UnCle SAM: Modeling Cloud Attacks with the Automotive Security Abstraction Model

Markus Zoppelt

Department of Computer Science
Nuremberg Institute of Technology
Nuremberg, Bavaria 90489
Email: markus.zoppelt@th-nuernberg.de

Ramin Tavakoli Kolagari

Department of Computer Science
Nuremberg Institute of Technology
Nuremberg, Bavaria 90489
Email: ramin.tavakolikolagari@th-nuernberg.de

***Abstract*—Driverless (autonomous) vehicles will have greater attack potential than any other individual mobility vehicles ever before. Most intelligent vehicles require communication interfaces to the environment, direct connections (e.g., Vehicle-to-X (V2X)) to an Original Equipment Manufacturer (OEM) backend service or a cloud. By connecting to the Internet, which is not only necessary for the infotainment systems, cars could increasingly turn into targets for malware or botnet attacks. Remote control via the Internet by a remote attacker is also conceivable, as has already been impressively demonstrated. This paper examines security modeling for cloud-based remote attacks on autonomous vehicles using a Security Abstraction Model (SAM) for automotive software systems). SAM adds to the early phases of (automotive) software architecture development by explicitly documenting attacks and handling them with security techniques. SAM also provides the basis for comprehensive security analysis techniques, such as the already available Common Vulnerability Scoring System (CVSS) or any other attack assessment system.**

Keywords–Automotive Security; Automotive Software Engineering; Security Modeling; Cloud Attacks; OTA Updates.

I. INTRODUCTION

Modern cars are interconnected networks, with potentially more than 150 Electronic Control Units (ECUs) in luxury models communicating with one another and with the environment (V2X communication). In recent years, car manufacturers produced vehicles that are connected to the Internet and are providing cloud services, e.g., Tesla's mobile app, BMW iDrive or Audi Connect. In most cases, the user can even monitor or control parts of the vehicle using a mobile application or cloud service. These convenience features are designed to attract new customers but may impede some of the security goals by downright enabling a barrage of possible attack vectors. Attackers do not target cars in the same way as they would attack standard computer systems; cars use different networks, protocols and architectures [1], [2]. Moreover, cars carry burdensome legacy mechanisms with insecure and unencrypted protocols (e.g., CAN, Controller Area Network) in their system design and were originally not designed in line with today's security principles [3], [4]. Secure automotive network architectures were not prioritized in the past due to the general preconception in the last three decades that cars are secure because of their technical complexity (security by obscurity). The goal is to establish the principle of security by design, not only for automotive software systems but for cloud services as well. However, numerous attack vectors [5], [6], [7] on cars and their network of ECUs, actuators and sensors exist. In contrast to desktop computers, human lives are at stake when these "driving computers" are the target of an attack.

In an earlier publication, we introduced SAM: a Security Abstraction Model for automotive software systems [8]. The examples discussed in [8] are direct attack vectors. In this paper, however, we will focus on remote attack scenarios in the automotive domain considering cloud attacks and over-the-air (OTA) updates. Figure 1 illustrates the difference between direct attack vectors and cloud attack vectors. Cloud attack vectors target the vehicle indirectly over cloud infrastructure, e.g., the OEM's server.

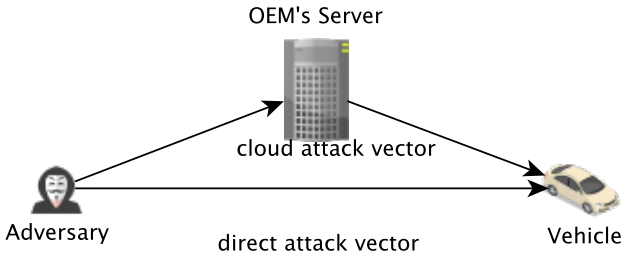

Figure 1. Direct Attack Vector vs. Cloud Attack Vector

In this paper, we show:

- A list of certain cloud attack vectors that cause major threats to automotive systems.
- A revamped version of SAM, featuring the ability to use any type of scoring system for attack rating.
- An explanation of how to use a well-known security scoring system (like CVSS) with SAM.
- A practical case study, applying the new version of SAM to the discussed cloud attacks from our list.

