Future Cars: Necessity for an Adaptive and Distributed Multiple Independent Levels of Security Architecture

Alexander Camek
fortiss GmbH
Guerickestr. 25
Munich, Germany
camek@fortiss.org

Christian Buckl
fortiss GmbH
Guerickestr. 25
Munich, Germany
buckl@fortiss.org

Alois Knoll
Technical University Munich
Boltzmannstr.3
Garching, Germany
knoll@in.tum.de

ABSTRACT

Current automotive systems contain security solutions provided as singular solutions. Security mechanisms are implemented for each automotive function individually. This individual security design leads to several problems: combining several functions that are for its own secure may not result in a secure system. Furthermore, the combination of functions might also lead to situations, where mechanisms erroneously detect a security threat. This paper argues that new features, such as Car-2-Car communication or autonomous driving, will result in new information and communication technology (ICT) architectures of cars. The paper will outline basic properties of this architecture and summarize resulting security threads. We will argue that security needs to be treated in a holistic way and that the design must be suitable for adaptive, multiple independent levels of security (MILS) architecture.

Categories and Subject Descriptors

C.2.0 [**General**]: Security and protection (e.g., firewalls); C.2.4 [**C.2.4 Distributed Systems**]: Distributed Applications; D.4.6 [**Security and Protection**]: Security Kernels

General Terms

Security

Keywords

Security; Plug&Play; adaptivity; distributed MILS; secure product lifecycle; automotive

1. INTRODUCTION

Over the past 30 years, information and communication technology (ICT) has made significant innovations in automotive construction possible: from the anti-lock braking system in 1978 to electronic stability control in 1995 and emergency brake assist in 2010. Accordingly, ICT, and especially its software, has expanded significantly, from about 100 lines of code (LOC) in the 1970s to as much as ten million LOC.

This massive usage of software has however also downsides. Different research groups have reported security vulnerabilities of cars. A major reason is that security is not treated as a first-class design issue across the whole ICT architecture. In contrast, the decision if and which security functions are applied is made for each automotive function individually. This individual security design leads to several problems: combining several functions that are for its own secure may not result in a secure system. Furthermore, the combination of functions might also lead to situations, where mechanisms erroneously detect a security threat.

For a recent study [5], we interviewed 240 experts world wide, how the automotive ICT architecture will evolve in the next 20 years. The study was focused on electric cars, but its results can be mapped to cars with internal combustion engines. The study identified several architectural changes. Additionally, it identified functions of future cars that will increase the vulnerability of cars if no adequate measures are applied. All these lead to more severe consequences of successful attacks. This paper will analyze these changes and motivate the necessity of an adaptive, multiple independent layers of security (MILS) architecture.

The paper starts with an overview on the security architecture in today's cars in section 2. Section 3 will then summarize the societal and technological drivers for the upcoming change of the ICT architecture, as identified in the study. The cornerstones of the resulting architecture will be explained in section 4. The main contribution is the analysis described in section 5, which security threats will be the result of this architectural change and the motivation of requirements on the security architecture in section 6. The paper is summarized in section 7.

2. CURRENT AUTOMOTIVE SECURITY SYSTEMS

Currently produced cars contain a lot of electronic control units (ECU). For each functionality, from electronic stability control to high end driver assistance systems, suppliers provide an own ECU. These ECUs are targets to influence the system, to access the whole car, or to conduct a financial damage of the car owner. To avoid such attacks ECU suppliers stock their products with embedded suitable security mechanisms. For example, the immobilizer is such a mecha-

HiCoNS'13, April 9–11, 2013, Philadelphia, Pennsylvania, USA.

nism, which is directly integrated in the motor control unit and hinders a non-authorized person to start the engine.

As a first version, the security mechanisms were coded directly in the control software of a given ECU. After attacks, these mechanisms were improved and moved to specific hardware components [20], which can be added to standard automotive microcontroller. One of the solutions is the secure hardware extension (SHE). It was developed to provide protection of cryptographic keys and secure boot, among others. In the motor control unit SHE is implemented to secure the immobilizer.

