTET_STRING_t& pdu);

  bool get(bstring& pdu) const;
  bool set(const bstring& pdu);

  bool get(NAS_PDU& nas_pdu) const;
  bool set(const NAS_PDU& nas_pdu);

 private:
  bstring pdu_bstring;
  // ^ private member that needs initializing (bstring is a pointer type)
};
```

`src/ngap/ngapIEs/NAS-PDU.cpp:28`:

```
NAS_PDU::NAS_PDU() {}
// ^ missing `: pdu_bstring(nullptr)` initialization; field left uninitialized
```

CVE-2024-24446 (VULN-B09):

An uninitialized pointer dereference in OpenAirInterface CN5G AMF up to v2.0.0 allows attackers to cause a Denial of Service (DoS) via a crafted InitialContextSetupResponse message sent to the AMF.

`src/ngap/ngapMsgs/InitialContextSetupResponse.hpp`:

```
class InitialContextSetupResponseMsg : public NgapUEMessage {
public:

private:
    Ngap_InitialContextSetupResponse_t* initialContextSetupResponseIEs;
    // ^ correctly initialized during `decodeFromPdu()`
    std::optional<PDUSessionResourceSetupListCxtRes>
        pduSessionResourceSetupResponseList;
    // ^ Uninitialized if the Initial Context Response message doesn't contain the IE
    std::optional<PDUSessionResourceFailedToSetupListCxtRes>
        pduSessionResourceFailedToSetupResponseList;
    // ^ Uninitialized if the Initial Context Response message doesn't contain the IE
};
```

```
bool InitialContextSetupResponseMsg::decodeFromPdu(
        Ngap_NGAP_PDU_t* ngapMsgPdu) {
    ngapPdu = ngapMsgPdu;

    if (ngapPdu->present == Ngap_NGAP_PDU_PR_successfulOutcome) {
        if (ngapPdu->choice.successfulOutcome &&
            ngapPdu->choice.successfulOutcome->procedureCode ==
                Ngap_ProcedureCode_id_InitialContextSetup &&
            ngapPdu->choice.successfulOutcome->criticality ==
                Ngap_Criticality_reject &&
            ngapPdu->choice.successfulOutcome->value.present ==
                Ngap_SuccessfulOutcome__value_PR_InitialContextSetupResponse) {
            initialContextSetupResponseIEs = &ngapPdu->choice.successfulOutcome->value
```

```
                                                .choice.InitialContextSetupResponse;
        } else {
            Logger::ngap().error("Check InitialContextSetupResponse message error");
            return false;
        }
    } else {
        Logger::ngap().error("MessageType error");
        return false;
    }
    for (int i = 0; i < initialContextSetupResponseIEs->protocolIEs.list.count;
        i++) {
        // ... decoding of IEs done here
    }

    // An Initial Context Response message with no IEs is considered valid
    return true;
}

vim src/ngap/ngap_app/ngap_message_callback.hpp:

int ngap_amf_handle_initial_context_setup_response(
        const sctp_assoc_id_t assoc_id, const sctp_stream_id_t stream,
        struct Ngap_NGAP_PDU* message_p) {
    Logger::ngap().debug("Handling Initial Context Setup Response");

    InitialContextSetupResponseMsg* init_cxt_setup_response =
        new InitialContextSetupResponseMsg();
    if (!init_cxt_setup_response->decodeFromPdu(message_p)) {
        Logger::ngap().error("Decoding InitialContextSetupResponse message error");
        return RETURNerror;
    }

    std::vector<PDUSessionResourceSetupResponseItem_t> list;
    if (!init_cxt_setup_response->getPduSessionResourceSetupResponseList(list)) {
        // ^ getter accesses uninitialized memory if the field was never in the packet
        Logger::ngap().debug(
            "Decode PduSessionResourceSetupResponseList IE error or this IE is not "
            "available");
        return RETURNok;
    }

    // ...
}
```

CVE-2024-24443 (VULN-B10):

The `pduSessionResourceFailedToSetupResponseList` field in the `PduSessionResourceSetupResponseMsg` class is an optional that is default-initialized to none. However, it is subsequently used as if it is initialized when decoding an NGAP PDU. The uninitialized values may trigger an out-of-bounds vector write when `push_back()` is called.

`src/ngap/ngapMsgs/PduSessionResourceSetupResponse.cpp:321`:

```
case Ngap_ProtocolIE_ID_id_PDUSessionResourceFailedToSetupListSURes: {
    if (pduSessionResourceSetupResponseIEs->protocolIEs.list.array[i]
                ->criticality == Ngap_Criticality_ignore &&
        pduSessionResourceSetupResponseIEs->protocolIEs.list.array[i]
                ->value.present ==
            Ngap_PDUSessionResourceSetupResponseIEs__value_PR_PDUSessionResourceFailedToSetupListSURes) {
    PDUSessionResourceFailedToSetupListSURes tmp = {};
    if (!pduSessionResourceFailedToSetupResponseList->decode(
    // ^ called from an optional value of None means this is an uninitialized dereference
            &pduSessionResourceSetupResponseIEs->protocolIEs.list
                .array[i]
                ->value.choice
                .PDUSessionResourceFailedToSetupListSURes)) {
        Logger::ngap().error(
            "Decoded NGAP PDUSessionResourceFailedToSetupListSURes IE "
            "error!");
        return false;
    }
    pduSessionResourceFailedToSetupResponseList =
        std::optional<PDUSessionResourceFailedToSetupListSURes>(tmp);
    // ^ now the field is properly initialized, but at this point it's too late

    // ...
}
```

CVE-2024-24447 (VULN-B11):

A buffer overflow in the `ngap_amf_handle_pdu_session_resource_setup_response` function of oai-cn5g-amf up to v2.0.0 allows attackers to cause a Denial of Service (DoS) via a PDU Session Resource Setup Response with an empty Response Item list.

`src/ngap/ngap_app/ngap_message_callback.hpp`:

```
int ngap_amf_handle_pdu_session_resource_setup_response(
    const sctp_assoc_id_t assoc_id, const sctp_stream_id_t stream,
    struct Ngap_NGAP_PDU* message_p) {
  Logger::ngap().debug("Handle PDU Session Resource Setup Response");

  PduSessionResourceSetupResponseMsg* pdu_session_resource_setup_resp =
      new PduSessionResourceSetupResponseMsg();
  if (!pdu_session_resource_setup_resp->decodeFromPdu(message_p)) {
```

```
    Logger::ngap().error(
        "Decoding PduSessionResourceSetupResponseMsg message error");
    return RETURNerror;
  }

  std::vector<PDUSessionResourceSetupResponseItem_t> list;
  if (!pdu_session_resource_setup_resp->getPduSessionResourceSetupResponseList(
          list)) {
    Logger::ngap().error(
        "Decoding PduSessionResourceSetupResponseMsg "
        "getPduSessionResourceSetupResponseList IE error");
    // return RETURNerror;
  } else {
    // TODO: for multiple PDU Sessions
    itti_nsmf_pdusession_update_sm_context* itti_msg =
        new itti_nsmf_pdusession_update_sm_context(TASK_NGAP, TASK_AMF_SBI);
    long amf_ue_ngap_id = pdu_session_resource_setup_resp->getAmfUeNgapId();
    std::shared_ptr<nas_context> nct = {};
    if (!amf_n1_inst->amf_ue_id_2_nas_context(amf_ue_ngap_id, nct)) {
      Logger::ngap().error(
          "No UE NAS context with amf_ue_ngap_id (0x%x)", amf_ue_ngap_id);
      return RETURNerror;
    }
    itti_msg->supi           = conv::imsi_to_supi(nct->imsi);
    itti_msg->pdu_session_id = list[0].pduSessionId;
    // ^ list could have 0 elements (out-of-bound array index)
    itti_msg->n2sm           = blk2bstr(
        list[0].pduSessionResourceSetupResponseTransfer.buf,
        list[0].pduSessionResourceSetupResponseTransfer.size);
    // ^ list could have 0 elements (out-of-bound array index)
```

Open5GS (5G)

Affected versions: Open5GS <= 2.6.4

NAS Protocol Vulnerabilities

VULN-A01:

A malformed SUCI within the NAS 5GMM message can lead to a parsing error and reachable assertion.

`src/amf/context.c`:

```
amf_ue_t *amf_ue_find_by_message(ogs_nas_5gs_message_t *message)
{
    // ...

    suci = ogs_nas_5gs_suci_from_mobile_identity(mobile_identity);
    // ^ malformed mobile identity causes this to fail...
    ogs_assert(suci);
    // ^ ... leading to this reachable assertion.

    // ...
}
```

CVE-2024-24428 (**VULN-A02**):

A zero-length NAS 5GMM packet triggers a reachable assertion.

`lib/nas/5gs/decoder.c`:

```
int ogs_nas_5gmm_decode(ogs_nas_5gs_message_t *message, ogs_pkbuf_t *pkbuf)
{
    // ^ received NAS payload of 0 bytes in length...
    int size = 0;
    int decoded = 0;

    ogs_assert(pkbuf);
    ogs_assert(pkbuf->data);
    ogs_assert(pkbuf->len);
    // ^ ...triggers this assertion
```

2024-24427 (**VULN-A04**):

A malformed SUCI within the NAS 5GMM message can lead to a parsing error and reachable assertion.

`src/amf/context.c`:

```
void amf_ue_set_suci(amf_ue_t *amf_ue,
        ogs_nas_5gs_mobile_identity_t *mobile_identity) {
    // ...

    suci = ogs_nas_5gs_suci_from_mobile_identity(mobile_identity);
    // ^ malformed mobile identity causes this to fail...
    ogs_assert(suci);
    // ^ ... leading to this reachable assertion.

    // ...
}
```

NGAP Protocol Vulnerabilities

VULN-A03:

Malformed `SON Configuration Transfer` information element can trigger a reachable assertion when sending a Downlink SON Configuration Transfer.

`src/amf/ngap-build.c:1768`:

```
ogs_pkbuf_t *ngap_build_downlink_ran_configuration_transfer(
    NGAP_SONConfigurationTransfer_t *transfer)
{
    // ...

    rv = ogs_asn_copy_ie(&asn_DEF_NGAP_SONConfigurationTransfer,
            transfer, SONConfigurationTransfer);
    // ^ a malformed SONConfigurationTransfer can cause this to fail...
    ogs_assert(rv == OGS_OK);
    // ^ ...leading to reachable assertion failure.

    // ...
}
```

VULN-A05:

Malformed `SON Configuration Transfer` information element can trigger a reachable assertion when handling an Uplink SON Configuration Transfer.

