What Today's Serious Cyber Attacks on Cars Tell Us: Consequences for Automotive Security and Dependability

Markus Zoppelt and Ramin Tavakoli Kolagari

Nuremberg Institute of Technology, Hohfederstr. 40, 90489 Nuremberg, Germany
{markus.zoppelt,ramin.tavakolikolagari}@th-nuernberg.de

Abstract. Highly connected with the environment via various interfaces, cars have been the focus of malicious cyber attacks for years. These attacks are becoming an increasing burden for a society with growing vehicle autonomization: they are the sword of Damocles of future mobility. Therefore, research is particularly active in the area of vehicle IT security, and in part also in the area of dependability, in order to develop effective countermeasures and to maintain a minimum of one step ahead of hackers. This paper examines the known state-of-the-art security and dependability measures based on a detailed and systematic analysis of published cyber attacks on automotive software systems. The sobering result of the analysis of the cyber attacks with the model-based technique SAM (Security Abstraction Model) and a categorization of the examined attacks in relation to the known security and dependability measures is that most countermeasures against cyber attacks are hardly effective. They either are not applicable to the underlying problem or take effect too late; the intruder has already gained access to a substantial part of the vehicle when the countermeasures apply. The paper is thus contributing to an understanding of the gaps that exist today in the area of vehicle security and dependability and concludes concrete research challenges.

Keywords: Automotive Security · Automotive System Architecture · Dependability · Model-Driven Engineering Methodologies

1 Introduction

The development of automobiles has ever been a subject to constant change. None of these changes, however, were as striking as the incorporation of software. Since the turn of the millennium, scientific contributions on software security of cars have been published [5, 13, 37]. Earlier publications are practically non-existent due to the scarcity of software in vehicles back then. One of the first systematic analyses of attacks on automotive (software) security [38, p. 6f] describes the prevailing attacks in the automotive sector as either theft or modification of critical components: for example, an attacker would like to achieve financial gain by stealing the car or valuable components. Modification refers to the car owner that would like to change components (tuning), for example in order

to increase the value of the car (reduced mileage) or decrease it for taxation reasons (increased mileage). The analysis also mentions that attackers want to steal competitors' expertise and intellectual property. The cyber attacks against vehicles presented in this paper show that the attack potential has increased considerably in the last decade due to the interconnectivity and architecture of modern vehicles. The described attack motivations and attacks from the early days of the rise of software in vehicles make up only a small fraction of today's hackers' motivations and attacks. In comparison to those of the past, attacks on modern vehicles are particularly worrying because attackers can take control of the entire vehicle. This often requires no or only short physical access. Our society, which is on the threshold of autonomous mobility, takes this challenge seriously. Therefore, research is very active in the field of vehicle IT security and partly also in the field of functional safety (dependability) in order to develop effective countermeasures. Cyber attack protection does not initially imply dependability; this paper will argue, though, that at the interface between dependability and security research, innovative protection mechanisms emerge just as capable of providing protection against malicious attacks as established security measures. The countermeasures published so far are manifold adaptations of classical IT security approaches in the area of automotive security, for example [3, 4, 7, 15, 18, 19, 23, 26, 30].

The remainder of this paper is structured as follows: Section 2 reviews the most effective attacks on automotive software systems. In Section 3 we describe typical automotive security and dependability mechanisms and analyse their protection potential with respect to the published attacks described earlier. Section 4 gives an overview of related work in this field. In our conclusion in Section 5, we provide indications for future research challenges in the area of automotive security and dependability.

2 Attacks on Automotive Software Systems

In this section we first present the Security Abstraction Model (SAM) [40] and the Common Vulnerability Scoring System (CVSS) [20] for evaluating and analysing attack vectors on automotive software systems. Afterwards, we give an overview of today's serious attacks on modern, highly connected vehicles.

2.1 SAM and CVSS

In an earlier publication, we introduced SAM, a Security Abstraction Model for automotive software systems. SAM allows for a security analysis of automotive attack vectors. Systematic security analyses can be used to quantify the required effort for a potential attack. The approach tightly couples security management and model-based systems engineering by an abstract description of automotive security modeling principles. The resulting SAM language specification is based on security requirements elicited from common industrial scenarios. It is a suitable solution for representing attack vectors on vehicles and provides a thorough security modeling for the automotive industry. SAM has a close connection to

