Adapting Organic Computing Architectures to an Automotive Environment to Increase Safety & Security

Kevin Lamshöft, Robert Altschaffel, Jana Dittmann
Otto von Guericke Universität Magdeburg

31.05.2017

Overview

- Introduction
  - Automotive components today
  - Organic Computing & adaptive systems
- Adapting OC to automotive bus networks
  - Why should be do this?
  - How can do this?
- Exemplary Implementation
- Summary & Outlook

Introduction - Automotive IT

- Dumb components
  - Parts without electronics
- Smart components
  - Sensors
  - Actuators
  - Electronic Control Units (ECU)
  - Direct analogue cable connections
  - Shared Digital Bus Systems
- Growing number of interfaces to the environment

  -> Growing number of attack vectors
- Growing number of smart components

  -> Growing complexity

  Problem: a combination of legitimate micro behavior can led to unwanted macro behavior

Introduction - Organic Computing

- [Wu08]: *In organic computing, the only task humans hold on to is the setting of goals. As the machine is autonomously organizing, detailed communication between programmer and machine is restricted to the fundamental algorithm, which is realizing system organization.*

- AUTOSAR implements some of this as a middleware which take of the internal communication

- Mechanism to enforces that the system behavior adheres to the goals is required

[Wu08] R.P. Würtz (ed.): Organic Computing Understanding Complex Systems, doi: 10.1007/978-3-540-77657-4 1, © Springer-Verlag Berlin Heidelberg 2008

Observer [Ri06]

- **O1 Monitor** gains raw data from the underlying SuOC
- **O2 Log file** saves history data from each iteration
- **O3 Pre-Processor** prepares data for analysis and prediction
- **O4 Data analyzer** applies a set of detectors on the pre-processed data; result is he current state of the SuOC
- **O5 Predictor** predicts future system states based on raw data, history data and analyzer data
- **O6 Aggregator** accumulates data to pass it to the controller

[Ri06] Richter, U.; Mnif, M.; Branke, J.;Christian Muller-Schloer, C.;Schmeck, H.:Towards a generic observer/controllerarchitecture for Organic Computing.

Adapting OC to automotive bus networks

- Problem: a combination of legitimate micro behavior can led to unwanted macro behavior
- Increase robustness against attacks and malfunctions
- Achieve desired and disrupt undesired emergent behavior
- Detection and reaction on irregularities of meta functionality

-> Take mechanisms from OC and adapt them to automotive systems

Observer for automotive bus networks

- **Some changes on conceptual level:**
- **O1 Monitor** monitors communication on the various bus systems
- **O2 Log file** saves history data from each iteration
- **O3 Pre-Processor** prepares data for analysis and prediction
- **O4 Data analyzer** detects unusual/illogical behavior and rule violations → anomaly detection for bus networks
- **O5 Predictor** predicts future system states based on raw data, history data and analyzer data
- **O6 Aggregator** accumulates data to pass it to the controller

Observer for automotive bus networks – O1

- **O1 Monitor** monitors communication on the various bus systems
- External sensors would be best to capture the behavior of actuators
- But not completely practical at this moment
- Instead: observation of bus communications
  - Actuators are very dumb and expected to act (only) on commands received via bus communication
  - Communication should contain information on all actions
  - Exception: defunct actuators
  - Would need external sensors (possibly via C2x)

Observer for automotive bus networks – O4

- **O4 Data analyzer** detects unusual/illogical behavior and rule violations → anomaly detection for bus networks
- Detection of unusual/illogical behavior is based on anomaly detection
  - Triggering a certain function repeatedly in a short time can point to the function being defunct (e.g. turning lights on)
  - Toggling a certain function repeatedly in a short time can also point to a problem (e.g. opening and closing windows)
  - 'normal behavior' needs to be learned
- Rule definitions describe certain pre-defined rules
  - e.g. Vehicle should not move while the door is opened

Controller for automotive bus networks

- Takes action
  - Based on information received from the observer
  - Sends commands to ECUs, informs the driver
  - Uses a set of classifiers (rules) to map states into actions
  - Each classifier has a fitness value which is updated once the controller gets feedback on the effect of the classifier
  - Classifiers are adaptive

Implementation – Observer (Technical details)

- Raspberry Pi 3
- CANtact Hardware
- SocketCAN / can-utils
- Direct access to CAN

Implementation – Observer

- **O1 Monitor** monitors communication on CAN using SocketCAN, CANtact and python-can
- **O2 Log file** saves history data from each iteration, e.g status of ignation and doors
- **O3 Pre-Processor** strips frames to CAN ID and payload only
- **O4 Data analyzer** entropy-based anomaly detection, combination of whole bus and dedicated calculations for specific IDs. Analyzer needs a learning phase
- **O5 Predictor** is not included in this demonstrator
- **O6 Aggregator** accumulates data to pass it to the controller

Implementation - Controller

state =

[{Entropy Anomaly Detector -> Affected ID: 0x172},
{ECU Fingerprint Anomaly Detector -> Anomaly Source: central lock ECU},
{Aggregator -> Result: doors do not open}]

→ Mapping

actions = [{flash central lock ECU}, {open windows},
{open trunk}]

Implementation – the MALfunctions

- Use of an experimental automotive ransomware
- Physical and remote attack possible
- Automotive ransomware sends spoofed messages on the bus to implement different attack scenarios
  - **S1 Lockout** waits for a signal to unlock the doors, sends signal to lock doors
  - **S2 Nuisance** randomly triggers warning and turning lights, changes volume of multimedia system
  - **S3 Confinement** waits for car to be turned off, then applies central lock, closes windows and starts DOS on these controllers
- All these attacks work on the Q7 test environment

Evaluation

- **S1 Lockout**
  - Incident alone does not raise alarm
  - repeated tries to open doors
  - Passengers are asked via CLI if they are trying to open the door
  - Controller might open the doors
- **S2 Nuisance**
  - Detection does not work well (either many false-positives or many false-negatives, depending on parameters)
  - If the detection works the Controller can take action
- **S3 Confinement**
  - Attack is easily detected
  - Controller informs passengers of attack and opens trunk

Summary & Outlook

- This works presents an approach to handle the growing complexity of automotive systems
- It focuses on the reaction to complex threats
- Main contribution is exploration of adaption to automotive environments
- Exemplary implementation and evaluation of 3 attacks

- More work needed, especially on the analyzer
  - Use of other metrics than entropy
- Larger data base needed, especially for the learning phase
- Evaluation of more attacks / malfunctions
- Evaluation on Trucks / Bus
  - Currently adopting ransomware

Thanks for your attention!

Partly funded by German Research Foundation, project **ORCHideas** (DFG GZ: 863/4-1)