Topic: Security of In-vehicle Communication

Julian Pfeifer

***Abstract*— While IoT devices get more and more widely used, embedded boards are already a central part of every car for several years now. Sensors and embedded boards for car control are connected via a multitude of networks. The most widely used of these networks, the Controller Area Network (CAN), was designed in the eighties with no security in mind because these networks were closed off. But nowadays vehicles have a multitude of interfaces to outside networks, so there is a dire need to modernise the in-vehicle networks with appropriate security measures....tbd**

I. Introduction

The Internet of things (IoT) is a continuously growing network of commonplace devices. They can be physical like wireless sensors and smart phones or virtual like services. The IoT transforms how we interact with our environment. Today almost any electronic device can be bought as a smart version like microwaves, washing machines, light bulbs, door locks and many other ”things”. All these belong to the IoT network. The IoT has several application domains. It can be used personally or in enterprises [9]. The personal and social application domain is for connecting people to their environment and to other people. The industries and enterprises domain enables different activities in and between organizations of for example the finance and banking sector. Also IoT can be used in for example breeding or energy management where it is used for service and utility monitoring. The last application domain is transportation. Which encompasses smart cars and infrastructure like traffic lights.

Ever since there were the first electronic systems to assist drivers in control of their cars there were Electronic Control Units (ECUs) included in these vehicles. ECUs consist of micro-controllers and sensors. Nowadays cars have not only normal Driver Assistance Systems like antilock breaking systems. They also have Advanced Driver Assistance Systems (ADAS) which help people with safety-critical functionality like parking or emergency breaking. To realize all these functions a multitude of in-vehicle networks (IVNs) is needed to connect all the ECUs and the sensors. These IVNs were originally closed off and had no connection to other networks. So they were designed with no security in mind whatsoever. Recently, cars have been connected to a multitude of networks like 3G/4G mobile phone networks, Bluetooth and vehicular ad hoc networks (IEEE 802.11p). So the IVNs have become vulnerable to different types of attacks while joining the IoT. The most used type of IVN today is still the Controller Area Network (CAN). It was introduced in 1983 by the Robert Bosh GmbH. Because in these days IVNs were closed of there were no security features included in the CAN protocol like encryption or message authentication [2]. This absence of inherent security features in the protocol led to successful demonstrations of security breaches [7]. It was possible to reprogram ECUs via physical and wireless connections. The car could even be monitored and controlled remotely. Since then there have been many automotive attack demonstrations [3]–[5].

This report focuses on recent CAN authentication research. Two initially proposed authentication protocols named LeiA and vatiCAN secure the CAN network against adversaries that do not control code execution on ECUs in the network [13], [15]. Later these protocols were improved by VulCAN which is an efficient approach to implement secure distributed automotive control software on lightweight trusted computing platforms [16]. VulCAN adds another layer of security by relying on trusted hardware and a minimal software Trusted Computing Base (TCB). Additionally it was shown that VulCAN provides sufficient performance to be used in automotive real-time applications.

In the following chapter II I will provide an overview of different IVNs like CAN, Time-Triggered Networks, Low-Cost Automotive Networks and more. Also I will introduce the general types of security measures in IVNs. Afterwards chapters III and IV will explain LeiA and vatiCAN and how each of them work in general. Chapter V will talk in depth about VulCAN. I will explain how it achieves improvements over LeiA and vatiCAN using Sancus 2.0. Finally there will be a discussion in chapter VI connecting security in IVNs to security in the general IoT sector and the world wide web.

II. In-vehicle Communication Networks and Security

The different functions of a vehicle have very different requirements in performance and safety needs. Therefor the quality of service needed from the communication system varies (e.g. response time or bandwidth). Normally there are different functional domains which divide the in-car embeeded systems [11]. There are the safety-critical domains “power-train” (e.g. engine control) and “chassis” (e.g. steering) that need a deterministic real-time behavior. The functions of the “body” domain that controls for example dashboard, wipers, lights and windows need to exchange many informations of small size between each other. Other domains like “telematics” and “multimedia” have for example increased requirements in bandwidth and confidentiality.

A multitude of different networks resulted out of this diversity of requirements. Therefor the Society of Automotive Engineers (SAE) created in 1994 a classification for automotive communication protocols. This classification is based on data transmision speed and functionality. There were 4 different classes defined that are labeled class A to class D [1]. Class A networks have a speed lower than 10 kb/s. They are used for convenience features such as trunk release or

electric mirror adjustment. Examples for Class A networks are LIN and TTP/A. Class B has a medium speed of 10 to 125 kb/s. Networks of this classification are for general information between ECUs from for example sensors. Main representatives of this class are J1850 and low-speed CAN. High speed networks of class C have a speed between 125 kb/s and 1 Mb/s and are used for real time control like the power train or vehicle dynamics. High-speed CAN falls into this classification. Above the high speed classification C there is class D. Every communication protocol faster than 1 Mb/s fall into this category. They are normally used either for multimedia applications (e.g. MOST) or for hard real time critical functions like X-by-Wire applications (e.g. TTP/C or FlexRay). Networks this fast, like FlexRay, can also be used as gateways between sub-systems.