the architecture description via the coupling of *Item* from the architecture model. SAM attack models express all important criteria of attack vectors—from an adversary's motivation up until a breach—to allow for system's modeling in an early software engineering phase. Besides attack motivations, SAM also describes all intrinsic and temporal properties of an attack, e.g., impact on security goals (confidentiality, availability, integrity, etc.), attack complexity, affected item and the attackable property. SAM can be used with generic security scoring systems for attack rating like, e.g., the Common Vulnerability Scoring System (CVSS). The CVSS is an acclaimed industry standard for rating vulnerabilities in computer systems and proposes three different metric groups for calculating the vulnerability scores. The Base Metric Group reflects the intrinsic properties of an attack: from SAM's automotive-oriented perspective, this group therefore indicates the characteristics that result when the attack in question is aimed at the automotive domain in general. The Temporal Metric Group allows for adjustment of the score after more information of the exploited vulnerability is available. The CVSS provides an online calculator [1] where specific vulnerabilities can be referenced with a unique CVSS vector string. We will provide those vector strings below every SAM model of the respective attack. Readers who are interested in the attack properties of specific attacks are able to check them on the online calculator. The additional benefit of having SAM models compared to directly giving the properties and a vulnerability score is that not only the CVSS (or scoring systems in general) is used, but also the possibility to construct attack trees via sub-attacks and follow-up attacks. SAM is also a method for hierarchical processing of attack vectors. In terms of substance, this goes beyond the classic attack rating. SAM makes the scoring system available to the software architect or in other words: SAM's strength lies in its ability to integrate with existing automotive architectures. What is brought together are architectural considerations with pure security considerations as regards the attack itself (attack vectors that can be derived from it, motivations, target areas) and all scoring systems that are known, which can derive all necessary information from the properties.

2.2 Overview of the Attacks

Scientific contributions on software security in cars publish a large number of attacks on automotive software systems. Table 1 gives an overview of the most serious of the published attacks on modern, highly connected vehicles. The selection of the attacks was made strictly according to the following attack characteristics: The selected attacks 1) are aiming at a broad range of security goals, ideally all security goals, and 2) have high severity levels (CVSS Temporal Score greater than 4.0). The CVSS Vector String is omitted in the table, but is shown below each of the figures of the attack models later in this paper. For the purposes of this study, we differentiate between *gateway attacks* and *follow-up attacks*. Gateway attacks usually change the extent of a vulnerability and typically serve as door openers, enabling the adversary to launch one or more follow-up attacks. A typical follow-up attack would be to reverse Controller Area Network (CAN) bus messages to learn how the vehicle's Electrical Control Units (ECUs)

Table 1. The most serious attacks, sorted by CVSS [20] Temporal Score

Attack	AttackableProperty	Item	Score
Tesla Remote Control	Webkit Browser	Autopilot ECU	8.0 / 7.2
SecurityAccess via UDS	Substandard ciphers	Body Control Module	7.1 / 6.7
CAN Message Injection	(Multiple)	Pow. Steer. Contr. Mod.	7.0 / 6.5
BMW Remote Diagnostics	NBT Backdoor	Infotainment Domain	7.1 / 6.4
Control via OBD Injection	Clear CAN traffic	Diagnostics	7.7 / 6.3
Telematics Attack	SSH, SMS	Telematics Control Unit	6.4 / 6.1
Remote Keyless Entry	Rolling Code	Remote Keyless Entry	5.7 / 5.4
CAN DoS Attack	CAN Protocol	Any CAN bus	4.6 / 4.5

communicate and injecting malicious messages into the system. Some follow-up attacks are just as trivial as starting Denial-of-Service (DoS) attacks. Many follow-up attacks are fairly high-level though, e.g., remotely driving/steering the car, disturbing functions of the vehicle or disabling driver-assisting systems.

2.3 Tesla Remote Control Attack

One of the most serious attacks is the Tesla Remote Control Attack [24, 25, 35]. This gateway attack enables an adversary to break into the AutoPilot ECU (APE) via the Webkit Browser of the infotainment unit. The researchers of Tencent Keen Security Lab [35] have demonstrated how to remotely control and steer the vehicle, disturbing the autowipers by confusing the machine learning (ML) component with a technique called adversarial examples [11, 29] and eliminating the lane detection of the vehicle. The following is a brief explanation of the entities shown in Figure 1: The adversary in this scenario is a remote attacker with the attack motivation to harm car occupants by crashing the vehicle. The attack is possible when the mode of the vehicle is "Slow or Standing". The exploited vehicle feature is Tesla's Autopilot, specifically the item AutoPilot ECU (APE). The exploited vulnerability is the Webkit browser framework of the infotainment unit which offers the JSArray function. This function is the attackable property the adversary is looking for, i.e., his anchor of the attack. After analysing with the attack properties via the CVSS metrics, one can calculate the base score and temporal score of the attack and derive the requirement: code signing protection for over-the-air (OTA) updates.

For the remaining attacks in this paper, further textual explanation of the models is omitted. Readers might refer to the explanation of the entities of this attack or look at SAM / CVSS references.

2.4 Security Access via UDS Attack

Many security-sensitive preferences or functions of a vehicle are secured via the Unified Diagnostic Service (UDS). Getting advanced security access to an ECU makes it possible for an adversary to fully reprogram the respective ECU or get confidential information out of the ECU's secure memory, what makes this

