Towards bridging the gap between modern and legacy automotive ECUs:

A Software-based Security Framework for Legacy ECUs

Ashok Samraj Thangarajan[1], Mahmoud Ammar[1], Bruno Crispo[1,2], and Danny Hughes[1]

[1]imec-DistriNet, KU Leuven
E-mail: {firstname.lastname}@cs.kuleuven.be
[2]University of Trento, Italy
E-mail: bruno.crispo@unitn.it

Abstract—**Modern automotive architectures are complex and often comprise of hundreds of electronic control units (ECUs). These ECUs provide diverse services including infotainment, telematics, diagnostics, advanced driving assistance, and many others. The availability of such services is mainly attained by the increasing connectivity with the external world, thus expanding the attack surface. In recent years, automotive original equipment manufacturers (OEMs) and ECU suppliers have become cautious of cyber attacks and have begun fortifying the most vulnerable systems, with hardware-based security modules that enable sandboxing, secure boot, secure software updates and end-to-end message authentication. Nevertheless, insecure legacy ECUs are still in-use in modern vehicles due to price and design complexity issues. Legacy ECUs depend on simple microcontrollers, that lack any kind of hardware-based security. This makes it essential to bridge the gap between modern and legacy ECUs through software-based security by which cyber attacks can be mitigated, thus enhancing the security of vehicles. This paper provides one more step towards highly secure vehicles by introducing a lightweight software-based security framework which provides legacy ECUs with software-based virtualization and protection features along with custom security services. We discuss the motivation for pure software-based approaches, explore the various requirements and advantages obtained, and give an initial insight of the design rationale. Furthermore, we provide a proof of concept implementation and evaluation with a demonstrative use case illustrating the importance of such framework in delivering new diagnostics security services to legacy ECUs.**

Index Terms—**Automotive Security, Legacy ECU, Memory Protection, Trusted Computing.**

I. INTRODUCTION

Modern vehicles have become very complex and sophisticated after having been equipped with multiple on-board microcontrollers, that are identified as Electronic Control Units (ECUs). The increasing deployment of ECUs along with the potential availability of various communication technologies in vehicles has significantly changed the means of transportation by providing passengers with dozens of services that enhance safety, provide entertainment, and keep connectivity with the external world. However, this fast evolution has brought various security and privacy challenges to the automotive domain as connected vehicles have become attractive targets to a wide spectrum of cyber attacks that can be serious enough to threaten people's lives [1]–[5]. Abstractly speaking, the current automotive standards along with the complex automotive architectures, complicate the task of adding security to all ECUs. Nevertheless, efforts are going on by both the research community and industry to address security issues.

Problem statement. While some automotive ECUs provide hardware-based security mechanisms, with their cost sensitive design, developing a vehicle that is totally composed of such kind of ECUs is less-than-desirable. Furthermore, it is well known that automotive OEMs spend several years in the design of new automotive platforms. In the fast evolving security world, this period is long enough to see radical changes in the security landscape. However, redesigning nearly-completed models for security purposes with the consideration of automotive standards is time consuming and very expensive. Moreover, redesigning critical ECUs introduces complex testing and regulatory concerns. This leads to selectively reusing legacy ECUs, resulting in automotive platforms carrying a combination of legacy and new ECUs.

Due to the clear security issues at every layer of vehicle architectures, a significant attention has been paid to secure both inter-vehicle [6]–[9] and intra-vehicle [10]–[15] communications. To the best of our knowledge, all of these proposals would either build atop modern (secure) ECUs or assume trusted ECUs and focus only on the communication medium (e.g. CAN bus) [16]. Since ECUs are the building blocks of vehicle platforms, any security proposal that would not consider realistic assumptions of secure ECUs (regardless their class) is incomplete and thus inefficient [1], [3], [17]. On the other hand, the importance of securing legacy ECUs in an automotive platform has received comparatively little attention by proposals [18] that demand unrealistic changes to all legacy ECUs in vehicles to support hardware extensions.