```
void ngap_handle_uplink_ran_configuration_transfer(
        amf_gnb_t *gnb, ogs_ngap_message_t *message, ogs_pkbuf_t *pkbuf)
{
    // ...

    ogs_assert(OGS_OK ==
        ngap_send_downlink_ran_configuration_transfer(
            target_gnb, SONConfigurationTransfer));
    // ^ reachable assertion when malformed SONConfigurationTransfer received

    // ...
}
```

Magma (5G)

Affected versions: Magma <= 1.8.0 (fixed in v1.9 commit 08472ba98b8321f802e95f5622fa90fec2dea486)

NAS Protocol Vulnerabilities

CVE-2024-24425 (**VULN-C01**):

Magma v1.8.0 and OAI EPC Federation v1.20 were discovered to contain an out-of-bounds read in the amf_as_establish_req function at /tasks/amf/amf_as.cpp. This vulnerability allows attackers to cause a Denial of Service (DoS) via a crafted NAS packet.

`lte/gateway/c/core/oai/tasks/amf/amf_as.cpp`:

```
static status_code_e amf_as_establish_req(amf_as_establish_t* msg,
                                          int* amf_cause) {
  OAILOG_FUNC_IN(LOG_AMF_APP);
  amf_security_context_t* amf_security_context = NULL;
  amf_nas_message_decode_status_t decode_status;
  memset(&decode_status, 0, sizeof(decode_status));
  int decoder_rc = 1;
  status_code_e rc = RETURNerror;
  tai_t originating_tai = {0};
  amf_nas_message_t nas_msg = {0};
  ue_m5gmm_context_s* ue_m5gmm_context = NULL;
  ue_m5gmm_context = amf_ue_context_exists_amf_ue_ngap_id(msg->ue_id);
  if (ue_m5gmm_context == NULL) {
    OAILOG_ERROR(LOG_AMF_APP,
                 "ue context not found for the ue_id=" AMF_UE_NGAP_ID_FMT "\n",
                 msg->ue_id);
    OAILOG_FUNC_RETURN(LOG_AMF_APP, rc);
  }

  amf_context_t* amf_ctx = NULL;
  amf_ctx = &ue_m5gmm_context->amf_context;

  if (amf_ctx) {
    if (IS_AMF_CTXT_PRESENT_SECURITY(amf_ctx)) {
      amf_security_context = &amf_ctx->_security;
    }
  }

  if ((msg->nas_msg->data[1] != 0x0) && (msg->nas_msg->data[9] == 0x5c)) {
    // ^ data length not checked before accessed--OOB read
    for (int i = 0, j = 7; j < blength(msg->nas_msg); i++, j++) {
      msg->nas_msg->data[i] = msg->nas_msg->data[j];
    }
    msg->nas_msg->slen = msg->nas_msg->slen - 7;
  }
```

NGAP Protocol Vulnerabilities

CVE-2024-24426 (**VULN-C02-VULN-C20**):

Reachable assertions in the NGAP_FIND_PROTOCOLIE_BY_ID function of OpenAirInterface Magma v1.8.0 and OAI EPC Federation v1.2.0 allow attackers to cause a Denial of Service (DoS) via a crafted NGAP packet. These assertions manifest in 19 locations that require conditional branches to handle.

The following method will assign `ie` a pointer to the Information Element requested:

`lte/gateway/c/core/oai/tasks/ngap/ngap_common.h`:

```
  NGAP_FIND_PROTOCOLIE_BY_ID(Ngap_NGSetupRequestIEs_t, ie, container,
                             Ngap_ProtocolIE_ID_id_GlobalRANNodeID, true);
```

Magma will assert within the NGAP_FIND_PROTOCOLIE_BY_ID macro if the requested field is absent from the decoded ASN.1 payload. This macro is dispersed across 19 different locations in the Mamga AMF; each of these locations required a check on whether the `ie` is null in addition to removing the assertion in order to properly mitigate this threat.

Magma/OpenAirInterface (LTE)

Note: Magma and OAI use the same software for portions of their LTE core

Affected versions: Magma <= 1.8.0 (fixed in v1.9 commit 08472ba98b8321f802e95f5622fa90fec2dea486)

NAS Protocol Vulnerabilities

CVE-2023-37024 (**VULN-D01**):

The Magma MME contains a reachable assertion when handling the `Emergency Number List` field of a received NAS packet. This assertion is due to an unfinished routine in parsing the field.

`lte/gateway/c/core/oai/lib/3gpp/3gpp_24.008_mm_ies.c`:

```
int decode_emergency_number_list_ie(
    emergency_number_list_t* emergencynumberlist,
    const bool iei_present,
    uint8_t* buffer,
    const uint32_t len
) {
    // ...

    for (int i = e->lengthofemergencynumberinformation - 1;
        i < EMERGENCY_NUMBER_MAX_DIGITS; i++) {
        e->number_digit[i] = 0xFF;
    }
    Fatal("TODO emergency_number_list_t->next");
    // ^ Reachable assertion

    return decoded;
}
```

CVE-2023-37029 (**VULN-D06**):

The S1AP handling routines of Magma MME contain an assertion failure when a received NAS packet is larger than 1000 bytes in length. This reachable assertion can be trigggered by an adversary sending an unexpectedly large NAS payload to the MME.

`lte/gateway/c/core/oai/tasks/s1ap/s1ap_mme_itti_messaging.cpp`:

```
void s1ap_mme_itti_s1ap_initial_ue_message(
    const sctp_assoc_id_t assoc_id, const uint32_t enb_id,
    const enb_ue_s1ap_id_t enb_ue_s1ap_id, const uint8_t* const nas_msg,
    const size_t nas_msg_length, const tai_t* const tai,
    const ecgi_t* const ecgi, const long rrc_cause,
    const s_tmsi_t* const opt_s_tmsi, const csg_id_t* const opt_csg_id,
    const gummei_t* const opt_gummei,
    const void* const opt_cell_access_mode,           // unused
    const void* const opt_cell_gw_transport_address,  // unused
    const void* const opt_relay_node_indicator)       // unused
{
    MessageDef* message_p = NULL;

    OAILOG_FUNC_IN(LOG_S1AP);
    AssertFatal((nas_msg_length < 1000), "Bad length for NAS message %lu",
                nas_msg_length);
    // ^ oversized NAS payload triggers reachable assertion

    // ...
}
```

CVE-2023-37032 (**VULN-D09**):

The Magma MME uses a fixed-size buffer with no length check when handling emergency numbers listed in the `Emergency Number List` field of a received NAS packet. Note that the threat of this attack is limited to Denial of Service due to **CVE-2023-37024**; however, the two vulnerabilities have unique causes and involved distinct fixes.

lte/gateway/c/core/oai/lib/3gpp/3gpp_24.008_mm_ies.c:

```
int decode_emergency_number_list_ie(
    emergency_number_list_t* emergencynumberlist,
    const bool iei_present,
    uint8_t* buffer,
    const uint32_t len
) {
    // ...

    e->lengthofemergencynumberinformation = *(buffer + decoded);
    // ^ if this valaue is > 20...
    decoded++;
    emergencynumberlist->emergencyservicecategoryvalue =
        *(buffer + decoded) & 0x1f;
    decoded++;
    for (int i = 0; i < e->lengthofemergencynumberinformation - 1; i++) {
        e->number_digit[i] = *(buffer + decoded);
        // ^ then this will overflow the bounds of `number_digit`
        decoded++;
    }
    for (int i = e->lengthofemergencynumberinformation - 1;
        i < EMERGENCY_NUMBER_MAX_DIGITS; i++) {
        e->number_digit[i] = 0xFF;
    }

    // ...
}
```

CVE-2024-24419 (**VULN-D16**):

An `Initial UE Message` packet containing a `Bearer Resource Modification Request` with a a malformed Traffic Flow Template packet filter can cause an out-of-bounds read past the end of a buffer.

lte/gateway/c/core/oai/lib/3gpp/3gpp_24.008_sm_ies.c:

```
static int decode_traffic_flow_template_packet_filter(
        packet_filter_t* packetfilter, const uint8_t* const buffer,
        const uint32_t len) {
    int decoded = 0, j;
    // ...

    packetfilter->identifier = *(buffer + decoded) & 0x0f;
    // ^ no bound check prior to pulling bytes off of wire
    decoded++;

    /*
     * Packet filter evaluation precedence
     */
    IES_DECODE_U8(buffer, decoded, packetfilter->eval_precedence);
    /*
     * Length of the Packet filter contents field
     */
    uint8_t pkflen;

    IES_DECODE_U8(buffer, decoded, pkflen);
    // ^ likewise, more bytes read from wire prior to check
    /*
    * Packet filter contents
    */
    int pkfstart = decoded;
    while (decoded - pkfstart < pkflen) {
        // ^ length field of packet not checked

        // Additional data pulled from `buffer` without length checks...
    }

    return decoded;
}
```

CVE-2024-24416 (**VULN-D17**):

An `Initial UE Message` containing a NAS payload with a malformed Access Point Name IE will trigger a buffer overflow.

lte/gateway/c/core/oai/lib/3gpp/3gpp_24.008_sm_ies.c:

```
int decode_access_point_name_ie(access_point_name_t* access_point_name,
                                bool is_ie_present, uint8_t* buffer,
                                const uint32_t len) {
    int decoded = 0;
    uint8_t ielen = 0;

    *access_point_name = NULL;

    if (is_ie_present > 0) {
        CHECK_PDU_POINTER_AND_LENGTH_DECODER(buffer,
                                        ACCESS_POINT_NAME_IE_MIN_LENGTH, len);
        CHECK_IEI_DECODER(SM_ACCESS_POINT_NAME_IEI, *buffer);
        decoded++;
    } else {
        CHECK_PDU_POINTER_AND_LENGTH_DECODER(
            buffer, (ACCESS_POINT_NAME_IE_MIN_LENGTH - 1), len);
```

```
    }

    ielen = *(buffer + decoded);
    decoded++;
    CHECK_LENGTH_DECODER(len - decoded, ielen);

    if (1 <= ielen) {
        int length_apn = *(buffer + decoded);
        // ^ length field read from packet, not checked against buffer length
        decoded++;
        *access_point_name = blk2bstr((void*)(buffer + decoded), length_apn);
        // ^ resulting string reads past the end of `buffer`
        decoded += length_apn;

        // ...
    }
    return decoded;
}
```

CVE-2024-24424 (**VULN-D18**):

An `Initial UE Message` S1AP packet containing a NAS payload with a malformed Access Point Name IE will trigger an assertion during decoding.

`lte/gateway/c/core/oai/lib/3gpp/3gpp_24.008_sm_ies.c`:

```
int decode_access_point_name_ie(access_point_name_t* access_point_name,
                                bool is_ie_present, uint8_t* buffer,
                                const uint32_t len) {
    int decoded = 0;
    uint8_t ielen = 0;

    *access_point_name = NULL;

    if (is_ie_present > 0) {
        CHECK_PDU_POINTER_AND_LENGTH_DECODER(buffer,
                                             ACCESS_POINT_NAME_IE_MIN_LENGTH, len);
        CHECK_IEI_DECODER(SM_ACCESS_POINT_NAME_IEI, *buffer);
        decoded++;
    } else {
        CHECK_PDU_POINTER_AND_LENGTH_DECODER(
            buffer, (ACCESS_POINT_NAME_IE_MIN_LENGTH - 1), len);
    }

    ielen = *(buffer + decoded);
    // ^ ielen determined by packet
    decoded++;

    // ...