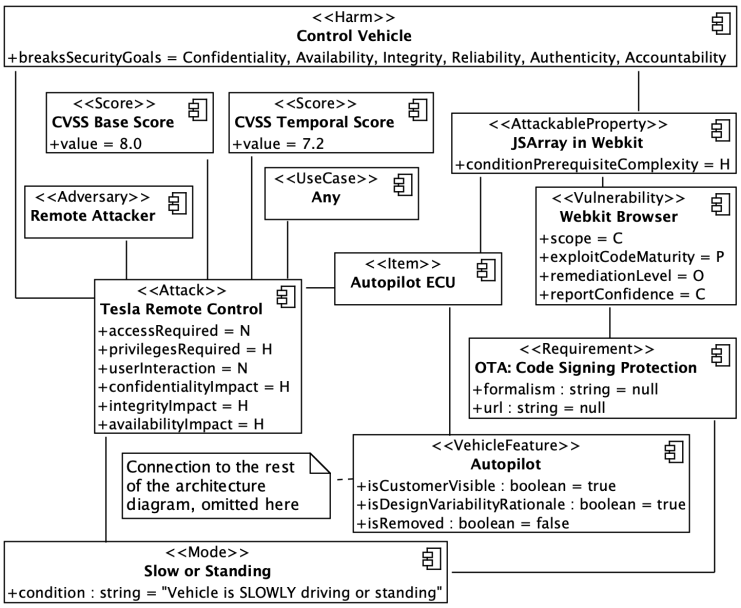

Fig. 1. SAM model of Tesla Remote Control Attack—CVSS v3.0 Vector String: CVSS:3.0/AV:N/AC:H/PR:H/UI:N/S:C/C:H/I:H/A:H/E:P/RL:O/RC:C

attack a gateway attack. In contrast to the Tesla Remote Control attack, the attack motivation is product modification. The Security Access via UDS Attack as shown by den Herrewegen [12] is illustrated in the SAM model in Figure 2.

2.5 CAN Message Injection Attack

Miller and Valasek's [36] attack on an unaltered passenger vehicle [21] was widely discussed in research and press. A CAN Message Injection Attack [22] is one of the logical consequences after a successful gateway attack. After an adversary has gained access to the powertrain, he can reverse engineer the messages communicated via the bus and inject his own malicious messages of choice. Once an adversary has the ability to send arbitrary network messages (e.g., via CAN) he is able to control the braking system, engine behaviours or the air vents, (un-)lock the doors, etc. Therefore, there is a strong need to secure the vehicle before the adversary can even gain access to the bus as then it is already too late. Figure 3 illustrates the CAN message injection attack in SAM.

2.6 BMW Remote Diagnostics Attack

The BMW Remote Diagnostics Attack [34] is the second attack by Tencent Keen Security Lab on our list. It is a hybrid of the Tesla Remote Control attack and the UDS Security Access attack. The researchers were able to control a BMW

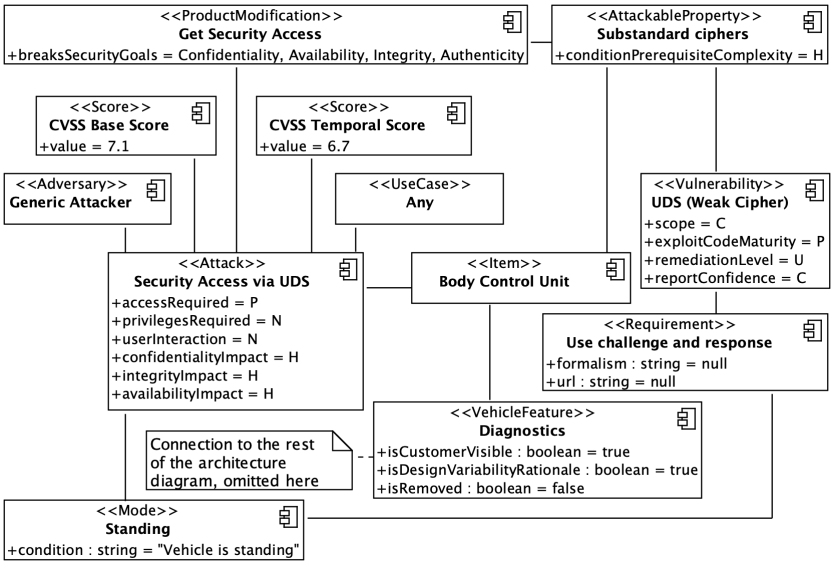

Fig. 2. SAM model of Security Access via UDS Attack—CVSS v3.0 Vector String: CVSS:3.0/AV:P/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H/E:P/RL:U/RC:C

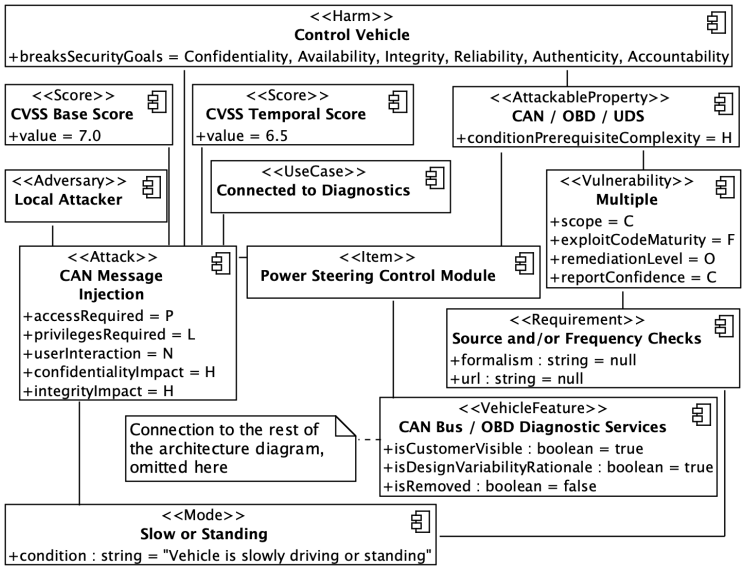

Fig. 3. SAM model of CAN Message Injection Attack—CVSS v3.0 Vector String: CVSS:3.0/AV:P/AC:H/PR:L/UI:N/S:C/C:H/I:H/A:H/E:F/RL:O/RC:C

after exploiting a back door in the in-vehicle infotainment system (also known as NBT Head Unit). This was possible because UDS was not locked at high speed. This gateway attack is shown as a SAM model in Figure 4.

