Towards an Integrated In-Vehicle Isolation and Resilience Framework for Connected Autonomous Vehicles

Khaled Mahbub, Mohammad Patwary, Antonio Nehme
Birmingham City University
Birmingham, United Kingdom
Email:{firstname.lastname}@bcu.ac.uk

Marc Lacoste, Sylvain Allio, Yvan Rafflé
Orange Labs
France
Email:{firstname.lastname}@orange.com

***Abstract*—Connected Autonomous Vehicles (CAV) have attracted significant attention, specifically due to successful deployment of ultra-reliable low-latency communications with Fifth Generation (5G) wireless networks. Due to the safety-critical nature of CAV, reliability is one of the well-investigated areas of research. Security of in-vehicle communications is mandatory to achieve this goal. Unfortunately, existing research so far focused on in-vehicle isolation or resilience independently. This short paper presents the elements of an integrated in-vehicle isolation and resilience framework to attain a higher degree of reliability for CAV systems. The proposed framework architecture leverages benefits of Trusted Execution Environments to mitigate several classes of threats. The framework implementation is also mapped to the AUTOSAR open automotive standard.**

Keywords - Isolation; Resilience; ECU; Monitoring; Trusted Execution Environment; AUTOSAR; Certification.

I. INTRODUCTION

Despite considerable progress in the last decade, the development of fully self-driving vehicles is still largely under research and experimentation. In such safety-critical systems, the resilience of in-vehicle and inter-vehicle communication is a key element to ensure the security of the vehicle. While in-vehicle relates to on-board communication between Electronic Control Units (ECUs) acting based on inputs from different sensors, inter-vehicular communications enable data exchange with the external environment including other vehicles, broadband clouds and roadside-infrastructures [1].

In this system of systems model, the diversity, autonomy and connectivity of vehicles mean vulnerabilities at the level of the vehicle affect the larger environment [2]. While both types of communication enable safety-critical decision-making, in-vehicle communication requires special attention. The disparity of coding practices among the diversity of specialised vendors in different functionalities (e.g., infotainment, braking and steering assistance), and the trust model induced by the high degree of connectivity and unrestricted interactions between vehicle components to enable comfort features (e.g., adjusting the sound volume according to the velocity) widen the attack surface [1].

Internal security barriers to detect and react to an intrusion are therefore needed to limit the impact of a compromise and to mitigate its propagation to different subsystems within a vehicle [1]. Moreover, the adoption of new technologies in the automotive domain is opening new safety and security challenges. For example, the advent of new generations of ECUs that are virtualized as lightweight execution environments (e.g., virtual machines, containers) on different types of virtualization platforms, (e.g., OKL4 Microvisor, Proteus Hypervisor, ETAS STA-HVR [3]) may face system-level isolation challenges such as side-channels.

This short paper introduces our approach to detect and limit the impact of intrusions for in-vehicle networks that can compromise the safety of autonomous vehicles. This will be a step towards enhancing the robustness of in-vehicle communications through the isolation of ECUs, the detection of and recovery from intrusions. Focusing on spoofing, replay, and side-channel attacks, we present principles of a framework for in-vehicle isolation and resilience and discuss technical considerations for its implementation according to the AUTomotive Open System ARchitecture (AUTOSAR) open standard.

This paper is structured as follows: Section II presents related work. Section III introduces our framework and Section IV discusses considerations to adhere to the AUTOSAR standard. We conclude our paper in Section V.

II. RELATED WORK

A significant body of work focuses on improving resilience of connected autonomous vehicles. Solutions against threats can be categorised as i) Proactive Defence, ii) Active Defence and iii) Passive Defence [4]-[8]. We give next a brief overview of each family of techniques.

A. Proactive Defence

Proactive defence is underpinned by the "security by design" principle practiced in the software industry [6],[7]. Integration of common security practices, public key encryption and hash-based message authentication fall under this category [4],[9].

 ISBN: 978-1-61208-795-5

B. *Active Defence*

Active defence mitigates threats as they occur. For instance, continuous monitoring can be applied to detect intrusions and preserve the security hygiene of the vehicle and take adequate remediation actions [10]; in this sense, real-time monitoring enables the identification and isolation of faulty applications in safety-critical systems [11].

