Social Engineering Exploits in Automotive Software Security: Modeling Human-targeted Attacks with SAM

Matthias Bergler

Computer Science, Technische Hochschule Nürnberg, Germany. E-mail: matthias.bergler@th-nuernberg.de

Juha-Pekka Tolvanen

MetaCase, Finland. E-mail: jpt@metacase.com

Markus Zoppelt

Computer Science, Friedrich Alexander Universität Erlangen, Germany. E-mail: markus.zoppelt@fau.de

Ramin Tavakoli Kolagari

Computer Science, Technische Hochschule Nürnberg, Germany.
E-mail: ramin.tavakolikolagari@th-nuernberg.de

Security cannot be implemented into a system retrospectively without considerable effort, so security must be taken into consideration already at the beginning of the system development. The engineering of automotive software is by no means an exception to this rule. For addressing automotive security, the AUTOSAR and EAST-ADL standards for domain-specific system and component modeling provide the central foundation as a start. The EAST-ADL extension SAM enables fully integrated security modeling for traditional feature-targeted attacks. Due to the COVID-19 pandemic, the number of cyber-attacks has increased tremendously and of these, about 98 percent are based on social engineering attacks. These social engineering attacks exploit vulnerabilities in human behaviors, rather than vulnerabilities in a system, to inflict damage. And these social engineering attacks also play a relevant but nonetheless regularly neglected role for automotive software. The contribution of this paper is a novel modeling concept for social engineering attacks and their criticality assessment integrated into a general automotive software security modeling approach. This makes it possible to relate social engineering exploits with feature-related attacks. To elevate the practical usage, we implemented an integration of this concept into the established, domain-specific modeling tool MetaEdit+. The tool support enables collaboration between stakeholders, calculates vulnerability scores, and enables the specification of security objectives and measures to eliminate vulnerabilities.

Keywords: automotive systems, social engineering attacks, design, model-based development, modeling, security.

1. Introduction

The importance of security grew as hacks on cars and other connected devices became widespread and reported in the general public. Unfortunately, these are not rare special cases in certain car models or their particular components: “Hackers can clone millions of Toyota, Hyundai, and Kia keys” Greenberg (2020), “A new wireless hack can unlock 100 million Volkswagens” Greenberg (2016b), “Helpless in Jeep Cherokee” Timberg (2015). Since modern cars are computers on wheels, security challenges can be found from numerous systems, such as from anti-theft systems, tire pressure monitoring systems, remote keys, Bluetooth, radios (3G, 4G, 5G) and telematics access functionality. Also, infotainment systems tend to provide access to 3rd party applications as well as internet access. Social engineering attacks are also becoming increasingly popular due to increasing digitalization. The advancing digital exchange has made it easier for attackers to gain access to confidential data and thus smuggle malicious software into the development process PurpleSec (2021). The headlines on successful attacks are not only embarrassing, but customer concerns and lawsuits on vulnerabilities push the automotive industry to change Timberg (2015). Developing and maintaining secure systems, however, is not easy. First, security cannot be an afterthought, i.e., a component that may be added to or removed from an existing system. Second, security goals and measures to cope with vulnerabilities cannot be isolated from the rest of the design and development work. Instead, security must be designed in already from the very beginning of the system design SAE (2016). Third, the actual development of secure and trustworthy

Proceedings of the 31st European Safety and Reliability Conference
Edited by Bruno Castanier, Marko Cepin, David Bigaud, and Christophe Berenguer
Published by Research Publishing, Singapore.
ISBN: 978-981-18-2016-8; doi:10.3850/978-981-18-2016-8_720-cd