Hardening ECUs at the hardware level was one step. Next, ECUs were hardened at the software level by secure programming. Therefore, three major topics have been identified. At first, ECUs must be programmed only by authorized subjects, e.g., certified ECU manufacturers. This is mostly solved by setting up processes, which guarantees the compliance of regulations. Besides that, the ECUs must be loaded with authorized objects. Here, a lot of effort, standards in specific groups, and research [11] was done. And as third, standard cryptographic algorithms were defined, which shall be used by ECU manufacturers.

But, in modern cars ECUs cannot provide their functionality without data exchange. For example, the immobilizer needs a command, which authorize the ignition of the engine. This command is a combination of unlocking the door and pushing the ignition button or turning a key. It will be sent to the immobilizer through an in-vehicle communication network, such as LIN, CAN, FLEXRAY, or MOST, for data exchange.

These networks become more and more a target for attackers [9]. In the past, it was simple to wiretap and to manipulate the exchanged data, because the communication was not secured at all, as shown in [12]. Later, the communication for specific applications, such as the immobilizer or diagnostics, and their data exchange became encrypted. For example EVITA HSM [1] is a solution, which is used only for vehicle communication. Furthermore, in-vehicle communication systems are topics of ongoing researches, e.g., OVERSEE [2].

Diagnosis is another application, which uses the network to collect information from log files. To get access, an On-Board Diagnostics (OBD) port exists. But, this port is not only used to read reports. It can also be used to update the firmware of an ECU. This capability was a target of attacks [7], which tried to manipulate the messages and install manipulated firmware. After these attacks were published, new standards were introduced to drive secure diagnosis with access control.

All these security mechanisms are single solutions and independent from each other. This can also be seen by analyzing the AUTOSAR standard [1]. Here, the standard only specifies the Crypto Service Manager, a component that offers generic access to standardized cryptographic routines. Higher-level services are not part of the standard and the whole security management has to be implemented application level. As a results, the mechanisms are implemented for each automotive function individually. This individual security design leads to several problems as can be seen from reported attacks: combining several functions that are for its own secure may not result in a secure system. Furthermore, the combination of functions might also lead to situations, where mechanisms erroneously detect a security threat.

[1] http://evita-project.org/

[2] https://www.oversee-project.com

3. TRENDS IN AUTOMOTIVE DOMAIN

This section summarizes the trends identified in the beforementioned study [5]. Customers will ask essentially for three capabilities: zero accidents, Plug&Play and always-on.

To decrease the number of car accidents and simultaneously enable mobility for the growing number of elderly people, cars will be equipped with more intelligent advances driver assistance systems and even be capable for autonomous driving [19]. These functions impose a twofold challenge to the ICT architecture: first of all they are absolutely safety-critical. A failure of these functions might lead to car damage and even threaten live of the passengers. The second challenge is about the close interconnection of these functions with already existing functions and sensors. Today, the functions can mainly be implemented and integrated into the car in isolated fashion and via additional electronic control units (ECU). For the new interconnected functions, this approach will not be feasible. Therefore, new functions will be rather integrated on a central platform computer.

The second capability of future cars is Plug&Play. In today's cars it is nearly impossible to add new functions after production of the car, except from the infotainment domain. Due to the fast technological progress and the long life cycle of a car, this fact will be no longer tenable. Customers will for example require an update of their five year's old car to the latest ADAS technology.

The third trend is interconnectivity of the car leading to an always-on car. The passengers of the car will require access to their data and to the Internet similar to the smart phone domain. Furthermore the ADAS functions of the car will more and more rely on data from outside the car leading to car-to-car and car-to-infrastructure communication.

Car manufacturers have already picked up these trends and started technological changes to the ICT architecture. The most important trend is to introduce a standardized run-time system called AUTOSAR [1], that enables the integration of new functions at the software component level rather than at the hardware or ECU level. The main goal is to allow several functions to execute on one computer with the final vision of a central platform computer. The required computational power of such computers is anticipated by introducing multi-core technologies.

