Secure Automotive Gateway – Secure Communication for Future Cars

Stefan Seifert
Regensburg University of Applied Sciences, Germany
Continental Automotive GmbH, Body & Security, Germany
stefan.seifert@oth-regensburg.de

Roman Obermaisser
University of Siegen, Germany
roman.obermaisser@uni-siegen.de

***Abstract*—The main focus of the paper is to secure the on-board communication of automobiles. The current trend in the automotive domain is to incorporate technologies known from the consumer segment (e.g., WLAN, Ethernet) into the car. This makes it easier for an attacker to attack the on-board networks of the car, even without having physical access. To detect attacks against the automotive networks such as CAN and FlexRay, we introduce the concept of the so called "security gateway" which is part of the automotive architecture and is located on transition points, where different networks connect with each other. A language was created to specify the correct application behavior and to configure the security gateway. Using this representation of the application behavior the security gateway not only detects failures caused by an attacker but also detects failures caused by malfunctions.**

***Index Terms*—Automotive, FlexRay, CAN, Security, Simulation**

I. Introduction

Vehicle safety and reliability of the electronic car systems have always been major topics in the automotive industry. The complexity of nowadays automobiles has risen over the past few years and top of the shelf automobiles have up to 100 different Electronic Control Units (ECUs). These ECUs are interconnected with each other over different bus systems such as FlexRay, CAN and most recently Ethernet (using BroadR Reach as the physical layer [1]).
Also the number of external interfaces has increased over the past years. Wireless interfaces include mobile Broadband, WLAN, Bluetooth, and others. Examples of interfaces where physical access is necessary are OBD-II, USB and DVD.

Over the last years, various papers have identified different vulnerabilities in the current automotive architectures as well as the used bus systems. Automotive systems need to increase their security as various attacks are already possible [2], [3]. Different research papers have also shown that the common automotive bus systems lack fundamental security properties [4] [5] [6] [7] [8].

In addition, research revealed the ability for an attacker to control different subsystems of the car, e.g., steering, braking, acceleration and display [9] when physical access to the bus systems is available.

We introduce a so called "Security Gateway" that secures the automotive network from outside attacks as well as attacks from the inside. The gateway is centralized and ideally located on a transition point, where different networks connect with each other. Our goal is a classification of communication by distinguishing between correct application behavior as well as failures and anomalies. There is a huge amount of data sent in a car and it is not feasible to store every transmission to analyze it at a later stage. Therefore, the decision whether the application behavior is incorrect has to be made in real time by the "Security Gateway".

To determine abnormal application behavior the current application state as well as the system state and the time domain need to be considered together. We introduce a generic model to specify the correct application behavior based on generic communication types, while also taking the current system state of the car into consideration.

In conventional IT networks firewall and IDS/IPS systems are common, but those systems are not useable for automotive networks, as they do not support real-time requirements. In desktop/server applications the time constraints are different to real-time automotive networks, where the messages have to be processed within very strict time boundaries. Furthermore, the approach of regular firewall and IDS/IPS systems is different from our proposal.

In contrast, our approach would not be feasible in conventional IT environments, because in general the systems and the networks are dynamic and change rapidly. Therefore, it would not be practical to create rules for every application and network communication, as those networks change fast and people can easily install new software or software updates. In the automotive domain we have a static system structure, which will not or only slightly change during its lifetime.

We introduce an approach to model the system behavior in the car. Instead of one large monolithic model, we propose a model that can be split into smaller sub-models describing different subsystems and their communication. These models are also aware of the different states and modes of operation a car can be in. Studnia et al. [10] is describing a rather similar research proposal, where the state of the ECUs is modeled and the focus lies on the CAN bus. In contrast, our attention is on the communication between the ECUs with generic support for different communication protocols. With our modeling approach we aim to have the possibility to describe the communication in a generic way. We abstract from the specific communication protocol and the implementation

978-1-4799-4905-2/14/$31.00 ©2014 IEEE

of the application by focusing on the relations and timings of different messages. Therefore, we are not only able to detect attacks but also failures caused by malfunctions. For example, if a safety system violates its correct application behavior the gateway is able to detect this failure.

The paper is structured as follows. Section II describes the related work, section III gives an overview about the attacker model, section IV outlines the system model, section V gives an overview about our concrete realisation. In section VI we discuss our results and the limitations of our gateway. Section VII outlines future work. The paper is concluded by section VIII.

II. Related Work

The combination of safety and security issues is still a main research focus and many different projects have formed around the goal to create a secure automobile. One of these projects is SEIS, its main focus is to integrate IP-based communication into the vehicle network. Security is considered since the beginning of the project to provide secure in-vehicle communication. To accomplish this goal Glass et al. [11] propose a security zone model for automobiles. Different subsystems are grouped into zones according to their criticality.

