KEEPING YOUR ECU SECURE AGAINST COMPROMISED ADAPTIVE APPLICATIONS
**FACTS & FIGURES**

1994
year of establishment

100%
subsidiary of Robert Bosch GmbH since 2017

Bernd Gohlicke
CTO

Michael Englert
Founder & CEO

10 locations in Germany
Ruelzheim | Friedrichshafen | Munich | Ingolstadt | Stuttgart | Frankfurt | Lollar | Cologne | Braunschweig | Berlin

4 locations worldwide
Vienna | Barcelona | Tokyo | Wuxi

>25 years
of experience in custom-made systems and software engineering in various industries

1,300 employees worldwide
OUR BUSINESS MODEL

BENEFITS OF COLLABORATION

WHITEBOX BUSINESS MODEL
What we develop for you belongs to you.

INDUSTRY-SPANNING EXPERTISE
Why reinvent the wheel, if a problem was solved in another domain?

PRODUCT INDEPENDENCE
As we do not offer products, our engineering services are unbiased.
AGENDA

1. Motivation
3. Overview of the IAM solution
4. Remote Access Control
5. Conclusion
MOTIVATION
Adaptive AUTOSAR allows many applications of multiple vendors to share not only the same hardware but also the same OS.
SECURITY IMPLICATIONS OF APPLICATION CO-LOCATION

- Compromised applications can directly influence other applications.
- Compromised applications can indirectly influence other applications via AUTOSAR functions.

The weakest application determines the level of security of overall system.
WHY SHOULD WE CARE ABOUT SECURITY?

- Legal and industry requirements
  - UNECE Cybersecurity Regulation R155/R156
  - ISO 21434
  - General Data Protection Regulation (GDPR)

- Consequences of security breaches must be confined

- Security concepts need platform support
PROBLEM STATEMENT
DO WE REALLY HAVE TO ASSUME COMPROMISED APPLICATIONS IN THE ADAPTIVE AUTOSAR SCENARIO?

Different security quality requirements at different companies

High probability of vulnerabilities due to the amount of code that runs on one platform

Potentially malicious actors in the supply chain

vs.

supply chain
WHAT CAN WE DO ABOUT POTENTIALLY COMPROMISED APPLICATIONS FROM A PLATFORM PERSPECTIVE?

Application isolation:
Process isolation and containerization of applications → “Protected Execution Environment” concept

But: Some applications legitimately have to interact to fulfill their functions

Least-privilege principle:
“Every system component has to be able to do the things required for the intended functionality but nothing more.”

→ “Identity and Access Management” (IAM) concept
**FOR WHICH AUTOSAR COMPONENTS DOES IAM HAVE TO BE APPLIED?**

"Every system component has to able to do the things required for the intended functionality but nothing more."

– Principle of Least Privilege –

**AUTOSAR resources and services that are currently covered by IAM:**

<table>
<thead>
<tr>
<th>Service-oriented communication</th>
<th>Persisted Data</th>
<th>Diagnostics Interface</th>
<th>Raw Data Stream, [...]</th>
</tr>
</thead>
<tbody>
<tr>
<td>allow applications to use or provide specific service-methods, fields, or events</td>
<td>restrict access to application’s own storage</td>
<td>allow application to access Diagnostic Manager</td>
<td>further services and functionality in ARA</td>
</tr>
<tr>
<td>Platform Health Management</td>
<td>allow application to report specific CheckPoints</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© ITK Engineering GmbH | 9/17/2021 | public
SOLUTION
Subjects (e.g., applications) access resources (e.g., SOME/IP methods, events or fields)

- Least Privileges Paradigm: Only allow accesses that are required to guarantee the functionality of the system
- Reference Monitor is distinct from untrusted subject

PEP: Policy Enforcement Point
**PRECONDITION FOR IAM: ISOLATION OF APPLICATIONS**

- **Isolation from other applications:**
  It must not be possible for applications to **directly influence** other applications.
  
  *Example: CPU/RAM consumption, interrupts, ...*

- **Isolation from AUTOSAR runtime:**
  It must not be possible for applications to **manipulate the PEPs** in the AUTOSAR Runtime.

  *Example: applications that run in the same OS-process as the AUTOSAR Runtime*

- **Isolation from the OS:**
  It must not be possible for applications to **sidestep the PEPs** in the AUTOSAR Runtime by using OS functions directly.