The challenge derived from increasing connectivity demands is met by introducing high-bandwidth communication technologies such as Ethernet. Using Ethernet as communication backbone has also the advantage of easier integration with the Internet and car-to-X functionality.

4. CORNERSTONES OF FUTURE ICT ARCHITECTURES

Figure 1 depicts a future scenario for future ICT architectures in cars. Advanced functions will be executed on a central platform computer, which may itself consist of several controllers. Smart sensors or actuators will on the other hand read sensor values, preprocess the data and execute local control functions. The preprocessing step will allow a reduction of the required bandwidth. The local control functions will implement control loops with real-time

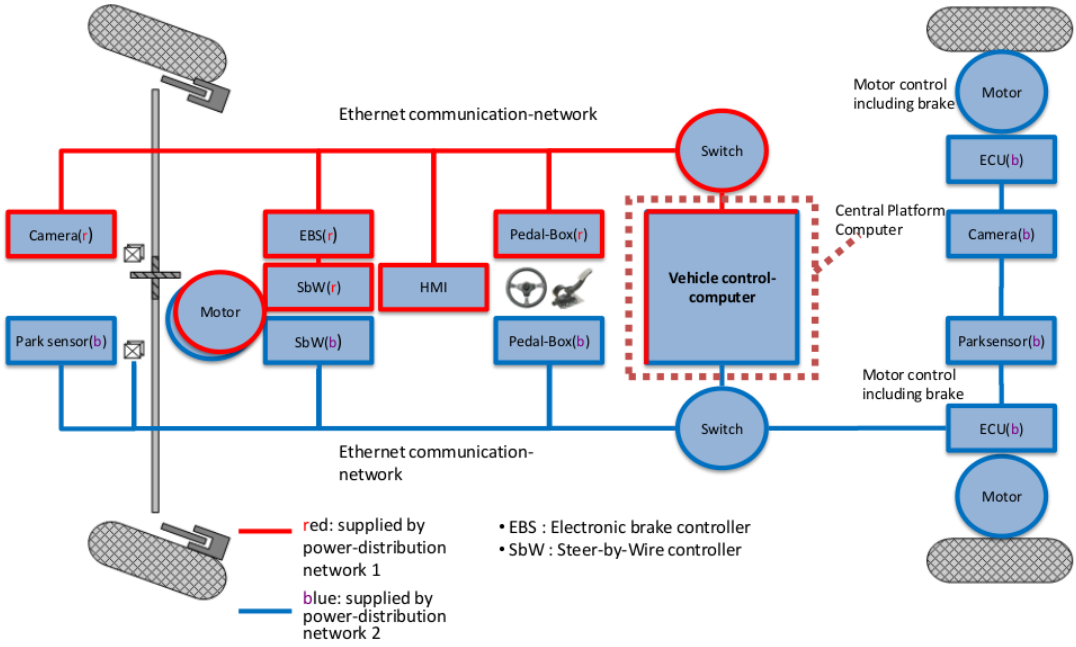

Figure 1: Possible Future ICT Architecture ©RACE, www.race-projekt.de

requirements that cannot be achieved in distributed fashion.

A communication backbone, most probably a real-time Ethernet variant, will connect the smart sensors and actuators. For safety reasons, the communication backbone will be based on physical redundancy. Sensors and actuators that are required for functions with fail-operational execution semantics will be connected to both channels.

All controllers will execute a standardized run-time system. This run-time system will enable the execution of functions with different levels of criticality on one computer. The run-time system will offer services for fault-tolerant execution and other extra-functional requirements including security.

Furthermore, it will provide functionality for Plug&Play. On the one hand, this includes the support to add software functions to the system. On the other hand, it must be possible to detect and integrate new hardware components, such as additional sensors, actuators, or computational units for the central control computer.