EVITA tries to protect security-relevant components against tampering and to secure the communication inside the vehicle. Wolf et al. [12] propose a so called HSM (Hardware Security Module) for secure communication between the ECUs in the vehicle network.

GENIVI was founded by different OEMs with the goal to create a reference infotainment platform. Security is considered during the creation of the reference platform [13].

Sagstetter et al. [14] describe the challenges to create a secure hardware and software architecture for automobiles. They emphasize that security of the car has to be ensured over the complete life-cycle. Also Camek et al. [15] analyze the current automotive architectures and propose a MILS (Multiple Independent Levels of Security) architecture, which is based on separation. Muter et al. [16] present different ”automotive detection sensors” to recognize security threats of in-vehicle networks. For example, one sensor is a formality sensor which determines if a message has the correct size, header, checksum, etc.

Even if a secure software and hardware architecture can be established, the security of the system also has to be maintained. Not only is it necessary to secure the automobile from remote attacks, e.g. over the Internet but also from ”near” attacks. CE-Devices are quite popular nowadays and those devices are also getting integrated in the car. For example they are used for listening to music or use the car’s own Internet connection to surf the web. Attacking the car through those devices might be a worthwhile approach. Bouard et al. [17] introduce an automotive security proxy to counteract the mentioned problems. Therefore, CE-Devices can securely communicate with the car and attacks can be mitigated. A similar approach is presented by Joy et. al. [18], namely a so called ”smart gateway” is added to the on-board network which all mobiles devices have to authenticate themselves with.

III. Attacker model

An attacker classification considering the attacker’s knowledge can be seen in table I. Attackers with a different knowledge level can achieve different goals and it is very hard to anticipate the knowledge level of an attacker. Even though in our case the thief might have a ”low” knowledge level, he still is able to buy the equipment engineered by for example a hacker with a high knowledge level, who is able to circumvent a sophisticated keyless entry system. This will then allow the thief to achieve his goal of stealing the car. Security problems only have to be found once and can be reused indefinitely – until they are fixed.

Attacker	Knowledge	Goal
Tuner	medium - high	Unlock features/modify car
Hacker/Cracker	high	Recognition
Competition	high	Examine the car
Thief	low - medium	Steal the car
”Terrorist”	low - high	Harm passenger/people

TABLE I
Classification based on the attackers knowledge

Table II defines the different entry points. Each row in table II needs its own protection mechanisms against attackers. The goal is to have as many different layers of protection as possible. Therefore, an attacker has to circumvent all the different layers of security to accomplish a successful attack.

An attacker who has physical access to the bus system is able to directly send messages, but there is still the security mechanism on that particular level, for example the Security Gateway. Incidentally, for this case an appropriate countermeasure might also be to physically harden the on-board network.

Based on the range of the different interfaces, we can distinguish two important types of attacks. Far range attacks can target any car as long as it is somehow remotely accessible (e.g., the car has a public IP address). Those kind of attacks might be most problematic as many cars can be easily attacked.

Secondly there are near range attacks, where the attacker has to be in the vicinity of the car to attack it. For example, when using wireless interfaces like WLAN or Bluetooth the attacker has to get into the vicinity of the car. Therefore, the attack is limited to a specific car or just a few cars.

Some interfaces allow both near range and far range attacks. For example, if the car is at home and is connected to the home WLAN it still might be accessible through the Internet, even though it might not have a cellular interface. But as long as WLAN is used to only integrate CE-Devices into the car, it is still a near accessible interface.

For this paper we focus on securing the internal networks. In our opinion this is an important step for a secure on-board network, as the current widely used automotive bus systems lack security properties [2] [3] [4] [6] [8] [7] [19]. For internal networks different attack vectors are possible to send malicious

Type	Layout	Range	Example
Physical	Internal (Networks)	Near	CAN, FlexRay, ...
Physical	Internal (Interfaces)	Near	CD/DVD, USB, OBD-II, ...
Wireless	External	Near	NFC, Bluetooth, WLAN, ...
Wireless	External	Far	Mobile Broadband, GPS, Radio, CAR2X, ...

TABLE II
CLASSIFICATION BASED ON ENTRY POINTS

messages into the desired network. An example attack tree for such an attack can be seen in figure 1. This attack tree is not complete, but it gives a brief overview about different possibilities for an attacker.

