Security Analysis of Automotive Architectures using Probabilistic Model Checking

Philipp Mundhenk, Sebastian Steinhorst, Martin Lukasiewycz
TUM CREATE, Singapore
<firstname.lastname>@tum-create.edu.sg

Suhaib A. Fahmy
School of Computer Engineering, Nanyang Technological University, Singapore
sfahmy@ntu.edu.sg

Samarjit Chakraborty
TU Munich, Germany
samarjit@tum.de

ABSTRACT

This paper proposes a novel approach to security analysis of automotive architectures at the system-level. With an increasing amount of software and connectedness of cars, security challenges are emerging in the automotive domain. Our proposed approach enables assessment of the security of architecture variants and can be used by decision makers in the design process. First, the automotive Electronic Control Units (ECUs) and networks are modelled at the system-level using parameters per component, including an exploitability score and patching rates that are derived from an automated or manual assessment. For any specific architecture variant, a Continuous-Time Markov Chain (CTMC) model is determined and analyzed in terms of confidentiality, integrity and availability, using probabilistic model checking. The introduced case study demonstrates the applicability of our approach, enabling, for instance, the exploration of parameters like patch rate targets for ECU manufacturers.

Categories and Subject Descriptors: C.3 [Special-purpose and application-based systems]: Real-time and embedded systems
General Terms: Algorithms, Design, Security
Keywords: Security, Automotive, Model checking, Networks

1. INTRODUCTION & RELATED WORK

Within the past two decades, the amount of electronics and software in automotive systems has increased rapidly. Today, top-of-the-range vehicles comprise up to 100 Electronic Control Units (ECUs) and several heterogeneous bus systems, implementing a variety of applications ranging from comfort to active safety functions.

While functional and safety requirements were always fundamental considerations in the design of automotive architectures, security is becoming a major challenge [1] for many emerging applications. Consumers and car manufacturers understand the benefits of cloud-connected services in vehicles, making modern infotainment systems, adaptive route planing, or over-the-air updates of software possible. However, it is also well understood that these applications present the risk of vulnerability to hacking attacks.

This work was financially supported in part by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

DAC '15 June 07 - 11, 2015, San Francisco, CA, USA

http://dx.doi.org/10.1145/2744769.2744906.

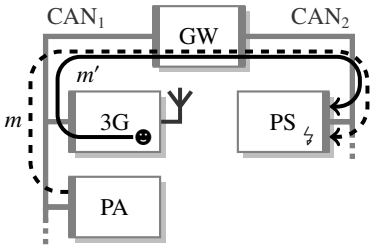


Figure 1: *Illustration of a possible exploit in an automotive architecture. In normal operation, the park assist (PA) controls the power steering (PS) with the message stream m that is sent via the gateway (GW). If the telematics module (3G) is hacked (☻), a message stream m′ with an identical identifier ID(m) = ID(m′) can control the steering (ϟ).*

In current approaches, vehicle networks are shielded with firewalls at the external interfaces to ensure security. However, internal security is often not considered. Upon breach of a firewall, the network is openly accessible to the attacker. Such an example attack is illustrated in Figure 1.

Studies show that traditional automotive architectures that were not designed with security in mind are highly vulnerable [2, 3]. In particular, hacking a safety-critical control function can have severe and even fatal consequences. In [4], different attacks and their influence on a control function are modeled, showing that it is easily possible to tamper with driver assistance functions if their communication is not secured. Novel techniques are developed to integrate safety and security, protecting in-vehicle communication networks such as Controller Area Network (CAN) [5] and FlexRay [6]. In the automotive domain, encryption and authentication for internal buses has to be implemented efficiently, an approach to authentication has been proposed in [7].

While current state-of-the-art approaches for security in automotive systems consider separate components, our approach is targeted at the system-level. Towards the analysis of systems, model checking has been applied to verify computer networks [8] and protocols [9]. These approaches only detect system vulnerabilities if each single component vulnerability is modeled. However, vulnerabilities are often not known at design time and, therefore, a probabilistic approach, as proposed in [10] for Denial-of-Service (DoS) exploits, can abstract this uncertainty. In [10] an approach to analyzing security protocols with probabilistic model checking based on attack cost is proposed. By contrast, we are quantizing the security of all traffic, based on the underlying automotive architecture, including DoS attacks on protocols.

Contributions of the paper. In this paper, we present a methodology for the security analysis of automotive architectures at the system-level. We take advantage of the fact that ECU topologies, communication networks and message streams are known at design time. Using probabilistic model checking enables us to quantify

the security of automotive architectures in terms of confidentiality, integrity and availability.

In Section 2, we outline the proposed framework which comprises the following three steps:

1. The considered automotive architecture is transformed into a Markov model.
2. An assessment of components is performed to determine the transition rates for the Markov model.
3. The definition of a property for the model checker is carried out to determine an appropriate security value.

With the complete Markov model and property, a probabilistic model checker determines security values for the architecture under consideration.