systems requires effort. It takes time, requires expertise from multiple fields and stakeholders, and needs to be linked with other development processes and practices. These challenges call for apt security design practices supporting companies to smoothly integrate security in their development processes. We propose a language-based approach and present a modeling language, called Security Abstraction Model (SAM) and its new extension for social engineering attacks with tool support for specifying security aspects. Unlike other approaches Cheah et al. (2017); Macher et al. (2016); Pattaranantakul et al. (2018); Matulevičius (2017); Microsoft-Corporation (2005), SAM is fully integrated into a standardized architecture description language, in this case the automotive-specific systems modeling language EAST-ADL Blom et al. (2013). It not only focuses on attacks, vulnerabilities and motivations for attacks but relates them with relevant parts of the automotive system design. Because of this integration, SAM also enables to start security design early on — already when the first high-level features of the vehicle are defined. This way, security is not an isolated afterthought but an integral part of the system development. Together with the tool support, SAM enables collaboration of system engineers with security engineers, traceability with system features, requirements, hazards as well as calculating vulnerability scores. SAM also helps in specifying security goals and measures to solve attacks. We demonstrate these via a case study along with identified benefits on applying SAM. We start by introducing the relevance of security in automotive and its current status. Then we go into social engineering attacks in automotive and the resulting danger. Afterwards we present SAM and the changes for covering social engineering, followed by the detailed tool implementation for all of its parts. Section 6 demonstrates the tool use with a practical example before the lessons learned and directions for future research are discussed.

2. Security in Automotive

Automotive system development has always had to adapt to the latest state of research and economic factors. Therefore, modern vehicles became interconnected computer networks in which many electronic control units (ECUs) communicate with one another and with the environment (Vehicle-to-X communication). Car manufacturers were producing vehicles that feature advanced desktop-grade software components. These vehicles have advanced algorithms for assistance systems and other computer-dominant extensions that can provide entry points and powerful tools for malicious attackers. It is thus not surprising that the number of scientific publications on automotive security has increased drastically Amendola (2004); Hubaux et al. (2004); Wolf et al. (2004). Considering the fact that autonomous vehicles will continue rather than reverse the trend towards more communication interfaces for reasons of functionality, safety and comfort, collective research efforts in the field of vehicle security are reasonable; after all, human lives are at stake every time these "driving computers" are the target of attacks. Combining state-of-the-art software components with legacy interfaces and hardware infrastructure decisions results in a risky set up from an IT security perspective. Legacy mechanisms like insecure and unencrypted protocols (e.g., Controller Area Network (CAN)) were originally not designed in accordance with today's security principles. Secure automotive network architectures were not prioritized in the past due to the general prejudice of cars' security due to their technical complexity (security by obscurity). Sluggish development processes, lack of standard guidelines and low societal pressure lead to a rather slow transformation of automotive development processes taking the security-by-design principle systematically into consideration. Most existing countermeasures against cyber-attacks, e.g., the use of message cryptography for encrypting, authenticating or randomizing vehicle-level network messages focus on concrete attacks and do not consider the complexity of the access options offered by modern vehicles Zoppelt and Kolagari (2019). This is mainly due to a solution-oriented approach to security problems. Defining and enforcing security goals for the automotive system helps to improve overall security. In the presented work, we address the security goals integrity, authenticity, confidentiality, reliability, availability and accountability. Many attack vectors often affect multiple security goals at once. Some of the attack vectors known to cause major threats to automotive systems include:

- Gaining remote control access to the vehicle using the OEMs cloud and/or mobile application's infrastructure Nie et al. (2018); Lab (2019); Nie et al. (2017); Lab (2018).
- Getting SecurityAccess via Unified Diagnostic Services (UDS) Van den Herrewegen and Garcia (2018).
- Controlling the car via Onboard Diagnostic (OBD) injection Zhang et al. (2016).
- Remotely breaking into the telematics unit Foster et al. (2015).
- Infecting the system with ransomware Ring (2015).