Contribution. To this end, we propose SECURE, a pure software-based security framework, that bridges the gap between legacy ECUs and modern ones. SECURE guarantees a base level of security by providing memory protection. Currently, the code in legacy ECUs executes within a single shared address space without any form of protection or isolation. Accordingly, it is easy to inject a malware that is capable of tampering with the functionality of corresponding ECU, thus compromising the security of entire vehicle system, as demonstrated in [19]. Consequently, there is a great need for a memory protection mechanism to isolate and protect the trusted code from other untrusted software modules that could run within the same address space, thus preventing faulty or malicious ones from affecting the security of vehicle. In nutshell, memory protection is a fundamental security property that is pivotal to implement any other security mechanisms (e.g. CAN authentication). It provides a way to control memory access rights and prevent damage or leakage of private data through unauthorized access that could occur by a software bug or malware infection. Given the difficulty of realizing the memory protection feature though hardware support due to the aforementioned reasons, software-based solutions can handle this issue, especially in the automotive domain, in which, the vast majority of attacks are performed remotely [2], [3], [19].

Paper outline. The remainder of this paper is organized as follows. Section II describes the design of SECURE. The importance of SECURE with regards to an automotive usecase is discussed in Section III. Implementation and evaluation results are reported in Section IV. Section V discusses related work and Section VI concludes.

II. Design Rationale of SECURE

Overview. The vast majority of legacy ECUs are based on microcontrollers that are designed to fulfill automotive standards, with safety and regulatory compliance as prime goals and being optimized for cost. Hence, these ECUs do not provide any form of hardware-based protection. We overcome this limitation by introducing SECURE, a pure software-based solution that allows several security features (e.g. secure OTA updates, remote attestation, etc.) to be added to legacy ECUs. The design of SECURE shares similarities with the Software-based Fault Isolation (SFI) approach [20] that has been proposed for computer systems to prevent faults in untrusted software modules from corrupting other software on processors with a single shared address space and no memory protection. Analogously, SECURE uses selective software virtualization and machine-level code verification to ensure full isolation and protection of the TCM (Trusted Computing Module) from untrusted application software. Figure 1 provides a high level overview of SECURE. It composes of two main tiers, where the top one targets the ECU side, and the other one represents the ECU development environment (e.g. the toolchain).

A. Adversary model

We assume an adversary with a full access to the network, but cannot physically tamper with any of the ECUs. The adversary may try to remotely compromise any of the ECUs through one of the potential attack vectors in the vehicle. She can inject a malware, re-flash the software, or perform any illegal operation through the network. We rule out all kinds of side channel and DoS attacks. We assume that the software architecture of SECURE is trusted and bug-free.

B. Architecture of SECURE

SECURE targets legacy ECUs with microcontrollers having the following characteristics: (i) no memory protection, (ii) single thread of execution, (iii) global interrupts disabling, and

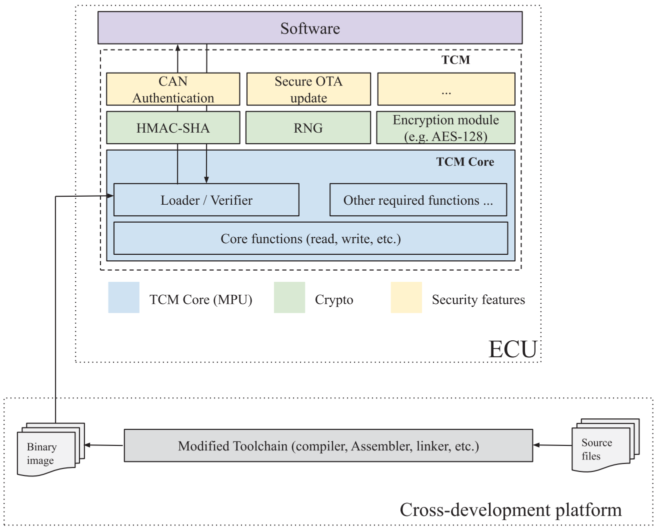
Software
TCM
CAN Authentication
Secure OTA update
...
HMAC-SHA
RNG
Encryption module (e.g. AES-128)
TCM Core
Loader / Verifier
Other required functions ...
Core functions (read, write, etc.)
TCM Core (MPU)
Crypto
Security features
ECU
Binary image
Modified Toolchain (compiler, Assembler, linker, etc.)
Source files
Cross-development platform

Fig. 1: An overview of SECURE.

