TFM PSA IAT RA system

TrustedFirmware-M Platform Security Architecture Initial Attestation Token Remote Attestation system

---

1. 개요

1.1. 목적

Trusted Firmware-M의 기능중 PSA IAT를 사용하여 초기 검증 토큰 (IAT)를 생성하고 이를 Micro ROS에서의 xrce-DDS를 통해 원격 환경으로 전송하여 키로 검증하는 시스템을 구축

- 보드 코드
  https://github.com/tmdals010126/Micro-ROS-Zephyr-TFM/tree/tfm-attestation-base
- Agent 코드
  https://github.com/tmdals010126/Micro-XRCE-DDS-Agent/tree/TF-M-test
  https://github.com/tmdals010126/Micro-ROS-Zephyr/tree/TFM-agent

1.2. 구성 요소

보드는 zephyr RTOS에서 Micro ROS module을 설치하고 module 내부 XRCE-DDS Client 코드를 수정하여 구현
Agent는 Micro ROS를 따로 설치 후 XRCE-DDS Agent의 내부 코드를 수정하여 구현

- NUCLEO-L552ZE-Q https://os.mbed.com/platforms/ST-Nucleo-L552ZE-Q/
- zephyr RTOS https://github.com/zephyrproject-rtos/zephyr
- Micro ROS module
  https://github.com/micro-ROS/micro_ros_zephyr_module/tree/humble#
- XRCE-DDS Client
  https://github.com/eProsima/Micro-XRCE-DDS-Client?tab=readme-ov-file
- ROS2 humble
- TFM https://github.com/zephyrproject-rtos/trusted-firmware-m/tree/main
- Micro ROS https://github.com/micro-ROS/micro_ros_setup
- XRCE-DDS Agent https://github.com/eProsima/Micro-XRCE-DDS-Agent/tree/master
- IAT 검증 https://github.com/TrustedFirmware-M/tf-m-tools/tree/main