In Section 3, a detailed description of our methodology is given. We increase the granularity of an architecture under test to the submodule level by including all network interfaces, bus systems, ECUs and messages. These submodules are converted into a Markov model to transform the architecture into a graph. The edges of the graph are weighted by probabilistic rates, representing exploitability and patching rates of each submodule. We propose to determine the rates by a standardized security assessment of every submodule separately. Using the Markov model with assigned rates, we define properties to evaluate the architecture. Our framework allows the definition of properties for any submodule in the architecture, thus enabling us to evaluate every security aspect relevant while considering the complete architecture.

The results of this analysis for a set of example architectures are presented in Section 4. For three architecture alternatives, we obtain and compare results. We also show the possibility to explore different exploitability and patching rates for a given architecture. Finally, we discuss the general applicability and scalability of the proposed approach before concluding the paper in Section 5.

2. FRAMEWORK

In the following, we give an overview of our analysis approach. After the problem description, we explain the main steps of our analysis flow.

2.1 Problem Description

Automotive networks consist of a high number of ECUs that perform dedicated tasks such as sensing, computation, or actuation. Functions are implemented in a distributed fashion that requires communication via different shared bus systems like CAN and FlexRay. Particularly, the communication via shared buses using predefined and often unencrypted message streams becomes a major issue once the system is compromised.

Generally, components or functions are developed independently in the automotive domain and are subsequently integrated into the architecture. While security aspects might be considered for use cases at subsystem-level, vulnerabilities at system-level are often not taken into account. As a remedy, the proposed framework is designed to answer the following questions during design and integration:

- What influence do component vulnerabilities have on the security of a specific function?
- Is a certain architecture design decision beneficial in comparison to an alternative in terms of security?
- How much effort should be invested in the consideration of security during implementation of specific components?

Note that we are considering system-level security, abstracting aspects of internal ECU security, such as secure boot, key storage, etc.

Since automotive architectures are highly heterogeneous systems comprising many different components, it is mandatory to model all important aspects at the system-level. For instance, communication systems used in vehicles differ greatly in their support of security aspects, particularly in terms of availability. Thus, appropriate models for these communication systems are required. ECUs might also be connected to multiple communication buses, resulting in different potential attack paths with varying probabilities. Finally, to characterize the security properties of ECUs and other components, it is necessary to quantify the rates at which they can be exploited and how fast they can, in turn, be patched. For this purpose, security assessment and Automotive Safety Integrity Level (ASIL) values should be taken into account when modeling the system.

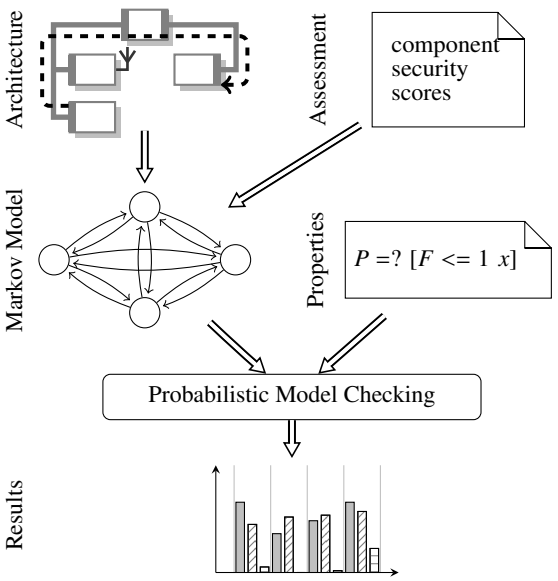


Figure 2: *Illustration of the proposed framework: (1) The architecture is transformed to a Markov model, (2) transition rates are determined by a security assessment per component and (3) a property for the model checker is defined. Finally, the probabilistic model checker returns quantified results that support decision making at the system-level. system-level.*

2.2 Analysis Flow

Our security analysis approach for automotive architectures is outlined in Figure 2. It comprises three steps: Model transformation, component assessment, and property definition. Finally, the security of an architecture is determined with a probabilistic model checker.

Model Transformation. To be able to process the architecture in terms of security, we break it down into the relevant components. These include buses, ECUs, messages, interfaces, etc. For each component, a Markov model defines the exploitability and patching rate. Finally, the Markov models are combined into a single system that enables probabilistic model checking. Depending on the system under consideration, this transformation can be flexibly extended or adapted. We propose a specific transformation approach in Section 3.1.

Component Assessment. Correspondingly to the transformation of the architecture, it is necessary to assess the individual components in terms of their exploitability and patching rates. These values determine the transition rates of the Markov model. For exploitability, we propose to use the Common Vulnerability Scoring System (CVSS) to assess the vulnerability of components relying on an established standard [11]. Considering the ASIL values of components, we observe that patching rates strongly depend on safety requirements of components that might require costly re-testing and validation in case of software changes. These rates can be adapted by the user, depending on development processes and existing data. While the assessment should ideally be performed on subcomponent level, incorporating ECU security elements, we use a simplified component-based approach. The granularity of the approach can be scaled up to the required level (see Section 5). The component assessment with CVSS and ASIL is detailed in Section 3.2.