  *Example: opening network sockets*
IAM SOLUTION: POLICY DECISION AND ENFORCEMENT

1. Request resource
2. Is application 1 authorized?
3. Yes.
4. Perform request

- Manifests are deployed with applications
- Manifests are trustworthy due to digital signature
- No prior knowledge about application needed
EXAMPLE: ADAPTIVE APPLICATIONS AS SERVICE PROVIDER

1. Request service

2. Enforce authorization: consumer allowed?

3. Enforce authorization: provider allowed?

4. Process service request

- Policy Enforcement is executed in layer of ARA::COM
- This allows service instances in adaptive applications without changes
EXAMPLE: OPERATING SYSTEM PROVIDES PEP

- PEP might be provided by operating system
- System is setup according to manifests

1. Configure file permissions based on manifest
2. Request file access
3. Enforce permissions

Trusted reference monitor: operating system access control
MODELING
Basic principle of capabilities is known from smartphone applications.

- Mobile application comes with signed list of requested permissions.
- During installation, the user confirms or rejects requested permissions.
ASSIGNING PERMISSIONS TO ADAPTIVE APPLICATIONS: ROLES

Software Development
requests capabilities for SW components by **Intent** in Model

Application Design
composes Adaptive Application from components

Integration
deployment of Adaptive Application with configuration and accepted **Intents**
ASSIGNING CAPABILITIES TO ADAPTIVE APPLICATIONS:
SOFTWARE DEVELOPMENT

- Adaptive Applications define ports
- Ports are configured with lists of ComSpecs
- ComSpec defines ClientIntent: request to access a method

Simplified modelling of **Intents**
ASSIGNING CAPABILITIES TO ADAPTIVE APPLICATIONS: APPLICATION DESIGN

- Application designer collects capabilities of components and links to executables with **GrantDesign**

Simplified modelling of **GrantDesign**
ASSIGNING CAPABILITIES TO ADAPTIVE APPLICATIONS: INTEGRATION

- Integrator defines **Process** from executables via **ProcessDesign**
- The integrator finally accepts the requested permission to call the specific method with Grant linking to the **GrantDesign** elements
- This permission is bound to a specific deployment
ASSIGNING CAPABILITIES TO ADAPTIVE APPLICATIONS: ROLES

Software Development
Capabilities are defined by ComSpec

Application Design
Application designer creates list of capabilities with GrantDesign elements

Integration
Integrator accepts the GrantDesign requests with Grant elements

All this will be supported by toolchains!
REMOTE IDENTITY AND ACCESS MANAGEMENT

- **Goal 1**: Prevent compromised applications on the local system from using functionalities they are not authorized for.
- **Goal 2**: Prevent Denial-of-Service on the bus by a compromised application.
- **Goal 3**: Prevent compromised applications on remote systems from using functionalities on the local system they are not authorized for.

Manifests are synchronized (not specified by AUTOSAR) or managed via Update and Configuration Management (Concept SCREIAM).
Peers are identified and authenticated with support from network binding security
Specified for SecOC, TLS and IPSec
Example TLS: Peers authenticated with certificate or pre-shared keys
ASSIGNING PERMISSIONS TO REMOTE ECU EXAMPLE: SCREIAM WITH IPSEC

- Integrator defines **IPSecRemoteSubject**
- The integrator finally accepts the requested permission to call the specific method with **Grant** linking to the **RemoteSubject** elements
- **IPSecRule** defines keys, certificates etc.
- **Grant** links to Service Instance
CONCLUSION
Adaptive AUTOSAR Identity and Access Management allows to:

- enforce access policies on interfaces and according data flow
- restrict attacker’s potential to control the overall system
- consider security in all stages from development to deployment
- define access policies on well-known modelling elements with little overhead
CONCLUSION

IAM alone does not cover everything, but constitutes an essential building block for a secure system
- Other security mechanisms are required (e.g., proper application isolation, Secure Communication, Secure Boot and Secure Update)
- Capabilities have to be modelled correctly and checked for effectiveness

Usability
- The Intent model makes sense from a modeling perspective, but not from a usability perspective. It is up to the tooling to improve this

Future Challenges
- “A fool with a tool is still a fool” → Correct configuration of IAM during integration is essential