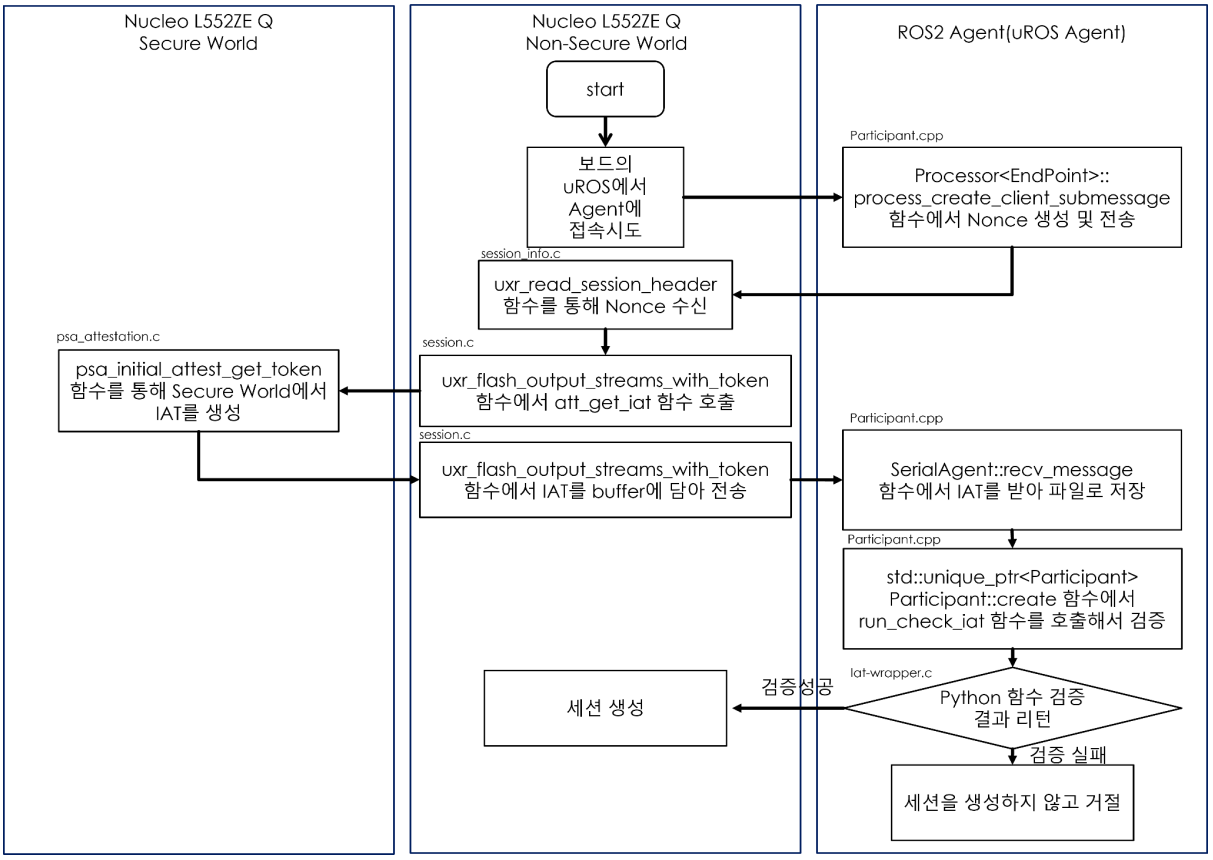
1.3. 주요 기능

1. 부트로더의 서명을 활용한 Secure booting으로 펌웨어를 보호 (TFM)
   - TrustZone-M을 활용한 secure world
   - Root-Of-Trust 구현
2. uROS를 통해 Agent 접근시 IAT로 펌웨어를 Remote Attestation
   - 개발자가 제조한 펌웨어인지 검증
   - replay attack을 방지하기위해 Nonce 포함

2. High-Level

2.1. 동작 흐름

1. TFM 기능으로 Bootloader에서 보안 부팅
2. Non-Secure world에서 xrce-DDS 기반 Micro ROS가 동작
3. xrce-DDS에서 agent(Server)에 접속을 시도
4. Agent에서는 접속 관련 정보와 Nonce값을 전송
5. 보드에서는 Nonce값으로 PSA IAT를 생성
6. PSA IAT를 생성할 때 관련 함수들은 Secure world(TrustZone)에서 동작
7. 생성한 PSA IAT를 agent로 전송
8. agent에서는 전송 받은 PSA IAT를 검증
9. 검증 결과가 유효하면 session을 생성



2.2. 구성 요소

Nucleo-L552ZE-Q (Client)

- Non-Secure-World
  - uROS
  - Nonce 수신
  - PSA IAT 발급 함수 호출

    - PSA IAT 전송
- Secure-World
    - 보안 부팅
    - secp256r1 (usage: ecdsa-with-SHA256)기반 PSA IAT 생성
        - PUF는 사용하지 않음. 하드웨어 별로 다르지 않다.

ROS2 Agent (Agent)

- Nonce 생성 및 전송
- Client PSA IAT 수신
- secp256r1 (usage: ecdsa-with-SHA256)기반 Client PSA IAT 검증

---

3. Module-Level

3.1. Nucleo-L552ZE-Q (Client)

3.1.1. zephyr_transport_open

uROS에서 통신 연결을 위해 설정을 하는 함수
메모리 맵에 정의된 LPUART_1 으로 연결하도록 수정

Parameter	Value	설명
struct uxrCustomTransport * transport	transport	통신 설정을 담고있는 구조체

3.1.2. rclc_node_init_default

node create를 Agent에 요청하는 함수

Parameter	Value	설명
rcl_node_t * node	&node	생성한 node를 저장할 구조체
const char * name	"zephyr_int32_publisher"	node name
const char * namespace_	""	node namespace
rclc_support_t * support	&support	메모리 관리 구조체

3.1.3. uxr_read_session_header

recv_message 함수에서 구동되어 Agent로 부터 받은 데이터에서 header를 파싱하는 함수
Nonce값을 파싱하는 구현이 포함. info->nonce에 저장.

Parameter	Value	설명

uxrSessionInfo* info	&session->info	session의 정보를 포함
ucdrBuffer* ub	ub	raw 데이터 버퍼
uint8_t* stream_id_raw	&stream_id_raw	ub에서 파싱된 stream id 가 저장될 포인터
uxrSeqNum* seq_num	&seq_num	메시지 순서를 저장

3.1.4. uxr_flash_output_streams_with_token

node create를 요청할 때 Agent로 session을 통해 보낼 buffer를 생성하는 함수
해당 함수 내에 PSA IAT 생성 요청 함수가 존재