Property Definition. By defining specific properties, we are able to investigate certain aspects of the architecture in terms of security. In contrast to a steady-state analysis, which analyzes convergence to a steady state of the system at some point in time, this analysis is significantly more powerful. Thus, a property can for instance be the cumulated time that a function is exploitable within a period of 10 years. The definition of appropriate properties is detailed in Section 3.3.

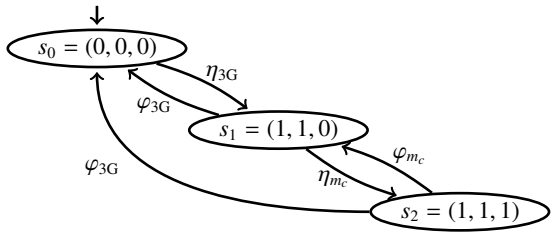


Figure 3: *Illustration of a simplified Markov model that analyzes the exploitability of message m in the architecture in Fig. 1 in terms of confidentiality with $n_{\max} = 1$. The states are composed by $s = (s_{3G}, s_{CAN_1}, s_{m_{conf}})$. Each atomic state s_i represents how many exploits for the components 3G, CAN_1, and m exist.*

3. METHODOLOGY

In this Section, we describe the steps required to create a Markov model from an architecture (Section 3.1), weight the edges of the model by a component assessment (Section 3.2) and define properties to analyze the model (Section 3.3).

When creating a model from an architecture, we model ECUs and interfaces by the number of exploits existing. New exploits are discovered (e.g., by security researchers) with rate η and exploits are patched (e.g., by Over-The-Air (OTA) updates) with rate φ. Networks and buses are considered passive, depending on the least secure state of all attached ECUs. Messages in turn are modeled based on the security principles of confidentiality, integrity and availability:

- *Confidentiality* describes the protection from reading messages by an unauthorized entity.
- *Integrity* describes the protection from creation and modification of messages.
- *Availability* describes protection from interruption or removal of messages.

Each of the principles is analyzed separately, depending on the protection of the message (none, cryptographic hash, encryption) and the communication networks employed for transmission.

Example. To illustrate this process, consider a minimal architecture with a telematics ECU (*3G*) attached to a CAN bus (compare *3G* in Figure 1). The CAN bus is used to transmit a message m which is neither sent nor received by the telematics ECU. After transformation of this architecture into a Markov model, we obtain the model shown in Figure 3. This is a very simplified transformation, not considering all ECUs, interfaces or message states. Further, we only consider one exploit per module.

Analyzing this model, we see that from a secure state (s_0), an exploit is discovered in the telematics unit with rate η_{3G}. Once an exploit is discovered, the CAN bus is immediately considered to be exploitable (s_1). The telematics unit can be patched with rate φ_{3G}. If this does not happen and an exploit for the protection of message m, which the telematics ECU does not have the keys to, is discovered (with rate η_{m_c}), we advance to state s_2 and the message can be exploited. From now on, the contents of the message m cannot be considered confidential any longer, as the contents could be altered. To remedy this situation, the message protection and the telematics unit should be patched. In our example model this happens with rates φ_{m_c} and φ_{3G}, respectively. Alternatively, the access for attackers can be denied by patching the telematics unit with rate φ_{3G}.

This is a very simplified example, requiring only a small subset of our proposed modeling techniques. In the following, we will explain the full set of transformation rules, as well as the assessment of components and the analysis of the model with probabilistic properties.

3.1 Model Transformation

Transformation of an automotive communication architecture into a Markov model is required to be able to analyze the communication architecture with probabilistic methods. Our transformation considers all communication participants (i.e. ECUs) and interconnections (such as bus systems and networks). We further separate ECUs into interfaces for every communication system in order to analyze the impact of different communication systems. Furthermore, we analyze messages transmitted in the system.

Terminology. To model an architecture for security analysis, we require the set of all ECUs $e \in E$ and all internal buses $b \in B$. We also require the set of all interfaces I_e of every ECU e. An interface $i_b \in I_e$ connects an ECU e with a bus or an external network $b \in B_e$. Thus, we define an ECU as $e = \{I_e, B_e\}$ and a bus as $b = \{E_b\}$, where E_b is the set of ECUs on bus b. To analyze a message m, the sending ECU s_m, the set of receiving ECUs R_m and the set of buses B_m over which the message is transmitted, are required: $m = \{s_m, R_m, B_m\}$. This data is obtained from the architecture, including the fully scheduled set of messages. The maximum number of exploits considered for every module at one point of time is defined as $n_{\max}$.