5. SECURITY THREATS FOR NEW ARCHITECTURE

Let our new architecture, given in Section 4, serve as a basis for a security analysis.

5.1 Attacker Model

Before we take a closer look of potential security holes, we will setup an attacker model at the beginning. The attackers are inspired by briefly described attacks in Section 2. Therefore, we classify possible attackers into different categories. Such categorizations have been studied in related work.

Howard and Longstaff [10] provide a common model to describe computer security incidents. Here, an incident is a relation of an attacker to an objective. Their seven attacker categories are primarily defined from the perspective of computer and network security. In contrast to our categorization attackers are not divided into internal and external physical access to the system.

In [4] attackers are categorized according to their physical access to the target and their skills. There, the attackers are not specific for automotive systems, and therefore they do not match for us. Bless et al. [6] classified their attackers based on several motivations. But, they do not classify their attackers according to their objectives and their access to the system.

Given that, the available attacker classifications do inadequately suit our kind of architecture. Thus, we propose a suitable attacker classification for automotive systems, which is a combination of the above with additional objectives. We classify our attackers into four skill classes, as given in Table 1:

- **Class A** includes technically unskilled persons searching for a low-risk opportunity to steal and sell some devices, or to attack and change systems. This attacker possesses low technical equipment and uses by experts pre-crafted tools. The members of this category have only minor financial resources. The car owner is part of that category. They want to improve the car's performance without extra payment for the additional functionality or install additional components. We also add the petty criminal to this category. He normally steals devices and wants to sell these. Script kiddies [14] who attack systems only for fun without special knowledge fit in this category. They want only to improve their reputation, but their attacks will harm the system and after those attacks the system is left in an unusable mess.

- **Class B** contains technically low skilled persons who have already attacked systems with weak security, and may have been caught for one of their attacks. This

Table 1: Attacker Classification

Skill Class	Class A	Class B	Class C	Class D
Automotive knowledge	medium	high	high	medium
IT knowledge	low	medium	high	high
Technical equipment	low	medium	medium	high
Financial resources	low	low	medium	high
Possible roles	car owner, petty criminal, or script kiddie	manufacturer, motorcar mechanic, or tuner	device developer, motorcar mechanic, or hacker	terrorist or organized crime

kind of attacker is cunning and experienced. The person only attacks smaller systems or systems with weak security. They possess some technical equipment, where special tools are part of. The members of this category have only minor financial resources. Device or original end manufacturers are part of that category. We also add motorcar mechanics to this category, because they know about the system and possess the needed special tools. In this category we also put tuners who improve the performance or appearance of a car.

- **Class C** covers technically high skilled persons who are specialized in attacking systems. Sometimes they use members of our other classes as a help. We put normal developers of components in this category. Sometimes these people want to increase their reputation or learn something new, known as hackers. But mostly these people attack for money or harming the company. Additionally, motorcar mechanics who are paid by car owners to tune or tweak the system are also part of this category. They know the system and can use insider knowledge. They have the tools and special equipment to perform the attack. The goal of this attacker class is to use their knowledge to improve their personal benefit.

- **Class D** includes technically high skilled or trained persons who have substantial resources. Additionally, the attacker uses members of our other classes as help. One can interpret this category as terrorists or mafia members. We define this category for criminal organizations. The goal of this attacker class is to maximize the impact by harming passengers, and getting the highest possible outcome, such as money or terror.

Furthermore, we distinguish between *restricted* and *unrestricted* physical access of the attacker (Table 2). An attacker with unrestricted access to a car can manipulate, replace, or remove components directly. Attackers with restricted access can manipulate the system only through the ICT infrastructure.

The main paradigm of security is based on the the protection of most valuable assets from the most capable attackers by a single component under the most difficult circumstances. This paradigm matches our Class D attacker. But, defending a system against a Class D attacker is costly and consumes high efforts of mechanisms. Therefore, in the upcoming analysis we will mainly focus on attackers of Class A-C.