According to the SAE J 3601 "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems" SAE (2016), security affects the entire development, production and operation process of automotive systems. This is described in explicit analogy to ISO 26262-1:2018 (2018) the functional

safety standard in the automotive sector, and results directly from the rule that security must be considered at the system design stage (”security by design”). With regard to concrete instructions in terms of the techniques to be used, the standard is fairly reserved, but in relation to model-based automotive system development, SAE (2016) refers in Appendices A-C to techniques for threat analysis and risk assessment, threat modeling and vulnerability analysis (e.g., attack trees) and explains when these should be used. The referenced techniques are relevant to the early stages of development in that they can be linked to requirements and design specifications by their illustrative (attack trees Cheah et al. (2017)), table-based Macher et al. (2016), use-case-based Pattaranantakul et al. (2018) and misuse case-based Matulevičius (2017) character. The development of threats and related information is typically performed by the STRIDE Threat modeling technique Microsoft-Corporation (2005) aiming to identify early possible security problems that may happen during the operation of a system. This approach is helpful even today, but what is true for this approach is equivalent to security modeling for enterprise systems: These approaches are not integrated into the design of the respective domain. Thus, it is not possible to identify the iterative cross-relationships between the designed system and security.

3. Social Engineering Attacks in Automotive

Thanks to the advanced digitization in communication, it is now possible to network and exchange ideas at any time. Especially during the COVID-19 pandemic, digitization is a key element in economy continuity. However, these digital communication systems are easily attackable and offer a weak point for engineering attacks, as we cannot identify our counterpart directly. Social engineering attacks in particular become increasingly popular with attackers, as they often lead to success despite the comparatively greater effort involved. This is because they are specially designed to exploit human weaknesses in behavior in order to obtain unauthorized access or sensitive data Mitnick and Simon (2003). The greater effort with some attacks depends on the fact that they have to be specially adapted to individual people, such as an employee of a certain company. Based on the technical report by the PurpleSec Association, the number of all kind of cyber-attacks has increased sixfold since the beginning of the pandemic and 98 percent of the cases are social engineering attacks PurpleSec (2021).
There are very different types of such attacks. These can be human-based or computer-based Xiangyu et al. (2017). Human-based means that there is direct interaction with a person, e.g. fake service calls or the well-known grandchildren’s trick. Computer-based attacks are carried out using computer or email programs, e.g. phishing attacks. Even if the types of attack differ from one another, social engineering attacks always have the same pattern Mouton et al. (2016):

(i) Collect information about the target.
(ii) Develop relationship with the target.
(iii) Exploit the available information and execute the attack.
(iv) Exit with no traces.

In the automotive sector too, social engineering attacks may lead to success and create vulnerabilities for further attacks. These weaknesses have to be taken into account when developing the vehicle components. Although a vehicle cannot become a direct victim of a social engineering attack, successfully carried out attacks on employees in vehicle development or even a private person can lead to an attacker gaining access to individual vehicle parts or the entire vehicle. In many cases, these attacks can only be discovered but not prevented, e.g. in the case of a quid pro quo attack, in which a victim provides information or access in exchange for other services.
In addition to social engineering attacks on employees at a vehicle parts manufacturer in order to gain access to systems, private car owners are a popular target. There may be different motivations for the attack. Either an attempt can be made to take control of the vehicle himself by using social engineering attacks to trick the owner into installing malicious software or hardware in the vehicle, as described in Costantino et al. (2018), or valuable information about the driver can be obtained using vehicle data for further social engineering attacks on targets related to the victim, e.g. the victim’s employer. An attack could look like this:
First of all, the vehicle type of the victim is spied out. The attacker then contacts the victim with the identity of a service employee and tries to find out more about the vehicle, the infotainment system and its usage behavior in a service conversation. He then offers the victim a free security update via CD, USB stick or mobile phone app, which the victim can download from a fake website or receive by post. This update actually installs malicious software that enables the attacker to read the victim’s mobile phone data or to access the microphone of the hands-free system in order to record conversations and send them to the attacker via the mobile data connection of the mobile phone or vehicle. Alternatively, software can also be installed that gives the attacker control of the vehicle as in Greenberg (2016a). In order to prevent such attacks, possible points of attack via social engineering attacks must be identified and taken into account during the development of the

components. With the latest expansion of SAM it is now also possible to map social engineering attacks and develop a counter strategy.