Detection approaches for the in-vehicle network can be categorised as i) Signature-Based Detection, ii) Anomaly-Based Detection and iii) Hybrid Approach [12]-[15]:

- Signature-Based Detection: These approaches use information about attacks (signatures) as a pattern characterizing known threats, comparing signatures against observed events to identify possible attacks.
- Anomaly-Based Detection: These approaches are based on continuous monitoring of system activities, checking against a reference model (e.g., profile of the system). An alarm is raised if deviation from the reference model is observed. Various mechanisms can be applied to derive the reference model, such as machine learning [16],[17], frequency-based [18]-[20], and statistical-based methods [21],[22].
- Hybrid Approach: This family of approach comprises several intrusion detection techniques (e.g., signature- and anomaly-based detection).

In addition to in-vehicle intrusion detection, several approaches explore detection of side channel attacks for the automotive domain - at the physical layer [23], using cache-based [24] or interface-based approaches [25],[26].

C. *Passive Defence*

Passive defence mainly focuses on detecting, responding to, and recovering from an attack once it has occurred. This type of defence is notably suitable to prevent malwares and code injection and modification threats. Therefore, passive defences are not suitable for safety-critical systems, like autonomous vehicles, as these approaches do not facilitate detection and mitigation of adversaries in real-time [8],[10].

It should be noted that proactive defence and passive defence are not suitable to handle adaptive security requirements, very common in the cyber and the automotive domains. Proactive defence recommends designing control features to meet the security objectives at system design time and embedding such features in the system. However, this approach is unable to cover new types of threats once the system has been developed. On the other hand, passive defences alone are not suitable for safety-critical systems, such as autonomous vehicles, as these approaches detect the attack once it has occurred. Also the active defence techniques approaches found in the literature apply continuous monitoring to detect anomalies, but did not consider their application to secure execution environments for ECUs. As described in the next sections, our approach aims to address these limitations.

III. IN-VEHICLE ISOLATION AND RESILIENCE

We adopt the active defence approach to improve in-vehicle resilience: security properties related to the communication among ECUs will be continuously monitored in order to detect security threats, and actions will be taken to mitigate the impact of and gracefully recover from the detected threat. Recovery in our context consists of rolling back (or forward) to a stable state to overcome intrusions [27].

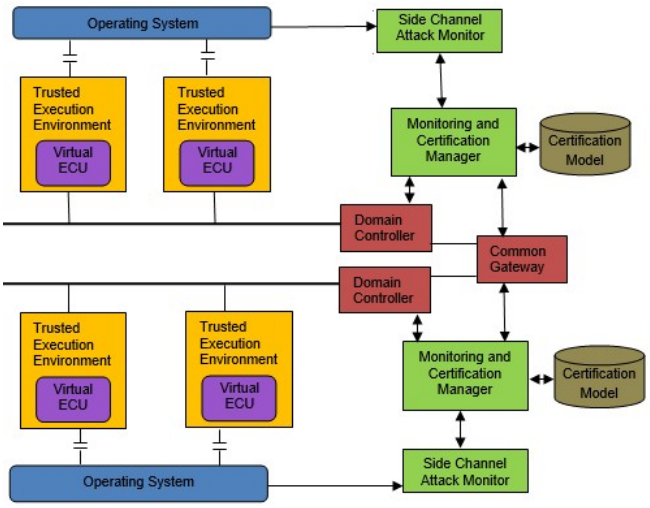

Figure 1. Reference Architecture of Isolation and Resilience Framework

Figure 1 shows the proposed reference architecture for threat detection and mitigation in the in-vehicle network. ECUs are grouped into different domains according to similarity of functionalities. All ECUs in a domain are connected to the same communication bus and activities of each ECU in a domain are controlled by one domain controller. Domain controllers are connected through a common gateway in order to enable communication among the ECUs belong to different domains [4],[9]. The major components of the architecture are:

A. *Trusted Execution Environment (TEE)*

Trusted Execution Environments enable to specify isolated execution environment in the main processor [28],[29]. The TEE provides security features such as isolated execution, integrity of applications executing in the TEE, and confidentiality of application assets. Several hardware vendors provide embedded technologies that can be used to support TEE implementations, including AMD PSP [30], ARM TrustZone [31], EVITA Hardware Security Modules (HSM) [32] and Intel SGX Software Guard Extensions [33]. We aim to explore if TEEs could be applied as secure execution environment for ECUs, thereby ensuring secure/isolated communication from ECU to ECU.