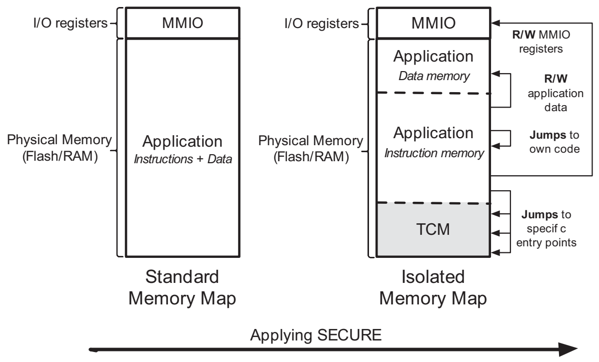

Fig. 2: A layout of secure memory after applying SECURE.

(iv) having a sufficient flash memory to store the TCM. It ensures memory protection and other custom security services at two levels: (i) the ECU itself, and (ii) a modified cross-development toolchain that is used for compiling binaries before deployment.

In particular, SECURE reserves part of the memory of an ECU for the TCM, which is always installed using a physical programming device, e.g. JTAG, prior to the integration of ECU in the vehicle. Deployment of the application software should go through the TCM, where the TCM Core has a function called *verifier*, to guarantee that none of the instructions of the deployed application performs illegal operations (e.g. accessing secure memory area) or causes crashes, as shown in Figure 1. In Figure 1, the base layer at the ECU side exemplifies the TCM that should be loaded on the ECU before deploying the binary code in order to provide a root of trust. The two middle layers are optional and can be added to each ECU as needed. Therefore, the entire design and implementation of SECURE is component-based where trusted software components and security features can be loaded and unloaded as required. The top layer represents the application software that should run on the ECU. This software is compiled through the modified toolchain (in the other tier) and deployed though the *verifier* component to double check that it does not violate the rules enforced by the TCM. The rules represent the access rights for each part of the memory. We visualize these rules in Figure 2. As such the TCM code is considered to be immutable and resides in a *virtual* ROM memory.

The TCM code is subject to no restrictions, whereas the untrusted application software has some rules as follows. First, jump operations are only allowed within the application instruction memory. A controlled interaction with the TCM is allowed through specific entry points, i.e. the *verifier* function. Second, read and write operations can be performed in the data memory or Mapped IO (MMIO) locations. Accordingly, read or write operations cannot be executed in the application instruction or TCM memory. Third, deployment or secure over the air updates (OTA) of any software should only occur through the TCM. This property is enforced by second rule that disallows an application to write in its own instruction memory, only the TCM is allowed to do so. All restrictions are enforced at the assembly level, where incoming applications are verified by the TCM at load-time to ensure that they adhere to the rules listed above.

C. *SECURE Toolchain*

The modified toolchain consists of a set of cross-compilers along with standard libraries for each considered architecture (e.g. Harvard, Von Neumann, etc.). The goal of the toolchain is to produce a safe binary code that should run on the corresponding ECU. Along with the TCM, this toolchain forms the backbone of providing security to legacy ECUs. In a standard toolchain, the process of producing binaries goes through compiling the source code to produce assembly files, which are further processed by the assembler to produce binary object files, and then ends up with the linker that combines all object files in one file. Some instruction (e.g. indirect jump) could contain dynamic addresses that are identified at the runtime. Such instruction can be unsafe. Nonetheless, they are essential for normal operations. SECURE adds a post-processor in the toolchain between the compiler and assembler phase, which substitutes all unsafe dynamic assembly instructions with calls to their secure virtualized ones. At runtime, the TCM checks whether the address calculated is located within the boundaries of the application instruction memory. If so, it allows this operation as it does not violate the rules. Otherwise, it performs soft reset to prevent it. Please note that the security properties of SECURE-enabled ECUs are maintained even when adversaries use their own toolchains or write hand-crafted assembly by load-time verification on the ECU itself.