ECUs & Interfaces. To model the architecture, every ECU needs to be split into interfaces. For each interface i_b, the exploitability $\varepsilon(i_b) \geq 0$ is analyzed separately, based on the probabilistic exploitability rate η_{i_b} increasing the number of exploits n if the bus is exploitable:

$$\varepsilon(i_b) = n \xrightarrow{\eta_{i_b}} \varepsilon(i_b) = n + 1 \quad \text{if } \varepsilon(b) > 0, \text{with } i_b \in I_e, 0 \leq n < n_{\max} \tag{1}$$

The resulting value for exploitability describes the number of parallel exploits $\varepsilon(e)$ existing in the ECU, $\varepsilon(i)$ in the interface or $\varepsilon(m)$ for the message. We use $n_{\max}$ to limit the maximum number of exploits and thus reduce the complexity of our model. The inverse of this relationship defines the patching of a security flaw in an interface i_b, based on the patching rate φ_{i_b}:

$$\varepsilon(i_b) = n + 1 \xrightarrow{\varphi_{i_b}} \varepsilon(i_b) = n \quad \text{if } \varepsilon(b) > 0, \text{with } i_b \in I_e, 0 \leq n < n_{\max} \tag{2}$$

The exploitability of each ECU $\varepsilon(e)$ is based on the exploitability of all interfaces of this ECU:

$$\varepsilon(e) = \bigvee_{i \in I_e} \varepsilon(i) \tag{3}$$

Buses & Networks. Similarly, the exploitability $\varepsilon(b_c)$ of a CAN bus b_c is dependent on the exploitability of every attached ECU:

$$\varepsilon(b_c) = \bigvee_{e \in E_{b_c}} \varepsilon(e) \tag{4}$$

In case a FlexRay bus b_f is used, additionally, the bus guardian i_{bg} needs to be exploited, before an ECU e can transmit freely on the bus:

$$\varepsilon(b_f) = \bigvee_{e \in E_{b_f}} \varepsilon(e) \wedge \varepsilon(i_{bg}) \tag{5}$$

Networks or buses directly connected to the internet, such as 3G, are always considered to be exploitable, as attackers have continuous access to these. We set a constant exploitability value of 1 to model this property:

$$\varepsilon(b_{3G}) = 1 \tag{6}$$

Messages. The exploitability of a message is subdivided into the impact categories availability A, integrity G and confidentiality C. While availability is highly dependent on the employed communication system, integrity and confidentiality are based on the message protection. Specifically, cryptographic hashing and encryption address these categories. When using a CAN bus, availability can not be guaranteed if any of the buses used for message transmission are exploited:

$$A(m) = \neg \bigvee_{b \in B_m} \varepsilon(b) \tag{7}$$

Confidentiality and integrity of a message m can not be guaranteed if the sending or receiving ECUs are exploited, even if the message is encrypted. To ensure real-time performance, we assume symmetric encryption, such that the key for message encryption is stored both on sending and receiving ECU. Secure key storage is not analyzed here, but could be integrated as a submodule into the ECU module. Confidentiality C and integrity G behave similarly, but depend on the protection mechanisms used for the respective message m. In the remainder of this subsection, we show the transformations for confidentiality. The rules apply identically for message integrity by replacing $C(m)/\eta_C/\varphi_C$ in the equations by $G(m)/\eta_G/\varphi_G$.

Confidentiality of a message can be violated if the sender or receivers are exploitable:

$$C(m) = \neg \bigvee_{e \in \{s_m, R_m\}} \varepsilon(e) \tag{8}$$

Furthermore, confidentiality can be attacked by any ECU on any of the buses used for transmission of m. The probability of such an attack depends on the strength of the employed encryption algorithm and its implementation. To describe this behavior, we define the following relation:

$$C(m) = 1 \xrightarrow{\eta_C} C(m) = 0 \quad \text{if} \bigvee_{b \in B_m} (\varepsilon(b)) = 1 \tag{9}$$

Consequently, a message is not exploitable, if all ECUs on all transmitting networks are not exploitable.

The inverse of the above operations describes the patching of a flaw in the message encryption/hashing, and is described as:

$$C(m) = 0 \xrightarrow{\varphi_C} C(m) = 1 \quad \text{if} \bigvee_{b \in B_m} (\varepsilon(b)) = 1 \tag{10}$$

This concludes the transformation of the architecture into the states of a Markov model.

3.2 Component Assessment

To be able to analyze the Markov model generated in the previous section, we need to assign weights to the edges. These rates represent the probability over time that a device is exploitable and the rate in which a device can be patched.

Exploitability Rates. In this work, we determine rates via an adjusted version of the CVSS [11]. CVSS has been developed as an open standard to assess vulnerabilities in software systems and is maintained by the National Institute of Standards and Technology (NIST). It can generate a score for the security of a component, based on the assessment of units in multiple subscores and categories. The criteria used in the exploitation subscore are similar to the criteria required for interfaces in the automotive domain. We utilize the exploitation subscore (see Table 1) and adjust it further to adapt it to the automotive domain. Based on the scores for the subcategories, we calculate the exploitability score

$$\sigma = 20 \cdot AV \cdot AC \cdot Au. \tag{11}$$

Based on the exploitability score σ, we calculate the rate

$$\eta = \sigma - 1.3. \tag{12}$$

We normalize the exploitability rate to 1 year.