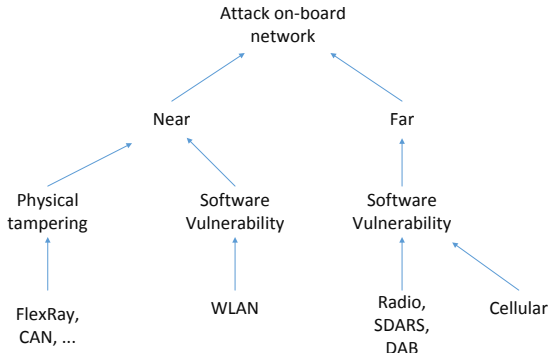

Fig. 1. Example attack tree

If the goal is to attack the on-board network, the attacker has to use any of the mentioned entry points. He can either choose to go near the targeted vehicle, if he decides to use physical tampering to get direct access to the bus system. He can then directly connect to the CAN bus and send messages to the network. Alternatively, he gains access to the WLAN of the car where he might be able to exploit vulnerabilities which make it possible to send message to the internal networks, e.g. over the infotainment system, which has already been demonstrated by Checkoway et al. [20].

If the attack is carried out from afar, cellular services can be used. For example if the car is equipped with remote unlocking capabilities via a telematics unit, a software vulnerability could be exploited to unlock the car. Such an attack has also already been demonstrated by Don Bailey [21]. Also far attacks via the radio system are possible [22].

IV. System Model

The general idea of the security gateway is to create a modular software component, which monitors the attached networks such as FlexRay, CAN, Ethernet and Real-Time Ethernet (see figure 2). Due to its modularity it can be easily extended to support other protocols or interfaces. The gateway can also run side-by-side with other software on an ECU. The gateway has to be at a central junction point of the whole architecture in order to monitor the different networks. FlexRay and CAN communication is monitored based on their broadcast communication. Ethernet on the other hand is a fully switched topology with point-to-point communication. One possible solution would be to integrate the gateway into the switch or to mirror the communication to the security gateway.

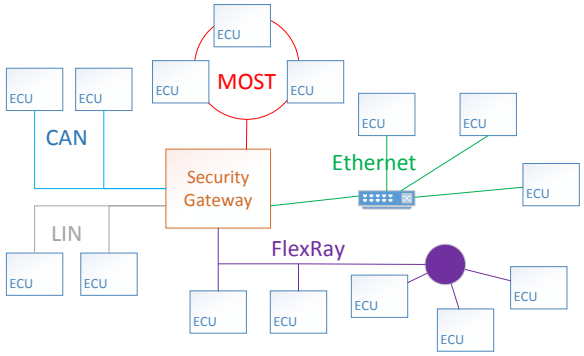

Fig. 2. Example car architecture

Given that the security gateway is located at a central junction point, it can detect failures in the attached networks. It does currently not distinguish between intentional and accidental failures. Hence a detected failure can be an actual attack but it can also be a system malfunction. The security gateway will categorize both as a failure and will act upon it.

In any specific mode of operation (e.g. driving, engine off, etc.) only a subset, of all the available messages in the car, is valid on a specific interface. Everything not part of this set is classified as a failure. This is only correct for stateless communication behaviors, communication behaviors with states can be described as graphs. A failure is detected when no transition is possible by consuming the received message.

After the successful detection of a communication failure, the gateway has to initiate a proper countermeasure or it can report the incident to another security entity which has knowledge about all systems and can determine relations between different incidents. This module might also be part of the gateway. To actually detect communication failures the security gateway has access to two different databases. Apart from those databases the gateway needs to process the received messages and detect failures in real-time. In case of stream communications, which are meant for the user (e.g. surfing the web or map navigation), real-time processing within strict time boundaries is typically not necessary. Slight time variances in the processing are hardly noticed by the user, similar to existing IDS/IPS solutions. The contrary is the case when processing e.g. FlexRay communication. In this case the gateway has to process the messages in real-time and has to react in a timely manner to successfully detect a failure and to prevent disruption of the communication. The reason is that in the modern automobile we have electronic safety systems communicating via the real-time buses. For example, messages for the ABS system have to be forwarded within strict time boundaries, to assure the system can react to safety incidents.

A. Communication Log Database

The "Communication Log Database" is used to store the recent communication history. Each packet received by the gateway, is timestamped and stored in the database. Information stored in the database has to be trimmed regularly, after a set threshold is reached.

B. Intentional Application Behavior Specification

This database describes the valid application behavior. It contains a generalized model of the application behavior, which is derivable from protocol specification files such as Fibex for FlexRay or CANDB+ for CAN. The information about the intentional application behavior is stored in an XML file. We created an XSD schema to reflect different communications types. This XSD schema is the semantic representation which is used to describe the communication. The security gateway is able to to read and execute the resulting XML files, which describe the application behavior.

The abstract definition of the different communication behaviors (see table III) enables us to define the application behavior without deeper knowledge about the underlying bus system or protocols. Therefore, also the processing in the gateway does not need any deeper knowledge about the protocol.