Parameter	Value	설명
uxrSession* session	session	session 정보를 저장한 구조체

3.1.5. att_get_iat

PSA IAT를 발급받기 위한 handler 함수
psa_initial_attest_get_token 함수를 호출

Parameter	Value	설명
uint8_t *ch_buffer	session->info.nonce	nonce 값
uint32_t ch_sz	64	nonce 크기
uint8_t *token_buffer	iat_buf	생성된 iat가 저장되는 buffer
uint32_t *token_sz	&iat_sz	iat 토큰 크기 저장

3.1.6. psa_initial_attest_get_token

psa api를 호출하여 iat를 생성
psa_initial_attest_get_token 함수는 TrustZone을 활용한 Secure World에서 동작

Parameter	Value	설명
const uint8_t *auth_challenge	session->info.nonce	nonce 값
size_t challenge_size	64	nonce 크기
uint8_t *token_buf	iat_buf	생성된 iat가 저장되는 buffer
size_t token_buf_size	ATT_MAX_TOKEN_SIZE	buf 크기 0x240
size_t *token_size	&iat_sz	iat 토큰 크기 저장

3.2. ROS2 Agent (Agent)

3.2.1. Processor<EndPoint>::process_create_client_submessage

node 생성 요청 전에 Client에게 Agent가 존재함을 알려주는 과정에서 답장을 보내주는 함수
Nonce를 header에 포함하여 전송하도록 구현

Parameter	Value	설명
InputPacket<EndPoint>& input_packet	input_packet	packet의 정보를 담고 있는 구조체. message, header, source 등의 정보를 포함

3.2.2. SerialAgent::recv_message

시리얼(Serial) 통신을 통해 Client로부터 들어오는 메시지를 수신하고 이를 상위 로직에서 처리할 수 있는 InputPacket 형태로 만드는 함수
PSA IAT 토큰을 받아서 로컬파일로 저장하도록 코드를 수정

Parameter	Value	설명
InputPacket<SerialEndPoint> &input_packet	std::vector<InputPacket<MultiSerialEndPoint>> input_packet	packet의 정보와 메시지를 담는 구조체
int timeout	RECEIVE_TIMEOUT	timeout 시간을 설정
TransportRc &transport_rc	TransportRc transport_rc = TransportRc::ok	처리 성공 / 실패 여부를 저장

3.2.3. std::unique_ptr<Participant> Participant::create

Participant(Client 측 node)를 생성하는 함수
run_check_iat를 호출하여 PSA IAT를 검증하는 코드가 포함되도록 수정
검증 실패시 null 포인터를 반환 / 성공시 생성된 Participant를 반환

Parameter	Value	설명
const dds::xrce::ObjectId& object_id	object_id	생성할 Participant의 id (node id)
const std::shared_ptr<ProxyClient>& proxy_client	shared_from_this()	Participant 생성을 요청한 Micro XRCE-DDS 클라이언트를 대리하는 객체
const dds::xrce::OBJK_PARTICIPANT_Representation& representation	representation	생성하려는 Participant에 대한 정보

3.2.4. run_check_iat

python코드를 호출하여 PSA IAT의 유효성을 대리 검증하는 함수
내부적으로 python 코드를 호출하고 return 값을 받아서 반환

Parameter	Value	설명
int argc	custom_argc	python호출 시 인자의 개수
char *argv[]	custom_argv	python호출 시 전달될 인자들을 포함. main함수의 argv와 같은 형식

---

4. Source-Level

4.1. Nucleo-L552ZE-Q (Client)

4.1.1. zephyr_transport_open

`micro_ros_tfm/modules/libmicroros/microros_transports/serial/microros_transports.c`
uROS에서 통신 연결을 위해 설정을 하는 함수
메모리 맵에 정의된 LPUART_1 으로 연결하도록 수정

```
bool zephyr_transport_open(struct uxrCustomTransport * transport){
    zephyr_transport_params_t * params = (zephyr_transport_params_t*)
transport->args;
    char uart_descriptor[20];
    sprintf(uart_descriptor, "LPUART_1");
    params->uart_dev = device_get_binding(uart_descriptor);
    if (!params->uart_dev) {
        printf("Serial device not found\n");
        return false;
    }
    ring_buf_init(&in_ringbuf, sizeof(uart_in_buffer), uart_out_buffer);
    uart_irq_callback_set(params->uart_dev, uart_fifo_callback);
    /* Enable rx interrupts */
    uart_irq_rx_enable(params->uart_dev);
    return true;
}
```