As an example, we illustrate the assessment of the 3G interface of the telematics ECU. As this device is connected to the internet, the Access Vector is across multiple networks ($AV = 1$). Due to its open surface, we assume that the device is hardened against attacks, thus the access complexity is high ($AC = 0.35$). To access the device, we further assume that multiple authentication steps are required ($Au = 0.45$). From Equation (11), we determine $\sigma = 3.15$. Based on Equation (12), the exploitability rate can be determined as $\eta = 1.85$.

In the remainder of the paper, we use these CVSS-based assessments for the interfaces of all ECUs.

Patching Rates. The amount and rate in which patches can be supplied to the vehicle is dependent on many factors. Firstly, the development time for a patch sets an upper bound to the patching rate. Additionally, the amount of tests required can be extensive, if a safety-related function needs to be altered to implement the security patch. Thus, we base our assignment of patching rates on the ASIL level of the functionality to be patched. The ASIL-dependent patching rates are shown in Table 2. While a safety-related device, such as the gateway (*GW*) can only be patched at relatively long intervals, non-safety-related functions, such as a telematics unit (*3G*), can be patched at short intervals, as fewer tests are required.

Using such transition rates in the Markov model allows us to analyze it on a time basis, resulting in a Continuous-Time Markov Chain (CTMC).

3.3 Property Definition

After creating a CTMC model from the architecture and assigning transition rates as discussed in the previous subsection, we need to define the goals of our analysis. These goals are defined as properties of the model. A common property evaluated on CTMC models is a steady-state analysis. Here, the probabilities to be in each state at any sampled point in time can be calculated with conventional matrix operations. Consider the example in Figure 3. With rates $\varphi_{3G} = \varphi_{m_c} = 52$ (weekly) and $\eta_{3G} = \eta_{m_c} = 2$ (bi-annually), we obtain the transition rate matrix:

Category	**Subcategory** (Description)	**Value**
Access Vector (AV)	L (Local) Accessible only on device	0.395
	A (Adjacent Network) Accessible via directly attached bus	0.646
	N (Network) Accessible via any number of networks	1
Access Complexity (AC)	H (High) Device is generally secured	0.35
	M (Medium) Device is partially secured	0.61
	L (Low) Device is not secured	0.71
Authentication (Au)	M (Multiple) Multiple authentication steps required	0.45
	S (Single) One authentication step required	0.56
	N (None) No authentication is required	0.704

Table 1: *The categories for the CVSS exploitation subscore and our interpretation of these for automotive networks (adapted from [11]).*

$$Q = \begin{pmatrix} -s_0 s_1 - s_0 s_2 & s_0 s_1 & s_0 s_2 \\ s_1 s_0 & -s_1 s_0 - s_1 s_2 & s_1 s_2 \\ s_2 s_0 & s_2 s_1 & -s_2 s_0 - s_2 s_1 \end{pmatrix} \tag{13}$$

$$= \begin{pmatrix} -2 & 2 & 0 \\ 52 & -54 & 2 \\ 52 & 52 & -104 \end{pmatrix} \tag{14}$$

Note that the values on the diagonal of the matrix are the negative sum of the rates of the row. The steady-state solution $\pi Q = 0$ yields a stationary distribution

$$\pi = \begin{pmatrix} 0.96296 & 0.036338 & 0.000699 \end{pmatrix}. \tag{15}$$

Consequently, at any given point in time, the probability to be in state s_2 where m is exploitable, is 0.0699%. This stationary information, however, is not conclusive for practical security questions. Thus, we are interested, e.g., in the probability that the model reaches state s_2 at least once within one year. To express this for CTMCs, we need to define a reward-based property, counting each occurence of the state. We follow the syntax in [12] and define: $R\{s_2\} =? [F < 1]$.

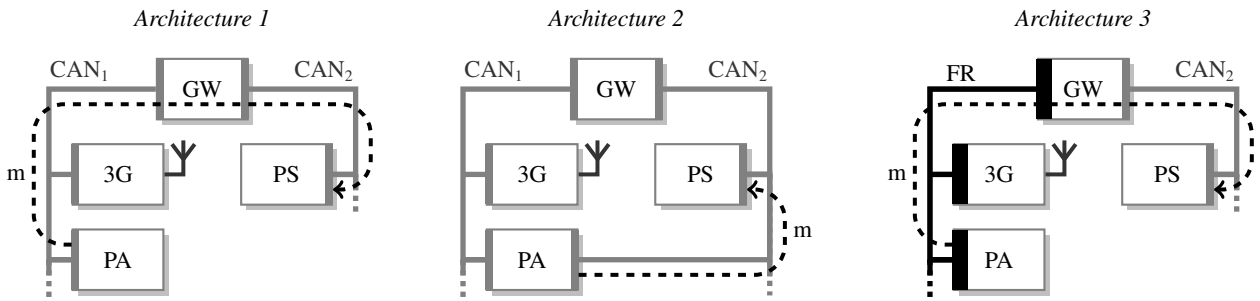
To evaluate such properties, model checking algorithms have to be applied. Using these algorithms, we can analyze every state of each submodule generated in the above steps for reachability or probability. Reachability constraints, as used in conventional model verification, can only define security in absolute terms (e.g. $P =?[\text{F ECU}_1 > 0]$ or *Can ECU 1 be exploited at any point in time?*). Security, however, can not be defined in absolute terms. Assuming absolute security of any device is not realistic. Thus, we have to define properties which allow us to draw conclusions based on the underlying time basis of the model.