 ISBN: 978-1-61208-795-5

B. *Side Channel Attack Monitor*

While TEEs aim to provide secured execution environments, they are vulnerable to side-channel attacks [34],[35]. This component focuses on runtime detection of the variants of side-channel attacks (e.g., SGX interface-based attacks) that are relevant in a vehicular context.

C. *Monitoring and Certification Manager*

The responsibility of this component is to perform real-time monitoring of security properties related to components (e.g., ECUs) in the in-vehicle network to detect security threats. This component applies the hybrid approach (including frequency-based, statistical-based, and deep packet inspection approaches) to detect spoofing and replay attacks. Based on the validity of the security properties, this component also maintains the certificates (detailed in Section IV.B) that certify the valid state of the ECUs.

IV. IMPLEMENTATION CONSIDERATIONS

We adopt the AUTOSAR open standard for automotive software architecture and framework to implement the architecture presented in Section III. The AUTOSAR consortium was formed by major automotive OEMs like BMW, Ford, Daimler and Chrysler to standardize the automotive software architecture and framework, thereby facilitating scalability, reusability and interoperability across the products lines from different OEMs [36]. The use of AUTOSAR in the implementation would therefore inherently enable the prototype to have the same benefits.

Next, we briefly introduce AUTOSAR, and then discuss the mapping between our framework and AUTOSAR. In AUTOSAR, the ECU software is abstracted in a layered architecture, built on top of the underlying micro-controller hardware [37]. As shown in Figure 2, this architecture is composed of three layers, namely Basic Software (BSW), Runtime Environment (RTE), and Application Layer.

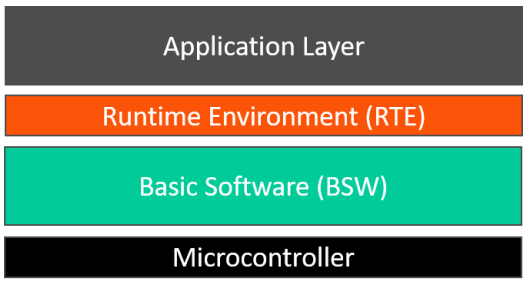

Figure 2. Overview of AUTOSAR components [37]

Basic Software Layer (BSW) is the bottom layer of the architecture and provides core system functionalities. This layer has 3 sub-layers. First, the Micro-controller Abstraction Layer (MCAL) contains internal driver modules that access the underlying micro-controller and internal peripherals directly. Second, the ECU Abstraction Layer (ECUAL) interfaces the drivers of the MCAL and offers an API for accessing the peripherals and external devices, thus making higher software layers independent of the hardware layout. And third, the Services Layer (SL) provides top-level services (e.g., operating system functionality, communication services, management services, memory services, etc.) to application software components.

The Run-Time Environment (RTE) layer provides communication services to the application software, acting as a bridge between the application and the BSW layer.

The Application Layer mainly consists of software components (SWC) interconnected to other SWCs and BSW modules. This layer is component-based, which enhances SWC scalability and re-usability.

Figure 3 shows the mapping of the major components of our framework to the AUTOSAR architecture. We propose to add an ECU that takes the role of monitoring existing ECUs in the system, and to isolate ECUs.

The left side (yellow box) of the Figure 3 shows the deployment of virtual ECUs within the TEE, following the AUTOSAR architecture. The right side of the Figure 3 shows the Domain Controller, Monitoring & Certification Manager and Side Channel Attack Monitor components of the framework developed as SWCs in the AUTOSAR application layer, i.e., these software components will reside within a trusted virtual ECU and will collect the data transmitted among the virtual ECUs of the in-vehicle network. Such data will be used for monitoring the security properties related to different ECUs.