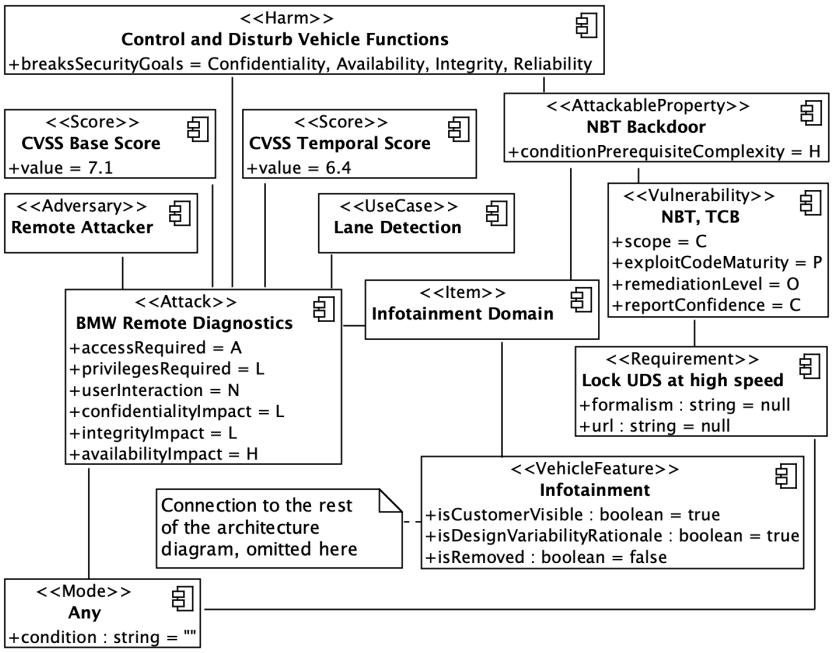

Fig. 4. SAM model of BMW Remote Diagnostics Attack—CVSS v3.0 Vector String: CVSS:3.0/AV:A/AC:H/PR:L/UI:N/S:C/C:L/I:L/A:H/E:P/RL:O/RC:C

2.7 OBD Injection Attack

On-board diagnostics (OBD) is a vehicle's self-diagnostic and reporting capability for vehicles. Over the OBD port, which is easily accessible inside the vehicle, many simple attack vectors are possible, especially in older car models, where OBD injection attacks [39] are astoundingly easy to perform gateway attacks. The SAM model of such an attack is omitted here. CVSS v3.0 Vector String: CVSS:3.0/AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:H/E:U/RL:W/RC:U

2.8 Telematics Attack

A large part of the remote attack surface of a modern vehicle is determined by telematics units. A potential adversary might use software defined radios or similar tools for remote exploitation of said telematics to obtain access to a device connected to the CAN bus or similar powertrain as a gateway attack. Foster [8] describes an example telematics attack. The SAM model is omitted here. CVSS v3.0 Vector String: CVSS:3.0/AV:N/AC:H/PR:L/UI:N/S:U/C:H/I:L/A:L/E:P/RL:U

2.9 Remote Keyless Entry Attack

Almost every modern vehicle has the ability for "keyless entry" or "keyless start engine". Those convenience features raise security risks as they provoke a gateway attack as shown by Garcia et al. [9]. With common hardware and low-level software skills, potential adversaries are able to unlock, open or start foreign vehicles after capturing and decoding radio signals for a Remote Keyless Entry Attack. The SAM model for this attack is omitted here as this particular attack is extensively described in various literature. CVSS v3.0 Vector String:
CVSS:3.0/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N/E:H/RL:W/RC:R

2.10 CAN DoS Attack

One of the simplest but highly safety-critical follow-up attack is a CAN DoS Attack as described by Palanca et al. [28]. Due to the CAN protocol definition, CAN bus messages are arbitrated by ID. Lower IDs, i.e., with more starting zeros, have higher priority than higher IDs. In a simple sense, spamming the bus with messages that have a lower ID leads to network constipation and is the equivalent to a classic denial-of-service attack. The SAM model for this attack is omitted here as well, as its vulnerability and attackable property are widely known. CVSS v3.0 Vector String:
CVSS:3.0/AV:P/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H/E:H/RL:W/RC:C

3 Countermeasures and Analysis

In this section we describe typical automotive security and dependability mechanisms and analyse their protection potential with respect to the published attacks described in Section 2. The result of the analysis of the relationship between attacks on vehicle security and the protection potential of countermeasures is presented in Table 2. It shows that message cryptography, as a popular representative of software security, is only effective for a small part of the attacks as a protection measure. Lesser known representatives can partially compensate, but overall, it can be stated that there are no adequate security protection mechanisms for some serious and well-known attacks on automotive software.