An example question for cumulative probabilities could be: *How long is message m potentially exposed to an availability attack within 1 year?* For this question we are looking for the cumulative time of all occurrences of $A(m)$, expressed through rewards:

$$R\{A(m) > 0\} =? [F < 1] \tag{16}$$

4. EXPERIMENTAL RESULTS

In this section, we discuss our experimental setup and show the potential of our analysis methodology by applying the framework to a case study based on the three architectures illustrated in Figure 4. The analysis of the architectures, modeled with the methodology



GW - gateway, PS - power steering, PA - park assist, 3G - telematics unit, CAN - controller area network, m - message stream, FR - FlexRay

Figure 4: *Three architectures used to compare different approaches to security. Architecture 1 transmits message m on the same bus as the telematics unit is located. Architecture 2 adds a dedicated connection for message m. Architecture 3 introduces a FlexRay bus for the transmission of message m.*

introduced in Section 3, is performed with the probabilistic model checking tool PRISM [12].

The computations for one property in the architectures and with $n_{\max} = 2$ require between 15 minutes and 1.5 hours on a conventional desktop computer with an Intel Xeon quad-core processor at 3.2 GHz. Every PRISM computation runs as a single process with negligible memory usage. The runtime correlates with the number of states in the generated Markov model. The models for our example architectures in Figure 4 have between 400,000 and 1.2 million states. All results are given as the percentage of time the message *m* is exploitable within 1 year.

4.1 Architecture Evaluation

We evaluate three different architectures that are based on real-world systems. We only consider a relevant subsystem of the complete vehicle architecture as illustrated in Figure 4. The considered architectures comprise two bus systems connected by a gateway, a telematics unit, and an automatic parking assistance application, including 2 ECUs. The automatic parking assistant has been reduced to one message stream transmitted between the parking assistant controller and the power steering of the vehicle.

Architectures and Component Assessment. Architecture 1 resembles the example in Figure 1. Since it is assumed that the security of this system is not optimal, we also consider two alternative architectures. In Architecture 2, the message stream *m* is sent via an additional connection directly on CAN_2, avoiding an exposure of the stream on the bus that is directly connected to the telematics unit. In Architecture 3, we replace the CAN bus with a time-triggered FlexRay bus where a schedule is defined at design time with a fixed assignment of slots to devices.

The exploitation and patching rates for our examples can be found in Table 2. We assume that critical ECUs, such as the gateway and the telematics unit, are hardened against attacks. The exploitation rate of message *m* depends on the security features used. We consider three variants: Unencrypted, Cipher-based Message Authentication Code (CMAC) with 128 bit key and Advanced Encryption Standard (AES) with 128 bit key. While an unencrypted message is instantly exploitable, a CMAC protected message provides integrity, while an AES protected message can ensure confidentiality.

The patching rates for all devices are based on the ASIL evaluation. The resulting values are listed in Table 2.

Results. The results of our analysis are illustrated in Figure 5. It can be observed that cryptographic hashing with CMAC 128 only improves security in terms of integrity while encryption with AES 128 is effective for integrity and confidentiality. This validates our results as cryptographic hashing only prevents creation of messages while encryption also prevents reading.

In general it can be observed that neither cryptographic hashing nor encryption improves the security values significantly. This is a counter-intuitive result that can be explained as follows: By hacking the Park Assist (PA), the cryptographic hashing and encryption, respectively, are compromised and lose their effectiveness. Particularly the relatively low patching rate of the PA due to its ASIL values results in a high exploitability of the device.