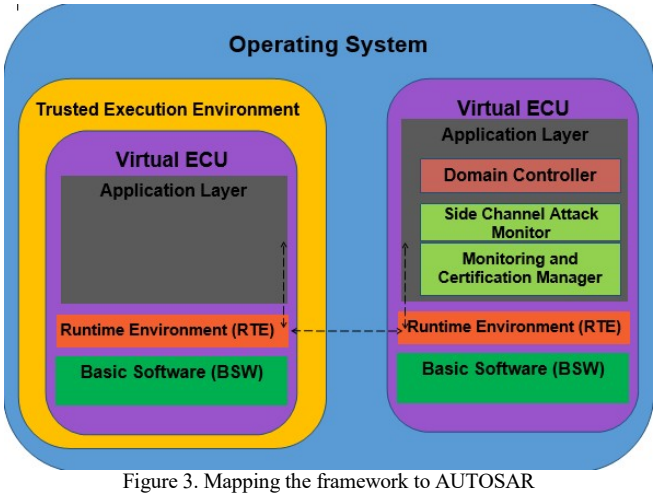

Figure 3. Mapping the framework to AUTOSAR

A. *Monitoring*

As shown in Figure 3, the Monitoring and Certification Manager and the Side Channel Attack Monitor will collect the data transmitted among the virtual ECUs of the in-vehicle network. A sub-component, namely *DataCollector*, will be deployed for transforming the data from a legacy format (e.g., CAN bus, being the most widely used protocol in the automotive industry [38]) to a format that used for monitoring.

 ISBN: 978-1-61208-795-5

This design may help to support multiple communication standards in the framework (e.g., Automotive Ethernet) by implementing a dedicated *DataCollector* (e.g., converting in-vehicle data from Automotive Ethernet to network monitoring format). Using multiple monitoring ECUs (e.g., for each sets of Domain Controllers) may help addressing safety constraints to avoid single points of failure.

B. Certificate Model

The Monitoring & Certification Manager and Side Channel Attack Monitor perform real-time monitoring of security properties related to ECUs to detect security threats and produce a certificate for the in-vehicle network. Monitoring is driven by security properties expressed as condition-action rules verified by a rule engine (e.g., CLIPS [39]) against runtime facts (i.e., runtime data). Monitoring results are accumulated to produce a certificate that certifies the state of the monitored components (e.g., ECUs).

The certificate structure includes the following elements:

1) CertificateID: represents the unique identifier of a generated certificate during the monitoring process.

2) MonitoringResultAggregator: aggregates monitoring results produced by monitoring different components (e.g., ECUs, CAN bus etc.) of the in-vehicle network. This element contains the following sub-elements:

- *AggregationTime:* denotes the time of aggregation.
- *Duration:* specifies the timespan between monitoring results considered for aggregation.
- *ToMLis:* specifies a list of TargetOfMonitoring considered for the aggregation operation.
- *AggregationRule:* defines how monitoring results should be aggregated, e.g., for results with numerical values by applying statistical methods.
- *AggregationResult:* stores the aggregation result.

3) TargetOfMonitoring (ToM): a component (e.g., ECU, CAN bus, etc.) monitored to identify security threats associated with it.

The *ToM* has the following sub-elements

- *ToMType:* the type of component to be monitored.
- *ToMID:* a unique identifier of the component in the in-vehicle network.
- *MonitoringRule:* the security property related to this component to be monitored.
- *MonitoringEvidenceAggregator:* contains the aggregation of results by monitoring the *MonitoringRule* related to this component.

The *MonitoringEvidence Aggregator* contains the following sub-elements:

- *AggregationTime;* denotes the time of aggregation.
- *Duration:* specifies the time span between in-vehicle network data considered for monitoring.
- *AggregationRule:* defines how monitoring results should be aggregated, e.g., for results with numerical values by applying statistical methods.
- *AggregationResult:* stores the aggregation result.

V. Conclusion and Outlook

This position paper provided an overview of defence strategies to mitigate common threats to in-vehicle networks. We proposed an architecture and framework to enhance in-vehicle isolation and resilience focusing on spoofing, replay and side-channel attacks. The framework follows an active defence strategy to detect and react to intrusions on the in-vehicle network and to recover from attacks. This recovery may be rolling back to a stable state to overcome an intrusion (e.g., in [27]), or to estimate the stable state by applying different techniques (e.g., in [5]). This framework may also be used to detect anomaly or misbehavior, which are not necessarily resulting from cyberattacks but simply from system faults and to limit their propagation in such a system of systems (e.g., in [40]).

Next steps are to implement the framework, and to evaluate its isolation and resilience benefits in a simple setting, first using simulations, before a possible testbed implementation.