3.1 Message Cryptography (MC)

Message cryptography entails encryption, authentication and verification of messages communicated over the vehicle's bus, e.g., CAN, LIN, Flexray, Automotive Ethernet, etc. Message cryptography is an immensely large field of research and a big amount of apparent solutions does exist [3, 4, 19, 23, 26, 30]. Unfortunately, reliable and adaptive key distribution in heterogeneous automotive bus networks is a difficult challenge. Keys need to be distributed, updated and revoked in case of some soft- or hardware-updates. For some attacks, even properly implemented cryptography would offer just a partial protection, e.g., authenticity for CAN messages but no confidentiality due to the network topology. If an attacker gains access to an ECU, she might also retrieve the cryptographic keys. Cryptographically verifying messages would pose an obstacle for connecting rogue devices to the bus but would not mitigate remote attack scenarios.

3.2 ID Hopping (IDH)

ID Hopping is a technique to obfuscate network bindings or messages by changing ("hopping") between arbitration IDs without changing the actual arbitration. Order preserving encryption (OPE) is also considered as ID Hopping. This technique is also widely explored in the research field of automotive network security [15, 18, 19] and hinders adversaries to easily reverse engineer network messages.

3.3 Challenge and Response (CR)

Challenge and response is a common technique used widely in the security and network domain, though it is disturbingly unpopular in the automotive domain. Physical car keys (keyfobs) mostly still use a rolling code system when transmitting, enabling adversaries with mediocre skills and a software defined radio to perform replay or relay attacks. Those could easily be mitigated by using a challenge and response mechanism. Unfortunately, keyfobs are not equipped with the necessary hardware components due to financial reasons in the automotive industry.

3.4 ECU Hardening (ECUH)

ECU Hardening stops the adversary to change the state of the flashed software in any way. A popular application of ECU hardening is "Autonomous Security" and "Karamba Carwall" by Karamba Security [7] which hardens ECUs based on factory settings, eliminating the risks of false positives, detection delays, and performance drag issues. ECU hardening relies heavily on static analysis of the factory settings and firmware. It seems that this security mechanism is not really inquired by researchers but popular in the industry, as it is easy to implement and does not require increased effort.

3.5 Run Time Correctness (RTC)

Synergies between safety and security are exploited in the area of fault tolerance and software protection by tamper-tolerant software [16]. Shared approaches are developed to get programs run-time error free [10, p. 9ff]. While dependability aims at protection against systematic errors and random errors caused by malfunction or unintended interference, security additionally wants to protect against targeted, intended and possibly malicious manipulation. According to Kriha [17, p. 13f], security attacks are input or output related. This can be made verifiable by, e.g., validation frameworks. It must be stated, though, that the availability of complete frameworks for validation is generally rather deficient. Today's security vulnerabilities rarely lie in cryptographic algorithms or protocols but are almost always implementation-related, e.g., wrongly chosen (weak) ciphers or keys, memory safety, the inability to update software over-the-air, wrongly configured network interfaces, and more. Lists like the "Recent Vulnerability Notes" [2] demonstrate that vividly. Techniques like Voting could mitigate attacks that happen at random or are bound by probability, e.g., botnet attacks.

3.6 Integrity Protection (IP)

Dependability measures insist on maintaining integrity through redundancy checks, i.e., repeatedly or concurrently sending messages on the bus, checking ECU state, double computing, etc. Integrity checks for data through redundancy requires an adversary to compromise more individual pinpoints in order to break the security goal integrity.

3.7 Virtualisation (V)

Virtualisation as a dependability and security measure, e.g., running applications of different automotive safety integrity levels (ASIL) on different virtual computers can be used to virtually draw a line between applications and networks to seal off applications who are safety-critical from functionally unrelated applications. Glas et al. [10, p. 12f] show that virtualisation may serve as a measure for both dependability and security. Rosenstatter [32, p.4] and Othmane [27] also describe virtualisation and Virtual Local Area Networks (VLANs) as a possible solution for access control. The biggest benefit of virtualisation is, that it limits the scope of a vulnerability to U (unchanged), as it is isolated in the virtualised sandbox. The scope of a vulnerability is changed, if an attack impacts more than the vulnerable component. That means, that if the scope is unchanged, an attacker is not able to start a successful follow-up attack. Furthermore, lean use of virtualisation could obliterate lacking CAN authenticity, as standard CAN messages alone have no assignable source identifier.

3.8 Analysis and Mapping of Countermeasures on Attacks

The analysis of the respectively selected attacks (cf. Section 2.2) does not have to be very detailed in order to reveal the obvious mismatch between largely proposed countermeasures and their effective protection in practice. In this section we describe this mentioned mismatch by mapping effective countermeasures proposed by academic research to the selected attacks. This mapping shows that the most researched countermeasure (MC) is only effective in less than half of the attacks. The first discussed attack (Tesla Remote Control Attack) is analyzed both in text and in an illustrating figure. Due to the strict space limits, illustrations for the other attacks are omitted, but the analyses are always textually described.