    // apn terminated by '.' ?
    if (length_apn > 0) {
        AssertFatal(ielen >= length_apn,
                    "Mismatch in lengths remaining ielen %d apn length %d",
                    ielen, length_apn);
        // ^ attacker can select `ielen` value that will trigger this assertion
        bcatblk(*access_point_name, (void*)(buffer + decoded), length_apn);
        decoded += length_apn;
        ielen = ielen - length_apn;
    }

    // ...

    return decoded;
}
```

CVE-2024-24420 (**VULN-D19**):

An `Initial UE Message` S1AP packet containing a NAS payload containing a `Linked TI IE` will trigger a fatal assertion.

`lte/gateway/c/core/oai/lib/3gpp/3gpp_24.008_sm_ies.c`:

```
int decode_linked_ti_ie(linked_ti_t* linkedti, const bool iei_present,
                        uint8_t* buffer, const uint32_t len) {
    Fatal("TODO Implement decode_linked_ti_ie");
    // ^ reachable by sending packet containing a `Linked TI IE`
    return -1;
}
```

CVE-2024-24418 (**VULN-D20**):

An `Initial UE Message` S1AP packet containing a NAS payload containing a malformed PDN Address will trigger a buffer overflow.

`lte/gateway/c/core/oai/tasks/nas/ies/PdnAddress.cpp`:

```
int decode_pdn_address(PdnAddress* pdnaddress, uint8_t iei, uint8_t* buffer,
                       uint32_t len) {
  int decoded = 0;
  // ^ `len` unsigned, `decoded` signed
  uint8_t ielen = 0;
  int decode_result;

  if (iei > 0) {
```

```
    CHECK_IEI_DECODER(iei, *buffer);
    decoded++;
  }

  // Suppose len == 0. Then this will address out of bounds...
  ielen = *(buffer + decoded);
  decoded++;
  CHECK_LENGTH_DECODER(len - decoded, ielen);
  // ^ len is unsigned, so `len - decoded` leads to integer underflow
  pdnaddress->pdntypevalue = *(buffer + decoded) & 0x7;
  // More out-of-bounds accesses...
  decoded++;

  // ...
}
```

CVE-2024-24421 (VULN-D21):

An `Initial UE Message` S1AP packet containing a NAS payload containing a malformed EMM Attach Request will trigger a buffer overflow.

When `nas_message_decode` decodes a packet, it checks its external header. However, the packet may be encrypted, such that the internal header is ESM and is decoded as such, leading to type confusion when the union is later referenced as an EMM packet.

`lte/gateway/c/core/oai/tasks/nas/emm/sap/emm_as.cpp`:

```
static status_code_e emm_as_recv(mme_ue_s1ap_id_t ue_id,
                                 tai_t const* originating_tai,
                                 ecgi_t const* originating_ecgi, bstring msg,
                                 size_t len, int* emm_cause,
                                 nas_message_decode_status_t* decode_status) {

    // ^ code prior to this method reads the outer header of the packet as being EMM

    // ...

    decoder_rc =
        nas_message_decode(msg->nas_msg->data, &nas_msg, blength(msg->nas_msg),
                           emm_security_context, &decode_status);
    // ^ During decode, the inner payload is decrypted. This may return either an EMM OR an ESM struct.
    bdestroy_wrapper(&msg->nas_msg);

    if (decoder_rc < 0) {
        if (decoder_rc < TLV_FATAL_ERROR) {
            *emm_cause = EMM_CAUSE_PROTOCOL_ERROR;
            OAILOG_FUNC_RETURN(LOG_NAS_EMM, RETURNerror);
        } else if (decoder_rc == TLV_MANDATORY_FIELD_NOT_PRESENT) {
            *emm_cause = EMM_CAUSE_INVALID_MANDATORY_INFO;
            REQUIREMENT_3GPP_24_301(R10_5_5_1_2_7_b__1);
        } else if (decoder_rc == TLV_UNEXPECTED_IEI) {
            *emm_cause = EMM_CAUSE_IE_NOT_IMPLEMENTED;
            REQUIREMENT_3GPP_24_301(R10_5_5_1_2_7_b__2);
        } else {
            *emm_cause = EMM_CAUSE_PROTOCOL_ERROR;
            REQUIREMENT_3GPP_24_301(R10_5_5_1_2_7_b__4);
        }
    }

    // BUGFIX: check for internal header consistency
    if (nas_msg.plain.emm.protocol_discriminator != EPS_MOBILITY_MANAGEMENT_MESSAGE) {
        // The NAS message had external EMM header and internal encrypted ESM header--discard
        return RETURNerror;
    }

    /*
     * Process initial NAS message
     */
    EMM_msg* emm_msg = &nas_msg.plain.emm;
    // ^ The (internal) plain message struct is automatically assumed to be EMM. If the payload
    // specified an inner ESM message, this will incorrectly reinterpret it as an EMM struct.

    switch (emm_msg->header.message_type) {
        // ^ Subsequent field accesses will confuse pointers and data fields between EMM/ESM
        case ATTACH_REQUEST:
            // ...
    }

    // ...
}
```

CVE-2024-24423 (VULN-D22):

An `Initial UE Message` S1AP packet containing a NAS payload containing a malformed ESM Message will trigger a buffer overflow.

`lte/gateway/c/core/oai/tasks/nas/ies/EsmMessageContainer.cpp`:

```
int decode_esm_message_container(EsmMessageContainer* esmmessagecontainer,
                                 uint8_t iei, uint8_t* buffer, uint32_t len) {
    int decoded = 0;
    int decode_result;
    uint16_t ielen;

    OAILOG_FUNC_IN(LOG_NAS_ESM);
```

```
    if (iei > 0) {
        CHECK_IEI_DECODER(iei, *buffer);
        decoded++;
    }

    DECODE_LENGTH_U16(buffer + decoded, ielen, decoded);
    // ^ doesn't check `decoded` len before pulling bytes
    CHECK_LENGTH_DECODER(len - decoded, ielen);
    // ^ if len < 2, this check is useless due to unsigned integer underflow

    if ((decode_result = decode_bstring(esmmessagecontainer, ielen,
                                        buffer + decoded, len - decoded)) < 0) {
        // ^ out-of-bounds read, up to attacker-specified `ielen` bytes
        OAILOG_FUNC_RETURN(LOG_NAS_ESM, decode_result);
    } else {
        decoded += decode_result;
    }

    OAILOG_FUNC_RETURN(LOG_NAS_ESM, decoded);
}
```

CVE-2024-24417 (VULN-D23):

An `Initial UE Message` S1AP packet containing a NAS payload containing a malformed Protocol Configuration Options field will trigger an out-of-bounds read.

`lte/gateway/c/core/oai/lib/3gpp/3gpp_24.008_sm_ies.c`:

```
int decode_protocol_configuration_options(
        protocol_configuration_options_t* protocolconfigurationoptions,
        const uint8_t* const buffer, const uint32_t len) {
    int decoded = 0;
    int decode_result = 0;

    if (((*(buffer + decoded) >> 7) & 0x1) != 1) {
        return TLV_VALUE_DOESNT_MATCH;
    }

    /*
    * Bits 7 to 4 of octet 3 are spare, read as 0
    */
    if (((*(buffer + decoded) & 0x78) >> 3) != 0) {
        return TLV_VALUE_DOESNT_MATCH;
    }

    protocolconfigurationoptions->configuration_protocol =
        (*(buffer + decoded) >> 1) & 0x7;
    // ^ access buffer without checking length fields--out-of-bound read when len == 0
    decoded++;

    // ...
}
```

CVE-2024-24422 (VULN-D24):

An `Initial UE Message` S1AP packet containing a NAS payload containing a malformed Protocol Configuration Options field will trigger a write buffer overflow.

`lte/gateway/c/core/oai/lib/3gpp/3gpp_24.008_sm_ies.c`:

```
int decode_protocol_configuration_options(
        protocol_configuration_options_t* protocolconfigurationoptions,
        const uint8_t* const buffer, const uint32_t len) {
    // ...

    while (3 <= ((int32_t)len - (int32_t)decoded)) {
        DECODE_U16(
            buffer + decoded,
            protocolconfigurationoptions
                ->protocol_or_container_ids[protocolconfigurationoptions
                                            ->num_protocol_or_container_id]
                .id,
            decoded);
        // ^ this will write beyond the end of `protocol_or_container_ids[]`...
        DECODE_U8(
            buffer + decoded,
            protocolconfigurationoptions
                ->protocol_or_container_ids[protocolconfigurationoptions
                                            ->num_protocol_or_container_id]
                .length,
            decoded);

        // ...

        protocolconfigurationoptions->num_protocol_or_container_id += 1;
        // ^ ... if this iterates more than 30 times.
    }

    return decoded;
}
```

S1AP Protocol Vulnerabilities

CVE-2023-37025 (VULN-D02):

A malformed “Reset” S1AP packet missing a `Reset Type` field will cause Magma/OAI due to a null pointer dereference.

`lte/gateway/c/core/oai/tasks/s1ap/s1ap_mme_handlers.cpp`:

```
status_code_e s1ap_mme_handle_erab_setup_response(
        oai::S1apState* state, const sctp_assoc_id_t assoc_id,
        const sctp_stream_id_t stream, S1ap_S1AP_PDU_t* pdu) {
    // ...

    S1AP_FIND_PROTOCOLIE_BY_ID(S1ap_ResetIEs_t, ie, container,
                               S1ap_ProtocolIE_ID_id_ResetType, true);
    // ^ ie assigned null pointer if ResetType field not present

    S1ap_ResetType_t* resetType = &ie->value.choice.ResetType;
    // ^ Null dereference on ie
    switch (resetType->present) {
        // ...

    }
```

CVE-2023-37026 (VULN-D03):

A malformed `E-RAB Release Response` S1AP packet missing a required `MME_UE_S1AP_ID` field will cause Magma/OAI to crash due to a null pointer dereference.

`lte/gateway/c/core/oai/tasks/s1ap/s1ap_mme_handlers.cpp`:

```
status_code_e s1ap_mme_handle_erab_rel_response(oai::S1apState* state,
                                                const sctp_assoc_id_t assoc_id,
                                                const sctp_stream_id_t stream,
                                                S1ap_S1AP_PDU_t* pdu) {
    // ...

    S1AP_FIND_PROTOCOLIE_BY_ID(S1ap_E_RABReleaseResponseIEs_t, ie, container,
                           S1ap_ProtocolIE_ID_id_MME_UE_S1AP_ID, true);
    // ^ ie assigned null pointer if MME_UE_S1AP_ID field not present
    mme_ue_s1ap_id = ie->value.choice.MME_UE_S1AP_ID;
    // ^ Null dereference on ie

    // ...
}
```

CVE-2023-37027 (VULN-D04):

A malformed `E-RAB Modification Indication` S1AP packet missing a required `MME_UE_S1AP_ID` field will cause Magma/OAI to crash due to a null pointer dereference.

`lte/gateway/c/core/oai/tasks/s1ap/s1ap_mme_handlers.cpp`:

```
status_code_e s1ap_mme_handle_erab_modification_indication(
        oai::S1apState* state, const sctp_assoc_id_t assoc_id,
        const sctp_stream_id_t stream, S1ap_S1AP_PDU_t* pdu) {
    // ...

    S1AP_FIND_PROTOCOLIE_BY_ID(S1ap_E_RABModificationIndicationIEs_t, ie,
                         container, S1ap_ProtocolIE_ID_id_MME_UE_S1AP_ID,
                         true);
    // ^ ie assigned null pointer if MME_UE_S1AP_ID field not present
    mme_ue_s1ap_id = ie->value.choice.MME_UE_S1AP_ID;
    // ^ Null dereference on ie

    // ...
}
```

CVE-2023-37028 (VULN-D05):

A malformed `E-RAB Modification Indication` S1AP packet missing a required `eNB_UE_S1AP_ID` field will cause Magma/OAI to crash due to a null pointer dereference.

`lte/gateway/c/core/oai/tasks/s1ap/s1ap_mme_handlers.cpp`:

```
status_code_e s1ap_mme_handle_erab_modification_indication(
        oai::S1apState* state, const sctp_assoc_id_t assoc_id,
        const sctp_stream_id_t stream, S1ap_S1AP_PDU_t* pdu) {
    // ...

    S1AP_FIND_PROTOCOLIE_BY_ID(S1ap_E_RABModificationIndicationIEs_t, ie,
                           container, S1ap_ProtocolIE_ID_id_eNB_UE_S1AP_ID,
                           true);
    // ^ ie assigned null pointer if eNB_UE_S1AP_ID field not present
    enb_ue_s1ap_id =
        (enb_ue_s1ap_id_t)(ie->value.choice.ENB_UE_S1AP_ID & ENB_UE_S1AP_ID_MASK);
    // ^ Null dereference on ie

    // ...
}
```

CVE-2023-37030 (VULN-D07):

A malformed `Initial UE Message` S1AP packet missing a `ENB-UE-S1AP-ID` field will cause Magma/OAI to crash due to a null pointer dereference.

`lte/gateway/c/core/oai/tasks/s1ap/s1ap_mme_nas_procedures.cpp`:

```
status_code_e s1ap_mme_handle_initial_ue_message(oai::S1apState* state,
                                                 const sctp_assoc_id_t assoc_id,
                                                 const sctp_stream_id_t stream,
                                                 S1ap_S1AP_PDU_t* pdu) {
    // ...