D. *Advantages of SECURE*

In principle, the advantage of SECURE is multi-fold. First, it is cost effective, thus enhancing the security of vehicles from different price segments. Second, it is easier to be applied on the currently on-road vehicles and thus patching common vulnerabilities without any significant overhead compared to hardware-based solutions. Third, the vast majority of cyber attacks on vehicles are performed remotely, as physical attacks are unlikely to appear and are practically infeasible and not scalable [2], [3], [19]. Finally, as mentioned, the targeted ECUs do not support any kind of hardware extensions, leaving no choices other than software-based solutions. A high level of reliability and trust can be guaranteed by leveraging formal verification techniques to formally verify some properties of the framework's software architecture (e.g. memory-safety, freedom from crashes, semantic correctness, etc.).

III. Automotive Usecase: Securing the OBD-II

In this section, as a demonstrative usecase, we highlight the vulnerability of On-Board Diagnostics (OBD-II) port and show how SECURE can be leveraged to protect it.

A. *OBD-II: Problem Statement*

The OBD-II port is a mandatory interface in all vehicles in the U.S. and Europe, which enables a diagnostics tool to be physically connected to the intra-vehicle network in order to quickly analyze and perform diagnostics of the vehicle. As standardised, the OBD-II port provides full access to the intra-vehicle network and allows programming the various connected ECUs. In modern and connected vehicles, this privilege is supported through a remote access (i.e. through cellular dongles) too, making OBD-II port the most critical automotive interface that should be secured [21]. In the last decade, we have witnessed various attacks that are performed either remotely or physically through OBD-II port, affecting the entire vehicle security and safety [1], [2], [22]–[24].

Service identifier (SID) 27 in ISO 14229-1 [25] provides a security service for protecting the vehicle from unauthorized access during diagnostics sessions. This service depends on a simple challenge-response protocol, called the seed-key protocol, in which the ECU that delivers extended diagnostics access through the OBD-II port, challenges the connected device with a seed in order to generate the correct response for authentication, taking into account that both parties share in advance a secret key and a cryptographically secure function. Apart from being inaccurate and loose, the standard is outdated and not evolving to cope with current emerging threats, as attacks have been reported over this weak security technique [26]. As a result, AUTOSAR has released version 4.4 specification for the classic platform [27], where they implement a solution on SID29 that delivers authentication service, based on the Draft International Standard (DIS) of Unified Diagnostics Service (UDS) ISO 14229-1-2018 [28]. In that, they use a certificate-based protocol with a role-based access control model, where the various OBD-II-connected devices should be authenticated depending on their roles by the edge ECUs. The main limitation of this proposal is that it is only applicable to the currently being designed modern automotive architectures where they require using several hardware-based cryptographic modules for managing secret keys and performing the various security operations. Thus, authentication can be performed only by modern and powerful ECUs. Hence, we leverage SECURE, to overcome this limitation, so that the AUTOSAR SID29 implementation [27] can also be applied on the currently on-road vehicles and legacy ECUs.

B. *OBD-II: Security Contribution*

We have evaluated our approach by implementing a simplified version of AUTOSAR protocol [27], for legacy ECUs. The main difference is that we do not consider exchanging full certificates as validating them is time consuming process, but we consider exchanging only signatures. Furthermore, edge ECUs maintain only public keys and permissions, where all private keys are secured at an Internet-connected OEM-owned cloud server that is accessible by any OBD-II device. For further details about the protocol, we refer the reader to the AUTOSAR specifications [27].