The essential element of the Tesla Remote Attack is the access to the APE via exploiting Webkit, which must be capable of being updated from offboard. MC cannot help here, because it is about the exchange of an entire component. ECUH could completely prevent this attack, but does not offer any flexibility with regard to updates. V could at least limit the scope of the vulnerability from an unacceptable C to a U (see CVSS), requiring much more attack effort.

Although, some countermeasures prevent adversaries from accessing or successfully attacking a system, some attacks, engineer the attack vector around the countermeasure applied by targeting a sub-component of a system via unprotected communication channels and systematically traverse the system from

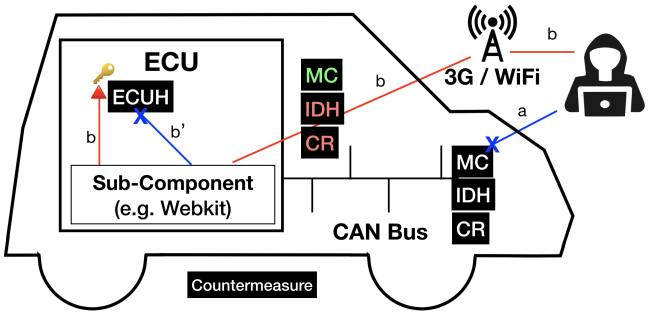

Fig. 5. Adversary attacking an ECU by-passing countermeasures (a) via an unprotected communication channel (b). ECUH (b') would have prevented the attack

the inside out. For example, in the Tesla remote attack, the adversaries could not directly access the secret key embedded in the ECU via network (a) so they first compromised the unprotected and vulnerable webkit framework (b) and got access to the key by-passing other countermeasures. In the (b) scenario, MC is used but does not stop the attacker because it is used as intended. IDH cannot be used because the channel is not a bus. CR cannot be applied either. If ECUH was in place for the (b) scenario then it would have prevented the attack (illustrated as (b') in Figure 5). The dependability measures are not depicted because they are too complex to be captured in an illustrating figure.

The SecurityAccess via UDS attack is possible because a weak cipher for the security challenge was chosen. A strong cipher in the UDS protocol would mitigate such an attack. In addition, IP could at least determine, that an ECU has been tampered with as integrity checks fail.

CAN message injection is one of the most researched attacks because the topology of the CAN bus allows rogue participants of the network to send arbitrary messages to the bus. Unfortunately, MC quickly reaches its limits due to the complex network topology and then can only if at all be used for partial encryption or authentication. V, however, could virtually separate critical applications, thereby mitigating cross-ECU message injection, maybe even offering some authenticity on top of the CAN protocol. V can also help in the case of the BMW Remote Diagnostics Attack for the same reason. Plus, ECUH would prevent attackers from tampering with the ECU firmware through diagnostic protocols. RTC can be used to limit the ability to corrupt software functions over diagnostic protocols.

The Control via OBD Injection attack shows many parallels to the CAN Message Injection attack, though it is in this case possible to prevent the attack with IDH or IP. IDH makes it harder for the attacker to reverse-engineer and send valid messages to the car, while IP would recognize that some injected messages are outliers.

Telematics attacks are very similar to CAN Message Injection attacks, except that their gateway attack vector, i.e., the actual telematics unit, can be secured better with MC, because their protocols are not entirely automotive specific.

The Remote Keyless Entry attack is actually already a solved problem, using CR. Unfortunately, industry (hardware) pricing policies prevent this solution from being used. Coming up with a software-only solution is a much sought-after research challenge.

CAN DoS attacks are possible because of CAN's arbitration characteristic. They cannot be stopped with MC or other security countermeasures. It is possible, though, to use RTC techniques, e.g., watchdogs to prevent such attacks.

Table 2. Analysis of automotive security and dependability countermeasures with respect to attacks against automotive software systems. We distinguish between security (left) and dependability (right) countermeasures. The X indicates a feasible protection; (X) indicates a partial protection

Attack	**MC**	**IDH**	**CR**	**ECUH**	**RTC**	**IP**	**V**
Tesla Remote Control [24, 25, 35]				X	X	X	X
SecurityAccess via UDS [12]	X					(X)	
CAN Message Injection [21, 22, 36]	(X)						X
BMW Remote Diagnostics [34]				X	X		X
Control via OBD Injection [39]	X	X				X	X
Telematics Attack [8]	X						X
Remote Keyless Entry [9]			X				
CAN DoS Attack [28]					X		

Table 2 shows that MC—where the majority of research is conducted—does not mitigate all of the top attacks. The attacks and countermeasures discussed in up-to-date research papers are—while being interesting in academia—not feasible in industrial automotive software. The attacks most successful in practice are usually not prevented by typical published security research results. Fortunately, dependability measures (RTC, IP, V) would offer some remarkable protections against the majority of our investigated attacks.