The rest of this paper is structured as follows: Section II reviews the state of the art on remote / cloud attacks on modern vehicles. Section III discusses possible remote attack scenarios and the security challenges of cloud attacks and OTA updates in the automotive domain. Section IV presents the current version of SAM and how to use any generic scoring system for attack rating. Section V illustrates two examples of remote

 ISBN: 978-1-61208-703-0

and OTA update attacks using SAM for security modeling. Section VI reviews related work on security architectures for automotive software systems. Section VII concludes the paper and gives an outlook on future work.

II. State of the Art

Modern vehicles communicate critical and safety relevant commands over a shared powertrain between different types of ECUs. The most popular broadcast network used for communication is the CAN bus. CAN bus messages are unencrypted and unsigned by default, because this just wasn't an issue when CAN was designed. Remote exploitation of a single ECU item on the CAN bus causes a major security threat because it allows an attacker to send valid (and potentially harmful) messages over the bus to critical parts of the vehicle's ECU network. Various attacks [7] have shown that adversaries are able to cause serious threats by compromising a vehicle's ECU (or adding an external device) and sending malicious CAN commands to the devices listening on the bus. Once the adversary has the ability to send arbitrary CAN messages, she is able to control the braking system, engine behaviour, the air vents, (un-)locking the doors, etc. Therefore there is a strong need to secure the vehicle before the adversary even can gain access to the CAN bus. If the adversary has access to the powertrain it is already too late.

Modern vehicles have a tremendous amount of remote attack surfaces like wireless protocols, mobile application support and more. Examples of specific remote technologies are the passive anti-theft system (PATS), tire pressure monitoring systems (TPMS), remote keyless entry (RKE), Bluetooth, radio data systems (3G, 4G, LTE, 5G, etc.), Wi-Fi and telematics. Miller and Valasek describe numerous remote exploits targeting said technologies [7]. Typically, infotainment systems tend to feature Internet access and support for third-party applications. If one or some of these applications or services become vulnerable to hacking attacks over the network, an adversary might be able to control a crucial participant in the physical network of the vehicle: the CAN bus. Automotive Ethernet is a new approach in the automotive domain to connect ECUs in the vehicle, though, it is not expected to fully replace the CAN bus. CAN will continue to exist as a low-cost component, for example for connecting low-cost and computationally weak actuators and sensors with their corresponding ECUs or gateways, rather than be used as the main powertrain. As of today, the LIN-bus (Local Interconnect Network) is used for this type (low-cost, low-risk) of connection.

Although the CAN specification describes CAN as unencrypted by default, a sound solution for encryption and authentication is necessary to ensure a safe and secure distribution of critical new software over this public channel. In the automotive domain, there are not only software updates to consider, but hardware updates as well. If a workshop, for instance, replaces one of the brakes in a vehicle, they might also replace the corresponding ECU. In that scenario, how will the new cryptographic key (for message cryptography) be obtained? Common key distribution techniques like the Diffie-Hellman key exchange are difficult to implement, since many of the smaller network participants are low-cost and computationally weak ECUs. These ECUs often do not feature enough memory or CPU power to perform those cryptographic algorithms and methods. Cost is a limiting factor as well, when it comes to implementing expensive hardware into the vehicle. Automobile manufacturers prefer to spend more money on the salaries of programmers (fixed costs; used for entire fleet) rather than spending a cent more on a hardware part of a vehicle (variable costs; for each vehicle) because of the huge market scale. This means that hardware modules like TPMs (Trusted Platform Modules) are unattractive (cost, weight, space) as a key storing solution for each and every communicating part in the vehicle. Message cryptography on the CAN bus is not only hard to realize due to the strong network complexity, where key distribution is a difficult problem, but because an adversary in control of an ECU also gets access to the keys stored on that device.

III. Automotive Attack Scenarios

This section describes the motivation of our approach. This motivation is necessary to highlight the threats and dangers of automotive attack scenarios when considering cloud attack vectors. The claim of this section is to demonstrate what kinds of cloud attacks are possible and how they should be generally assessed. Section IV will describe how to assess them in more detail with SAM. A majority of remote attack vectors targeting automotive systems lead to accessing and tampering with the CAN bus, i.e., altering, sending or blocking CAN frames. Therefore it is necessary to improve the security of the remote access systems before a potential adversary even gets to the powertrain. OTA updates are most often pulled and received via the infotainment unit, which has access to a 4G, LTE or 5G broadband connection. From there, each and every ECU that needs to receive an update has to get the new firmware or software patch from the infotainment unit via the CAN bus. Rolling out sensitive data, especially new firmware or security patches in case of OTA updates over the CAN bus is incredibly critical and a major liability. OEM updates must be checked and validated before they can be deployed to the range of ECUs connected to the CAN bus. Faulty network configurations and the lack of authentication checks for OTA updates and patches can increase the risk of cloud and botnet attacks, e.g., Mirai [9].

All of this information needs to be documented in a system model that takes attack modeling for automotive software systems into account. The latest version of SAM [10] introduces new attributes for rating these kinds of attacks.

The following is a non-exhaustive list of cloud attack vectors that cause major threats to automotive software systems:

- Rolling out malicious (possibly unsigned) firmware to ECUs.
- Gaining remote control access to the vehicle using the OEMs cloud and mobile application's infrastructure.
- Infecting the system with ransomware.

The above attacks were chosen because they break the security goals integrity and authenticity. These security goals are especially important to make sure that the safety critical software of the vehicle stays untampered. Once the adversary has gained remote access to the vehicle she can start follow-up attacks as she already has access to the powertrain. The following is a list of automotive attack vectors regarding the CAN bus, assuming the adversary already has gained access via remote attack:

 ISBN: 978-1-61208-703-0

- Reverse engineering of CAN frames by filtering by arbitration IDs and identifying frames via tools like cansniffer or other can-utils [11].
- Injection of CAN frames from ECUs that were taken over after the remote attack (e.g., replay attacks, spamming attacks, etc.).
- Denial of Service (DoS) attacks, e.g., as shown by Palamanca et al [12].

Basically, cloud features and OTA updates have to be considered skeptical from the start. Even if the distribution source of the software is the OEM, attacks are still possible. A potential attacker might have found a way to distribute his malware over the OEM's infrastructure (e.g., their servers) and as a result a trust problem arises. It is fair to assume that any kind of roll-out (software updates, cloud data) is untrusted until the key distribution problem described earlier has been solved. Even if a solution for key distribution in heterogeneous CAN bus networks is developed, the number of remote attack vectors will rise harshly in comparison to the number of direct attack vectors. Hence, it is important to have a framework for modeling safe and secure automotive software systems with a system architecture model that takes even cloud attacks and remote attack vectors into account. The changes to the SAM meta model presented in this paper are a tangible solution for this kind of security analysis and security by design.

IV. Using Generic Scoring Systems for SAM

The current version of SAM introduces many new attributes to the modeling entities which allow for using well-known security scoring systems like CVSS [13]. In order to be able to keep SAM up-to-date and gain, some flexibility by not making a strong commitment to one particular system, we designed SAM to use any generic scoring system. When modeling attack scenarios, users of SAM can choose among their favorite. In this paper, we will use the CVSS. The latest version of SAM is available open source [10]. The architecture description has been completed to the extent that common scoring systems are now able to find the necessary information and thus perform their analyses. Inspired by the CVSS, which is an acclaimed industry standard for rating vulnerabilities in computer systems, we added new attributes to some of SAM's entities. The CVSS proposes three different metric groups for calculating the vulnerability scores. In the following, an explanation of the interplay between SAM and the metrics is given. The assignment of the attributes to the meta entities and partly their naming does not come from CVSS, but was developed by the athors.