    S1AP_FIND_PROTOCOLIE_BY_ID(S1ap_InitialUEMessage_IEs_t, ie, container,
                               S1ap_ProtocolIE_ID_id_eNB_UE_S1AP_ID, true);
    // ^ ie assigned null pointer if eNB_UE_S1AP_ID field not present

    OAILOG_INFO(
    LOG_S1AP,
    "Received S1AP INITIAL_UE_MESSAGE ENB_UE_S1AP_ID " ENB_UE_S1AP_ID_FMT
    " assoc-id:%d \n",
    (enb_ue_s1ap_id_t)ie->value.choice.ENB_UE_S1AP_ID, assoc_id);
    // ^ Null dereference on ie

    // ...
}
```

CVE-2023-37031 (VULN-D08):

A malformed `eNB Configuration Transfer` S1AP packet missing a required `Target eNB ID` field will cause Magma/OAI to crash due to a null pointer dereference.

`lte/gateway/c/core/oai/tasks/s1ap/s1ap_mme_handlers.cpp`:

```
status_code_e s1ap_mme_handle_enb_configuration_transfer(
        oai::S1apState* state, const sctp_assoc_id_t assoc_id,
        const sctp_stream_id_t stream, S1ap_S1AP_PDU_t* pdu) {
    // ...

    S1AP_FIND_PROTOCOLIE_BY_ID(S1ap_ENBConfigurationTransferIEs_t, ie, container,
                         S1ap_ProtocolIE_ID_id_SONConfigurationTransferECT,
                         false);
    // ^ ie assigned null pointer if SONConfigurationTransferECT field not present

    // ...

    targeteNB_ID = &ie->value.choice.SONConfigurationTransfer.targeteNB_ID;
    // ^ Null dereference on ie

    // ...
}
```

CVE-2023-37033 (VULN-D10):

A malformed “Initial UE Message” S1AP packet missing a `EUTRAN CGI` field will cause Magma/OAI to crash due to a null pointer dereference.

`lte/gateway/c/core/oai/tasks/s1ap/s1ap_mme_nas_procedures.cpp`:

```
status_code_e s1ap_mme_handle_initial_ue_message(oai::S1apState* state,
                                                 const sctp_assoc_id_t assoc_id,
                                                 const sctp_stream_id_t stream,
                                                 S1ap_S1AP_PDU_t* pdu) {
    // ...

    S1AP_FIND_PROTOCOLIE_BY_ID(S1ap_InitialUEMessage_IEs_t, ie, container,
                           S1ap_ProtocolIE_ID_id_EUTRAN_CGI, true);
    // ^ ie assigned null pointer if EUTRAN_CGI field not present
    if (!(ie->value.choice.EUTRAN_CGI.pLMNidentity.size == 3)) {
        // ^ Null pointer dereference of `ie`
        OAILOG_ERROR(LOG_S1AP, "Incorrect PLMN size \n");
        return RETURNerror;
    }

    // ...
}
```

CVE-2023-37034 (VULN-D11):

A malformed `Initial UE Message` S1AP packet missing a required `TAI` field will cause Magma/OAI to crash due to a null pointer dereference.

`lte/gateway/c/core/oai/tasks/s1ap/s1ap_mme_nas_procedures.cpp`:

```
status_code_e s1ap_mme_handle_initial_ue_message(oai::S1apState* state,
                                                 const sctp_assoc_id_t assoc_id,
                                                 const sctp_stream_id_t stream,
                                                 S1ap_S1AP_PDU_t* pdu) {
    // ...

    S1AP_FIND_PROTOCOLIE_BY_ID(S1ap_InitialUEMessage_IEs_t, ie, container,
                           S1ap_ProtocolIE_ID_id_TAI, true);
    // ^ ie assigned null pointer if TAI field not present
    OCTET_STRING_TO_TAC(&ie->value.choice.TAI.tAC, tai.tac);
    // ^ Null pointer dereference of `ie`
```

```
    // ...
}
```

CVE-2023-37035 (VULN-D12):

A malformed `S1 Setup Request` S1AP packet missing a required `Global eNB ID` field will cause Magma/OAI to crash due to a null pointer dereference.

`lte/gateway/c/core/oai/tasks/s1ap/s1ap_mme_handlers.cpp`:

```
status_code_e s1ap_mme_handle_s1_setup_request(oai::S1apState* state,
                                               const sctp_assoc_id_t assoc_id,
                                               const sctp_stream_id_t stream,
                                               S1ap_S1AP_PDU_t* pdu) {
    // ...

    S1AP_FIND_PROTOCOLIE_BY_ID(S1ap_S1SetupRequestIEs_t, ie, container,
                           S1ap_ProtocolIE_ID_id_Global_ENB_ID, true);
    // ^ ie assigned null pointer if Global_ENB_ID field not present
    if (ie->value.choice.Global_ENB_ID.eNB_ID.present ==
        S1ap_ENB_ID_PR_homeENB_ID) {
        // ^ Null pointer dereference of `ie`
        // ...
    }

    // ...
}
```

CVE-2023-37036 (VULN-D13):

A malformed “Uplink NAS Transport” S1AP packet missing a `ENB_UE_S1AP_ID` field will cause Magma/OAI due to a null pointer dereference.

`lte/gateway/c/core/oai/tasks/s1ap/s1ap_mme_nas_procedures.cpp`:

```
status_code_e s1ap_mme_handle_uplink_nas_transport(
        oai::S1apState* state, const sctp_assoc_id_t assoc_id,
        __attribute__((unused)) const sctp_stream_id_t stream,
        S1ap_S1AP_PDU_t* pdu) {
    // ...

    S1AP_FIND_PROTOCOLIE_BY_ID(S1ap_UplinkNASTransport_IEs_t, ie, container,
                           S1ap_ProtocolIE_ID_id_eNB_UE_S1AP_ID, true);
    // ^ ie assigned null pointer if eNB_UE_S1AP_ID field not present
    enb_ue_s1ap_id = (enb_ue_s1ap_id_t)ie->value.choice.ENB_UE_S1AP_ID;
    // ^ Null pointer dereference of `ie`

    // ...
}
```

CVE-2023-37037 (VULN-D14):

A malformed `S1 Setup Request` S1AP packet missing a required `Supported TAs` choice in the `Supported TAs` field will cause Magma/OAI to crash due to a null pointer dereference.

`lte/gateway/c/core/oai/tasks/s1ap/s1ap_mme_handlers.cpp`:

```
status_code_e s1ap_mme_handle_s1_setup_request(oai::S1apState* state,
                                               const sctp_assoc_id_t assoc_id,
                                               const sctp_stream_id_t stream,
                                               S1ap_S1AP_PDU_t* pdu) {
    // ...
    S1AP_FIND_PROTOCOLIE_BY_ID(S1ap_S1SetupRequestIEs_t, ie_supported_tas,
                           container, S1ap_ProtocolIE_ID_id_SupportedTAs,
                           true);
    // ^ ie assigned null pointer if Global_ENB_ID field not present

    ta_ret =
        s1ap_mme_compare_ta_lists(&ie_supported_tas->value.choice.SupportedTAs);
    // ^ Null pointer dereference of `ie`

    // ...
}
```

CVE-2023-37038 (VULN-D15):

A malformed “Uplink NAS Transport” S1AP packet missing a `MME_UE_S1AP_ID` field will cause Magma/OAI due to a null pointer dereference.

`lte/gateway/c/core/oai/tasks/s1ap/s1ap_mme_nas_procedures.cpp`:

```
status_code_e s1ap_mme_handle_uplink_nas_transport(
        oai::S1apState* state, const sctp_assoc_id_t assoc_id,
        __attribute__((unused)) const sctp_stream_id_t stream,
        S1ap_S1AP_PDU_t* pdu) {
    // ...

    S1AP_FIND_PROTOCOLIE_BY_ID(S1ap_UplinkNASTransport_IEs_t, ie, container,
                            S1ap_ProtocolIE_ID_id_MME_UE_S1AP_ID, true);
    // ^ ie assigned null pointer if MME_UE_S1AP_ID field not present
    mme_ue_s1ap_id = (mme_ue_s1ap_id_t)ie->value.choice.MME_UE_S1AP_ID;
    // ^ Null pointer dereference of `ie`
```

```
    // ...
}
```

CVE-2023-37039 (**VULN-D25**):

Absent mandatory NAS_PDU field in Initial UE Message leads to null dereference.

`lte/gateway/c/core/oai/tasks/s1ap/s1ap_mme_nas_procedures.cpp`:

```
status_code_e s1ap_mme_handle_initial_ue_message(oai::S1apState* state,
                                                 const sctp_assoc_id_t assoc_id,
                                                 const sctp_stream_id_t stream,
                                                 S1ap_S1AP_PDU_t* pdu) {
    // ...

    S1AP_FIND_PROTOCOLIE_BY_ID(S1ap_InitialUEMessage_IEs_t, ie, container,
                               S1ap_ProtocolIE_ID_id_NAS_PDU, true);
    // ^ if this NAS_PDU IE is absent...
    S1AP_FIND_PROTOCOLIE_BY_ID(S1ap_InitialUEMessage_IEs_t, ie_cause, container,
                               S1ap_ProtocolIE_ID_id_RRC_Establishment_Cause,
                               true); if (!ie_cause) { printf("VULN-D25\n"); abort();}
    s1ap_mme_itti_s1ap_initial_ue_message(
        assoc_id, eNB_ref.enb_id(), ue_ref->enb_ue_s1ap_id(),
        ie->value.choice.NAS_PDU.buf, ie->value.choice.NAS_PDU.size, &tai,
        &ecgi, ie_cause->value.choice.RRC_Establishment_Cause,
        ie_e_tmsi ? &s_tmsi : NULL, ie_csg_id ? &csg_id : NULL,
        ie_gummei ? &gummei : NULL,
        NULL,  // CELL ACCESS MODE
        NULL,  // GW Transport Layer Address
        NULL   // Relay Node Indicator
    );
    // ^ Then the ie->value dereferences in here will null derefs

    // ...
}
```

OpenAirInterface (LTE)

GTP Protocol Vulnerabilities

VULN-J01:

openair-spgwu-tiny will abort when a GTP message is received with a TEID field with the value of 4294967295 (or 2^32 - 1).

`src/spgwu/simpleswitch/pfcp_switch.cpp`:

```
bool pfcp_switch::get_pfcp_ul_pdrs_by_up_teid(
    const teid_t teid,
    std::shared_ptr<std::vector<std::shared_ptr<pfcp::pfcp_pdr>>>& pdrs) const {
  folly::AtomicHashMap<
      teid_t, std::shared_ptr<std::vector<std::shared_ptr<pfcp::pfcp_pdr>>>>::
      const_iterator pit = ul_s1u_teid2pfcp_pdr.find(teid);
      // ^ raises abort when teid is equal to 2^32 - 1
  if (pit == ul_s1u_teid2pfcp_pdr.end())
    return false;
  else {
    pdrs = pit->second;
    return true;
  }
}
```

VULN-J02:

openair-spgwu-tiny contains a reachable, uhandled exception when parsing an IP address TLV value in a GTP packet that contains a length other than 4 or 16.

`src/gtpv1u/3gpp_29.281.hpp`:

```
void load_from(std::istream& is) {
    // tlv.load_from(is);
    if (tlv.get_length() == 4) {
        ipv4_address_load_from(is, ipv4_address);
    } else if (tlv.get_length() == 16) {
        ipv6_address_load_from(is, ipv6_address);
    } else {
        throw gtpu_tlv_bad_length_exception(
        GTPU_IE_GTP_U_PEER_ADDRESS, tlv.length);
    }
}
```

Open5GS (LTE)

Affected versions: Open5GS <= 2.6.4

NAS Protocol Vulnerabilities

CVE-2024-24431 (**VULN-F24**):

A reachable assertion in the ogs_nas_emm_decode function of Open5GS v2.7.0 allows attackers to cause a Denial of Service (DoS) via a crafted NAS packet with a zero-length EMM message length.

lib/nas/eps/decoder.c:

```
int ogs_nas_emm_decode(ogs_nas_eps_message_t *message, ogs_pkbuf_t *pkbuf)
{
    int size = 0;
    int decoded = 0;

    ogs_assert(pkbuf);
    ogs_assert(pkbuf->data);
    ogs_assert(pkbuf->len);

    // ...
}
```

CVE-2024-24429 (VULN-F25):

A malformed `Attach Request` NAS packet with a 0-length `ESM Message Container` field will cause Open5GS to crash due to assertion failure.