References

[1] M. Faezipour, M. Nourani, A. Saeed, and S Addepalli, “Progress and challenges in intelligent vehicle area networks,” Communications of the ACM, vol. 55, no. 2, pp. 90-100, Feb. 2012.

[2] J. Boardman and B. Sauser, “System of Systems-the meaning of,” 2006 IEEE/SMC International Conference on System of Systems Engineering, pp. 6-pp, Apr. 2006, IEEE.

[3] A.K. Rajan, A. Feucht, L. Gamer, and I. Smaili, “Hypervisor for Consolidating Real-Time Automotive Control Units: Its Procedure, Implications and Hidden Pitfalls,” Journal of Systems Architecture, vol. 82, pp. 37-48, Jan. 2018.

[4] K. Daimi, M. Saed, S. Bone, and John Robb, “Securing Vehicle’s Electronic Control Units,” Twelfth International Conference on Networking and Services, 2016.

[5] V. Marquis et al., “Toward attack-resilient state estimation and control of autonomous cyber-physical systems,” Systems and Information Engineering Design Symposium (SIEDS), pp. 70-75, 2018.

[6] D. A. Brown et al. “Automotive Security Best Practices: Recommendations for security and privacy in the era of the next-generation car,” McAfee White Paper, Aug. 2016.

[7] A. Chattopadhyay and K. Lam, “Autonomous Vehicle: Security by Design,” Oct 2018, arXiv:1810.00545v1 [Retrieved: 13-05-2020].

[8] M. Saed, K. Daimi, and S. Bayan, “A Survey of Autonomous Vehicle Technology and Security,” VEHICULAR 2019.

[9] M. S. Ul Alam, “Securing Vehicle Electronic Control Unit (ECU) Communications and Stored Data,” Master of Science Thesis, School of Computing, Queen's University Kingston, Ontario, Canada, Sep. 2018.

[10] V. L. Thing and J. Wu, “Autonomous Vehicle Security: A Taxonomy of Attacks and Defences,” 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 164-170, Dec. 2016.

 ISBN: 978-1-61208-795-5

[11] B. Motruk, J. Diemer, R. Buchty, R. Ernst, and M. Berekovic, "IDAMC: A many-core platform with run-time monitoring for mixed-criticality," 2012 IEEE 14th International Symposium on High-Assurance Systems Engineering, pp. 24-31, Oct. 2012, IEEE.

[12] G. Dupont, J. Hartog, S. Etalle, and A. Lekidis. (2019). "Network intrusion detection systems for in-vehicle network." Technical report, https://arxiv.org/abs/1905.11587 [Retrieved: 13-05-2020]

[13] S. F. Lokman, A. T. Othman, and M. H. Abu-Bakar, "Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review," EURASIP Journal on Wireless Communications and Networking. 2019, doi: 10.1186/s13638-019-1484-3.

[14] P Kaur, M. Kumar, and A. Bhandari "A review of detection approaches for distributed denial of service attacks" Systems Science & Control Engineering. pp. 301-320, Dec. 2016.

[15] A. Tomlinson, J. Bryans, and S. Shaikh, "Towards Viable Intrusion Detection Methods For The Automotive Controller Area Network," 2nd ACM Computer Science in Cars Symposium, pp. 1-9, Sep. 2018.

[16] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN based Intrusion Detection System for In-Vehicle Network," 2018 16th Annual Conference on Privacy, Security and Trust (PST), pp. 1-6, 2018, doi: 10.1109/PST.2018.8514157.

[17] M. J. Kang and J. W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," PLoS One, vol. 11, no. 6, 2016.

[18] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," 2015 World Congress on Industrial Control Systems Security (WCICSS), pp. 45-49, 2015, doi:10.1109/WCICSS.2015.7420322.

[19] C. Young, H. Olufowobi, G. Bloom, and J. Zambreno, "Automotive Intrusion Detection Based on Constant CAN Message Frequencies Across Vehicle Driving Modes," pp. 9-14, Mar. 2019, doi:10.1145/3309171.3309179.

[20] H. S. Sánchez, D. Rotondo, T. Escobet, V. Puig, J. Saludes, and J. Quevedo, "Detection of replay attacks in cyber-physical systems using a frequency-based signature," Journal of the Franklin Institute, vol. 356, no. 5, 2019.