Table 2: Physical Access

Physical Access	unrestricted	restricted
Possible roles	car owner, tuner, petty criminal, manufacturer, device developer, or motorcar mechanics	script kiddie, hacker, organized crime, or terrorist

5.2 Analysis of Classical Architectures

As shown in Section 2, a lot of attacks have been assaulted against classical architectures. These attacks were done against single functions or network components, especially at the infotainment domain. This section summaries already known threats that must be also considered in the new architecture. Here, attackers are Class B and Class C ones, who can access the system restricted and unrestricted.

Bypassing of security mechanism is a fundamental problem. For example, this allows tampering of memory content or changes in an execution path.

In [17] communication networks of embedded systems are described to support only rudimentary capabilities or are isolated. To get the best out of these limitations, most data is broadcast as plain text. Such an information leakage allows an attacker to read and manipulate messages.

Covert channels [13] are another problem in embedded systems. Here, information is transferred between two components, which is normally prohibited by the security policy.

For a better administration embedded systems getting more connected to the Internet and isolation is broken up. Thus, they are faced with threats known from IT systems. Malwares, like Stuxnet, can subvert or compromise the system.

With the change from the internal combustion engine to the electric motor new components were added to the old architecture. Thus, attacks against the new components are done with techniques [16], which work mostly in the old architectures.

Other requirements given by attacks based on social engineering and physical access, such as unauthorized changes, malicious use, or insider threats, are out of scope here.

5.3 Analysis of New Architecture

The upcoming changes increase the attractiveness and potential of new attacks for several reasons. Attractiveness is increased due to the fact that more safety-critical functions

are introduced to the car and that access to data due to the centralized architecture becomes easier. The potential of new attacks increases as Plug&Play capability and the use of standardized networks offer a basis for new attacks.

We therefore analyzed the potential attacks and grouped them into categories with respect to attacks against the central computer, the smart sensors and actuators, the network, the data handling and the Plug&Play mechanism. In the following, we summarize the results.

Our architecture of a centralized system executes all important functions and applications. Thus, the centralized system will be the main target for an attacker to get access to a specific function. It is also possible to stall an execution or an execution order to get rid of a specific function. At a worst case, an attacker can shut down the entire system by exploiting a flaw. Additionally, functionality can be activated without extra payment by circumvent security mechanism of the centralized system.

Smart sensors and actuators are the smallest components of the new architecture. They are responsible to collect data and execute commands. Therefore, attackers will try to influence them by analyzing, tampering, and circumventing their software. This can be done whether in changing the firmware or providing special crafted data, former recorded data, or malformed data. For example, tuners fake sensor information of an oxygen sensor to manually override the air and fuel settings for an internal combustion engine [3]. Additionally, a sensor could be blinded by sending a jam signal [21]. Malformed data can lead to a denial of service (DoS), when the component is busy with checking the data and cannot execute other functions.

Components are interconnected by a communication network, above described attacks can be also done remote. Additional attacks are removing a component from the system, shutting down components, or collecting information about the system, such as topology or component interconnection. Besides that, an attacker can block a component by a denial of service (DoS) or can influence the temporal behavior by shifting the time basis. These attacks can base upon security holes of communication protocols or connection components, such as switches.

A new feature of our architecture provides the possibility to integrate components by Plug&Play. This adds a potential to attack the system. Software and hardware Plug&Play allows to integrate a component, which is owned by an attacker. Thus, a component can masquerade as another component, emulate to provide better services (aka Sybil attack [8]), denial of sleep of another component, or block another component (DoS). Plug&Play allows also an attacker to collect informations or listen to data exchange, known as man-in-the-middle attack. Additionally, removing a component can tricker a reconfiguration combined with a recalculation of resources, which may block the whole system, known as Chaos attack. It is also possible to integrate software or malware, which allows an attacker to open backdoors for further attacks [15].