In modern vehicles it is normal that there are many networks of different types. A BMW 7 series car from 2008 implements for example multiple LIN buses, a MOST and a FlexRay bus and additionally four CAN buses [6]. All of these networks are normally interconnected by gateways.

In-vehicle networks play a crucial role in keeping the embedded systems in a safe state. Depending on the network it can be for example more or less difficult to identify failed nodes. Also some networks can meet hard real-time constraints and some cannot. In general there are two main paradigms in automotive communication, event-triggered and time-triggered communication [11]. If the communication is consisting of asynchronous events, it follows the event-triggered paradigm. In such systems it is crucial to avoid conflicts while sending events from multiple sources in parallel. Event-triggered systems use the bandwidth effectively and are easy to extend with new network nodes. If network communication is synchronous, so every nodes sends at predefined time slot in a defined interval, then the communication follows the time-triggered paradigm. In general accessing a medium this way is called Time Division Multiple Access (TDMA). These systems are perfectly predictable but require to be statically defined up front. If there are new nodes introduced to the network the schedule has to be changed, so it is more complicated to extend these systems. Also the bandwidth is not used very efficiently and the response times are longer then in event-triggered systems. However missing messages can be identified rather easily. Because both paradigms have up and down sides. Normally both types of communications are needed for different features in an embedded system of a vehicle. Hence some networks like FlexRay provide both types of communication alongside each other.

In the following sub-chapters I will describe the CAN network and give an overview about some of the most representative other networks.

A. CAN Network

The CAN network was introduced in 1983 by the Robert Bosh GmbH [11]. It uses a twisted pair of copper wires as a bus between the nodes. Depending on the speed used it is either classified and used as a SAE class B or class C network. Aside the used speed there are two versions of CAN. Version 2.0A and 2.0B which differ in the length of the identifier used when sending data. CAN uses a Non-Return-to-Zero bit representation with a bit stuffing of 5 bits. Which means the bits will be represented by continuous levels of voltage. To be able to stay in sync when sequences of the same bit value are sent over the wire there will be the other bit value “stuffed” in after every 5 same consecutive values. The receivers know this and can properly “destuff” the bit sequence. So all CAN nodes can stay in sync using this bit stuffing.

For CAN to be able to bound the respond times it uses a priority system. The lower the identifier used to send data the higher the priority. To realize these priorities the physical layer needs to implement an “AND” scheme. Only when everyone simultaneously sends a 1 the resulting bus level is 1. If one node on the bus sends a 0, the resulting bus level has to be a 0. So the 0 is the dominant value on the bus and overrides a 1. With this mechanism it is easy to realize the priority system.

The normally used version of CAN is 2.0A because it provides with 2^{11} possibilities enough identifiers so I will focus on its specs in the following. Communication over the CAN bus is organized into frames of a maximum size of 135 bits if all overheads are included. A frame starts with a 1 bit Start Of Frame. Following this start there is an 18 bit header. The header contains the 11 bit identifier, the Remote Transmission Request (RTR) bit and the Data Length Code (DLC). The RTR distinguished between data frames and data request frames. The DLC provides the length of the data following after the header. This data can have at most 8 bytes. After the data follows a 15 bit Cyclic Redundancy Check (CRC) for ensuring data integrity. After the CRC follows the Acknowledgement field (Ack). With the Ack a sender is possible to know that a sender has received the frame. However it is not possible to distinguish who received it. At the end of the frame is the End Of Frame field followed by the intermission bits. The intermission bits are the minimal number of bits after which it is allowed to send a new data frame.

TODO: image of data frame composition

On an idle CAN bus every node can send a data frame at any time. If multiple nodes want to send a frame simultaneously and collide the priority based arbitration decides which node is allowed to send data. This works with the help of the physical layer implementing the “AND” scheme. So while sending identifier and RTR bit the nodes are observing the bus level. If the node observes a bit of its own to be overridden by a dominant bit (0 is dominant over 1) it stops sending data because another node with a smaller identifier plus RTR is sending at the same moment which has priority. The node which stopped sending needs to wait until the bus becomes idle again and then tries to send the data frame again.

For the priority-based arbitration to work the signal of a bit needs to propagate to all other nodes and back before sending the next bit. So the physical length of the bus restricts the speed with which data can be send. On a 40 meter bus a maximum speed of 1 Mbit/s is possible while only 250 Kbit/s can be achieved over a 250 meter long bus [11]. This restriction has lead to optimizations by the car manufacturers using “traffic shaping”. One example would be to use an offset

for important periodic messages so they will not interfere with each other [10]

The CAN protocol has different mechanisms to detect possible errors. One mechanism would be to use the CRC to validate the data integrity. If an error is detected by a node it will send six consecutive dominant bits which will cancel the current data frame. This will alert all nodes on the bus that an error occurred and a new arbitration phase can start. This delays the message sending and possible deadlines may be missed this way. The possible start of a new frame takes 17 to 31 bits after an error was detected. CAN also includes some fault-confinement mechanisms for failures for example on the hardware level of the micro-controller or communication controller. These mechanisms normally involve the counting of failures and successful delivery of frames. However each node is responsible itself to execute these mechanisms. So the relevance of these mechanisms is questionable [11]. If there is an error for example with the oscilloscope a node could be sending unknowingly many dominant bits. This would be one manifestation of the "babbling idiot" [14]. More mechanisms are needed if the protocol is used for safety-critical functions. There are lots of different solutions for different problems but there is no formal verification for these mechanisms used together [11].