Security Requirements. Secure storage is one of the various security requirements that should be met in the proposed protocol, in order to protected the public keys along with the permissions table from being tampered through arbitrary code execution or malware infection. In legacy ECUs, there is no way to enforce such protection. Thus, we have employed SECURE to provide memory protection and isolation. Hence, the permissions and public keys are only accessible through the responsible function of TCM.

IV. Implementation and Evaluation

A. *Implementation*

A prototype of SECURE has been implemented for one of the legacy ECUs that are employed in vehicles, DVK90CAN1 [29], which offers an 8-bit AT90CAN128 microcontroller [30] running at 8 MHz, with 4 KB of SRAM and 128 KB of flash memory. Atop SECURE, we have implemented the OBD-II security protocol. We have simulated the OEM server through connecting the OBD-II device (another development kit of type DVK90CAN1) to a local desktop computer through RS232 interface (in reality, a wireless connectivity is used). A CAN-to-OBD-II and OBD-II-to-CAN connectors are used to act as an intermediate bridge (OBD-II port) between the gateway and OBD-II-connected device. Furthermore, we have implemented the CAN stack (up to the session layer) with the support of an open source ISO-TP library [1]. Elliptic Curve Cryptography (ECC) (In particular: secp256rl for key pairs generation, Elliptic Curve Digital Signature Scheme (ECDSA) for signing and verifying signatures, and Elliptic Curve Diffie-Hellman (ECDH) for key agreement), and HMAC-SHA256 are used as cryptographic primitives.

B. *Evaluation*

Memory footprint. The memory overhead is limited to the flash memory and indicated in table I, where the core modules of SECURE do not exceed 1110 bytes, making it reasonable for high-constrained ECUs. our implementation incurs no static RAM overhead, which is used only as a stack to hold data temporarily during runtime.

[1]https://github.com/lishen2/isotp-c

TABLE I: The flash memory footprint of SECURE along with various cryptographic functions

Module	HMAC-SHA256	ECC	CAN	SECURE	Rest (UART, etc.)	Total
Size (Byte)	2378	17684	4560	1110	842	26574

TABLE II: Runtime overhead of the secure OBD-II protocol

Component	ECC				HMAC-SHA256
	KPGen	ECDSA sign	ECDSA verify	ECDH (KGen)	CAN MAC verification
AT90CAN128	6.84 s	7.274 s	8.123 s	6.838 s	23.77 ms

Runtime Overhead. After being deployed on the ECU, SECURE incurs no runtime overhead. It increases the deployment overhead of other applications for about 4.6% as it has to check the safety property of each instruction at the machine-level code. This percentage is measured by computing the average overhead of deploying four different applications with and without considering SECURE. Table II indicates the runtime overhead incurred by the various cryptographic primitives used in the secure OBD-II protocol. It shows that after passing the authentication phase, exchanging authenticated and authorized CAN messages consumes few milliseconds (about 23.77 ms).

Security Analysis. Assuming a memory-safe and error-free implementation of SECURE, the design of it shares similarities with the SFI approach [20], whose correctness has been formally verified in [31]. Thus, we assume that our framework is secure w.r.t our attacker model. The security of other security services built atop SECURE is limited to the security of their design and cryptographic modules used. In the case of secure OBD-II protocol, the design is standardized by AUTOSAR [27], and the minimum length of parameters and secret keys used adheres to the best security practices and recommendations mentioned by AUTOSAR [32].

V. Related Work

Multiple hardware-based virtualization techniques are proposed first for computer systems [33] and then for the IoT domain [34]. While such techniques have targeted high-end devices in each class, software-based approaches have been proposed for resource-constrained devices [35], [36]. However, they are all target, domain, task, or architecture specific.

In the automotive domain, virtualization-based security frameworks have been proposed at a large scale depending on hardware modules, targeting powerful ECUs [37]–[40]. We are unaware of any proposal targeting legacy ECUs except for [18] which demands unrealistic extension of hardware in legacy ECUs to support hardware-based isolation. To the best of our knowledge, we are the first to target the security of legacy ECUs using a software-based approach with a proof of concept implementation and evaluation.