To support Plug&Play we need a standardized interface, which are common to developers. These interfaces are a possible target for a first attack. Therefore, an attacker could use weaknesses of the interfaces [3] to get access to the system. This access enables an attacker to start a selective attack against system internals. Otherwise, interfaces are a problem of information leakage. Most interfaces provide more knowledge of itself or the underlying system as needed. By a simple interface probing an attacker could gather information and feed the interface with every possible alternative to analyze the reaction. Then the attacker will use the gathered information to get access to the system.

[3]http://www.ehow.com/how_5409757_fake-out-oxygen-sensor.html

6. REQUIREMENTS OF FUTURE SECURITY ARCHITECTURE

Security is a system property. Current security in vehicle systems is designed to protect either a function or a communication channel. As shown in Section 2 and 5, a lot of attacks exists. Hence, today's attacks and our centralized platform computer motivate holistic security architecture. This implies that a system must be designed with security in mind from the beginning. A decomposition of system functions is needed to generate successively simpler modules. This allows to support the paradigm, simpler is securer. There, it is possible to trust these simple modules to work under all conditions.

One cornerstone of our architecture is the central computer and its interconnection with smart sensors and actuators. These components must be secured by a holistic approach to support the execution of applications with different levels of security. Therefore, high-assurance security architecture is need.

Avionics, robotics, aeronautics, and military domains have already faced similar problems, and as a consequence have abandoned classical architecture approaches. The result was a new paradigm, which is known as multiple independent levels of security (MILS). Here, the concept is based on separation and information control flow. MILS uses three different levels, a separation kernel, middleware, and applications [18]. The security must ensure to be non-bypassable, evaluable, always invoked, and tamperproof. Based on the previous analysis, we believe that future cars must be based on a **MILS architecture**.

The separation kernel provides temporal and spatial partitions to separate parts of the system to avoid interferences. Between partitions the separation kernel establishes a secure transfer of control. Such kernels are very small with 4,000 lines of code. This allows to verify the correctness with mathematical or formal methods.

Upon the separation kernel a variety of middleware can reside in different partitions. These levels are responsible for creating application components. They run in user or non-privileged mode. This avoids to harm the whole system when a problem occurs, and will only affect their own partition. Additionally, it provides a secure end-to-end inter-object message flow.

The application level implements specific security functionality, such as firewalls or cryptographic modules. These components and non-security ones are mostly developed by other vendors. Consequently, MILS architectures are based on composition.

To ensure tamperproofness MILS runs self-tests during initialization. This shall ensure that the integrity of the platform is valid. When the tests fail a recovery mechanism must be provided. Additionally, the systems must isolate faults and must avoid a cascade of faults. Therefore, only the important part of the system, e.g., the separation ker-

nel, must run in privileged mode. All other functionality, such as partitioning communication system (PCS) or the middleware, will only run in user space.

However, implementing just a standard MILS architecture is not enough. As a lot of computational power is required, the central computer will be based on several computers. Therefore, a secure communication between these different computers and also with smart sensors and actuators is required. Hence, future cars will require a **distributed MILS architecture**. Distributed MILS architectures require that the communication between processors be managed by the MILS system [2]. The way how this can be achieved is the focus of several ongoing research projects, e.g., the projects EURO-MILS[4] and D-MILS[5] funded under the Seventh Framework Program.

But even a distributed MILS architecture will not be enough for future cars. As shown Section 4, Plug&Play is one main feature of future vehicle architecture. In contrast to classical MILS systems Plug&Play capable systems need to be more adaptable during runtime. Hence, the configuration of the system should be modifiable to support additional functionality. This allows also to keep an aging security system up-to-date or to equip a system with future security components. In short, Plug&Play motivates **adaptive and distributed MILS**.

With a later installation of components new dependencies are inserted. It must be checked whether these new dependencies break some old ones or whether connections are established, which are not allowed by the security policy. Furthermore, also resource constraints must be taken into account. During installation of a new component, the system must check whether enough resources are available for the new and the already installed applications. At run-time, the resource assumptions must be continuously checked to detect and omit violations.