The CAN protocol only defines the physical layer and the Data Link layer but there are higher level protocols like AUTOSAR which use the CAN protocol.

B. Time-Triggered Networks

C. Other Networks

Low-Cost Automotive Networks
Multimedia and Infotainment Networks
Automotive Ethernet

D. Security Measures

Controller Authentication, Encrypted Communication, Gateway Firewalls [8]

III. LeiA: Lightweight Authentication Protocol for CAN

Overview of LeiA [15]

IV. VatiCAN: Vetted, authenticated CAN bus

Overview of vatiCAN [13]

V. VulCAN: Vehicular Component Authentication and Software Isolation

In depth description/analysis of VulCAN [16] based on Sancus 2.0 [12]. Van Bulck et. al. "vulcanized" the LeiA and VatiCAN protocols to improve the protocol-level security guarantees and add several system-level security guarantees.

VI. Discussion

Key trade-offs & considerations on the presented technologies for in-vehicle communication security: VulCAN vs VatiCAN vs LeiA

maybe comparison to the internet world with tls etc. to set everything into context

VII. Conclusions

Wrapping up the presented technologies for in vehicle communication, especially VulCAN.

References

[1] F. Ali, Z. Sheng, and V. Ocheri. A Survey of Automotive Networking Applications and Protocols. Technical report, 2017.

[2] O. Avatefipour and H. Milik. State-of-the-Art Survey on In-Vehicle Network Communication "CAN-Bus" Security and Vulnerabilities. *International Journal of Computer Science and Network*, 6(6):720–727, 2017.

[3] M. Cheah, S. A. Shaikh, O. Haas, and A. Ruddle. Towards a systematic security evaluation of the automotive Bluetooth interface. *Vehicular Communications*, 9:8–18, jul 2017.

[4] 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 Security Symposium*, pages 77–92, 2011.

[5] T. Hoppe, S. Kiltz, and J. Dittmann. Security threats to automotive CAN networks—Practical examples and selected short-term countermeasures. *Reliability Engineering & System Safety*, 96(1):11–25, jan 2011.

[6] H. Kellermann, G. Németh, J. Kostelezky, K. L. Barbehön, F. El-Dwaik, and L. Hochmuth. Electrical and Electronic System Architecture. *ATZextra worldwide*, 13(8):30–37, nov 2008.

[7] 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 *2010 IEEE Symposium on Security and Privacy*, pages 447–462. IEEE, 2010.

[8] K. Lemke, C. Paar, and M. Wolf. *Embedded Security in Cars Securing Current and Future Automotive IT Applications*. 2006.

[9] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan. Internet of things (IoT) security: Current status, challenges and prospective measures. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 336–341. IEEE, dec 2015.

[10] N. Navet and F. Simonot-Lion. Scheduling Messages with Offsets on Controller Area Network: A Major Performance Boost. In *Automotive Embedded Systems Handbook*, chapter 14.1. 2009.

[11] N. Navet and F. Simonot-Lion. In-vehicle communication networks: A historical perspective and review. *Industrial Communication Technology Handbook, Second Edition*, 2013.

[12] J. Noorman, J. O. Van Bulck, J. T. Mühlberg, F. Piessens, P. Maene, B. Preneel, I. Verbauwhede, Iminds-Cosic, K. U. Leuven, J. G. Otzfried, M. Uller, F. Freiling, and F. Erlangen-Nürnberg. Sancus 2.0: A Low-Cost Security Architecture for IoT Devices. *ACM Transactions on Privacy and Security*, 0(0), 2017.

[13] S. Nürnberger and C. Rossow. vatiCAN: Vetted, authenticated CAN bus. In *Cryptographic Hardware and Embedded Systems – CHES 2016. CHES 2016. Lecture Notes in Computer Science*, volume 9813 LNCS, pages 106–124. Springer, Berlin, Heidelberg, 2016.

[14] J. Pimentel, J. Proenza, L. Almeida, G. Rodriguez-Navas, M. Barranco, and J. Ferreira. Dependable Automotive CAN Networks. In *Automotive Embedded Systems Handbook*, chapter 6. 2009.

[15] A. I. Radu and F. D. Garcia. LeiA: A lightweight authentication protocol for CAN. In *Computer Security – ESORICS 2016. ESORICS 2016. Lecture Notes in Computer Science*, volume 9879 LNCS, pages 283–300. Springer, Cham, 2016.

[16] J. Van Bulck, J. T. Mühlberg, and F. Piessens. VulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks. *33rd Annual Computer Security Applications Conference*, pages 225–237, 2017.

Appendix