The Base Metric Group reflects the intrinsic properties of Attack: from SAM's automotive-oriented perspective, this group therefore indicates the characteristics that result if the attack in question is aimed at the automotive domain in general. The entity AttackableProperty refers to the properties of the attacked item that are beyond the control of the attacker and must exist in order to exploit the vulnerability For example, in the case of a side channel attack, the use of shared caches within a multicore system. The attribute `conditionPrerequisiteComplexity` ("Low" and "High") in the AttackableProperty refers to the complexity of encountering or creating such conditions. For example, in the case of the side channel attack mentioned above, the `conditionPrerequisiteComplexity` is "Low" because shared caches are to be expected nowadays. It would be "High" if the attack made it necessary for all tasks on all cores to use one single common cache. When evaluating this property, all user interaction requirements for exploiting the vulnerability must be excluded (these conditions are recorded in the property `privilegesRequired` of Attack instead). If the `conditionPrerequisiteComplexity` is "Low", the attack is more dangerous than if the `conditionPrerequisiteComplexity` is "High". The property `privilegesRequired` describes the level of privileges an attacker must possess before successfully exploiting the vulnerability. This metric is greatest if no privileges are required. Also, the Attack entity has been extended with the attributes `accessRequired` and `userInteraction`. The attribute `accessRequired` describes the context in by which vulnerability exploitation is possible. Whether the user or driver of the vehicle needs to interact with the system in a certain way, e.g., by pressing a button, is captured in `userInteraction`. Attacks that do not require any user interaction increase the score of the attack. **The Temporal Metric Group** allows for adjustment of the score after more information of the exploited vulnerability is available. If, for example, `exploit code` has been published or the `report confidence` of a vulnerability is confirmed, the temporal score rises. In SAM, temporal metrics are part of the entity Vulnerability. **The Environmental Score Metrics** additionally enable the general CVSS Score (resulting from the Base Metric Group) to be adapted to the specific (automotive) company. The metrics are the modified equivalent of the base metrics weighting properties related to the concrete company's infrastructure and business risk. SAM offers a fully comprehensive basis to analyse the CVSS Base Metric Group, which means that SAM can also be used to evaluate the Environmental Metric Group. Environmental Metrics do not require any additional information beyond the Base Metrics, but merely a readjustment of the analysis perspective towards the concrete company. This means that the security scoring analysis can be carried out entirely by an analyst based on the available information provided by SAM.

The new changes to the SAM meta model (see Figure 2) allow the use of any security scoring or attack rating system, not only CVSS. This means that not all metrics and explanations of the CVSS have been transferred to SAM. This allows for more flexibility and SAM does not have to be adapted for any future CVSS updates. All attributes used for attack assessment are of the type String. This allows for SAM to be used with generic assessment techniques and is not tightly coupled with the CVSS attribute descriptions. In the model itself, or from the model itself, a CVSS score cannot be calculated automatically anyway. Doing so would happen in a behaviour model while SAM models are structure models. But if a security analyst is familiar with the CVSS, she will be able to calculate the CVSS score with all the information that is provided by the structure model. It is therefore still possible to find related information about the attribute types ("High" and "Low", etc.) in the notes of the meta model, but does not lead to problems in case of non-compliance.

V. Case Study on Cloud Attacks

SAM allows for a security analysis of cloud attacks. In the following we will show two examples: a remote attack