4.1.2. uxr_read_session_header

`micro_ros_tfm/modules/libmicroros/micro_ros_src/src/Micro-XRCE-DDS-Client/src/c/core/session/session_info.c`
recv_message 함수에서 구동되어 Agent로 부터 받은 데이터에서 header를 파싱하는 함수

Nonce값을 파싱하는 구현이 포함. info->nonce에 저장.

```
bool uxr_read_session_header(
        uxrSessionInfo* info,
        ucdrBuffer* ub,
        uint8_t* stream_id_raw,
        uxrSeqNum* seq_num)
{
    bool must_be_read = ucdr_buffer_remaining(ub) > MAX_HEADER_SIZE;
    if (must_be_read)
    {
        uint8_t session_id; uint8_t key[CLIENT_KEY_SIZE];
        uxr_deserialize_message_header(ub, &session_id, stream_id_raw,
seq_num, key);
        for (int i = 0; i < 64; i++)
        {
            info->nonce[i] = ((uint8_t *)ub->init)[i];
        }
        must_be_read = session_id == info->id;
        if (must_be_read)
        {
            if (SESSION_ID_WITHOUT_CLIENT_KEY > info->id)
            {
                must_be_read = (0 == memcmp(key, info->key,
CLIENT_KEY_SIZE));
            }
        }
    }
    return must_be_read;
}
```

4.1.3. uxr_flash_output_streams_with_token

micro_ros_tfm/modules/libmicroros/micro_ros_src/src/Micro-XRCE-DDS-Client/src/c/core/session/session.c

node create를 요청할 때 Agent로 session을 통해 보낼 buffer를 생성하는 함수
해당 함수 내에 PSA IAT 생성 요청 함수가 존재

```
       while (uxr_prepare_next_reliable_buffer_to_send(stream, &buffer,
&length, &seq_num))
        {
            uxr_stamp_session_header(&session->info, id.raw, seq_num,
buffer);
            uint32_t iat_sz = ATT_MAX_TOKEN_SIZE;
            uint8_t iat_buf[ATT_MAX_TOKEN_SIZE] = { 0 };

            /* String format output config. */
            struct sf_hex_tbl_fmt fmt = {
                .ascii = true,
                .addr_label = true,
                .addr = 0
            };
            psa_status_t err = PSA_SUCCESS;
            /* Request the IAT from the initial attestation service. */
            err = att_get_iat(session->info.nonce, 64, iat_buf,
&iat_sz);
            if (err == PSA_SUCCESS) {
                for (int i = 0; i < iat_sz; ++i) {
                    buffer[length + i] = iat_buf[i];
                }
                buffer[length + iat_sz] = iat_sz;
                buffer[length + iat_sz + 1] = iat_sz >> 8;
                buffer[length + iat_sz + 2] = iat_sz >> 16;
                buffer[length + iat_sz + 3] = iat_sz >> 24;
            }
            send_message(session, buffer, length + iat_sz +
sizeof(uint32_t));
        }
```

수정한 주요 로직 코드만 가져옴.
PSA IAT 함수를 호출하고 결과를 buffer에 담아 size에 반영하여 전송

4.1.4. att_get_iat

micro_ros_tfm/src/psa_attestation.c
PSA IAT를 발급받기 위한 handler 함수
psa_initial_attest_get_token 함수를 호출

```
      /* Request the initial attestation token w/the challenge data. */
      // LOG_INF("att: Requesting IAT with %u byte challenge.", ch_sz);
      err = psa_initial_attest_get_token(
            ch_buffer,      /* Challenge/nonce input buffer. */
            ch_sz,          /* Challenge size (32, 48 or 64). */
            token_buffer,   /* Token output buffer. */
            token_buf_size,
            token_sz        /* Post exec output token size. */
            );
      // LOG_INF("att: IAT data received: %u bytes.", *token_sz);
```