[21] A. A. Sivasamy, and B. Sundan, "A dynamic intrusion detection system based on multivariate Hotelling's T2 statistics approach for network environments," The Scientific World Journal, 2015, doi:10.1155/2015/850153.

[22] A. Qayyum, M. H. Islam, and M. Jamil, "Taxonomy of statistical based anomaly detection techniques for intrusion detection," IEEE Symposium on Emerging Technologies, pp. 270-276, doi:10.1109/ICET.2005.1558893.

[23] S. Jain, Q. Wang, M. T. Arafin, and J. Guajardo, "Probing Attacks on Physical Layer Key Agreement for Automotive Controller Area Networks," Asian Hardware Oriented Security and Trust Symposium, pp. 7-12, 2018.

[24] Y. Kulah, B. Dincer, C. Yilmaz, and E. Savas, "SpyDetector: An approach for detecting side-channel attacks at runtime," International Journal of Information Security, vol. 18, pp. 393–422, doi.org/10.1007/s10207-018-0411-7.

[25] J. Wang, Y. Cheng, Q. Li, and Y. Jiang, "Interface-Based Side Channel Attack Against Intel SGX," Oct. 2018, https://arxiv.org/abs/1811.05378 [Retrieved: 13-05-2020]

[26] N. Weichbrodt, P. Aublin, and R. Kapitza, "sgx-perf: A Performance Analysis Tool for Intel SGX Enclaves," 19th International Middleware Conference, pp. 201-213, doi:10.1145/3274808.3274824.

[27] A. Binun, A. Bloch, S. Dolev, M. R. Kahil, B. Menuhin, R. Yagel, T. Coupaye, M. Lacoste, A. Wailly. "Self-stabilizing virtual machine hypervisor architecture for resilient cloud," 2014 IEEE World Congress on Services pp. 200-207, June 2014. IEEE.

[28] M. Sabt, M. Achemlal and A. Bouabdallah, "Trusted Execution Environment: What It is, and What It is Not," 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, pp. 57-64, 2015 doi:10.1109/Trustcom.2015.357.

[29] "Trusted Execution Environment (TEE) 101: A Primer", Secure Technology Alliance, White Paper, Version 1.0, April 2018.

[30] AMD, AMD Secure Processor technology (AMD-SP), https://www.amd.com/en/technologies/security [Retrieved: 13-05-2020]

[31] "GlobalPlatform based Trusted Execution Environment and TrustZone-Ready: The foundations for trusted services", ARM, White Paper, October 2013.

[32] M. Wolf and T. Gendrullis, "Design, implementation, and evaluation of a vehicular hardware security module," International Conference on Information Security and Cryptology, pp. 302-318, Nov. 2011.

[33] J.-E. Ekberg, K. Kostiainen, and N. Asokan, "Trusted Execution Environments on Mobile Devices," ACM CCS 2013 Tutorial, https://www.cs.helsinki.fi/group/secures/CCS-tutorial/tutorial-slides.pdf [Retrieved : 13-05-2020].

[34] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A. Sadeghi, "Software Grand Exposure: SGX Cache Attacks Are Practical," WOOT'17 Proceedings of the 11th USENIX Conference on Offensive Technologies, 2017.

[35] M. Schwarz, S. Weiser, D. Gruss, C. Maurice, and S. Mangard, "Malware Guard Extension: Using SGX to Conceal Cache Attacks," In: Polychronakis M., Meier M. (eds) Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA 2017. Lecture Notes in Computer Science, vol 10327. Springer.

[36] AUTOSAR History, https://www.autosar.org/about/history/, [Retrieved : 10-02-2020].

[37] AUTOSAR, "AUTOSAR: Layered Software Architecture.", https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf [Retrieved: 13-05-2020]

[38] C. Schlegel, "The role of CAN in the age of Ethernet and IoT," International CAN Conference (iCC), 2017.

[39] "CLIPS, A Tool for Building Expert Systems," http://www.clipsrules.net/index.html [Retrieved: 13-03-2020].

[40] A. Wasicek, M. D. Pesé, A. Weimerskirch, Y. Burakova, K. Singh "Context-aware Intrusion Detection in Automotive Control Systems," 5th ESCAR Conference, 2017.

 ISBN: 978-1-61208-795-5