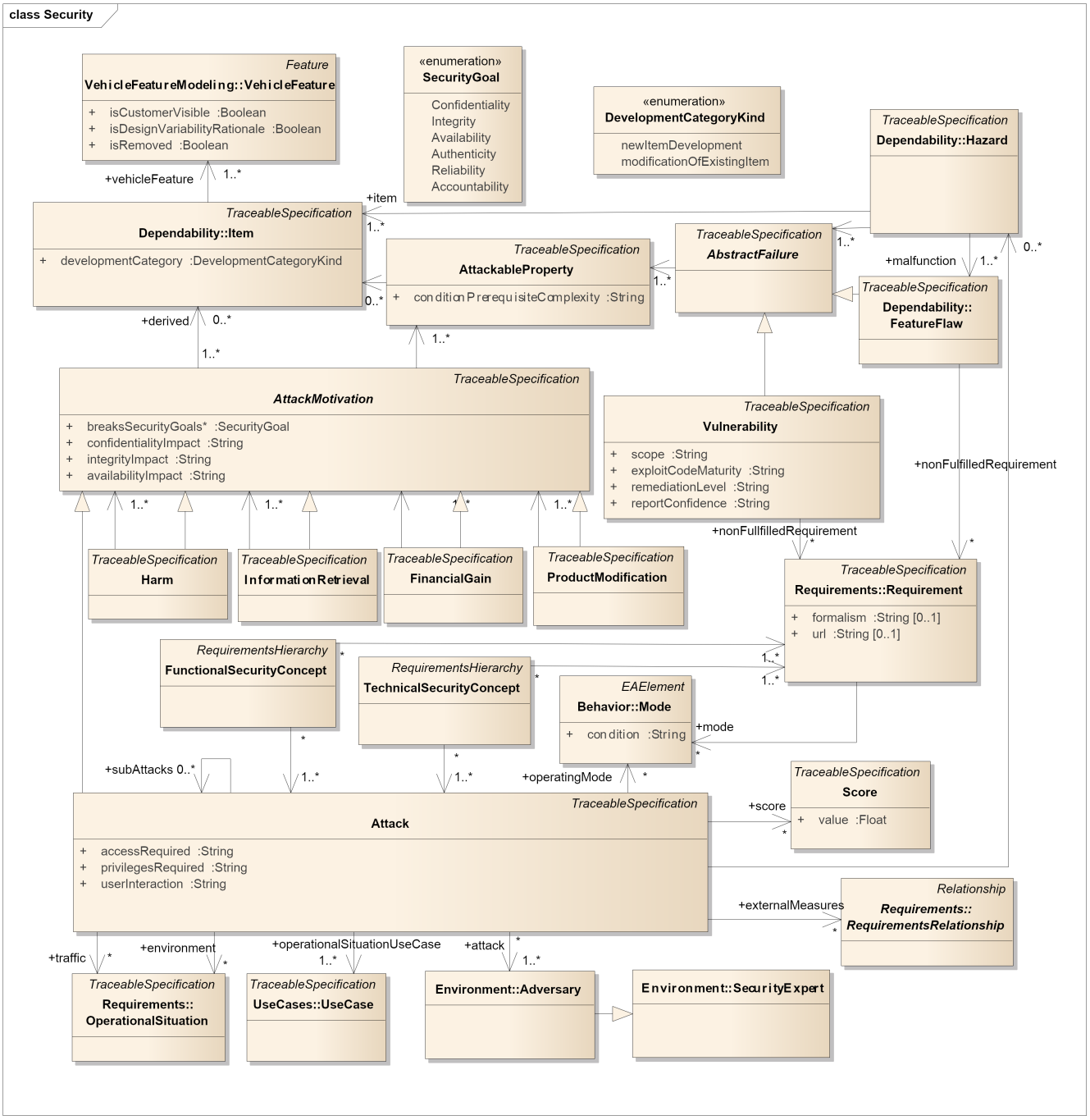

Figure 2. SAM Metamodel

to gain control of the vehicle (1) and an OTA update attack to install ransomware on the vehicle's infotainment unit (2). The attacks were chosen because they break the security goals authenticity and integrity. Cloud attack vectors often work only because of the absence of those security goals. The following examples are both in that category. (1) In the first part of the case study, we elaborate on a remote attack to gain control of a driving passenger vehicle as described by Miller and Valasek [7]. This kind of attack is one of the worst scenarios that can happen in theory and in practice, as an adversary does not need to have physical access to the target (`accessRequired` = N). Once the remote attack was successful, the adversary can perform numerous follow-up attacks as listed in Section III. Hence, the impact on the three security goals (confidentiality, integrity, availability) is *H*(igh). The adversary performing the attack is a **RemoteAttacker** and his attack motivation might be to *harm* the passengers or other road users (**CrashVehicle**). The exploited vulnerability is the **wrong D-Bus configuration**, specifically the **open D-Bus port** as the *AttackableProperty*. The vulnerability exists because of the *VehicleFeature* **BroadbandConnectivity** of the *Item* **InfotainmentUnit**. As the *Vulnerability* is already known because of the publication of Miller and Valasek and an official fix by the OEM is available, the tempo-