If components of different criticality levels are installed, the system must ensure that the different levels are separated and no security constraints are violated.

Finally, Plug&Play also enables update of security mechanisms itself. Systems are increasingly attacked, exploits are created for profit, and numbers of malwares are rising. But, simultaneously new technologies evolve and countermeasures are developed, which will lead to changes of security mechanisms. This is especially important for automotive systems, which last more than ten years in usage. However, MILS systems see security as part of the design and as a built into the system from the beginning. Thus, there will be no changes of the system's security mechanism during the lifecycle of MILS. Based on the lifecycle of a car and new arising attacks the architecture needs to be upgradeable. This motivates a secure product life cycle (SPLC), where system internals and important components can be patched or updated. This is another important aspect, which needs to be covered by an adaptive and distributes MILS architecture.

In summary, at least the following research questions need to be answered for a secure automotive future:

- Security Architecture: which mechanism can be implemented at which security layer and what are the appropriate interfaces? It must be clarified at which level (application, node, system) security mechanisms are implemented.
- Security Goals: which guarantees regarding security can be offered at the level of distributed systems and how can these guarantees be implemented? Furthermore, how can applications state their requirements independent of the concrete implementation? How are these requirements enforced? This includes securing the data flow between the different components according to confidentiality, integrity, and authenticity.
- Automatic Configuration: how can the system derive a valid configuration satisfying the security constraints of the applications? In traditional systems, a system developer takes over the role of the configurator. In the future, the Plug&Play capability including configuration must be offered as a service by the platform.
- Policy Management: how to implement policy management taking into account Plug&Play? In traditional systems, the communicating components are defined at design time. In a Plug&Play-capable system, the communication partners are determined at runtime. Therefore, policies must be formulated in a different fashion. Furthermore, new policies might be introduced and mechanisms must be defined to guarantee their correct and secure behavior within the system.
- Authentication: how to establish a secure communication and interaction in a dynamic distributed system? Mechanisms must be defined to detect and integrate new components. Additionally, hidden malicious components need to be detected and isolated.
- Intrusion Detection: how can the system detect malicious behavior of components and what are appropriate countermeasures? These mechanisms have to be implemented in a way that the platform can still provide a basic set of services to the remaining applications.
- Audit: how to save all data related to security during run-time to enable a retrospective analysis in case of incidents?
- Secure Product Lifecycle: what are the mechanisms to ensure an up-to-date security architecture? New arising attacks and evolving technologies imply the necessity to keep the architecture up-to-date. A concept for upgrading the architecture must be developed.
- Business Models: who is the owner of the data within the car and who is allowed to earn money with the data? In current cars, the access to the data is restricted: the car manufacturer controls access and usage of data ensuring national data privacy laws. In Plug&Play-capable cars, data access rules will be defined most probably by all stakeholders.

7. CONCLUSION

This paper summarized the results of a study on upcoming changes of the ICT architecture in the automotive area. In particular, we focused on the effects on how security is

[4]http://www.euromils.eu/

[5]http://www.fortiss.org/en/research/projects/distributed_mils/

achieved in cars. Based on the current state of the art to design security mechanisms for each automotive function individually, we discussed the cornerstones of future ICT architectures and analyzed the new security threats of such an architecture. As a result, we argue that future cars will need an adaptive and distributed MILS system. While MILS architectures are already state of the art in several domains such as avionics and distributed MILS is already an identified research topic, the design to support adaptivity will be an interesting research topic for the future.

Acknowledgments

This work is partially funded by the German Federal Ministry of Economics and Technology under grant no. 01ME12009 through the project RACE[6].

8. REFERENCES

[1] AUTomotive Open System ARchitecture (AUTOSAR) Release 4.0.

[2] J. Alves-Foss, W. S. Harrison, P. Oman, and C. Taylor. The mils architecture for high-assurance embedded systems. *International Journal of Embedded Systems*, 2:239–247, 2006.