VI. Conclusion

This paper describes SECURE, a pure software-based approach to provide legacy ECUs with MPU-like security guarantees with regards to remote-only attacks. Such approach is simple and cost efficient to bridge the gap between modern and legacy ECUs, a combination of which is usually found in modern vehicles. Additionally, design of SECURE enables advanced security solutions to be applied to on-road vehicles. We further exemplify the importance of SECURE through an implementation of an AUTOSAR-compliant authentication and access control mechanism to secure the OBD-II port, which run on top of SECURE securely. Considering the unlikelihood of physical attacks, SECURE can serve as a base to provide various security services, i.e. secure OTA software updates and attestation, in vehicles.

References

[1] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, *et al.*, "Experimental security analysis of a modern automobile," in *Security and Privacy (SP), 2010 IEEE Symposium on*, pp. 447–462, IEEE, 2010.

[2] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, *et al.*, "Comprehensive experimental analyses of automotive attack surfaces.," in *USENIX Security Symposium*, pp. 77–92, San Francisco, 2011.

[3] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," *black hat USA*, vol. 2014, p. 94, 2014.

[4] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *DEF CON 23*, p. 91, 2015.

[5] R. M. Ishtiaq Roufa, H. Mustafaa, S. O. Travis Taylora, W. Xua, M. Gruteserb, W. Trappeb, and I. Seskarb, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study," in *19th USENIX Security Symposium, Washington DC*, pp. 11–13, 2010.

[6] C. A. Kerrache, N. Lagraa, C. T. Calafate, and A. Lakas, "Tfdd: A trust-based framework for reliable data delivery and dos defense in vanets," *Vehicular Communications*, vol. 9, pp. 254–267, 2017.

[7] M. Durresi, A. Durresi, and L. Barolli, "Secure inter vehicle communications," in *Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on*, pp. 177–183, IEEE, 2012.

[8] N. J. Patel and R. H. Jhaveri, "Trust based approaches for secure routing in vanet: a survey," *Procedia Computer Science*, vol. 45, pp. 592–601, 2015.

[9] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung, and J.-P. Hubaux, "Secure vehicular communication systems: design and architecture," *arXiv preprint arXiv:0912.5391*, 2009.

[10] B. Groza, S. Murvay, A. Van Herrewege, and I. Verbauwhede, "Libracan: a lightweight broadcast authentication protocol for controller area networks," in *International Conference on Cryptology and Network Security*, pp. 185–200, Springer, 2012.

[11] S. Nürnberger and C. Rossow, "–vatican–vetted, authenticated can bus," in *International Conference on Cryptographic Hardware and Embedded Systems*, pp. 106–124, Springer, 2016.

[12] O. Hartkopp and R. M. SCHILLING, "Message authenticated can," in *Escar Conference, Berlin, Germany*, 2012.

[13] A. Hazem and H. Fahmy, "Lcap-a lightweight can authentication protocol for securing in-vehicle networks," in *10th escar Embedded Security in Cars Conference, Berlin, Germany*, vol. 6, 2012.

[14] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata, "Cacan-centralized authentication system in can (controller area network)," in *14th Int. Conf. on Embedded Security in Cars (ESCAR 2014)*, 2014.

[15] A.-I. Radu and F. D. Garcia, "Leia: A lightweight authentication protocol for can," in *European Symposium on Research in Computer Security*, pp. 283–300, Springer, 2016.

[16] M. Wolf, A. Weimerskirch, and C. Paar, "Security in automotive bus systems," in *Workshop on Embedded Security in Cars*, 2004.

[17] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive can networks–practical examples and selected short-term countermeasures," in *International Conference on Computer Safety, Reliability, and Security*, pp. 235–248, Springer, 2008.

[18] J. Van Bulck, J. T. Mühlberg, and F. Piessens, "Vulcan: Efficient component authentication and software isolation for automotive control networks," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, pp. 225–237, ACM, 2017.