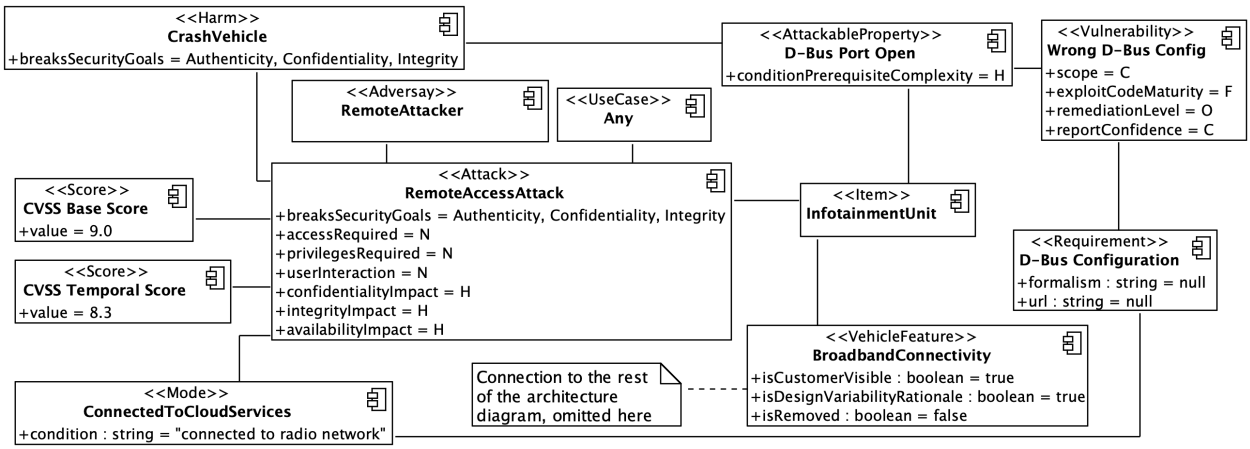

Figure 3. Exemplary architecture model for a cloud attack.
CVSS v3.0 Vector String: CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:C/C:H/I:H/A:H/E:F/RL:O/RC:C

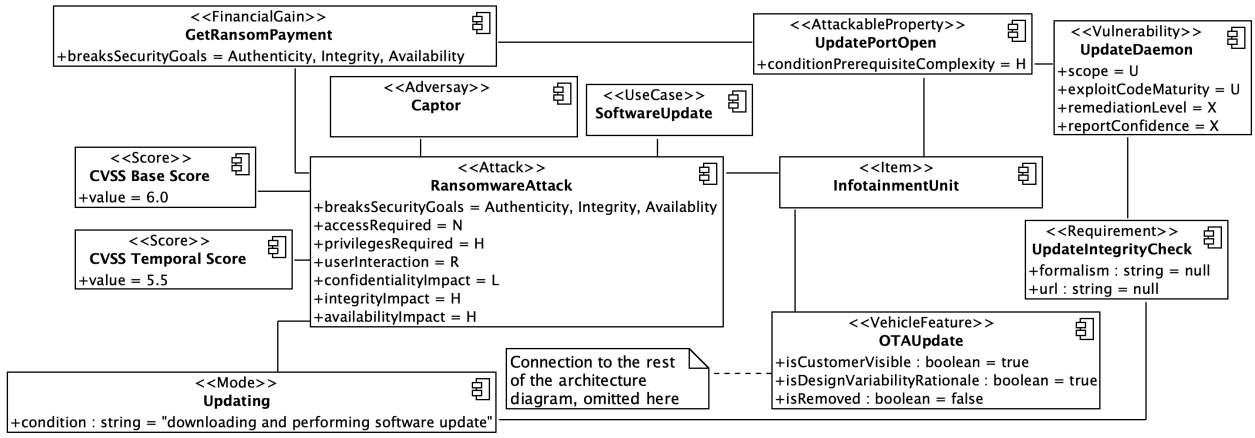

Figure 4. Exemplary architecture model for an OTA attack.
CVSS v3.0 Vector String: CVSS:3.0/AV:N/AC:H/PR:H/UI:R/S:U/C:L/I:H/A:H/E:U

ral metrics are *F*(unctional) for `exploitCodeMaturity`, *O*(fficial) Fix for `remediationLevel` and *C*(onfirmed) for `reportConfidence`. Hence, the remote attack scores a 9.0 as a **Base Score** and a 8.3 as a **Temporal Score**. This cloud attack is illustrated in Figure 3.