We use hierarchical timed automata [23] to describe the different communication behaviors. Each state of the automaton has an implicit error transition if the conditions for the transition are not met. Furthermore, in each state a transition to the state S_{Mode} is possible. This transition is executed whenever the mode of operation of the car changes. From this state it is possible to switch to the timed automata which is valid for the current mode of operation.

*1) **Event-triggered communication**:* In the event-triggered communication definition, for example for the CAN communication, we can define the minimum and maximum interarrival time. CAN, in contrary to FlexRay (static segment), does not have a deterministic message transmission. Therefore, the timing (minimum and maximum interarrival time) is typically defined for each message.

Those are the constraints when the message is being expected to be transmitted. Besides those boundaries, if a message violates its timing constraints, the priority of the messages (for example the ID of a CAN message) has to be considered to determine if this is an actual failure. It might be far worse if a safety-relevant message violates its timing constraint. For example, when a safety-relevant incident occurs unimportant messages like messages for the infotainment system might be suppressed, until the incident has been handled by the system. But if a message misses its defined deadline a failure is being detected regardless of the priority of the message.

Therefore, margins have to be defined manually for each message or the whole communication, to define how violations of the timing constraints are being handled by the gateway. Transitions can also be without temporal constraints, if the message or event can happen at any time without temporal boundaries. If a transition does not have temporal boundaries denial of service attacks are possible, hence this should be avoided.

Furthermore, there is the possibility to define references to other messages, therefore the gateway can verify and relate messages to one another. Also timing constraints between different messages can be created and it is possible to model the application behavior depending on the content of the message. Furthermore, the content of messages can be verified to be within certain boundaries.

An example can be seen in figure 3. The states S_1, S_2, S_4, S_7, are reached by consuming messages sent from the client to the server and the states S_3, S_6, S_8 are reached by consuming messages sent from the server to the client. The clock is represented by x.

We have an initial start state S_1, the message *M1* has to arrive while x is between 10ms and 20ms. From the state S_2 either message *M2* with *ID=42* or *ID=23* can be consumed to reach state S_3 or S_5. The transition from state S_2 to state S_5 has no time constraint, whereas to reach state S_3 message *M2* has to arrive while x is between 40ms and 45ms.

The transition from state S_2 to state S_5 will reset the clock. State S_8 defines the finite state where the communication ends, for example in a diagnose session other communications inside the car might not have a finite state as they are cyclic communications.

This communication model is also necessary for the dynamic segment of FlexRay, or any other event-triggered communication.

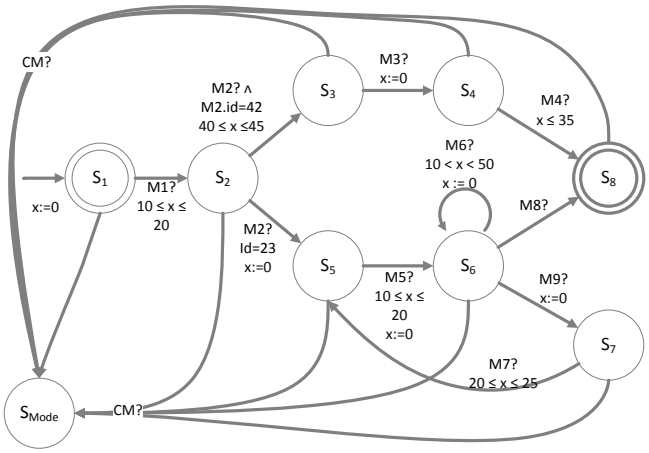

Fig. 3. Example event-triggered communication timed automaton

*2) **Time-triggered communication**:* For the static segment of FlexRay it is necessary to create a time-triggered definition and to define the slot, base cycle and cycle repetition in the automaton.

With this information the gateway is able to verify that the messages arrived in the correct slot and at the correct time.

To describe the time-triggered communication we also use a hierarchical timed automaton. The difference to the event-

Type	Example
Time-Triggered	FlexRay (Static segment), TTEthernet, TTCAN
Event-Triggered	FlexRay (Dynamic segment), CAN
Stream	Server/Client Communication (TCP/IP)

TABLE III
COMMUNICATION BEHAVIOR CLASSIFICATION

triggered automaton is that every transition must have a timing constraint, namely the specific time interval based on a global time base where the message is expected to be received (cf. Figure 4). In contrast to the event-triggered and stream automata, references to more than one message are not possible since the application behavior is determined solely by time. In addition, the content of the message can also be verified.