4 Related Work

Rosenstatter and Olovsson [32] provide a mapping between automotive security mechanisms and security levels in great detail. Auernhammer et al. [6] use a systematic mapping of published attacks on ML components on the security goals violated in autonomous vehicles. Their research shows that accountability (for ML) is not covered by literature as there have not yet been any attacks published, because accountability for ML is difficult to attack and the security goal is, therefore, not compulsory. Moreover, the work of Ray [31] lists practice and challenges in automotive security, discussing the need for extensibility and the constraints and considerations involved in achieving it. Huber's survey [14] shows how organizations from the automotive industry in the Euroregion tackle the challenge of integrating dependability and security aspects during system development. Their conclusion is that the utilization of a conceptual model unifying relevant documentation artifacts from requirements engineering, system modeling, risk assessment and evidence documentation could address these issues.

We addressed this by using SAM as a modeling technique. Finally, the six-step model for integrating autonomous vehicle safety and security analysis by Sabaliauskaite [33] achieves and maintains integration and alignment among safety and security artefacts throughout the entire AV life-cycle.

5 Conclusion

In this paper, we analyzed today's serious attacks with the model-based technique SAM and ranked them with the CVSS. The result of our study shows a revealing spectrum that research can actively take up and investigate. It is interesting that the majority of today's automotive security research is focused on (message) cryptography, which does not mitigate an essential part of the top attacks, although many other countermeasures offer advantages that should not be neglected. A highly topical research challenge is flexible extensibility, which at the same time provides protection against arbitrary manipulation (like ECUH) and would generally be a helpful approach for OTA updates. Moreover, it turned out that virtualisation is a promising countermeasure against attacks. A possible research challenge is a lean system for (embedded) virtualisation, as it limits the scope of the vulnerability and can obliterate some of the weaknesses in automotive security, e.g., CAN authenticity. Future research challenges should also focus on a combination of security and dependability countermeasures to provide adequate and flexible protection against cyber attacks on cars, e.g., mixing ECUH with extensibility or updatability enabled via cryptography. Autonomous vehicles in the future will probably be more susceptible to attacks than today's cars already are. Our work aims to offer the necessary insights and fundamentals to continue conducting relevant research in this domain.

Acknowledgment

This work is funded by the Bavarian State Ministry of Science and the Arts in the framework of the Centre Digitisation.Bavaria (ZD.B).
M.Z. was supported by the BayWISS Consortium Digitization.

References

1. Common Vulnerability Scoring System Version 3.0 Calculator. https://www.first.org/cvss/calculator/3.0, accessed: 2019-05-14
2. Vulnerability Notes Database. http://www.kb.cert.org/vuls/, accessed: 2014-10-29
3. vatiCAN: Vetted, authenticated CAN bus. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 9813 LNCS, pp. 106–124 (2016)
4. Agrawal, M., Huang, T., Zhou, J., Chang, D.: CAN-FD-Sec: Improving Security of CAN-FD Protocol. In: Hamid, B., Gallina, B., Shabtai, A., Elovici, Y., Garcia-Alfaro, J. (eds.) Security and Safety Interplay of Intelligent Software Systems. pp. 77–93. Springer International Publishing, Cham (2019)

5. Amendola, S.: Improving automotive security by evaluation—from security health check to common criteria. White paper, Security Research & Consulting GmbH **176** (2004)
6. Auernhammer, K., Tavakoli Kolagari, R., Zoppelt, M.: Attacks on Machine Learning : Lurking Danger for Accountability. In: Proceedings of the AAAI Workshop on Artificial Intelligence Safety 2019 co-located with the Thirty-Third AAAI Conference on Artificial Intelligence 2019 (AAAI 2019). p. 9. Honolulu, Hawaii (2019)
7. Barzilai, D.: Autonomous Security pp. 1–14 (2018)
8. Foster, I., Prudhomme, A., Koscher, K., Savage, S.: Fast and Vulnerable: A Story of Telematic Failures. 9th USENIX Workshop on Offensive Technologies (WOOT 15) (2015)
9. Garcia, F.D., Oswald, D., Kasper, T., Pavlidès, P.: Lock It and Still Lose It—On the (In)Security of Automotive Remote Keyless Entry Systems. Proceedings of the 25th USENIX Security Symposium pp. 929—-944 (2016)
10. Glas, B., Gebauer, C., Hänger, J., Heyl, A., Klarmann, J., Kriso, S., Vembar, P., Wörz, P.: Automotive safety and security integration challenges. Automotive-Safety & Security 2014 (2015)
11. Hayes, J., Danezis, G.: Machine Learning as an Adversarial Service: Learning Black-Box Adversarial Examples **2** (2017)
12. den Herrewegen, J., Garcia, F.D.: Beneath the Bonnet: A Breakdown of Diagnostic Security. In: European Symposium on Research in Computer Security. pp. 305–324. Springer (2018)
13. Hubaux, J.P., Capkun, S., Luo, J.: The security and privacy of smart vehicles. IEEE Security & Privacy (3), 49–55 (2004)
14. Huber, M., Brunner, M., Sauerwein, C., Carlan, C., Breu, R.: Roadblocks on the Highway to Secure Cars: An Exploratory Survey on the Current Safety and Security Practice of the Automotive Industry. In: Gallina, B., Skavhaug, A., Bitsch, F. (eds.) Computer Safety, Reliability, and Security. pp. 157–171. Springer International Publishing, Cham (2018)
15. Humayed, A., Luo, B.: Using ID-Hopping to Defend Against Targeted DoS on CAN. Proceedings of the 1st International Workshop on Safe Control of Connected and Autonomous Vehicles - SCAV'17 pp. 19–26 (2017)
16. Jakubowski, M.H., Saw, C.W.N., Venkatesan, R.: Tamper-Tolerant Software: Modeling and Implementation. In: Takagi, T., Mambo, M. (eds.) Advances in Information and Computer Security. pp. 125–139. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
17. Kriha, W., Schmitz, R.: Sichere Systeme: Konzepte, Architekturen und Frameworks. Springer-Verlag (2009)
18. Lukasiewycz, M., Mundhenk, P., Steinhorst, S.: Security-Aware Obfuscated Priority Assignment for Automotive CAN Platforms. ACM Transactions on Design Automation of Electronic Systems **21**(2), 1–27 (2016)
19. Madl, T., Brückmann, J., Hof, H.J.: CAN Obfuscation by Randomization (CANORa) A technology to prevent large-scale malware attacks on driverless autonomous vehicles (September), 1–7 (2018)
20. Mell, P., Scarfone, K., Romanosky, S.: Common Vulnerability Scoring System. IEEE Security & Privacy **4**(6) (2006)
21. Miller, C., Valasek, C.: Remote Exploitation of an Unaltered Passenger Vehicle. Defcon 23 **2015**, 1–91 (2015), http://illmatics.com/Remote Car Hacking.pdf
22. Miller, C., Valasek, C.: CAN Message Injection pp. 1–29 (2016), http://illmatics.com/can message injection.pdf

