Adding secure security to ROS2

• Filipe Rinaldi - filipe.rinaldi@arm.com
• September 2018
Introduction

About us:

Members of the Arm Open Source Software group. Part of a new team (a revamp of the Robotics team) with focus on the automotive area:

• Filipe Rinaldi (myself)
• Florian Depraz (present at ROSCon)
• Louis Mayencourt (present at ROSCon)
• Kurtis Charnock

Todays presentation:

Overview of a new project called **libddssec**: A library that implements the security portion of a DDS implementation using Arm TrustZone (Cortex-A profile) [1].
Agenda:

• Key concepts
  • DDS Nodes
  • DDS Security
  • Arm TrustZone

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

• Why securing DDS/ROS2?

• References
Key Concepts

DDS
DDS: Nodes

Node A

Publisher

Topic X

Subscriber

Node B

Publisher

Node C
DDS: Specs (cont.)

There is a collection of specifications [2] 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

What is Arm TrustZone?
What is Arm TrustZone?

- Provides a second virtual world allowing a different software stack to co-exist
- Physical address space is partitioned between these two worlds
- Orthogonal to the Exception/Privilege Levels
What is TrustZone? (cont.)

- It is all about who you **trust**
What is TrustZone? (cont.)
What is TrustZone? (cont.)

Non-Secure

EL0
- App

EL1
- OS

EL2
- Hypervisor

Secure

EL3
- Secure Monitor

Secure

- Trusted App

Non-Secure

- Trusted OS
What is TrustZone? (cont.)

Non-Secure

EL0
- App

EL1
- OS

EL2
- Hypervisor

Secure

EL0
- App

EL1
- OS

EL2
- Hypervisor

EL3
- Trusted Firmware A

- Trusted App

- OPTEE-OS
The libddssec
The libddssec: Goals

• Move all security assets into the Trusted Execution Environment (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 DDS implementations we came across use OpenSSL for the security support:

- Certificates in filesystem
- Operations in non-secure world
The libddssec: Overview (cont.)

Move security operations into a TEE

- **Generic low-level Security API**
  - **Auth**
  - **Access**
  - **Crypto**

- **TEE-backend**

- **OPTEE-Client**

- **OpenSSL**

- **OPTEE-OS**

- **DDS-Sec TA**

- **TF-A**
Isolate code into its own project
Under discussion: Implement plugins in the library
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

• 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 GlobalPlatforms) 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

- **ARMv8-M Support (microcontrollers)**
  - Retarget OPTEE and Trusted application to use Trusted-Firmware-M
  - Both using MBedTLS
The libddsec

- 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
謝謝
ありがとうございます
Gracias
Kiitos
감사합니다
धन्यवाद
תודה
References

[4] https://community.arm.com/dev-platforms/w/docs
The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks