Organic Principles to Counter Malware in Automotive Environments

Robert Altschaffel, Sven Kuhlmann, Jana Dittmann, Tobias Hoppe
Arbeitsgruppe Multimedia and Security
Otto-von-Guericke-Universität Magdeburg
Magdeburg, Germany
email: {Robert.Altschaffel|Sven.Kuhlmann|Jana.Dittmann|Tobias.Hoppe}@iti.cs.uni-magdeburg.de

***Abstract*—In an interconnected world malware is not only a topic in classical computer environments (information technology) anymore. Recent examples have shown which damage malware can cause in modern interconnected cyber-physical systems. Based on the increasing threat of malware we propose an approach to counter malware. In contrast to classical approaches like signature scanners or IDS (Intrusion Detection Systems), which focus on detection our approach focuses on the reaction on malware-caused incidents. While these classic approaches usually require manual intervention we focus our approach on an automatic, adaptive reaction. The approach proposed in this work is based on principles of Organic Computing and conceives a self-adapting system for malware reaction in automotive environments.**

Keywords - automotive security; adaptive systems; malware

I. INTRODUCTION

With the ubiquitousness of cyber-physical and interconnected systems a new field of potential targets for malware arises. This work focuses on the automotive domain as an example of an interconnected cyber-physical system. With its broad array of different Electronic Control Units (ECUs) and it's ever increasing range of means to communicate with its environment (Car2X), a modern car is a prime example for both categories. The automotive domain also carries the implication that malfunctions of an ECU could lead to accidents and serious physical harm. Therefore, not only the detection of a malware incident but also a timely, appropriate reaction upon such incidents becomes crucial. While interconnectivity and reliance on electronic components come with the downside of vulnerability, new technology also offers new possibilities to protect the automotive system from such attacks [1].

We propose an adaptive subsystem to support autonomous reactions against malware incidents in the automotive domain. After a brief introduction we give a short overview about anti-malware strategies from Classical computer environments, discuss the special properties of the automotive domain and show general approaches for adaptive systems. In the third section, we introduce our approach for an adaptive malware reaction system while the fourth section shows a first demonstrator of this approach. The fifth section closes with a summary and an outlook.

II. STATE OF THE ART

This section gives an overview about malware detection and reaction in classical computer environment, the characteristics of concurrent and upcoming automotive systems and adaptive systems.

A. Malware in Classical computer environments

In classical computer domains, like Desktop IT malware threats are a central challenge in the productive operation of IT systems. A lot of research went into reducing this threat. Malware analysis (like [2] or [3]) have determined the characteristics of common security threats. Appropriate countermeasures [4] [5] have been established. While protective mechanisms to prevent malware attacks are an important basis of established anti-malware concepts, the vast complexity of today's IT systems with diverse user and/or system interactions make establishing complete protection mere theory. Therefore, an important basis for the treatment of malware incidents is their detection. While classical, signature-based detection strategies are increasingly ineffective due to the ever increasing size of malware samples and bypassing strategies, evolving heuristic approaches still have to cope with false alarms. Consequently, in many cases the subsequent reaction to detected incidents still require manual interaction with human operators – which might be system administrators in professional environments or simply the user himself (e.g. in home environment). Also, in Classical computer environments, the mechanisms for autonomous reactions to malware incidents are a young field of research with little, immature approaches so far.

B. Characteristics of Modern Automotive Systems

Automotive systems differ from classical computer environments in a range of characteristics. In order to design an approach that covers the automotive domain it is necessary to discuss these characteristics and their impact. Important differences originate from the operational environment and the hardware architecture. For the operational environment, we identify the following aspects:

- Safety Implications: In contrast to typical operational environments of classical Classical computer environments, an incident in an automotive system can easily lead to threats to life and health of the user and even passengers and innocent bystanders.

 ISBN: 978-1-61208-427-5

- Technological Aspects: An automotive system is interacting with an analog environment. Therefore many central components within an automotive system have very strict real-time requirements.
- Organisational Aspects: The owner/driver of an automotive system in general is not an IT expert. The automotive system's design therefore must consider that the user is not able to administrate the system or handle user interactions requiring deep, specific knowledge.

These points show that some classic 'emergency reactions' from Classical computer environments - like rebooting or powering down - cannot be applied directly to an automotive environment. Other classical strategies, like software updates during runtime, can only be imported within limits. The hardware architecture itself also has direct impact:

- System Architecture: Like other embedded systems, automotive IT features a broad range of different hard- and software configurations. ECUs in an automotive environment have a much less standardized hardware and software architecture than in a classical computer environment. Also, the usage of processors with small word size and low clock rate is common in automotive environments. Combined with relatively small memory, this seriously restrains available resources. While using this kind of hardware is partially motivated by economic considerations, automotive hardware also needs to cope with various environmental effects virtually unknown in Classical computer environments. Automotive IT needs to be robust against wide variations in temperature, concussions, intruding fluids etc.
- Networking Architecture: The networking technologies and protocols used in automotive IT still differ widely from those utilised in Classical computer environments. In modern cars, the constantly growing number of ECUs require efficient means of interconnection to implement necessary communication use cases. While in the early years of electronic automotive systems, separate analogue lines were drawn for each signal, the complex automotive systems of today require communication via shared media in order to reduce the amount of cables needed (with the aim of reducing weight, costs, difficulties with handling, etc.). Different types of field bus systems are used for this like Controller Area Network (CAN), Local Interconnect Network (LIN), Media Oriented Systems Transport (MOST) and FlexRay. Usage of Ethernet within (parts of) automotive networks is increasingly discussed and tested in the automotive industry. A great difference to Classical computer environments is that typically, automotive networks follow a broadcast strategy, i.e. one bus message of a certain sender is simultaneously received by multiple receivers which evaluate and process its content (if required). Bus systems like CAN (currently the most widely used automotive bus system) are designed for simplicity and efficiency. Unfortunately, such bus systems, which do not provide sender/receiver addresses or even authentication "out of the box", can easily be attacked once access to the internal bus systems has been established. We demonstrated this in previous work [6] [7], as have other research groups like [8] and [9], who base their work on our aforementioned papers.

These characteristics must be taken into account when designing an approach, as discussed in this work.

C. Introduction to Adaptive Systems

In order to deal with as wide a variety of attacks as possible, we need a protective system which is able to adapt to various threats. This system should also be able to adapt to malware which is unknown at the point of the incident. Our approach to design such an adaptive system is based on principles of Organic Computing [10]. Organic Computing is a concept to deal with the ever-growing complexity of hardware, software and networking. While classic engineering approaches try to develop a system by examining all possibilities (like modelling all possible states), Organic Computing goes in a different direction to cope with those cases where the complexity simply grows too vast. Already, developers often do not know which components their system will ultimately interact with – and such cases will increase in future. Instead of preparing different modes for operation in lots of different environments (accepting the risk that the developers did not foresee some of them), a system based on the principles of Organic Computing is able to adapt to different environments by itself –an adaptive system: *„In organic computing, the only task humans hold on to is the setting of goals. As the machine is autonomously organising, detailed communication between programmer and machine is restricted to the fundamental algorithm, which is realising system organisation. Application-oriented mechanisms lose the status of algorithm and are treated as data, in analogy to the transcription factors in the ontogenetic toolbox."* [11]

Obviously, an organic system would need a set of specific properties to behave in such a manner. These properties are referred to as self-x-properties. Various self-x-properties have been proposed, but self-organisation is the core concept characterising adaptive systems. Further examples are self-configuration, self-optimisation or self-healing.

Some of these self-x-properties directly match our goal to deal with new malware trends. Self-healing would best describe our approach to find a fitting reaction to malware incidents. However, the ability for self-perception is the base of any adaptive system – since systems without any information about their state and capabilities could not fulfil any of the self-x-properties. Hence our approach needs to address self-perception, too.

 ISBN: 978-1-61208-427-5

III. ORGANIC PRINCIPLES TO COUNTER MALWARE IN AUTOMOTIVE ENVIROMENTS

In this section we discuss an adaptive system for the reaction against malware incidents in automotive environments.

Automotive environments bring up unique challenges to IT security. They have specific characteristics, and many security incidents carry a broad and deep threat to users and bystanders, including endangering life and limb. This leads to numerous challenges for dealing with malware in automotive environments.

From this perspective, different application scenarios for the implementation of a fitness function arise:

- While interconnectivity is on the rise it is by no means guaranteed that a vehicle will have network access at any time. It could also be possible that a vehicle does not have any access for an extended period of time and so would have no access to necessary updates to block malware.
- Probable safety impacts of an incident make it impossible to simply 'sit it out' or 'ignore it'.

To deal with these challenges we use principles from Organic Computing to design a system which both detects and reacts to malware incidents. Hence, it implements detection and reaction and establishes self-perception and self-healing in an automotive system. As discussed in Section II-B, the individual ECUs used in an automotive environment have serious resource constraints. Therefore, the changes to each of the ECUs need to be relatively simple. In general these ECUs only handle calculations – they get input from some ECUs (or sensors) and give instructions to other ECUs (or actuators). Many of these values are transferred via insecure lines, especially field bus system, throughout the vehicle. An attacker could plug into this system and falsify, or add false values. An input checking on each individual component could detect those tampering attempts. In addition it would distribute the effort to the individual components.

To achieve this we assign a confidence measure (CM) to each source of input value an ECU receives. If the input value seems to be tampered with, the ECU would reduce the confidence in this input source. In order for this to be effective the ECU needs to be able to verify the input given. This is done in two different ways. First, we assign various sources for different ECUs – an ECU that would usually only read speed information will also evaluate the positioning information in order to double check between those two sources. While this approach implements an inter-source sanity check, we also implement an intra-source sanity check, which detects unusual changes in the input data (like the GPS sensor implying that the vehicle moved 100 miles since the last update seconds ago) and reduces the CM accordingly.

The value the ECU works with is calculated from the weighted input values it receives from various sources. If the ECU completely distrusts a certain source it would completely disregard any data from this source.

In addition, the frequency of input messages of a certain type is also monitored. Usually these messages are transmitted in regular intervals - an increase in messages could point out the injection of falsified values. Hence if the number of received messages deviates from the expected frequency, the confidence measure would be decreased. On the other hand the CM of an input source would recover over time if the input is more in line with the other sources again.

IV. IMPLEMENTATION AND EVALUATION OF A FIRST DEMONSTRATOR

This section covers the implementation of a demonstrator to evaluate and refine the approach described in Section III. We introduce the components necessary for our demonstrator, the test setups used, and the evaluation results.

A. Exemplary Simulation Environment

In order to evaluate our chosen approach it was necessary to implement a simulated environment. This environment needs to fulfil a range of various requirements:

- Vehicles consist of a network of different ECUs and actuators. Hence the simulated vehicle needs to consist of multiple ECUs and means for them to communicate with each other.
- The simulated vehicle needs to be modular so more components can be added in order to increase the complexity and feasibility of the simulation. This also allows for different test cases with alterations to the adaptive subsystems, different attack vectors or malfunctions.
- The simulation environment must be able to support the evaluation by offering exhaustive output to ensure traceability of the simulation's results.

Our implementation follows a modular approach with different types of modules as shown in Figure 1. The main component is the simulator itself which handles the physics inside the simulation and contains the 'real' system state of the vehicle. Linked to the simulator is the logging subsystem. This subsystem records the information needed for the evaluation.

Our modelled vehicles themselves consist of actuators, sensors, assistance systems and a bus. Sensors get information about the current state from the simulator. Actuators on the other hand give feedback about the actions of the vehicle to the simulator. Assistance systems are the various ECUs which use sensor information to control actuators. The communication between sensors, actuators and ECUs is covered by a communication bus which mirrors a field bus used in automotive networks.

For our first exemplary simulation environment we implemented the simulator itself, the bus, various actuators, sensors and assistance systems. As assistance systems we implemented cruise control (CC), adaptive cruise control (ACC), lane-keeping assist, anti-lock braking systems (ABS) and park distance control (PDC).

 ISBN: 978-1-61208-427-5

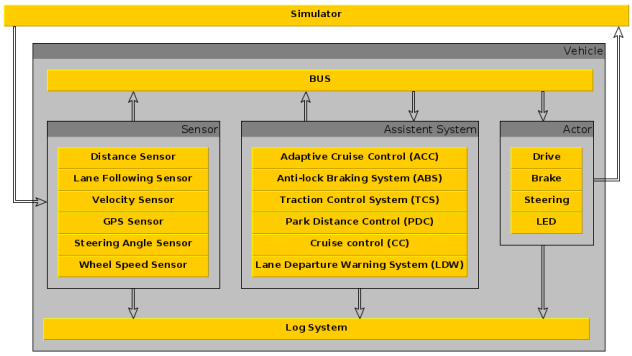

Figure 1. Architecture of our Demonstrator

B. *Exemplary Scenarios and Attacks*

The modular approach for our simulation environments allows for the easy simulation of various attacks or malfunctions within the simulated car (or cars). In order to evaluate the usefulness of our approach we formulated various scenarios:

- S1 - Defunct sensor: In this scenario the GPS sensor is manipulated in a way that it is not able to supply data anymore. This might be due to a malware attack, removal of the component or simple breakdown of the component.
- S2 - Malicious messages: In this scenario an additional component is added which adds false input values to the bus. In our case we choose the GPS sensor as main target and hence inject falsified GPS data. These injections occur at the same frequency the correct GPS data is transferred.
- S3 - Flooding attack: In this scenario we perform a basic Denial-of-Service (DoS) attack on the assistance systems, which rely on the input from the GPS sensor. We add a malicious component, which floods the bus with random GPS input data.