```
int emm_handle_attach_request(mme_ue_t *mme_ue,
        ogs_nas_eps_attach_request_t *attach_request, ogs_pkbuf_t *pkbuf)
{
    // ...

    ogs_nas_esm_message_container_t *esm_message_container =
                &attach_request->esm_message_container;

    char imsi_bcd[OGS_MAX_IMSI_BCD_LEN+1];

    ogs_assert(mme_ue);
    enb_ue = enb_ue_cycle(mme_ue->enb_ue);
    ogs_assert(enb_ue);

    ogs_assert(esm_message_container);
    ogs_assert(esm_message_container->length);
    // ^ 0-length ESM Message container causes assertion failure

    // ...
}
```

CVE-2024-24430 (VULN-F26):

A malformed `InitialUEMessage` S1AP packet with a 0-length `IMSI` field will cause Open5GS to crash due to assertion failure.

```
src/mme/mme-context.c

mme_ue_t *mme_ue_find_by_imsi(uint8_t *imsi, int imsi_len)
{
    ogs_assert(imsi && imsi_len);

    return (mme_ue_t *)ogs_hash_get(self.imsi_ue_hash, imsi, imsi_len);
}
```

CVE-2024-24432 (VULN-F27):

A malformed NAS packet containing a TAU Request that is a multiple of 256 bytes will cause Open5GS to crash due to assertion failure. This is due to a type cast from `int` to `uint8_t` in `ogs_kdf_hash_mme`.

```
src/mme/emm-handler.c

int emm_handle_tau_request(mme_ue_t *mme_ue,
    ogs_nas_eps_tracking_area_update_request_t *tau_request, ogs_pkbuf_t *pkbuf)
{
    // ...

    /* HashMME */
    ogs_kdf_hash_mme(pkbuf->data, pkbuf->len, mme_ue->hash_mme);
    // ^ if a packet has a length that is multiple of 256 bytes...

    // ...
}

lib/crypt/ogs-kdf.c

void ogs_kdf_hash_mme(uint8_t *message, uint8_t message_len, uint8_t *hash_mme)
{
    uint8_t key[32];
    uint8_t output[OGS_SHA256_DIGEST_SIZE];

    ogs_assert(message);
    ogs_assert(message_len);
    // ^ ...then this cast to uint8_t causes this value to be equal to 0
    ogs_assert(hash_mme);
```

An additional threat from this cast is that only up to the first 255 bytes are hashed using `ogs_kdf_hash_mme`; an attacker could leverage this to pad additional data to the end of a message without the hash changing.

S1AP Protocol Vulnerabilities

CVE-2023-37002 (VULN-F01):

A malformed “E-RAB Modification Indication” S1AP packet missing a required “MME UE S1AP ID” field causes Open5GS to crash as a result of failure to return an error response.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_e_rab_modification_indication(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        r = s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error);
        ogs_expect(r == OGS_OK);
        ogs_assert(r != OGS_ERROR);
        return;
    }

    // ...
}
```

CVE-2023-37003 (VULN-F02):

A malformed “E-RAB Setup Response” S1AP packet missing a required “MME UE S1AP ID” field causes Open5GS to crash as a result of failure to return an error response.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_ue_context_modification_failure(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        r = s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error);
        ogs_expect(r == OGS_OK);
        ogs_assert(r != OGS_ERROR);
        return;
    }

    // ...
}
```

CVE-2023-37004 (VULN-F03):

A malformed “Initial Context Setup Response” S1AP packet missing a required “MME UE S1AP ID” field causes Open5GS to crash as a result of failure to return an error response.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_initial_context_setup_response(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        r = s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error);
        ogs_expect(r == OGS_OK);
        ogs_assert(r != OGS_ERROR);
        return;
    }

    // ...
}
```

CVE-2023-37005 (VULN-F04):

A malformed “Initial Context Setup Failure” S1AP packet missing a required “MME UE S1AP ID” field causes Open5GS to crash as a result of failure to return an error response.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_initial_context_setup_failure(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        r = s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
```

```
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error);
        ogs_expect(r == OGS_OK);
        ogs_assert(r != OGS_ERROR);
        return;
    }

    // ...
}
```

CVE-2023-37006 (VULN-F05):

A malformed “Handover Request Ack” S1AP packet missing a required “MME UE S1AP ID” field causes Open5GS to crash as a result of failure to return an error response.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_handover_request_ack(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        r = s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error);
        ogs_expect(r == OGS_OK);
        ogs_assert(r != OGS_ERROR);
        return;
    }

    // ...
}
```

CVE-2023-37007 (VULN-F06):

A malformed “Handover Cancel” S1AP packet missing a required “MME UE S1AP ID” field causes Open5GS to crash as a result of failure to return an error response.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_handover_cancel(mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        r = s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error);
        ogs_expect(r == OGS_OK);
        ogs_assert(r != OGS_ERROR);
        return;
    }

    // ...
}
```

CVE-2023-37008 (VULN-F07):

A malformed ASN.1 S1AP packet may be used to overwrite a pointer to a struct with data beyond the end of an array (buffer overflow). This can lead to a Denial of Service attack by an attacker repeatedly sending these packets. Additionally, the struct contains function pointers that are subsequently called.

`lib/asn1c/common/constr_CHOICE_aper.c`:

```
value = per_get_few_bits(pd, 1);
// ...

elm = &td->elements[value];
```

`S1AP_Inter-SystemInformationTransferType.c`:

```
static asn_TYPE_member_t asn_MBR_S1AP_Inter_SystemInformationTransferType_1[] = {
    { ATF_POINTER, 0, offsetof(struct S1AP_Inter_SystemInformationTransferType, choice.rIMTransfer),
        (ASN_TAG_CLASS_CONTEXT | (0 << 2)),
        -1,     /* IMPLICIT tag at current level */
        &asn_DEF_S1AP_RIMTransfer,
        0,
        {
#if !defined(ASN_DISABLE_OER_SUPPORT)
            0,
#endif  /* !defined(ASN_DISABLE_OER_SUPPORT) */
#if !defined(ASN_DISABLE_UPER_SUPPORT) || !defined(ASN_DISABLE_APER_SUPPORT)
            0,
#endif  /* !defined(ASN_DISABLE_UPER_SUPPORT) || !defined(ASN_DISABLE_APER_SUPPORT) */
            0
        },
        0, 0, /* No default value */
        "rIMTransfer"
    },
};
```

The above is an array of only one element, yet `CHOICE_decode_aper` may index into element 1 (out of bounds) if certain conditions are met. This indexes into other structured data that may yield a valid pointer type, leading to type confusion of the data.

CVE-2023-37009 (VULN-F08):

A malformed “Handover Notification” S1AP packet missing a required “MME UE S1AP ID” field causes Open5GS to crash as a result of failure to return an error response.

src/mme/s1ap-handler.c:

```
void s1ap_handle_handover_notification(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        r = s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error);
        ogs_expect(r == OGS_OK);
        ogs_assert(r != OGS_ERROR);
        return;
    }

    // ...
}
```

CVE-2023-37010 (VULN-F09):

A malformed “eNB Status Transfer” S1AP packet missing a required “MME UE S1AP ID” field causes Open5GS to crash as a result of failure to return an error response.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_enb_status_transfer(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        r = s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error);
        ogs_expect(r == OGS_OK);
        ogs_assert(r != OGS_ERROR);
        return;
    }

    // ...
}
```

CVE-2023-37011 (VULN-F10):

A malformed “Handover Required” S1AP packet missing a required “MME UE S1AP ID” field causes Open5GS to crash as a result of failure to return an error response.

`src/mme/s1ap-handler.c:1906`:

```
void s1ap_handle_handover_required(mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        r = s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error);
        ogs_expect(r == OGS_OK);
        ogs_assert(r != OGS_ERROR);
        return;
    }

    // ...
}
```

CVE-2023-37012 (VULN-F11):

A malformed “Initial UE Message” S1AP packet missing a required “PLMN Identity” field causes Open5GS to crash on assertion failure.

```
void s1ap_handle_initial_ue_message(mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    pLMNidentity = &EUTRAN_CGI->pLMNidentity;
    ogs_assert(pLMNidentity && pLMNidentity->size == sizeof(ogs_plmn_id_t));

    // ...
}
```

CVE-2023-37013 (VULN-F12):

A specially-crafted oversized message will cause the Open5GS `s1ap_recv_handler` routine to crash due to assertion failure.

src/mme/s1ap-sctp.c:243:

```
void s1ap_recv_handler(ogs_sock_t *sock)
{
    // ...

    if (ogs_socket_errno != OGS_EAGAIN) {
        ogs_fatal("ogs_sctp_recvmsg(%d) failed(%d:%s-0x%x)",
                size, errno, strerror(errno), flags);
        ogs_assert_if_reached();
    } else {
        ogs_error("ogs_sctp_recvmsg(%d) failed(%d:%s-0x%x)",
                size, errno, strerror(errno), flags);
    }

    // ...
}
```

CVE-2023-37014 (VULN-F13):

A malformed “UE Context Release Request” S1AP packet containing an invalid `MME-UE-S1AP-ID` field and missing an `ENB-UE-S1AP-ID` causes Open5GS to crash as a result of failure to return an error response.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_ue_context_release_request(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    enb_ue = enb_ue_find_by_mme_ue_s1ap_id(*MME_UE_S1AP_ID);
    if (!enb_ue) {
        ogs_warn("No ENB UE Context : MME_UE_S1AP_ID[%d]",
                (int)*MME_UE_S1AP_ID);
        r = s1ap_send_error_indication(enb,
                MME_UE_S1AP_ID, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_radioNetwork,
                S1AP_CauseRadioNetwork_unknown_mme_ue_s1ap_id);
        ogs_expect(r == OGS_OK);
        ogs_assert(r != OGS_ERROR);
        return;
    }

    // ...
}
```

CVE-2023-37015 (VULN-F14):

A malformed “Path Switch Request” S1AP packet missing a required “MME UE S1AP ID” field causes Open5GS to crash as a result of failure to return an error response.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_path_switch_request(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        r = s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error);
        ogs_expect(r == OGS_OK);
        ogs_assert(r != OGS_ERROR);
        return;
    }

    // ...
}
```

CVE-2023-37016 (VULN-F15):

A malformed “UE Context Modification Response” S1AP packet missing a required “MME UE S1AP ID” field causes Open5GS to crash as a result of failure to return an error response.