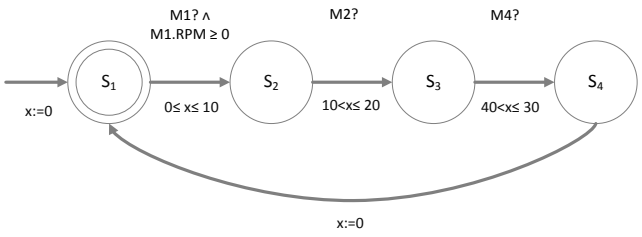

Fig. 4. Example time-triggered communication timed automaton

*3) **Stream communication**:* For a stream communication (Figure 5) we can also model dependencies between messages and verify the content of messages, just as described in the event-triggered communication. The difference to the previously described communication behaviors is that with the stream communication model the possibility exists to define data flow buffers and to mitigate buffer overflows.

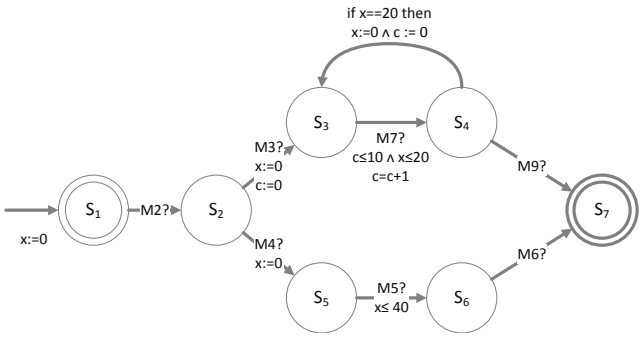

Fig. 5. Example stream communication timed automaton

The presented three specifications can be used to model the intentional application behavior. An example time-triggered definition describing a FlexRay communication is presented in listing 1. This description has to accurately describe the correct system behavior. Deviations during run-time will lead to the detection of a failure. Therefore, when the intentional application behavior changes the XML files have to be adjusted as well, if this is not the case the new functionality will be incorrectly detected as a failure, because the security gateway is working with an automaton that is not correctly reflecting the communication.

In addition, the loading of new application behavior definitions has to be protected, one possible solution can be to integrate key management and only allow signed application behavior definitions. In addition, automobiles have different *operational modes*, e.g. off, key in ignition, ignition on, engine running and so on. In all these different modes of operation, only a certain subset of all valid network messages between the different ECUs is allowed. For example, when the car is off sending ”start engine”-packets is not allowed or a specific protocol communication is only allowed in a certain mode. Needless to say the gateway has to be aware of the current mode of operation the car is in. For each mode appropriate application behavior definitions have to be created, so the correct definition for the current state will be used.

Listing 1. Intentional application behavior

```
<?xml version="1.0" encoding="utf-8"?>
<EventTriggeredCommunication
xsi:noNamespaceSchemaLocation=
"EventTriggeredCommunication.xsd">
<StateMachine>
 <State StateId="1">
  <Transitions>
   <NextState>
    <StateId>2</StateId>
    <Conditions>
     <MinimumArrivalTime>44
     </MinimumArrivalTime>
     <MaximumArrivalTime>56
     </MaximumArrivalTime>
     <Messages>
      <MessageId>ABSdata</MessageId>
     </Messages>
    </Conditions>
   </NextState>
   <NextState>
    <StateId>3</StateId>
    <Conditions>
     <MinimumArrivalTime>44
     </MinimumArrivalTime>
     <MaximumArrivalTime>56
     </MaximumArrivalTime>
     <Messages>
      <MessageId>EngineData</MessageId>
     </Messages>
    </Conditions>
   </NextState>
...
 <Bus Name="PowerTrain">
  <Messages>
   <Message MessageId="EngineData">
```

```
    <Header>
     <PacketType>EngineData</PacketType>
     <PacketID>64</PacketID>
     <PacketLength>8</PacketLength>
    </Header>
    </Message>
     <Message MessageId="ABSdata">
      <Header>
       <PacketType>ABSdata</PacketType>
       <PacketID>C9</PacketID>
       <PacketLength>8</PacketLength>
      </Header>
     </Message>
...
```

V. Realisation

The prototype is implemented as part of a simulation that supports FlexRay and CAN. For the simulation of the in-vehicle network we use CANoe v7.6. The included example "FlexRaySystemDemo" is used for FlexRay and "CanSystemDemo" for CAN. Therefore, the prototype is implemented in a simpler environment than a complete car network. Nevertheless, we are confident our approach can be transferred to a real car network and we even think it might be useful in other networks, e.g. in an airplane.

The gateway itself is implemented in C# and uses the Vectors XL Driver Library to send and receive messages on the supported bus systems. The gateway stores the message information of received messages as well as detected failures with timestamps in a SQLiTE database (cf. communication log database).

The application behavior definitions are generated from the Fibex/CANdb file of the example and stored as XML files. Those XML files (see listing 1) are loaded by the gateway at launch time and the object representation is generated. Whenever a message is received by the gateway, the gateway checks whether a transition to a new state is possible using the received message. If there is no possible transition the gateway detected a failure, also all other transitions are checked whether a different message violated its constraints. Besides the check after receiving a message, the gateway also checks regularly every possible transition from the current states of the automata, if any constraint is violated, for example if a message missed its deadline.

As mentioned before the gateway is designed as a modular system, therefore it has a plug-in system which makes it easy to support new interfaces. For every network interface a "InterfaceModule" has to be created. Those modules can be seen as the 'driver', from the view of the gateway, which is needed to access the particular network interface. To add new network interfaces at a later stage without changing the code of the gateway itself a plug-in system was integrated into the gateway. Every network interface module has to implement a basic set of functions for example, driver initialization, sending messages and throwing an event on the arrival of new messages. To make the gateway aware of a new module it has to be added to the XML configuration file of the gateway, by adding a new 'Interface' section in the configuration.

Apart from the driver module, for every communication type an applicable "Analyzer" is needed to interpret and execute the state machine. The analyzer gets instantiated for a specific interface and the state machine representing the intentional application behavior is loaded. Which type of analyzer (e.g. TimeTriggered, EventTriggered and StreamCommunication) shall be used for the network interface has to be specified together with the driver in the interface section of the security gateways configuration file. This plug-in system makes it possible to change the driver or the analyzer, the gateway is working with on a specified interface, to test new setups or new algorithms. Also new interfaces can be integrated in a fast and easy fashion.

We created a object structure representing a timed automaton and its transitions based on the XML schema. The XML description of the automaton can then be loaded into the structure and the analyzer can use this representation to determine whether the application behavior is correct. Hence the analyzer is not working with XML file itself but rather with its representation as an object model. Which has functions to determine the current state and transitions to following states can be executed on the model.

Our test setup is a single PC running CANoe 7.6 with an attached VN7600 Vector FlexRay interface. The simulation is started as a "real bus" simulation. Our security gateway is executed on the same PC accessing the VN7600 Vector Interface and therefore was acting as another "ECU" alongside the simulated ECUs in the Vector simulation.

For our tests we did not change the Vector simulation in any way. Therefore, the gateway is an addition to the car network and it can easily be integrated into existing networks. Our evaluation of the security gateway was separated into different stages, at first correct application behavior was tested for *CAN* and *FlexRay*.

VI. Discussion

The implementation of the security gateway was evaluated in a lab setup. We attacked the simulated automotive system to determine whether the gateway is working correctly and detects deviations from the intentional application behavior. The FlexRay or CAN buses in a car can be attacked through various attack vectors, as we described in section III. In this work we are focusing on two different scenarios.

The first attack vector describes an attack where a new ECU is added to the network which sends malicious CAN or FlexRay messages and tries to disrupt the communication. Also an existing ECU can be adjusted to act maliciously, this would correlate to an attack where an ECU in the car is flashed with a malicious firmware.

The second attack vector is to physically connect to the bus in the car and to send messages [20] [24] [25]. For this second scenario an attacker would physically connect itself to the VN7600 interface and send messages into the simulation.

For our test setup we used the XL Driver Library and created an ”attack application” which carries out the attack of sending malicious messages. As the second scenario would also alter the communication behavior of the application running on the ECU it will also be detected if the messages, introduced by our attacker application, are classified as failures. Therefore, we do not need a test case for the second scenario.

In section IV we stated that the gateway has to process the messages in real time, in our current implementation the gateway is not forwarding the received messages. It is a passive component only sniffing the packets, while it is capable of processing the received messages in real time, the other components are not aware of the security gateway.

With the current setup the described problem has no implications, while the gateway can easily be fit into the network without any constraints. If the security gateway is localized on the gateway, which is already in the car, it has to adhere strictly on the timing constraints defined during development or already defined for the (regular) gateway.

As for **FlexRay**, normal application behavior was tested and the security gateway classified the communication as correct. In the next step we executed the ”attack application” which sent FlexRay messages in cycles where, according to the application behavior specification, no messages should be transmitted. As a result there was a difference between the gateways definition for the application behavior and the actual communication between the nodes. The communication which was not defined in the application behavior specification was therefore correctly classified as faulty.

For **CAN** we also began first with testing the normal application behavior and the gateway classified the communication as correct. In the next step we also executed the ”attacker application”, which flooded the bus with messages to lower the window position. Conducting the attack against the electronic window lift described by Hoppe et al. [4]. The malicious packages which were sent by the ”attack application” were detected by the security gateway, and classified as a failure as well, because the messages violated timing constraints.

The results of our experiment meet our expectations and the goals we set in the introduction. We think we created a solid framework to conduct further research and to detect further types of failures. While we are aware that with the current state of our implementation the gateway will only detect very basic attacks and it is possible by carefully crafting the attack to circumvent our countermeasures. In our opinion such a system makes it hard for an attacker to craft a successful attack, the limitations of our current implementation as well as future enhancements to create a more holistic attack detection will be outlined in the next section.

VII. Future work

Even though our security gateway is currently able to detect deviations from normal application behavior, there are still some limitations to it. An attacker might be able to disguise his actions as normal system behavior. This might be possible, especially in a CAN network.

The gateway currently is not able to verify the content of the messages for plausibility. If the attacker has enough system knowledge and manages to capture an ECU he can hijack the communication. If this is the case messages can be sent with invalid content and the security gateway is unable to detect it – if the communication is conform according to the application behavior definition. We are well aware of the limitations of our system, but we are planning to further improve the system and overcome these limitations. For example with validating and putting the content of messages into the relation to one another we can go beyond only checking the boundaries of certain values.

Also at the moment only a reporting of detected failures is done. A proper handling of detected failures has to be established. This also includes the possibility of warning the driver, one possible solution to this problem is described by Hoppe et al. [26]

Not only attacks against the system but also attacks against the gateway itself are plausible. In the best case, the security gateway possesses its own ECU to increase the difficulty of attacks. Software running in parallel with the gateway might be prone to attacks, which can open the possibility for a successful attack against the security gateway. This would compromise the whole system. Therefore, the ECU containing the security gateway needs to be hardened with an attack surface as small as possible, it also has to be protected against physical tampering. But also the gateways itself has to be resilient against attacks. This leaves possibilities for future work, to create a ”secure ECU” the gateway can be placed on.

In our current work we only described a centralized security gateway, but a centralized system is not always possible or the best solution. Therefore, it might be necessary to create a decentralized/distributed system of multiple security gateways. Those gateways need to have the possibility to communicate, securely, with each other. Furthermore, they need to find a global consent about possible intrusions in a defined time - an attack consisting of different anomalies might not be detected by only one security gateway but rather different gateways.

It will also be necessary to update the gateway regularly with new definitions and anomaly information or even new rules for intentional application behavior after a system update of the car. First of all this has to be done securely and safely, as the operation of the vehicle cannot be interrupted. This also raises the question, how the system has to behave if there was no communication with the car manufactures backend for a certain amount of time or the last system update was weeks ago, for example in areas where no mobile broadband connection is available. The question is how to handle those situations - will it be necessary to disable certain subsystems of the car?

After all we think our approach gives a valuable contribution and we can solve different existing security problems of nowadays automotive bus-system, as we discussed in section VI.

VIII. Conclusion

We presented an approach to secure automotive bus-systems, with knowledge about the intentional application behavior which is already available during development. This knowledge is used by our security gateway to detect failures in the attached networks.

This paper's primary contribution are the abstract models we create for the communication modes: time-triggered communication, event-triggered communication and stream communication.

With these models we are able to describe the intentional application behavior and to handle different communication protocols without depending on the communication protocol. Therefore, it is even possible to change the underlying communication protocol without the need of changing the application behavior definition.

The security gateway was experimentally verified and it is able to detect different kinds of attacks against our test system.

Not only is our security gateway useful to detect attacks, but it might also be helpful during development, to test the correct implementation of the communication. Furthermore, we are not only able to detect security but also safety relevant failures. Since we currently do not distinguish between security and safety relevant failures.

References

[1] Open Alliance SIG, "Broadr-reach® automotive ethernet standard," accessed 24.01.2014. [Online]. Available: http://www.opensig.org/

[2] S. Seifert, M. Kucera, and T. Waas, "The need for security in distributed automotive systems," in *PECCS 2013 - Proceedings of the 3rd International Conference on Pervasive Embedded Computing and Communication Systems, Barcelona, Spain, 19-21 February, 2013*, César Benavente-Peces and Joaquim Filipe, Eds. SciTePress, 2013.

[3] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaâniche, and Y. Laarouchi, "A survey of security threats and protection mechanisms in embedded automotive networks," in *Proceedings of the 2nd Workshop on Open Resilient human-aware Cyber-physical Systems (WORCS-2013), co-located with the IEEE/IFIP Annual Symposium on Dependable Systems and Networks (DSN-2013)*, Budapest and Hungary, 2013, pp. 1–12.

[4] T. Hoppe and J. Dittmann, "Sniffing/replay attacks on can buses: A simulated attack on the electric window lift classified using an adapted cert taxonomy," *2nd Workshop on Embedded Systems Security (WESS' 2007)*, 2007.