함수 내부 중요 로직중 psa_initial_attest_get_token 함수를 호출하여 PSA IAT 생성 \

4.1.5. psa_initial_attest_get_token

../modules/tee/tf-m/trusted-firmware-m/interface/src/tfm_attest_api.c

psa api를 호출하여 iat를 생성

psa_initial_attest_get_token 함수는 TrustZone을 활용한 Secure World에서 동작

```
psa_initial_attest_get_token(const uint8_t *auth_challenge,
                             size_t         challenge_size,
                             uint8_t       *token_buf,
                             size_t         token_buf_size,
                             size_t        *token_size)
{
    psa_status_t status;
    psa_invec in_vec[] = {
        {auth_challenge, challenge_size}
    };
    psa_outvec out_vec[] = {
        {token_buf, token_buf_size}
    };
    status = psa_call(TFM_ATTESTATION_SERVICE_HANDLE,
TFM_ATTEST_GET_TOKEN,
                      in_vec, IOVEC_LEN(in_vec),
                      out_vec, IOVEC_LEN(out_vec));
    if (status == PSA_SUCCESS) {
        *token_size = out_vec[0].len;
    }
    return status;
}
```

추가 수정은 없음

4.2. ROS2 Agent (XRCE-DDS Agent)

4.2.1. Processor<EndPoint>::process_create_client_submessage

src/cpp/processor/Processor.cpp

node 생성 요청 전에 Client에게 Agent가 존재함을 알려주는 과정에서 답장을 보내주는 함수

Nonce를 header에 포함하여 전송하도록 구현

XRCE-DDS Agent 코드 내부에 존재. rand함수를 이용하여 랜덤값을 채워 header에 추가하여 전송. 64바이트 크기의 nonce 값.

```
            uint32_t nonce_buf[IAT_NONCE_SIZE/4] = {0,};
             // nonce_buf에 랜덤 값을 채웁니다.
             for (uint32_t i = 0; i < IAT_NONCE_SIZE/4; ++i)
             {
                 nonce_buf[i] = static_cast<uint32_t>(std::rand());
             }
             const size_t message_size =
status_header.getCdrSerializedSize() +

status_subheader.getCdrSerializedSize() +

status_agent.getCdrSerializedSize() + IAT_NONCE_SIZE;

             OutputPacket<EndPoint> output_packet;
             output_packet.destination = input_packet.source;
             output_packet.message = std::shared_ptr<OutputMessage>(new
OutputMessage(status_header, message_size));
             for(int i = 0; i < IAT_NONCE_SIZE/4; i++)
             {
                 output_packet.message->append_raw_uint32(nonce_buf[i]);
             }
output_packet.message->append_submessage(dds::xrce::STATUS_AGENT,
status_agent);
             server_.push_output_packet(std::move(output_packet));
```

4.2.2. SerialAgent::recv_message

src/cpp/transport/serial/SerialAgentLinux.cpp

시리얼(Serial) 통신을 통해 Client로부터 들어오는 메시지를 수신하고 이를 상위 로직에서 처리할 수 있는 InputPacket 형태로 만드는 함수

PSA IAT 토큰을 받아서 로컬파일로 저장하도록 코드를 수정

```
uint32_t token_size = 0;
uint8_t iat_token[1024] = {0,};
memcpy(&token_size, input_packet.message->get_buf() +
input_packet.message->get_len()-sizeof(uint32_t), sizeof(uint32_t));
```

```
if(input_packet.message->get_len() > token_size)
{
    memcpy(iat_token, input_packet.message->get_buf() +
input_packet.message->get_len()-token_size-sizeof(uint32_t),
token_size);
    // device_token을 파일로 저장합니다.
    FILE *fp = fopen("iat/attestation_token.dat", "wb");
    if (fp != NULL)
    {
        fwrite(iat_token, 1, token_size, fp);
        fclose(fp);
    }
    else
    {
        printf("iat_token file open failed.\n");
    }
}
```

4.2.3. std::unique_ptr<Participant> Participant::create

`src/cpp/participant/Participant.cpp`

Participant(Client 측 node)를 생성하는 함수
run_check_iat를 호출하여 PSA IAT를 검증하는 코드가 포함되도록 수정
검증 실패시 null 포인터를 반환 / 성공시 생성된 Participant를 반환
python 호출의 인자를 설정하고 run_check_iat 함수의 인자로 전달하여 검증

```
int iat_attest_return = 0;

char *custom_argv[] = {
    "main_executable",
    "-t",
    "PSA-IoT-Profile1-token",
    "-k",
    "iat/tfm_initial_attestation_key.pem",
    "iat/attestation_token.dat"
};
int custom_argc = sizeof(custom_argv) / sizeof(char *);

iat_attest_return = run_check_iat(custom_argc, custom_argv);
```

내부 검증 코드