Module	**Interface**	η **(CVSS v2 Vector)**	φ **(ASIL)**
Park Assistant (PA)	CAN_1/CAN_2 /FR	1.2 (AV:A/AC:H/Au:S)	12 (C)
Power Steering (PS)	CAN_2	1.2 (AV:A/AC:H/Au:S)	4 (D)
Gateway (GW)	CAN_1/CAN_2 /FR	1.2 (AV:A/AC:H/Au:S)	4 (D)
Telematics (3G)	CAN_1/FR 3G	3.8 (AV:A/AC:L/Au:S) 1.9 (AV:N/AC:H/Au:M)	52 (A) 52 (A)
FlexRay Bus Guardian (BG)	local	0.2 (AV:L/AC:H/Au:S)	4 (D)
Message (m) integrity	unencrypted CMAC128 AES128	∞ (instant) 1.2 (AV:A/AC:H/Au:S) 1.2 (AV:A/AC:H/Au:S)	- - -
Message (m) confidentiality	unencrypted CMAC128 AES128	∞ (instant) ∞ (instant) 1.2 (AV:A/AC:H/Au:S)	- - -

Table 2: *Results of the security assessment for exploitation rates and patching rates. Devices such as gateway, telematics unit and FlexRay Bus Guardian are specifically hardened against attacks. Message assessment depends on the mode to be evaluated (integrity, confidentiality). Message availability is addressed through the underlying bus system.*

It can further be observed that Architecture 2 does not improve the security significantly in comparison with Architecture 1 and in some cases it even becomes worse. Here, the low patching rates of the PA and the Gateway (GW) lead to high exploitabilities of these devices, resulting in an exposed CAN_2. At the same time, the PA in Architecture 2 is exposed to two buses resulting in higher exploitability. Connecting the PA to CAN_2 might even result in further security issues as the device might be exploited as a bridge to attack other functions. This would result in a significantly worse security value for Architecture 2 for functions that are implemented on CAN_2 only.

In FlexRay, the existing bandwidth is divided in a time-triggered manner. Devices can only transmit in their slot and are denied by the bus guardian to transmit in other slots. This safety measure also has a significant security impact as devices can not maliciously transmit messages in other timeslots. This leads to an overall reduction of the attack surface as devices which are part of the communication or the bus guardian need to be infiltrated to attack message *m*.

4.2 Parameter Exploration

To evaluate the effect of different exploitability rates and patching rates, we analyze the sensitivity of the exploitability of message *m* in Architecture 1 to these rates for the entrance point ECU *3G*. As the *3G* ECU is the entrance point to the architecture, the exploitation rate of message *m* heavily depends on the exploitation and patching rates for the *3G* ECU. We independently analyze exploitation and patching rates between once per decade ($\eta_{3G} = \varphi_{3G} = 0.1$) and once per hour ($\eta_{3G} = \varphi_{3G} = 8760$). When varying the exploitation rate, the patching rate is set to $\varphi_{3G} = 52$; when varying the patching rate, the exploitation rate is set to $\eta_{3G} = 1.9$. The results are shown in Figure 6. They exhibit exponential behavior. Hence, we can conclude that

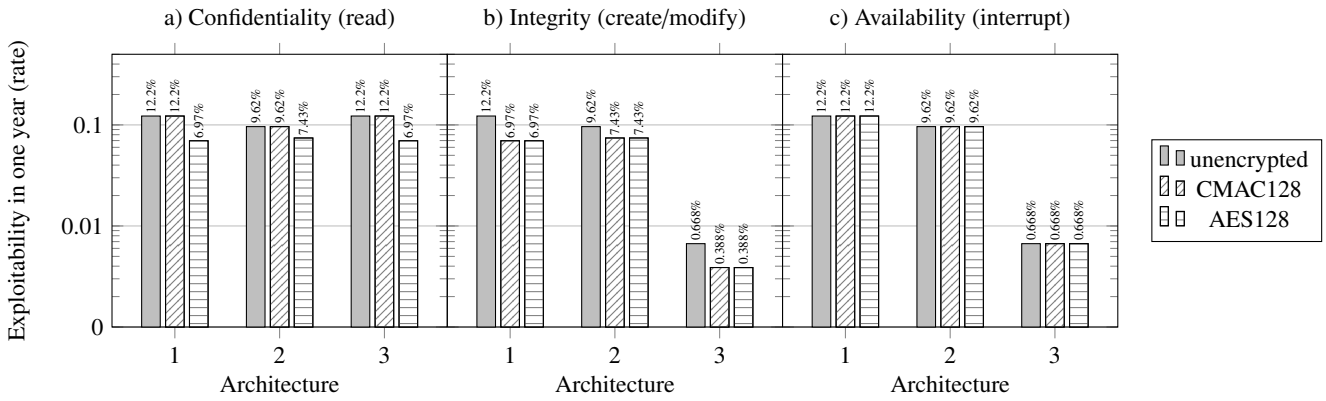


Figure 5: *Results of the analysis of the architectures shown in Figure 4. We analyze message m in terms of Confidentiality, Integrity and Availability for different protection mechanisms (unencrypted, CMAC 128, AES 128) across all three architectures. Results clearly show less exploitation potential for better encryption and more carefully designed architectures. In terms of availability, support from the bus system is required to ensure reasonable security.*

