Securing DDS with OPTEE
Introduction

Myself:
• Principal Software Engineer at Arm
• Tech lead of the Safety Critical Machines team (a revamp of the Robotics team) with focus on the automotive area

Today's presentation:

Overview of a new project called libDDSSec: A library that implements the security portion of a DDS implementation using a TEE based solution.
Agenda:

• Key concepts
  • What is DDS
  • DDS Security
  • The ROS Project

• The libDDSSec project
  • Overview
  • The current work
  • Main challenges
  • Future work

• Why securing DDS

• References
Key Concepts

What is DDS?
What is DDS?

DDS stands for **Data Distribution Service** [1] and it refers to a standard. There are many implementations of the standard specifications (both commercial and open source). Data-centric publish-subscribe model for distributed application communication and integration. Its purpose is to enable efficient and robust delivery of the right information to the right place at the right time.

- Network middleware to simplify networking programming
- Data is exchanged between nodes via **topics**
- Nodes can **publish** data on a specific topic as well as receive data when they **subscribe** to a topic
What is DDS? (cont.)

Node A

Node B

Node C
What is DDS? (cont.)
What is DDS? (cont.)
What is DDS? (cont.)

Publisher

Sensor 1

temperature

Subscriber

Monitor

Publisher

Sensor 2
What is DDS? (cont.)

There is a collection of specifications around the DDS split into different groups: Core, Extensions, Gateways and API).

This presentation focus on:

- **DDSI-RTPS protocol v2.2 [core]**: low-level interoperability wire protocol
- **DDS v1.4 [core]**: Data centric Push/Sub model
- **DDS-Security v1.1 [extension]**: Security model and plugin interface
DDS Security: Model

The DDS security model defines the **users of the system**, the **objects** that are being secured and the **operations** that are to be restricted.

- Securing DDS means providing:
  - **Confidentiality** of the data samples
  - **Integrity** of the data samples and the messages that contain them
  - **Authentication** of DDS writers and readers
  - **Authorization** of DDS writers and readers
  - **Message/Data origin** authentication
  - **Non-repudiation** of data
DDS Security: Threat Model

Specification details four categories of threats:

- Unauthorized subscription
- Unauthorized publication
- Tampering and replay
- Unauthorized access to data
DDS Security: Implementation

The DDS Security specification defines **plugins** to implement

- **Authentication**
  - Certificate management

- **Cryptography**
  - Crypto operations

- **Access control**
  - Policy enforcement
DDS Security: Implementation (cont.)

- Asymmetric key cryptography used mainly for discovery, authentication and shared-secret establishment phase.
- The use of cyphers, HMAC, or digital signatures is selectable on a per stream (Topic) basis.
Key Concepts
The ROS Project
The ROS Project

The Robotic Operating System [2] is a framework (libraries and tools) to develop robot applications.

ROS2 adopted DDS as its communication layer: Bundled by default with the open source DDS library called Fast-RTPS by eProsima but can be integrated with other DDS implementations (e.g. RTI Connext).

Client API:
- CPP API
- Python API
- C API

- ROS2 middleware (C API)
- DDS
- OS (Linux, MacOS, Windows)
ROS comes with a plethora of packages that can be interfaced with each other to build a robot.
The ROS Project

ROS comes with a plethora of **packages** that can be interfaced with each other to build a robot.
The LibDDSSSec
The LibDDSSec: Goals

• Move all security assets into the TEE
  • Certificates
  • Key generation
  • Security operations

• Limit attacks
  • E.g. No key leakage

• Provide a reference implementation on how to take advantage of the TrustZone IP to secure DDS (using OPTEE).
The LibDDSSec: Overview

The DDS implementations we came across use OpenSSL for the security support:

- Certificates in filesystem
- Operations in non-secure world
Move security operations into a TEE

- DDS Implementation
  - Auth
  - Access
  - Crypto
  - Generic low-level Security API
  - TEE-backend
  - OPTEE-Client
  - OpenSSL

- OPTEE-OS
- DDS-Sec TA
- TF-A

Non-secure world

Secure world
The LibDDSSec: Overview (cont.)

Isolate code into its own project

DDS Implementation
- Auth
- Access
- Crypto

LibDDSSec API

LibDDSSec
- TEE-backend
- OpenSSL-backend

TEE-backend
- OPTEE-Client

OpenSSL-backend
- OpenSSL

OpenSSL

OPTEE-OS

DDS-Sec TA

TF-A

Secure world

Non-secure world
The LibDDSSec: Overview (cont.)

Under discussion: Implement plugins in the library

DDS Implementation

Auth | Access | Crypto

LibDDSSec API

Auth | Access | Crypto

LibDDSSec

TEE-backend

OpenSSL-backend

OpenSSL

OPTEE-Client

OPTEE-OS

DDS-Sec TA

TF-A
The LibDDSSec: Current work

• Moving code from the prototype into the standalone library:
  • Reviewing prototype API whilst moving code into the new library
  • Adding unit tests instead of relying only on Fast-RTPS’s tests
  • Adding more

• Threat model
  • Ensure prototype design is sound
  • Ensure key deployment is safe
  • Ensure Non-secure interface is safe (or at least limit attacks)

• Investigating the new x.509 support in OPTEE 3.2
  • Current base code still uses OpenSSL for some of the operations, including handling of certificates
The LibDDSSec: Main challenges

• Latency:
  • One of the main trade-offs when using TEE will be the extra latency
  • Apex.ai recently released a benchmark tool to measure latency in DDS implementations that can be useful on this area.

• Vulnerabilities in the non-secure world could allow the secure assets to be used by potential attackers

• Key and certificate deployment
The LibDDSSec: Future work

Further areas that can be explored:

- Key and certificate deployment
  - Ideally using hardware ID to derive keys
  - As far as we are aware, OPTEE (or GlobalPlatform) has no API for deriving keys using hardware ID (yet)

- Evaluate the possibility of running “DDS Trusted Applications”
  - In other words, move a whole DDS application into the TEE
  - This means having a DDS layer in the TEE other sorts of complications
The LibDDSSec

- Under development and soon to be available in Github under the ARM-Software umbrella
- License: BSD (provisional)
- Language: C, C++
- Development using standard-ish ArmPlatforms [4] stack:
  - Base AEMv8-A Base Platform FVP
  - Linaro’s kernel-latest
  - OpenEmbedded
  - OPTEE (currently manually enabled)
Why are we doing this?

• DDS being adopted in mission critical applications
  • These usually have associated security requirements as well

• Increased adoption on the automotive area:
  • Autosar consortium [5] is adopting DDS in its specifications (Adaptive)
  • Baidu’s Apollo [3] uses ROS1 and is moving to another solution based on DDS
  • Autoware [6] currently ROS1 but planning to add support for ROS2

• We are aware of other automotive frameworks based on ROS2 project.
Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
תודה
References

[4] https://community.arm.com/dev-platforms/w/docs