(2) The second part of the case study illustrates an OTA update attack using ransomware. A potential attack might compromise the OTA update interface to install an adversary's version of firmware. The ransomware would take control of the car by, e.g., by blocking or weakening the braking system until the user whose car has been infected pays a ransom to gain back full control of their vehicle. The attack motivation of such attack would be *financial gain* in the SAM context with the adversary demanding the ransom. For the update to be installed, however, the user is required to approve the update, e.g., by pressing a confirmation button on the infotainment unit. In contrast to the attack described above, this attack example is merely an unproven concept as no such attack or real scenario is known yet. However, it might be in the future and SAM is able to create a threat model for such an attack scenario. The exploit code maturity is unproven and there is no remediation level or report confidence defined. Hence, this OTA update attack scores a 6.0 as a **Base Score** and a 5.5 as a **Temporal Score**. This OTA update attack is illustrated in Figure 4.

VI. Related Work

SAM utilizes common concepts of the listed projects and related work. A non-trivial foundation includes the work of Holm [14], featuring a Cyber Security Modeling Language (CySeMoL) for enterprise architectures, Mouratidis [15] (Secure Tropos), papers, such as Ngyuyen [16], Juerjens [17], featuring UMLSec, which allows to express security-relevant information within the diagrams in a system specification, INCOSE work on integrating system engineering with system security engineering [18], NIST SP 800-160 [19] and other NIST work on cyber-physical systems [20]. SAM's unique characteristic and advantage over those existing approaches is that it is already integrated into an existing system model, (i.e. EAST-ADL [21]). SAM uses existing entities of the EAST-ADL system model (e.g., `Environment`, `Hazard`, `Item`, etc.) and is therefore tightly coupled with the system model. This enables a seamless integration of a security model into a system model that is extensively used in the automotive industry.

 ISBN: 978-1-61208-703-0

Some approaches deal with OTA updates in the way of hardening ECU firmware. Karamba [22] proposes a solution called "Autonomous Security" which focuses on embedding native security through static code analysis of the ECU firmware and locking it to factory settings. Multi-dimensional whitelisting might be an effective approach to vehicle cybersecurity. As manufacturers strive to limit post-deployment modifications, hardening the ECUs offers the added benefit of a more stable environment that is easier to secure over the life of the vehicle.

PRESERVE was an "EU-funded project running from 2011 to 2015 and contributed to the security and privacy of future vehicle-to-vehicle and vehicle-to-infrastructure communication systems. It provides security requirements of vehicle security architectures" [23]. The EVITA project tries to "design, verify and prototype an architecture for automotive on-board networks where security-relevant components are protected against tampering and sensitive data are protected against compromise. It focuses on V2X (vehicle to anything) communications and provides a base for secure deployment of electronic safety applications" [24].

VII. Conclusion and Future Work

In this paper, we have presented an investigation on cloud attacks in the automotive domain. Doing so, we give a glimpse on possible cloud attack vectors and attack scenarios. Moreover, we revamped the Security Abstraction Model for automotive software systems with the ability to model even more precise attack vectors and attack scenarios by enabling the use of any generic security scoring system like CVSS. We showed the feasibility of our approach by giving a case study on cloud attacks applying the new model. SAM offers robust tooling for modeling security for automotive software systems. Future work will concentrate on the bottom-up approach, i.e., improving embedded security and network security on the application layer. Next steps need to develop automotive software solutions to actually be included in the technical and functional security concept. Our research focuses particularly on a lightweight crypto approach for authentication and encryption in the vehicle network and embedded software, including a suitable keys distribution solution. Our work aims to support security by design in the automotive industry and SAM offers 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] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin, "Intra-Vehicle Networks: A Review," pp. 534–545, 2015.

[2] W. Zeng, M. A. Khalid, and S. Chowdhury, "In-vehicle networks outlook: Achievements and challenges," IEEE Communications Surveys and Tutorials, vol. 18, no. 3, 2016, pp. 1552–1571.