23. Mundhenk, P., Paverd, A., Mrowca, A., Steinhorst, S., Lukasiewycz, M., Fahmy, S.A., Chakraborty, S.: Security in Automotive Networks: Lightweight Authentication and Authorization (2017)
24. Nie, S., Liu, L., Du, Y.: Free-fall: hacking tesla from wireless to can bus. Defcon pp. 1–16 (2017)
25. Nie, S., Liu, L., Du, Y., Zhang, W.: Over-the-Air : How We Remotely Compromised the Gateway , Bcm , and Autopilot Ecus of Tesla Cars. Defcon **1** (2018)
26. Nowdehi, N., Lautenbach, A., Olovsson, T.: In-Vehicle CAN Message Authentication: An Evaluation Based on Industrial Criteria. In: Vehicular Technology Conference (VTC-Fall), 2017 IEEE 86th. pp. 1–7. IEEE (2017)
27. Othmane, L.B., Weffers, H., Mohamad, M.M., Wolf, M.: A survey of security and privacy in connected vehicles. In: Wireless Sensor and Mobile Ad-Hoc Networks Vehicular and Space Applications, pp. 217–247 (2015)
28. Palanca, A., Evenchick, E., Maggi, F., Zanero, S.: A stealth, selective, link-layer denial-of-service attack against automotive networks, vol. 10327 LNCS, pp. 185–206. Springer International Publishing, Cham (2017)
29. Papernot, N., McDaniel, P., Goodfellow, I.: Transferability in Machine Learning: From Phenomena to Black-Box Attacks Using Adversarial Samples (2016)
30. Radu, A.I., Garcia, F.D.: LeiA: A lightweight authentication protocol for CAN. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 9879 LNCS, pp. 283–300 (2016)
31. Ray, S., Wen Chen, Bhadra, J., Al Faruque, M.A.: Extensibility in automotive security: Current practice and challenges. In: 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC). pp. 1–6 (jun 2017)
32. Rosenstatter, T., Olovsson, T.: Towards a Standardized Mapping from Automotive Security Levels to Security Mechanisms. IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC **2018-Novem**, 1501–1507 (2018)
33. Sabaliauskaite, G., Liew, L.S., Cui, J.: Integrating autonomous vehicle safety and security analysis using stpa method and the six-step model. International Journal on Advances in Security **11**(1&2), 160–169 (2018)
34. Tencent Keen Security Lab: Experimental Security Assessment of BMW Cars: A Summary Report (2018)
35. Tencent Keen Security Lab: Experimental Security Research of Tesla Autopilot p. 38 (2019)
36. Valasek, C., Miller, C.: Adventures in Automotive Networks and Control Units. Technical White Paper **21**, 99 (2013)
37. Wolf, M., Weimerskirch, A., Paar, C.: Security in automotive bus systems. In: Workshop on Embedded Security in Cars (2004)
38. Wolf, M., Weimerskirch, A., Wollinger, T.: State of the Art: Embedding Security in Vehicles. EURASIP Journal on Embedded Systems **2007**(1), 74706 (jun 2007)
39. Zhang, Y., Ge, B., Li, X., Shi, B., Li, B.: Controlling a Car Through OBD Injection. In: Proceedings - 3rd IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2016 and 2nd IEEE International Conference of Scalable and Smart Cloud, SSC 2016. pp. 26–29 (2016)
40. Zoppelt, M., Tavakoli Kolagari, R.: SAM: A Security Abstraction Model for Automotive Software Systems. In: Security and Safety Interplay of Intelligent Software Systems, pp. 59–74. Springer (2018)