[5] T. Hoppe, F. Exler, and J. Dittmann, "Ids-signaturen für automotive can-netzwerke," in *D-A-CH security 2011*. [Klagenfurt]: syssec, 2011, pp. 55–66.

[6] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive can networks: practical examples and selected short-term countermeasures," *Reliability engineering & system safety*, vol. 96, no. 1, pp. 11–25, 2011.

[7] Marko Wolf, André Weimerskirch, Christof Paar, and Most Bluetooth, "Security in automotive bus systems," in *in: Proceedings of the Workshop on Embedded Security in Cars (escar)'04*, 2004.

[8] D. Nilsson, U. Larson, F. Picasso, and E. Jonsson, "A first simulation of attacks in the automotive network communications protocol flexray," in *Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems CISIS'08*, ser. Advances in Soft Computing, E. Corchado, R. Zunino, P. Gastaldo, and Á. Herrero, Eds. Springer Berlin / Heidelberg, 2009, vol. 53, pp. 84–91.

[9] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *Defcon 21*, 2013.

[10] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaaniche, and Y. Laarouchi, "Security of embedded automotive networks: state of the art and a research proposal," in *Proceedings of Workshop CARS (2nd Workshop on Critical Automotive applications : Robustness & Safety) of the 32nd International Conference on Computer Safety, Reliability and Security*, Matthieu ROY, Ed., Toulouse and France, 2013, p. NA.

[11] M. Glass, D. Herrscher, H. Meier, M. Piastowski, and P. Schoo, "Seis - sicherheit in eingebetteten ip-basierten systemen," *ATZelektronik*, vol. 5, no. 1, pp. 50–55, 2010.

[12] 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*, ser. ICISC'11. Berlin and Heidelberg: Springer-Verlag, 2012, pp. 302–318.

[13] GENIVI, "Genivi alliance," accessed 24.01.2014. [Online]. Available: http://www.genivi.org/

[14] 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 *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '13. San Jose and CA and USA: EDA Consortium, 2013, pp. 458–463.

[15] A. G. Camek, C. Buckl, and A. Knoll, "Future cars: necessity for an adaptive and distributed multiple independent levels of security architecture," in *Proceedings of the 2nd ACM international conference on High confidence networked systems*, ser. HiCoNS '13. New York and NY and USA: ACM, 2013, pp. 17–24.

[16] M. Muter, A. Groll, and F. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *Information Assurance and Security (IAS), 2010 Sixth International Conference on*, 2010, pp. 92–98.

[17] A. Bouard, J. Schanda, D. Herrscher, and C. Eckert, "Automotive proxy-based security architecture for ce device integration," in *Mobile Wireless Middleware, Operating Systems, and Applications*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, C. Borcea, P. Bellavista, C. Giannelli, T. Magedanz, and F. Schreiner, Eds. Springer Berlin Heidelberg, 2013, vol. 65, pp. 62–76.

[18] J. Joy, A. Raghu, and J. Joy, "Architecture for secure tablet integration in automotive network," in *Proceedings of the FISITA 2012 World Automotive Congress*, ser. Lecture Notes in Electrical Engineering. Springer Berlin Heidelberg, 2013, vol. 194, pp. 683–692.

[19] K. Lemke, C. Paar, and M. Wolf, *Embedded Security in Cars: Securing Current and Future Automotive IT Applications*. Secaucus and NJ and USA: Springer-Verlag New York, Inc, 2005.

[20] 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*, ser. SEC'11. Berkeley and CA and USA: USENIX Association, 2011, pp. 6–6.

[21] A. Don Bailey, "War texting: Weaponizing machine 2 machine." [Online]. Available: http://www.isecpartners.com/storage/docs/presentations/isec_bh2011_war_texting.pdf

[22] A. Barisani and D. Bianco, "Hijacking rds-tmc traffic information signals," *BlackHat, Las Vegas USA, 1-2 Agust 2007*, 2007. [Online]. Available: http://www.phrack.org/issues.html?issue=64\&id=5#article

[23] L. Alfaro, T. Henzinger, and M. Stoelinga, "Timed interfaces," in *Embedded Software*, ser. Lecture Notes in Computer Science, A. Sangiovanni-Vincentelli and J. Sifakis, Eds. Springer Berlin Heidelberg, 2002, vol. 2491, pp. 108–122.

[24] 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 *Security and Privacy (SP), 2010 IEEE Symposium on*, 2010, pp. 447–462.

[25] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *Defcon 21*, 2013.

[26] T. Hoppe, S. Kiltz, and J. Dittmann, "Adaptive dynamic reaction to automotive it security incidents using multimedia car environment," in *Information Assurance and Security, 2008. ISIAS '08. Fourth International Conference on*, 2008, pp. 295–298.