Secure Software Update in Modern Software Architecture

University of Michigan - Dearborn

Abir Bazzi, Dr Di Ma, Dr Adnan Shaout
Trends in Automotive E/E architecture

Today

Tomorrow

Future
Most OEMs follow AUTOSAR Software standardized modular architecture:

- AUTOSAR is a worldwide development partnership of different manufacturers and companies within the automotive industry.
- The Classic Platform is AUTOSAR’s solution for embedded systems with hard real-time and safety constraints.
- The AUTOSAR Adaptive Platform is AUTOSAR’s solution for high-performance computing ECUs to build safety-related systems for use cases such as highly automated and autonomous driving.

Due to automotive industry growth and development, AUTOSAR has introduced a new concept “Flexible” Classic Autosar:

- Decompose the software into independent software clusters. This enables separation of Functions, shorter development cycles as well as independent development between software owners.
- Every changed software cluster is programmed independently from the others and the whole ECU is executable and functional with an intended combination of changed and unchanged clusters.

Virtualization is another innovation technology for hard real-time automotive applications.

- Multiple Virtual machines (VM) execute together over a shared hardware platform and in isolation from each other.
Modular Software Architecture

- Empowering of different software vendors contributing to same ECU
  - Enabling independent SW build units.
  - Enabling independent updating of Software parts and their execution.
  - Enabling independent set of library functions: Security, Safety, communication, memory….
  - Enabling Clustering of QM and safety related Software.
  - Enabling Clustering of SWCs and BSW modules.
Two types of Software Update:

- Local update: traditional software update technique, where cars are brought to the service center and a car mechanic updates the software using dedicated tools through OBD connection.
- Remote (Over-the-air) Update: Software is sent to the vehicle through wireless communication, while the vehicle is running.

The main drawback of local update method is time, resource consuming and can only be done at dealership:
- Resulting in high cost and customer dissatisfaction.
- SW update delay can lead to Security vulnerability.

OEM Benefits for Over-the-air (OTA) updates:
- Fast and Cost effective – no recall required.
- Ability to patch critical bugs/security vulnerabilities as early as possible.
- Revenue generation with new features.
OTA updates: A brief explanation

Step 1: Download while driving
- Software download to central storage
- Unnoticed by the user
- Vehicle safe and operational
- Service authentication:
  - Mutual authentication between car and OEM update server
  - Encrypted transport channel

Step 2: Update from internal storage
- Verification of the downloaded software
  - SW-IP protection (e.g. Encryption / Decryption)
  - Integrity Check (e.g. Hash)
- Update to the new Software image
- Confirmation to Backend
Secure software OTA update process

Software development
- **TIER 1**
  - Update Image
  - H - Hashing (image)
  - Signature relies on a **private** key associated with the public key used by the end ECU

OTA Software update
- **OEM / OTA Server**
  - Update Image
  - Encrypted Digital Signature
  - Signature relies on a private key issued by the update root certification authority
- **Host (TCU/GW)**
  - Update Image
  - Encrypted Digital Signature
  - Signature verified using public key stored in TCU
- **ECU Target**
  - Update Image
  - Encrypted Digital Signature
  - Signature relies on a **public** key associated with the private key used when building the image

Authentication and Download

2022 WLI WISH Conference
Connections are typically active all the time and thus OTA interfaces are potentially accessible for hackers.

SW OTA services provide doors for attacks which might endanger the operation of vehicles.

Already the potential to attack vehicles is threatening Tier1s and OEMs.
Existing Security mechanisms

- Update image repository
- Use application-level security, secure timestamps & metadata, and ensure redundancy (e.g., Uptane framework to secure the OTA ecosystem)
- Store private keys in a Secure Element (TPM)
- Harden telematics & server with Sec. Boot
- Secure Onboard Communication and end-to-end authenticated updates with HSM.

- Target Vehicle
- Target ECU(s)
- Telematics/Gateway
- OTA Servers

2022 WU WISH Conference
Existing secure SOTA solutions

- Categorized based on the security mechanisms used by researchers.
- Follow the same described process.
Secure Key deployment

- Create asymmetric key pair by OEM for each ECU:
  - Public/Private key using Asymmetric Algorithm (e.g. RSA/ECC).
  - Store public and private keys in Secure area / Protected Flash.
  - Public Key should be send to the ECU: mainly at the end of production line or later on the field using PKI system.

- Setup phase at the ECU side:
  - ECU receives public key.
  - Application imports the public key in cryptographic library.
  - Store public key in Protected Flash, mainly in TPM/HSM.
Software Release Process
Challenges of OTA approaches

- Using existing SOTA mechanisms, the complete application software image is updated together per ECU.
- Adaptive Autosar integrates the software update concept within the standard by standardizing the process and responsibility. However, Classic Autosar doesn’t have the same capabilities.
- Each ECU will need a unique set of keys per each software cluster shared between each developer and ECU Target.
- Increase complexity in key management, deployment, and storage.
- Limitations with current microcontrollers and Flash technologies due to performance and memory granularity (e.g. limited number of erase/write cycles).
SOTA Solution: New proposal

- Extend existing solution with features oriented for distributed systems
  - Uptane framework, Autosar platforms...
- Address dependencies and conflicts among the software entities.
- Logical implicit verification.
- Efficient Key deployment and management.
- Optimize code signing scheme and ECU resources.
- Generate unique signature based on Merkle-Tree concept.
- Solution flexible and applicable for various applications and security use-cases.
Proposed Approach for Signature generation

Height: \([\log n]\) Level: 0

Height: \([\log n] - 1\) Level: 1

Height: 1 Level: \([\log n]\) - \(1\)

Height: 0 Level: \([\log n]\)

Software = Software blocks \(\{SW_A + SW_B + SW_C + SW_D + \ldots + SW_{(N-1)} + SW_N\}\)
Updating Single Software entity

Software = Software blocks \{SW_A + SW_B + SW_C + SW_D + ... + SW_{(N-1)} + SW_N\}
Removing Software entity

Software = Software blocks \{SW_A + SW_B + SW_C + SW_D + \ldots + SW_{(N-1)} + SW_N\}
Different Merkle Tree types

(a) Full balanced

(b) Multi-Root

(c) Skewed
Summary

- New automotive platform is seen as the “App-Store” for automotive applications for installation and update over the air.
- The authenticity and integrity of update packages need to be protected to impede the import of forged and manipulated data.
- Unique signature generation based on Merkle Tree concept:
  - Smoothly update, add, and remove software image.
  - Security and safety of the update process.
  - Authenticity and integrity validation for the software image.
  - Reduce used key materials.
  - Enable local and remote software compatibility check.
  - Recovery of software after failure during update process.
  - Suitable for future cryptographic algorithms.
Q&A

Thank you!

Any questions?