`src/mme/s1ap-handler.c:981`:

```
void s1ap_handle_ue_context_modification_response(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        r = s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error);
        ogs_expect(r == OGS_OK);
        ogs_assert(r != OGS_ERROR);
        return;
```

```
    }

    // ...
}
```

CVE-2023-37017 (VULN-F16):

A malformed `S1Setup` S1AP packet missing a `Global-ENB-ID` field will cause Open5GS to crash due to assertion failure.

```
void s1ap_handle_s1_setup_request(mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    ogs_debug("S1SetupRequest");

    for (i = 0; i < S1SetupRequest->protocolIEs.list.count; i++) {
        ie = S1SetupRequest->protocolIEs.list.array[i];
        switch (ie->id) {
        case S1AP_ProtocolIE_ID_id_Global_ENB_ID:
            Global_ENB_ID = &ie->value.choice.Global_ENB_ID;
            break;
        case S1AP_ProtocolIE_ID_id_SupportedTAs:
            SupportedTAs = &ie->value.choice.SupportedTAs;
            break;
        case S1AP_ProtocolIE_ID_id_DefaultPagingDRX:
            PagingDRX = &ie->value.choice.PagingDRX;
            break;
        default:
            break;
        }
    }

    ogs_assert(Global_ENB_ID);

    // ...
}
```

CVE-2023-37018 (VULN-F17):

A malformed “UE Capability Info Indication” S1AP packet containing an invalid “ENB UE S1AP ID” field causes Open5GS to crash as a result of failure to return an error response.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_ue_capability_info_indication(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!ENB_UE_S1AP_ID) {
        ogs_error("No ENB_UE_S1AP_ID");
        ogs_assert(OGS_OK ==
            s1ap_send_error_indication(enb, MME_UE_S1AP_ID, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error));
        return;
    }

    // ...
}
```

CVE-2023-37019 (VULN-F18):

A malformed packet missing the “Supported TAs” S1AP field will cause Open5GS to crash due to assertion failure.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_s1_setup_request(mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    ogs_debug("S1SetupRequest");

    for (i = 0; i < S1SetupRequest->protocolIEs.list.count; i++) {
        ie = S1SetupRequest->protocolIEs.list.array[i];
        switch (ie->id) {
        case S1AP_ProtocolIE_ID_id_Global_ENB_ID:
            Global_ENB_ID = &ie->value.choice.Global_ENB_ID;
            break;
        case S1AP_ProtocolIE_ID_id_SupportedTAs:
            SupportedTAs = &ie->value.choice.SupportedTAs;
            break;
        case S1AP_ProtocolIE_ID_id_DefaultPagingDRX:
            PagingDRX = &ie->value.choice.PagingDRX;
            break;
        default:
            break;
        }
    }

    ogs_assert(Global_ENB_ID);

    ogs_s1ap_ENB_ID_to_uint32(&Global_ENB_ID->eNB_ID, &enb_id);
```

```
    ogs_debug("    IP[%s] ENB_ID[%d]", OGS_ADDR(enb->sctp.addr, buf), enb_id);

    if (PagingDRX)
        ogs_debug("    PagingDRX[%ld]", *PagingDRX);

    mme_enb_set_enb_id(enb, enb_id);

    ogs_assert(SupportedTAs);

    // ...
}
```

CVE-2023-37020 (VULN-F19):

A malformed “UE Context Release Complete” S1AP packet missing an “MME UE S1AP ID” field will cause Open5GS to crash due as a result of failure to return an error response.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_ue_context_release_complete(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        ogs_assert(OGS_OK ==
            s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error));
        return;
    }

    // ...
}
```

CVE-2023-37021 (VULN-F20):

A malformed “UE Context Modification Failure” S1AP packet missing a required “MME UE S1AP ID” field causes Open5GS to crash as a result of failure to return an error response.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_ue_context_modification_failure(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        ogs_assert(OGS_OK ==
            s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error));
        return;
    }

    // ...
}
```

CVE-2023-37022 (VULN-F21):

A malformed “UE Context Release Request” S1AP packet missing a required “MME UE S1AP ID” field causes Open5GS to crash as a result of failure to return an error response.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_ue_context_release_request(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        r = s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error);
        ogs_expect(r == OGS_OK);
        ogs_assert(r != OGS_ERROR);
        return;
    }

    // ...
}
```

CVE-2023-37023 (VULN-F22):

A malformed “Uplink NAS Transport” S1AP packet containing an invalid “ENB UE S1AP ID” field will cause Open5GS to crash due as a result of failure to return an error response.

`src/mme/s1ap-handler.c`:

```
void s1ap_handle_uplink_nas_transport(
        mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!MME_UE_S1AP_ID) {
        ogs_error("No MME_UE_S1AP_ID");
        r = s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error);
        ogs_expect(r == OGS_OK);
        ogs_assert(r != OGS_ERROR);
        return;
    }

    // ...
}
```

CVE-2024-34235 (VULN-F23):

A malformed “Initial UE Message” S1AP packet missing a required “NAS PDU” field causes Open5GS to crash on assertion failure.

```
void s1ap_handle_initial_ue_message(mme_enb_t *enb, ogs_s1ap_message_t *message)
{
    // ...

    if (!NAS_PDU) {
        ogs_error("No NAS_PDU");
        ogs_assert(OGS_OK ==
            s1ap_send_error_indication(enb, NULL, ENB_UE_S1AP_ID,
                S1AP_Cause_PR_protocol, S1AP_CauseProtocol_semantic_error));
        return;
    }

    // ...
}
```

GTP Protocol Vulnerabilities

VULN-F28:

A reachable assertion in Open5GS SGW enables an attacker that has established a PFCP session to crash the server.

```
lib/pfcp/path.c

void ogs_pfcp_send_g_pdu(
        ogs_pfcp_pdr_t *pdr, uint8_t type, ogs_pkbuf_t *sendbuf)
{
ogs_gtp_node_t *gnode = NULL;
    ogs_pfcp_far_t *far = NULL;

    ogs_gtp2_header_t gtp_hdesc;
    ogs_gtp2_extension_header_t ext_hdesc;

    ogs_assert(pdr);
    ogs_assert(type);
    ogs_assert(sendbuf);

    far = pdr->far;
    if (!far) {
        ogs_error("No FAR");
        ogs_pkbuf_free(sendbuf);
        return;
    }

    if (far->dst_if == OGS_PFCP_INTERFACE_UNKNOWN) {
        ogs_error("No Destination Interface");
        ogs_pkbuf_free(sendbuf);
        return;
    }

    // ...
}
```

VULN-F29:

A reachable assertion in Open5GS SGW enables an attacker that has established a PFCP session to crash the server.

```
src/sgwu/gpt-path.c

static void _gtpv1_u_recv_cb(short when, ogs_socket_t fd, void *data)
{
    // ...

    if (report.type.error_indication_report) {
        ogs_assert(far->sess);
        sess = SGWU_SESS(far->sess);
        ogs_assert(sess);


        ogs_assert(OGS_OK ==
            sgwu_pfcp_send_session_report_request(sess, &report));
```

```
        // ^ when report send fials, assertion is triggered
    }

    // ...
}
```

NextEPC (LTE)

Affected versions: NextEPC <= 1.0.1 (fixed in commit a8492c9c5bc0a66c6999cb5a263545b32a4109df)

NAS Protocol Vulnerabilities

CVE-2023-36998 (**VULN-E08**):

An S1AP packet containing a NAS message with an oversized Emergency Number List will cause Magma/OAI to crash due to a write buffer overflow.

`mme/lib/nas/nas_ies.c`:

```
c_int16_t nas_decode_emergency_number_list(nas_emergency_number_list_t *emergency_number_list, pkbuf_t *pkbuf)
{
    c_uint16_t size = 0;
    nas_emergency_number_list_t *source = pkbuf->payload;

    emergency_number_list->length = source->length;
    size = emergency_number_list->length + sizeof(emergency_number_list->length);

    d_assert(pkbuf_header(pkbuf, -size) == CORE_OK, return -1, "pkbuf_header error");
    memcpy(emergency_number_list, pkbuf->payload - size, size);
    // ^ Received input may exceed emergency number list type, lead to buffer overflow

    // ...
}
```

CVE-2023-36999 (**VULN-E09**):

A malformed NAS packet IMSI field will cause NextEPC to crash due to a buffer overflow write.

`src/mme/nas_conv.c`:

```
void nas_imsi_to_bcd(
    nas_mobile_identity_imsi_t *imsi, c_uint8_t imsi_len, c_int8_t *bcd)
{
    // ...

    bcd_len = imsi_len * 2 - 1;
    // ^ imsi_len is determined by NAS packet contents
    if (!imsi->odd_even) /* if bcd length is even */
    {
        if (bcd[bcd_len] != 0xf)
            d_warn("Spec warning : bcd[%d] = 0x%x", bcd_len, bcd[bcd_len]);
        (bcd_len)--;
    }

    bcd[bcd_len] = 0;
    // ^ Overwrites an arbitrarily-indexed byte with 0, and additionally makes bcd
    // not null-terminated, leading to read out-of-bounds
}
```

S1AP Protocol Vulnerabilities

CVE-2023-36997 (**VULN-E01**):

A malformed `Initial Context Setup Request` S1AP packet will cause NextEPC to crash due to an invalid read.

Note that this is not a null free; this is freeing uninitialized/out of range memory, and it is fundamentally caused by an off-by-one error between two statically-allocated arrays in asn1c.

Erroneous arrays located at:

- `lib/s1ap/asn1c/S1AP_ProtocolIE-Field.c:1386`: `asn_IOS_S1AP_InitialContextSetupRequestIEs_1_rows[]`
- `lib/s1ap/asn1c/S1AP_ProtocolIE-Field.c:25799`: `asn_MBR_S1AP_value_152[]`

CVE-2023-37000 (**VULN-E02**):

A malformed `Handover Command` S1AP packet will cause NextEPC to crash due to a buffer overflow due to an off-by-one error in statically-allocated arrays in asn1c.

The value of `elm->type->elements`, `asn_MBR_S1AP_value_80`, has only 8 elements. Meanwhile, the value of `asn_IOS_S1AP_HandoverCommandIEs_1_rows` has 9*4 elements. The first is missing `asn_VAL_22_S1AP_id_Target_ToSource_TransparentContainer_Secondary`, an optional secondary value of the same type as its immediately preceding element.

`OPEN_TYPE.c:440`:

```
inner_value =
    (char *)*memb_ptr2
```

```
    + elm->type->elements[selected.presence_index - 1].memb_offset;
    // ^ off-by-one buffer overflow occurs when selecting element type here
```

CVE-2024-24438 (VULN-E03):

A malformed `Handover Required` S1AP packet will cause NextEPC to crash due to a buffer overflow write.

`S1AP_ProtocolIE-Field.c:326`:

```
static const asn_ioc_set_t asn_IOS_S1AP_HandoverRequiredIEs_1[] = {
        { 14, 4, asn_IOS_S1AP_HandoverRequiredIEs_1_rows }
};
```

`S1AP_ProtocolIE-Field.c:22814`:

```
asn_TYPE_descriptor_t asn_DEF_S1AP_value_76 = {
        "value",
        "value",
        &asn_OP_OPEN_TYPE,
        0,      /* No effective tags (pointer) */
        0,      /* No effective tags (count) */
        0,      /* No tags (pointer) */
        0,      /* No tags (count) */
        { 0, 0, OPEN_TYPE_constraint },
        asn_MBR_S1AP_value_76,
        13,     /* Elements count */ // BUG: should be 14 elements
        &asn_SPC_S1AP_value_specs_76    /* Additional specs */
};
```

`asn_IOS_S1AP_HandoverRequiredIEs_1_rows` contains 14 fields, but its corresponding value struct `asn_MBR_S1AP_value_76` contains only 13 due to an error in asn1c compilation. This leads to type confusion in addition to an off-by-one buffer overflow, as the missing type shifts the types of the remaining indices in the struct.

CVE-2024-24439 (VULN-E04):

`OPEN_TYPE.c:440`:

```
inner_value =
    (char *)*memb_ptr2
    + elm->type->elements[selected.presence_index - 1].memb_offset; // Buffer overflow
```

`S1AP_ProtocolIE-Field.c:614`

```
static const asn_ioc_set_t asn_IOS_S1AP_HandoverRequestIEs_1[] = {
        { 26, 4, asn_IOS_S1AP_HandoverRequestIEs_1_rows }
};
```

`S1AP_ProtocolIE-Field.c:23479`

```
static /* Use -fall-defs-global to expose */
asn_TYPE_descriptor_t asn_DEF_S1AP_value_88 = {
        "value",
        "value",
        &asn_OP_OPEN_TYPE,
        0,      /* No effective tags (pointer) */
        0,      /* No effective tags (count) */
        0,      /* No tags (pointer) */
        0,      /* No tags (count) */
        { 0, 0, OPEN_TYPE_constraint },
        asn_MBR_S1AP_value_88,
        25,     /* Elements count */ // BUG: should have 26 elements, not 25
        &asn_SPC_S1AP_value_specs_88    /* Additional specs */
};
```

The value of `elm->type->elements`, `asn_MBR_S1AP_value_88`, has only 25 elements. Meanwhile, the value of `asn_IOS_S1AP_HandoverRequestIEs_1_rows` has 26*4 elements. The first is missing `MME-UE-S1AP-ID-2`, an optional secondary value of the same type as its immediately preceding element.

CVE-2024-24440 (VULN-E05):

A malformed `Bearers-SubjectToStatusTransfer` IE within an S1AP packet will cause NextEPC to crash due to a buffer overflow write.

`OPEN_TYPE.c:440`:

```
inner_value =
    (char *)*memb_ptr2
    + elm->type->elements[selected.presence_index - 1].memb_offset; // Buffer overflow
```

`lib/s1ap/asn1c/S1AP_ProtocolExtensionField.c`

```
static const asn_ioc_set_t asn_IOS_S1AP_Bearers_SubjectToStatusTransfer_ItemExtIEs_1[] = {
        { 6, 4, asn_IOS_S1AP_Bearers_SubjectToStatusTransfer_ItemExtIEs_1_rows }
};
```

`lib/s1ap/asn1c/S1AP_ProtocolExtensionField.c`

```
static /* Use -fall-defs-global to expose */
asn_TYPE_descriptor_t asn_DEF_S1AP_extensionValue_104 = {
        "extensionValue",
        "extensionValue",
```

```
        &asn_OP_OPEN_TYPE,
        0,      /* No effective tags (pointer) */
        0,      /* No effective tags (count) */
        0,      /* No tags (pointer) */
        0,      /* No tags (count) */
        { 0, 0, OPEN_TYPE_constraint },
        asn_MBR_S1AP_extensionValue_104,
        4,      /* Elements count */
        &asn_SPC_S1AP_extensionValue_specs_104  /* Additional specs */
};
```

The value of `elm->type->elements`, `asn_MBR_S1AP_extensionValue_104`, has only 4 elements. Meanwhile, the value of `asn_IOS_S1AP_Bearers_SubjectToStatusTransfer_ItemExtIEs_1_rows` has 6*4 elements. The first is missing `DLCOUNTValueExtended` and `DLCOUNTvaluePDCP-SNlength18`, both optional secondary values of the same type as their immediately preceding element.

CVE-2024-24441 (**VULN-E06**):

A certain `E_RAB Modification Confirmation` Information Element within an S1AP packet will cause NextEPC to crash due to a buffer overflow write.

`OPEN_TYPE.c`:

```
inner_value =
    (char *)*memb_ptr2
    + elm->type->elements[selected.presence_index - 1].memb_offset; // Buffer overflow
```

`S1AP_ProtocolIE-Field.c`:

```
static const asn_ioc_set_t asn_IOS_S1AP_E_RABModificationConfirmIEs_1[] = {
        { 7, 4, asn_IOS_S1AP_E_RABModificationConfirmIEs_1_rows }
};
```

`S1AP_ProtocolIE-Field.c`:

```
static /* Use -fall-defs-global to expose */
asn_TYPE_descriptor_t asn_DEF_S1AP_value_388 = {
        "value",
        "value",
        &asn_OP_OPEN_TYPE,
        0,      /* No effective tags (pointer) */
        0,      /* No effective tags (count) */
        0,      /* No tags (pointer) */
        0,      /* No tags (count) */
        { 0, 0, OPEN_TYPE_constraint },
        asn_MBR_S1AP_value_388,
        6,      /* Elements count */
        &asn_SPC_S1AP_value_specs_388   /* Additional specs */
};
```

The value of `elm->type->elements`, `asn_MBR_S1AP_value_388`, has only 6 elements. Meanwhile, the value of `asn_IOS_S1AP_E_RABModificationConfirmIEs_1_rows` has 7*4 elements. The first is missing `E_RABList`, an optional secondary value of the same type as its immediately preceding element.

CVE-2024-24437 (**VULN-E07**):

The presence of a `Path Switch Request Acknowledge` Information Element within an S1AP packet will cause NextEPC to crash due to a buffer overflow write.

`OPEN_TYPE.c:440`:

```
inner_value =
    (char *)*memb_ptr2
    + elm->type->elements[selected.presence_index - 1].memb_offset; // Buffer overflow
```

`S1AP_ProtocolIE-Field.c:137`

```
static const asn_ioc_set_t asn_IOS_S1AP_PathSwitchRequestAcknowledgeIEs_1[] = {
        { 14, 4, asn_IOS_S1AP_PathSwitchRequestAcknowledgeIEs_1_rows }
};
```

`S1AP_ProtocolIE-Field.c:34156`

```
asn_TYPE_descriptor_t asn_DEF_S1AP_value_108 = {
        "value",
        "value",
        &asn_OP_OPEN_TYPE,
        0,      /* No effective tags (pointer) */
        0,      /* No effective tags (count) */
        0,      /* No tags (pointer) */
        0,      /* No tags (count) */
        { 0, 0, OPEN_TYPE_constraint },
        asn_MBR_S1AP_value_108,
        13,     /* Elements count */
        &asn_SPC_S1AP_value_specs_108   /* Additional specs */
};
```

The value of `elm->type->elements`, `asn_MBR_S1AP_value_108`, has only 13 elements. Meanwhile, the value of `asn_IOS_S1AP_PathSwitchRequestAcknowledgeIEs_1_rows` has 14*4 elements. The first is missing `MME-UE-S1AP-ID-2`, an optional secondary value of the same type as its immediately preceding element.

SD-Core (LTE)

Affected versions: SD-Core <= 1.4.0 (fixed in Nucleus commit 8c0455035837ad97d5a55bd214d7ab7d35aebdf8)

NAS Protocol Vulnerabilities

CVE-2023-37043 (**VULN-H08**):

An `Uplink NAS Transport` S1AP packet containing an oversized NAS_PDU will cause Nucleus to crash due to a stack-based buffer overflow.

`src/s1ap/handlers/s1ap_msg_delegator.c`:

```
int convertUplinkNasToProtoIe(SuccessfulOutcome_t *msg, struct proto_IE* proto_ies)
{
    // ...

    switch(ie_p->id) {
        case ProtocolIE_ID_id_NAS_PDU:
        {
            NAS_PDU_t *s1apNASPDU_p = NULL;
            if(UplinkNASTransport_IEs__value_PR_NAS_PDU == ie_p->value.present)
            {
                s1apNASPDU_p = &ie_p->value.choice.NAS_PDU;
            }
            else
            {
                log_msg (LOG_ERROR, "Decoding of IE NAS PDU failed");
                return -1;
            }

            proto_ies->data[i].IE_type = S1AP_IE_NAS_PDU;
            memcpy(s1Msg->nasMsg.nasMsgBuf, (char*)s1apNASPDU_p->buf, s1apNASPDU_p->size);
            // ^ Copies potentially oversized packet buffer into static-sized local buffer
            s1Msg->nasMsg.nasMsgSize = s1apNASPDU_p->size;

        } break;

        // ...
    }

    // ...
}
```

CVE-2023-37044 (**VULN-H09**):

An `Initial UE Message` S1AP packet containing an oversized NAS_PDU will cause Nucleus to crash due to a stack-based buffer overflow.

`./s1ap/handlers/s1ap_msg_delegator.c`:

```
int convertInitCtxRspToProtoIe(SuccessfulOutcome_t *msg, struct proto_IE* proto_ies)
{
    // ...

    switch(ie_p->id) {
        case ProtocolIE_ID_id_NAS_PDU:
        {
            NAS_PDU_t *s1apNASPDU_p = NULL;
            if(InitialUEMessage_IEs__value_PR_NAS_PDU == ie_p->value.present)
            {
                s1apNASPDU_p = &ie_p->value.choice.NAS_PDU;
            }
            else
            {
                log_msg (LOG_ERROR, "Decoding of IE NAS PDU failed");
                return -1;
            }

            proto_ies->data[i].IE_type = S1AP_IE_NAS_PDU;
            memcpy(s1Msg->nasMsg.nasMsgBuf, (char*)s1apNASPDU_p->buf, s1apNASPDU_p->size);
            // ^ oversized NAS_PDU can overflow nasMsgBuf fixed-size bytearray
            s1Msg->nasMsg.nasMsgSize = s1apNASPDU_p->size;
            s1Msg->nasMsg.nasMsgSize = s1apNASPDU_p->size;
        } break;

        // ...
    }

    // ...
}
```

S1AP Protocol Vulnerabilities

CVE-2023-37040 (**VULN-H01**):

A malformed `Initial Context Setup Response` S1AP packet will cause Nucleus to crash due to a heap-based buffer overflow.

`vim include/common/s1ap_structs.h`:

```
struct eRAB_setup_ctx_SU {
    unsigned short eRAB_id;
    unsigned short dont_know_byte;
    unsigned int transp_layer_addr;
    // ^ transp_layer_addr has a fixed size of 4 bytes
    unsigned int gtp_teid;
};

src/s1ap/handlers/s1ap_msg_delegator.c:

int convertInitCtxRspToProtoIe(SuccessfulOutcome_t *msg, struct proto_IE* proto_ies)
{
    // ...

    if(s1apErabSetupItem_p->transportLayerAddress.buf != NULL) {
        memcpy(
            &(proto_ies->data[i].val.erab.elements[j].su_res.transp_layer_addr),
            s1apErabSetupItem_p->transportLayerAddress.buf,
            s1apErabSetupItem_p->transportLayerAddress.size);
        proto_ies->data[i].val.erab.elements[j].su_res.transp_layer_addr
            = ntohl(proto_ies->data[i].val.erab.elements[j].su_res.transp_layer_addr);
        // ^ transportLayerAddress size bound not checked; out-of-bounds write
    }

    // ...
}
```

CVE-2023-37042 (VULN-H02):

An off-by-one error in initializing memory pools leads to memory corruption when certain memory is allocated in the SD-Core Nucleus MME.

```
include/cmn/memPoolManager.h:

template <typename T>
class MemChunk
{
public:

    MemChunk(uint32_t numOfBlocks)
    {
        uint32_t  i = 1;
        for (; i < numOfBlocks; i++)
        // ^ should be i < (numOfBlocks - 1)...
        {
            blockArray_mpa[i-1].setNextMemBlock(&blockArray_mpa[i]);
        }
        blockArray_mpa[i].setNextMemBlock(NULL);
        // ^ otherwise this will index one beyond the end of the array

        // ...
    }

// ...
}
```

CVE-2024-24436 (VULN-H03):

A `Handover Ack` S1AP packet containing an oversized TargetToSource TransparentContainer will cause Nucleus to crash due to a stack-based buffer overflow.

```
int s1_handover_ack_handler(SuccessfulOutcome_t *msg)
{
    // ...

    switch (s1_ho_ack_ies.data[i].IE_type) {
        case S1AP_IE_TARGET_TOSOURCE_TRANSPARENTCONTAINER:
        {
            log_msg(LOG_INFO,
                    "Handover Request Ack S1AP_IE_TARGET_TOSOURCE_TRANSPARENTCONTAINER.");

            handover_ack.targetToSrcTranspContainer.count =
                    s1_ho_ack_ies.data[i].val.targetToSrcTranspContainer.size;

            memcpy(
                    handover_ack.targetToSrcTranspContainer.buffer,
                    s1_ho_ack_ies.data[i].val.targetToSrcTranspContainer.buffer_p,
                    s1_ho_ack_ies.data[i].val.targetToSrcTranspContainer.size);
            // ^ unchecked length of transportContainer overruns struct being copied into
        }
            break;
        // ...
    }

    // ...
}
```

CVE-2024-24435 (VULN-H04):

A malformed `Handover Ack` S1AP packet containing an oversized `TransportLayerAddress` will cause Nucleus to crash due to a stack-based buffer overflow.