C. *Evaluation results*

After implanting and testing the various scenarios we could evaluate them using the data supplied by the logging subsystem.

- S1 - Defunct sensor: Over the course of time the CM of the GPS sensor went down to zero and other sources (velocity, steering angle) were used by the assistance systems relying on position data.
- S2 - Malicious messages: The assistance systems start to notice that the input given by the GPS sensor is not plausible and therefore reduced the corresponding CM. Instead they relied more on the other sources (velocity sensor, steering angle) as input. When the CM of the GPS sensor went to zero it was ignored completely.
- S3 - Flooding attack: The strong divergence from the amount of expected messages led to a quick decay of the CM of the GPS sensor. After mere moments the sensor data was not used anymore. After the flooding stopped the CM slowly recovered.

V. Conclusion and Outlook

In this work we propose a novel approach to include an adaptive reaction system against malware incidents in automotive systems. Most current approaches as established in Classical computer environments only handle the detection of a malware incident and require a manual reaction. Furthermore they rely only on pre-defined knowledge from external sources for detection strategies. With the approach presented here, an automotive system is able to adapt to new malware threats by itself. This approach not only handles malicious manipulations but also unintentional malfunction like sensor failure, for example.

However, at this point this approach is very juvenile, and several points need to be addressed in the future. For example, further potential use cases will require more complex heuristics to achieve a useful determination of the CM of different input sources. Also, a more complete evaluation will be required to get a broader impression of the gains of the proposed approach. We consider doing so in future research, either by further extensions to the presented simulation environment or by implementing the methods proposed in this work in a real automotive (laboratory) system.

Acknowledgements

We like to thank our students from our course "Praktikum IT-Sicherheit 2014/2015" for their work concerning the presented demonstrator.

This work was partly supported by German Research Foundation, project ORCHideas (DFG GZ: 863/4-1).

This work was also partly supported (definition of the scenarios derived from high level project requirements) by European Research Foundation, project SAVELEC (FP7 - SEC-2011, Grant Agreement Number 285202).

References

[1] J. Dittmann, T. Hoppe and C. Vielhauer, "Multimedia Systems as Immune System to Improve Automotive Security?", SAFECOMP 2013, Toulouse, France, 2013.

[2] James M. Aquilina, Eoghan Casey and Cameron H. Malin, "Malware Forensics: Investigating and Analyzing Malicious Code", Elsevier, ISBN 987-1-59749-268-3, 2008.

[3] M. Sikorski and A. Honig, "Practical Malware Analysis – The Hands-On Guide to Dissecting Malicious Software", No Starch Press, San Francisco, ISBN 978-1-59327-290-6, 2013.

[4] E. Skoudis and L. Zeltser, "Malware – Fighting Malicious Code", Prentice Hall International, ISBN 978-0131014053, 2003.

[5] P. Mell, K. Kent and J. Nusbaum, "Guide to Malware Incident Prevention and Handling", National Institute of Standards and Technology Special Publication 800-83, November 2005. http://csrc.nist.gov/publications/nistpubs/800-83/SP800-83.pdf (last access: 14/01/2015), 2005.

[6] Tobias Hoppe, Stefan Kiltz and Jana Dittmann, "Security threats to automotive CAN networks – practical examples and selected short-term countermeasures", Computer Safety, Reliability, and Security, Proceedings of the 27th International Conference SAFECOMP 2008, Newcastle, UK, September 2008; Springer LNCS 5219; S. 235-248; Editors:

 ISBN: 978-1-61208-427-5

Michael D. Harrison, Mark-Alexander Sujan; ISBN 978-3-540-87697-7, 2008.

[7] Tobias Hoppe, Stefan Kiltz and Jana Dittmann, "Automotive IT-Security as a Challenge: Basic Attacks from the Black Box Perspective on the Example of Privacy Threats", Computer Safety, Reliability, and Security, Proceedings of the 28th International Conference SAFECOMP 2009, Hamburg, Germany, September 2009; Springer LNCS 5775; S. 145-158; Editors: Bettina Buth, Gerd Rabe, Till Seyfarth; ISBN 978-3-642-04467-0, 2009.

[8] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, et al.: "Experimental Security Analysis of a Modern Automobile", The IEEE Symposium on Security and Privacy, Oakland, CA, May 16-19, 2010.

[9] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, et al.: "Comprehensive Experimental Analyses of Automotive Attack Surfaces", In D. Wagner, ed., Proceedings of USENIX Security 2011. USENIX, Aug. 2011.

[10] C. Müller-Schloer, H. Schmeck, T.Ungerer, "Organic Computing — A Paradigm Shift for Complex Systems, Springer" , ISBN: 978-3-0348-0129-4, 2011.

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