[3] R. J. Anderson. What we can learn from api security. In B. Christianson, B. Crispo, J. A. Malcolm, and M. Roe, editors, *Security Protocols Workshop*, volume 3364 of *Lecture Notes in Computer Science*, pages 288–300. Springer, 2003.

[4] R. J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley computer publishing. Wiley, 2nd edition edition, 2008.

[5] M. Bernhard, C. Buckl, V. Döricht, M. Fehling, L. Fiege, H. von Grolman, N. Ivandic, C. Janelle, C. Klein, K.-J. Kuhn, C. Patzlaff, B. Riedl, B. Schätz, and C. Stanek. *The Software Car: Information and Communication Technology (ICT) as an Engine for the Electromobility of the Future, Summary of results of the "eCar ICT System Architecture for Electromobility" research project sponsored by the Federal Ministry of Economics and Technology*. ForTISS GmbH, March 2011.

[6] R. Bless, G. Grotewold, C. Haas, B. Hackstein, S. Hofmann, A. Jentzsch, A. Kiening, C. Krauß, J. Lamberty, M. Müter, P. Schoo, L. Völker, and C. Werle. A security model for future vehicular electronic infrastructures. In *8th Embedded Security in Cars (escar)*, 2010.

[7] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, pages 6–6, Berkeley, CA, USA, 2011. USENIX Association.

[8] J. R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, UK, 2002. Springer-Verlag.

[9] T. Hoppe, S. Kiltz, and J. Dittmann. Security threats to automotive can networks — practical examples and selected short-term countermeasures. In *Proceedings of the 27th international conference on Computer Safety, Reliability, and Security*, SAFECOMP '08, pages 235–248, Berlin, Heidelberg, 2008. Springer-Verlag.

[10] J. D. Howard and T. A. Longstaff. A Common Language for Computer Security Incidents. Samdia Report SAND98-8667, Sandia National Laboratories, Albuquerque, New Mexico 87185 and Livermore, California 94550, October 1998.

[11] M. S. Idrees, H. Schweppe, Y. Roudier, M. Wolf, D. Scheuermann, and O. Henniger. Secure automotive on-board protocols: a case of over-the-air firmware updates. In *Proceedings of the Third international conference on Communication technologies for vehicles*, Nets4Cars/Nets4Trains'11, pages 224–238, Berlin, Heidelberg, 2011. Springer-Verlag.

[12] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. Experimental security analysis of a modern automobile. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 447–462, Washington, DC, USA, 2010. IEEE Computer Society.

[13] B. W. Lampson. A note on the confinement problem. *Commun. ACM*, 16(10):613–615, Oct. 1973.

[14] N. Mead, E. Hough, and T. S. II. Security Quality Requirements Engineering. Technical Report CMU/SEI-2005-TR-009, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania,, 2005.

[15] T. Nash. An undirected attack against critical infrastructure a case study for improving your control system security. *US-Cert Control Systems Security Center*, 1.2, 2005.

[16] C. Paar, K. Schramm, A. Weimerskirch, and W. Burleson. Securing green cars: It security in next-generation electric vehicle systems.

[17] G. J. Pottie and W. J. Kaiser. *Principles of Embedded Networked Systems Design*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[18] J. M. Rushby. Design and verification of secure systems. In *Proceedings of the eighth ACM symposium on Operating systems principles*, SOSP '81, pages 12–21, New York, NY, USA, 1981. ACM.

[19] G. Silberg and R. Wallace. Self-driving cars: The next revolution. Technical report, KPMG, 2012.

[20] M. Wolf and T. Gendrullis. Design, implementation, and evaluation of a vehicular hardware security module. In *Proceedings of the 14th international conference on Information Security and Cryptology*, ICISC'11, pages 302–318, Berlin, Heidelberg, 2012. Springer-Verlag.

[21] W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming sensor networks: attack and defense strategies. *Network, IEEE*, 20(3):41 – 47, may-june 2006.

[6] http://www.projekt-race.de/