```
include/common/s1ap_structs.h:
```

```
typedef struct ERAB_admitted{
        uint8_t e_RAB_ID;
        uint32_t transportLayerAddress;
        uint32_t gtp_teid;
        uint32_t dL_transportLayerAddress;
        // ^ must be exactly 4 bytes in size
        uint32_t dL_gtp_teid;

}ERAB_admitted;
```

src/s1ap/handlers/s1ap_msg_delegator.c:

```
int convertHoAcklToProtoIe(SuccessfulOutcome_t *msg, struct proto_IE *proto_ies)
{
    // ...

    memcpy(
        &(proto_ies->data[i].val.erab_admittedlist.erab_admitted[0].dL_transportLayerAddress),
        eRabAdmittedItem_p->dL_transportLayerAddress->buf,
        eRabAdmittedItem_p->dL_transportLayerAddress->size);
    // ^ unchecked length overflows `uint32_t` field being copied into

    // ...
}
```

CVE-2024-24433 (**VULN-H05**):

A malformed `Handover Required` S1AP packet containing an oversized SourceToTarget TransparentContainer will cause Nucleus to crash due to a stack-based buffer overflow.

src/s1ap/handlers/handover_required.c:

```
int s1_handover_required_handler(InitiatingMessage_t *msg, int enb_fd)
{
    // ...

    switch (ho_required_ies.data[i].IE_type)
    {
        case S1AP_IE_SOURCE_TOTARGET_TRANSPARENTCONTAINER:
        {
            log_msg(LOG_INFO,
                    "handover required S1AP_IE_SOURCE_TOTARGET_TRANSPARENTCONTAINER.");

            ho_required.srcToTargetTranspContainer.count =
                    ho_required_ies.data[i].val.srcToTargetTranspContainer.size;

            memcpy(
                    ho_required.srcToTargetTranspContainer.buffer,
                    ho_required_ies.data[i].val.srcToTargetTranspContainer.buffer_p,
                    ho_required_ies.data[i].val.srcToTargetTranspContainer.size);
            // ^ unchecked length overruns struct being copied into
        }
            break;
        // ...
    }

    // ...
}
```

CVE-2024-24434 (**VULN-H06**):

A malformed `Handover Ack` S1AP packet containing an oversized GTP_TEID field will cause Nucleus to crash due to a stack-based buffer overflow.

include/common/s1ap_structs.h:

```
struct eRAB_setup_ctx_SU {
    unsigned short eRAB_id;
    unsigned short dont_know_byte;
    unsigned int transp_layer_addr;
    unsigned int gtp_teid;
    // ^ GTP_TEID should be exactly 4 bytes
};
```

src/s1ap/handlers/s1ap_msg_delegator.c:

```
int convertHoAcklToProtoIe(SuccessfulOutcome_t *msg, struct proto_IE *proto_ies)
{
    // ...

    proto_ies->data[i].val.erab_admittedlist.erab_admitted[0].e_RAB_ID =
                                (unsigned short) eRabAdmittedItem_p->e_RAB_ID;
    memcpy(
            &(proto_ies->data[i].val.erab_admittedlist.erab_admitted[0].gtp_teid),
            eRabAdmittedItem_p->gTP_TEID.buf,
            eRabAdmittedItem_p->gTP_TEID.size);
    // ^ unchecked length overruns `unsigned int` being copied into

    // ...
}
```

CVE-2023-37041 (**VULN-H07**):

A malformed `S1Setup Request` S1AP packet will cause Nucleus to crash due to memory corruption. The memory corruption happens during ASN.1 parsing and is manifest once structures are freed.

The precise cause could not be ascertained given that the ASN.1 encoding/decoding routines are precompiled as a static library.

Base64-encoded crashing examples:

- ABECAwAAABECAwAAAgIAVANkABECAwAAABECAwAAAgIAVANkAAACAG4AEQAAAgBUAAACRnUCgA== (causes crash during ASN_STRUCT_FREE_CONTENTS_ONLY)
- ABEALQAABAA7AAgAAPEQAAAZsAA8QAoDgHNyc2VuYjAxAEAfBwAAAcAD6BAAiUABQA== (causes crash during `free(val);`)

Athonet (LTE)

Note that source code is not available for Athonet, so we were not able to pin down the root programmatic cause of vulnerabilities.

Affected versions: Athonet vEPC MME <= v11.4.0

NAS Protocol Vulnerabilities

CVE-2024-24456 (**VULN-I08**):

An `E-RAB Release Command` packet containing a malformed `NAS PDU` will cause the Athonet MME to immediately crash, potentially due to a buffer overflow.

Samples:

- `AAdAgKEAAAoACEAEgLJ20AAAQAXAR8TVVwAAQAXA00ByNQAAQAXApwf9zwBC QAwgAc0/FPSAAWMm7QQAIUA/CQA+QAIcZgAjQAMOCEAAI0ACGogAI0ACHEAA I0ACACAAI0ACGIYAI0ACFiAAI0ACDIYAI0ADABCAACNAAghmAAhABIDA+5sA GkADAriXAABABcDZHHLJAEJADCAB5Y1Qz4ACFVKEiw==`

Counterexamples:

- `AAdAgKEAAAoACEAEgLJ20AAAQAXAR8TVVwAAQAXA00ByNQAAQAXApwf9zwBC QAwgAc0/FPSAAWMm7QQAIUA/CQAjQAIcZgAjQAMOCEAAI0ACGogAI0ACHEAA I0ACACAAI0ACGIYAI0ACFiAAI0ACDIYAI0ADAAGAACNAAghmAAhABIDA+5sA GkADAriXAABABcDZHHLJAEJADCAB5Y1Qz4ACFVKEiw==`

(Counterexample modifies the NAS field, does not lead to any resulting crash)

S1AP Protocol Vulnerabilities

CVE-2024-24454 (**VULN-I01**):

An invalid memory access when handling the ProtocolIE_ID field of E-RAB Modify Request messages in Athonet vEPC MME v11.4.0 allows attackers to cause a Denial of Service (DoS) to the cellular network by repeatedly initiating connections and sending a crafted payload.

CVE-2024-24459 (**VULN-I02**): An invalid memory access when handling the ProtocolIE_ID field of S1Setup Request messages in Athonet vEPC MME v11.4.0 allows attackers to cause a Denial of Service (DoS) to the cellular network by repeatedly initiating connections and sending a crafted payload.

CVE-2024-24455 (**VULN-I03**):

An invalid memory access when handling a UE Context Release message containing an invalid UE identifier in Athonet vEPC MME v11.4.0 allows attackers to cause a Denial of Service (DoS) to the cellular network by repeatedly initiating connections and sending a crafted payload.

CVE-2024-24457 (**VULN-I04**):

An invalid memory access when handling the ProtocolIE_ID field of E-RAB Setup List Context SURes messages in Athonet vEPC MME v11.4.0 allows attackers to cause a Denial of Service (DoS) to the cellular network by repeatedly initiating connections and sending a crafted payload.

CVE-2024-24452 (**VULN-I05**):

An invalid memory access when handling the ProtocolIE_ID field of E-RAB Release Indication messages in Athonet vEPC MME v11.4.0 allows attackers to cause a Denial of Service (DoS) to the cellular network by repeatedly initiating connections and sending a crafted payload.

CVE-2024-24453 (**VULN-I06**):

An invalid memory access when handling the ProtocolIE_ID field of E-RAB NotToBeModifiedBearerModInd information element in Athonet vEPC MME v11.4.0 allows attackers to cause a Denial of Service (DoS) to the cellular network by repeatedly initiating connections and sending a crafted payload.

CVE-2024-24458 (**VULN-I07**):

An invalid memory access when handling the ENB Configuration Transfer messages containing invalid PLMN Identities in Athonet vEPC MME v11.4.0 allows attackers to cause a Denial of Service (DoS) to the cellular network by repeatedly initiating connections and sending a crafted payload.

srsRAN (LTE)

Affected versions: srsRAN Project <= 23.5 (fixed in 23.10)

S1AP Protocol Vulnerabilities

CVE-2023-37001 (**VULN-G01**):

An ASN.1 parsing vulnerability was found in the srsRAN 4G EPC, where bounds constraints on certain integer types were not enforced.

`src/asn1/asn1_utils.cc`

```
template <class IntType>
SRSASN_CODE unpack_constrained_whole_number(IntType& n, cbit_ref& bref, IntType lb, IntType ub, bool aligned)
{
  if (ub < lb) {
    log_error("The condition lb <= ub ({} <= {}) was not met", (long)lb, (long)ub);
    return SRSASN_ERROR_DECODE_FAIL;
  }
  uint64_t ra = (uint64_t)(ub - lb) + 1; // NOTE: Can overflow if IntType is kept.
  if (ra == 1) {
    n = lb;
    return SRSASN_SUCCESS;
  }
  uint32_t n_bits = (uint32_t)ceilf(log2f((float)ra));
  if (not aligned) {
    // UNALIGNED variant
    HANDLE_CODE(bref.unpack(n, n_bits));
    n += lb;
    if (n > ub) {
      // ^ This check ensures that the number cannot exceed bound constraints...
      log_error("The condition lb <= n <= ub ({} <= {} <= {}) was not met", (long)lb, (long)n, (long)ub);
      return SRSASN_ERROR_DECODE_FAIL;
    }
  } else {
    // ALIGNED variant
    if (ra < 256) {
      HANDLE_CODE(bref.unpack(n, n_bits));
      // ^ ...but packed rules fail to check to ensure n <= ub.
    } else if (ra <= ASN_64K) {
      uint32_t n_octets = ceil_frac(n_bits, 8u);
      HANDLE_CODE(bref.align_bytes());
      HANDLE_CODE(bref.unpack(n, n_octets * 8));
      HANDLE_CODE(bref.align_bytes());
    } else {
      uint32_t n_bits_len = (uint32_t)ceilf(log2f(ceil_frac(n_bits, 8u)));
      uint32_t n_octets;
      HANDLE_CODE(bref.unpack(n_octets, n_bits_len));
      n_octets += 1;
      HANDLE_CODE(bref.align_bytes());
      HANDLE_CODE(bref.unpack(n, n_octets * 8));
    }
    n += lb;
  }

  // (n > ub) check should be moved here to account for both unaligned and aligned bounds checks.
  return SRSASN_SUCCESS;
}
```

As a direct consequence of this bug, the unpack_integer and unpack_length methods may return a number greater than the upper bound (up to the next power of 2). These in turn affect unpack_dyn_seq_of, which fills a fixed-sized buffer with data based on the length returned by unpack_length. A length value exceeding the upper bound restriction will cause a buffer overflow and write data from the packet into other fields of the dynamic sequence struct (including its length field). Once this is done, an arbitrary-length buffer overflow can occur from the overwritten length field. This is observed in the S1AP protocol specifically (in its SupportedTAs field, shown below).

`srsepc/src/mme/s1ap_mngmt_proc.cc`:

```
enb_ctx->tacs[i]                 = ntohs(enb_ctx->tacs[i]);
enb_ctx->nof_supported_bplmns[i] = tas.broadcast_plmns.size();
for (uint32_t j = 0; j < tas.broadcast_plmns.size(); j++) {
    // ^ tas.broadcast_plmns.size() exceeds bounds due to bad bounds check
    // BPLMNs
    ((uint8_t*)&enb_ctx->bplmns[i][j])[1] = tas.broadcast_plmns[j][0];
    ((uint8_t*)&enb_ctx->bplmns[i][j])[2] = tas.broadcast_plmns[j][1];
    ((uint8_t*)&enb_ctx->bplmns[i][j])[3] = tas.broadcast_plmns[j][2];
    // ^ out-of-bounds read/write on array due to oversized plmns list

    enb_ctx->bplmns[i][j] = ntohl(enb_ctx->bplmns[i][j]);
}
```

This vulnerability can also occur in any interface for which there exist dynamic sequence structs containing lower/upper bounds that are not an exact power of two. Based on a quick look-over of the other protocols mentioned, this vulnerability is likely remotely exploitable via the E1AP, E2AP and F1AP protocols as well.

Published with GitHub Pages