[19] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, p. 91, 2015.

[20] R. Wahbe, S. Lucco, T. E. Anderson, and S. L. Graham, "Efficient software-based fault isolation," *ACM SIGOPS Operating Systems Review*, vol. 27, pp. 203–216, dec 1993.

[21] C. Bernardini, M. R. Asghar, and B. Crispo, "Security and privacy in vehicular communications: Challenges and opportunities," *Vehicular Communications*, 2017.

[22] Argus Cyber Security, "A remote attack on an aftermarket telematics service." https://argus-sec.com/remote-attack-aftermarket-telematics-service/, 2014. [Online; accessed 07-January-2019].

[23] Argus Cyber Security, "A Remote Attack on the Bosch Drivelog Connector Dongle ." https://argus-sec.com/remote-attack-bosch-drivelog-connector-dongle, 2017. [Online; accessed 09-January-2019].

[24] ExtremeTech, "Hack the diagnostics connector, steal yourself a BMW in 3 minutes." http://www.extremetech.com/extreme/132526-hack-the-diagnostics-connector-steal-yourself-a-bmw-in-3-minutes, 2012. [Online; accessed 07-January-2019].

[25] ISO, "14229-1: 2013 road vehicles–unified diagnostic services (uds)–part 1: Specification and requirements," 2013.

[26] M. Ring, T. Rensen, and R. Kriesten, "Evaluation of vehicle diagnostics security–implementation of a reproducible security access," *SECURWARE 2014*, vol. 213, 2014.

[27] "Specification of diagnostic communication manager cp release 4.4.0," standard, AUTOSAR, 2018.

[28] ISO/DIS, "14229-1: 2018 road vehicles–unified diagnostic services (uds)–part 1: Application layer," 2018.

[29] Atmel, "DVK90CAN1 Hardware User Guide." http://ww1.microchip.com/downloads/en/devicedoc/doc4381.pdf, 2008. [Online; accessed 01-January-2019].

[30] Atmel, "8-bit AVR Microcontroller with 32K/64K/128K Bytes of ISP Flash and CAN controller." http://ww1.microchip.com/downloads/en/devicedoc/doc7679.pdf, 2008. [Online; accessed 01-January-2019].

[31] L. Zhao, G. Li, B. De Sutter, and J. Regehr, "Armor: fully verified software fault isolation," in *Proceedings of the ninth ACM international conference on Embedded software*, pp. 289–298, ACM, 2011.

[32] "Specification of secure onboard communication autosar cp release 4.4," specification, AUTOSAR, 2018.

[33] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar, "Innovative instructions and software model for isolated execution.," *HASP@ ISCA*, vol. 10, 2013.

[34] T. Alves, "Trustzone: Integrated hardware and software security," *White paper*, 2004.

[35] W. Daniels, D. Hughes, M. Ammar, B. Crispo, N. Matthys, and W. Joosen, "S μ v-the security microvisor: a virtualisation-based security middleware for the internet of things," in *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Industrial Track*, pp. 36–42, ACM, 2017.

[36] R. Kumar, E. Kohler, and M. Srivastava, "Harbor: Software-based Memory Protection For Sensor Nodes," in *International Symposium on Information Processing in Sensor Networks (IPSN)*, IEEE, 2007.

[37] "The automotive grade linux software defined connected car architecture," report, AGL, 2018.

[38] D. Reinhardt, D. Kaule, and M. Kucera, "Achieving a scalable e/e-architecture using autosar and virtualization," *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, vol. 6, no. 2013-01-1399, pp. 489–497, 2013.

[39] H. Dakroub, A. Shaout, and A. Awajan, "Connected car architecture and virtualization," *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, vol. 9, no. 2016-01-0081, pp. 153–159, 2016.

[40] D. Reinhardt and G. Morgan, "An embedded hypervisor for safety-relevant automotive e/e-systems," in *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014)*, pp. 189–198, IEEE, 2014.