[3] ISO/IEC, "ISO/IEC 15408-1:2009 - Evaluation Criteria for IT Security," vol. 2009, 2009, p. 64.

[4] A. Happel and C. Ebert, "Security in vehicle networks of connected cars," 15. Internationales Stuttgarter Symposium: Automobil- und Motorentechnik, no. March, 2015, pp. 233–246.

[5] C. Valasek and C. Miller, "Adventures in Automotive Networks and Control Units," Technical White Paper, vol. 21, 2013, p. 99.

[6] C. Miller and C. Valasek, "A Survey of Remote Automotive Attack Surfaces," Defcon 22, 2014, pp. 1–90.

[7] ——, "Remote Exploitation of an Unaltered Passenger Vehicle," Defcon 23, vol. 2015, 2015, pp. 1–91.

[8] M. Zoppelt and R. Tavakoli Kolagari, "SAM: A Security Abstraction Model for Automotive Software Systems," in Security and Safety Interplay of Intelligent Software Systems. Springer, 2018, pp. 59–74.

[9] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," Computer, vol. 50, no. 7, 2017, pp. 80–84.

[10] SAM repository on Bitbucket [retrieved: April, 2019]. [Online]. Available: https://bitbucket.org/east-adl/sam

[11] can-utils repository on GitHub [retrieved: April, 2019]. [Online]. Available: https://github.com/linux-can/can-utils

[12] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, "A stealth, selective, link-layer denial-of-service attack against automotive networks," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), M. Polychronakis and M. Meier, Eds. Cham: Springer International Publishing, 2017, vol. 10327 LNCS, pp. 185–206.

[13] Common Vulnerability Scoring System [retrieved: April, 2019]. [Online]. Available: https://www.first.org/cvss/

[14] H. Holm, M. Ekstedt, T. Sommestad, and M. Korman, "A Manual for the Cyber Security Modeling Language," 2013, p. 110.

[15] H. Mouratidis and P. Giorgini, "Secure Tropos: a Security-Oriented Extension of the Tropos Methodology," International Journal of Software Engineering and Knowledge Engineering, vol. 17, no. 02, 2007, pp. 285–309.

[16] P. H. Nguyen, S. Ali, and T. Yue, "Model-based security engineering for cyber-physical systems: A systematic mapping study," pp. 116–135, 2017.

[17] J. Jürjens, "UMLsec: Extending UML for Secure Systems Development," in International Conference on The Unified Modeling Language. Springer, 2002, pp. 412–425.

[18] INCOSE, "Systems Engineering Handbook," in Systems Engineering, no. August, 2000.

[19] R. Ross, M. McEvilley, and J. Carrier Oren, "Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems," vol. 160, no. November 2016, 2016.

[20] J. Lee, B. Bagheri, and H.-a. Kao, "A Cyber-Physical Systems architecture for Industry 4 . 0-based manufacturing systems," Manufacturing Letters, vol. 3, 2015, pp. 18–23.

[21] H. Blom, H. Lönn, F. Hagl, Y. Papadopoulos, M. Reiser, C. Sjöstedt, D. Chen, and R. Tavakoli Kolagari, "EAST-ADL–An Architecture Description Language for Automotive Software-Intensive Systems–White Paper Version 2.1. 12," Hyperlink: http://www.maenad.eu/public/conceptpresentations/EAST-ADL_WhitePaper_M2 [retrieved: December 2018], vol. 1.

[22] D. Barzilai, "Autonomous Security [retrieved: January 2019]," 2018, pp. 1–14. [Online]. Available: https://www.karambasecurity.com/approach

[23] N. Bißmeyer, S. Mauthofer, J. Petit, M. Lange, M. Moser, D. Estor, M. Sall, M. Feiri, R. Moalla, M. Lagana, and F. Kargl, "PREparing SEcuRe VEhicle-to-X Communication Systems," 2014.

[24] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, and B. Weyl, "Security requirements for automotive on-board networks," in 2009 9th International Conference on Intelligent Transport Systems Telecommunications, ITST 2009. IEEE, 2009, pp. 641–646.

 ISBN: 978-1-61208-703-0