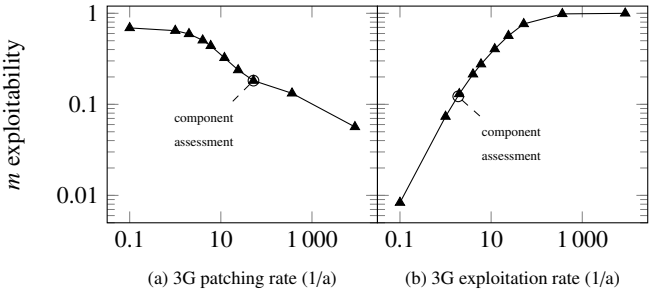


Figure 6: *Exploration of parameters to evaluate influence on complete architecture security. In (a), the patching rate of the entrance point is varied, while in (b), different strengths of security are employed in the telematics ECU forming the start point to a vehicle attack.*

while changes at the lower end of the exploitation resistance/patching spectrum have a rather large impact on the system, higher rates do not significantly help optimize security. Assuming a threshold of 0.5% exploitability, a reasonable patching rate for an internet-connected ECU without other access methods would be around $\varphi = 6$ (every 2 months). In case the exploitation rate is reduced by further securing the device, an exploitation rate of maximum $\eta = 12$ (once a month) needs to be guaranteed to keep exploitability under 0.5%.

Such an analysis can be performed for every element in the architecture. Thus, depending on the architecture, devices can either be hardened against attacks or patching rates can be contractually agreed upon between the Original Equipment Manufacturer (OEM) and suppliers.

4.3 Scalability

In verification approaches, the model size is usually a limiting factor due to the exponential characteristic of the state space. On the other hand, we showed that with an appropriate abstraction of a function that involves the reduction to relevant components and a single message stream, it is possible to evaluate realistic automotive architectures in a reasonable amount of time.

While more complex functions that involve a higher number of devices and components result in larger state spaces and runtimes, there is plenty of potential to reduce these values significantly. Currently, the PRISM model checker we use does not merge states with instantaneous transitions. These transitions are created when constructing the system model from component submodels and do not exist in the real world. Thus, they can be removed safely. Detecting states that can be merged within a preprocessing step can reduce complexity further as the number of states and runtime are strongly correlated in probabilistic model checking. In summary, we showed that probabilistic model checking is a feasible approach for quantifying security of automotive architectures and our future work will address the scalability of the approach.

5. CONCLUSION

In this work we have proposed a methodology for security analysis of automotive communication architectures. By analyzing the system-level architecture at design-time, we can quantify the security of every element in terms of confidentiality, integrity and availability. The results of this analysis can help in design decisions for the overall architecture and the security capabilities of components. For the first time, the impact of individual components on the overall security of an architecture can be determined. We analyze an architecture by generating a corresponding Markov model. Edge weights are determined through a security assessment of components. We define security properties and analyze these over the model with a probabilistic model checker. The results show that this method can be used to find security flaws in architectures, which cannot be identified at the component or subsystem level.

Future work will address scalability issues by the implementation of a targeted model checker. With improved performance, we will be capable of analyzing more fine grained models and more complex systems, e.g., comprising Ethernet. Furthermore, we will be able to generate a set of best practices for automotive architectures, based on our security analysis. A combination of security and reliability analysis, as well as the integration of ECU internal security measures is planned.

6. REFERENCES

[1] F. Sagstetter, M. Lukasiewycz, S. Steinhorst, M. Wolf, A. Bouard, W. R. Harris, S. Jha, T. Peyrin, A. Poschmann, and S. Chakraborty. Security challenges in automotive hardware/software architecture design. In *Prof. of DATE*, 2013.

[2] 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 *Proc. of SP*, 2010.

[3] 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 *Proc. of USENIX Security*, 2011.

[4] A. Wasicek, P. Derler, and E. A. Lee. Aspect-oriented modeling of attacks in automotive cyber-physical systems. In *Proc. of DAC*, 2014.

[5] C.-W. Lin, Q. Zhu, C. Phung, and A. Sangiovanni-Vincentelli. Security-aware mapping for CAN-based real-time distributed automotive systems. In *Proc. of ICCAD*, 2013.

[6] G. Han, H. Zeng, Y. Li, and W. Dou. SAFE: Security-aware flexray scheduling engine. In *Proc. of DATE*, 2014.

[7] R. Zalman and A. Mayer. A secure but still safe and low cost automotive communication technique. In *Proc. of DAC*, 2014.

[8] R. Ritchey and P. Ammann. Using model checking to analyze network vulnerabilities. In *Proc. of SP*, 2000.

[9] G. Lowe. Towards a completeness result for model checking of security protocols. *Journal of computer security*, 7(2):89–146, 1999.

[10] S. Basagiannis, P. Katsaros, A. Pombortsis, and N. Alexiou. Probabilistic model checking for the quantification of DoS security threats. *Computers & Security*, 28(6):450–465, 2009.

[11] M. Schiffman, G. Eschelbeck, D. Ahmad, and S. Romanosky. *CVSS: A Common Vulnerability Scoring System*. National Infrastructure Advisory Council (NIAC), 2004.

[12] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. of CAV*, 2011.