4. SAM: Automotive Domain Specific Security Modeling Approach

SAM is a modeling language for representing security-relevant properties of automotive software systems. It enables a security analysis of automotive attack vectors and conducts a thorough threat analysis. By means of systematic security analyses the effort for a potential attack can be quantified and appropriate counter measures can be modeled. The approach closely links security management and model-based system engineering by an abstract description of the principles of automotive security modeling. The resulting specification of SAM is based on security requirements that have been extracted from common industrial scenarios Zoppelt and Kolagari (2018). It aims to be a solution for representing attack vectors on vehicles and provide a thorough security modeling for the automotive industry.

4.1. *Feature-Targeted Attacks*

SAM has a close link to the architecture description via 'Item' entity (as in ISO 26262-1:2018 (2018) and Blom et al. (2013)) from the architecture model, playing the role of a 'Feature' from traditional security approaches. SAM aims to express all the important criteria of the attack vectors from the adversary's motivation up to the security breach—to enable modeling of the system in early software development phases. In addition to attack motivations, SAM also describes all intrinsic and temporal characteristics of an attack, e.g., effects on the security objectives (confidentiality, availability, integrity, etc.), the complexity of the attack, the affected object and the vulnerability. The reason why we extended EAST-ADL rather than other languages like SysML or AADL is that EAST-ADL already addresses relevant aspects of automotive systems, namely its product line nature by specifying explicitly features that are either visible to customers (e.g., lane detection or regenerative braking) or on technical level (e.g., power generation control). EAST-ADL also directly addresses functional safety and ISO26262 with its Dependability Model. SAM identifies the same Items, Requirements and Hazards from architecture and dependability modeling and related them to Attacks and Security Concepts. Although SAM is developed as part of the EAST-ADL, it is not necessarily bound to EAST-ADL. SAM models are meant to be used and applied by anyone wanting to conduct a threat analysis of attacks in the automotive domain. SAM models have no aspiration to completeness with the rest of the systems model and can be used even in the very first phases of the system engineering process. Though it is advised to comply with the rest of the EAST-ADL systems model, SAM models can be used standalone.

4.2. *Human-Targeted Attacks*

In order to enable the representation of social engineering attacks in SAM, we had to adapt the original metamodel (Zoppelt and Kolagari (2018)) to provide the necessary classes[a]. For this purpose, the new abstract class 'Target' was introduced. This class generalizes the Item class already known in SAM, which represents the connection to the EAST-ADL, and the new HumanActor class, which is required for social engineering, as this can only be carried out on the basis of a person. The new class HumanActor has two attributes of the String type, which represent the exploitable human properties of curiosity and helpfulness. The use of these properties is either not defined (X), none (N), i.e. not used, low (L), used slightly, or high (H), used to a high degree. In addition, an association of the Resilience class refers to the HumanActor class. This class represents the mental resistance to social engineering attacks and possesses the attributes cautiousness, contentment, courage, experience and knowledge. They all correspond to the ResilienceLevel type, with which the extent of the property, required to defend against the attack, is measured. The different levels are low (L), low intermediate (LI), intermediate (I), high intermediate (HI), high (H) and not defined (X). The higher the demands on the resilience of a victim, the more successfully a social engineering attack can be carried out and the more adequate security concepts must be designed to counteract it. In section 6 an example of a social engineering attack with SAM is demonstrated. A security assessment through social engineering attacks can be carried out independently of the assessment of an Item, but it should definitely be considered. With the extension, SAM remains backwards compatible, as the use of social engineering attacks is optional in the application. The Common Vulnerability Scoring System (CVSS) is not relevant for the assessment, as the values in the Resilience class show an assessment of the severity of the attack but a extended scoring system is under development.

5. Language Implementation

While it is possible to create tooling for SAM from scratch, we applied MetaEdit+ MetaCase (2018b), a commercial language workbench, providing most of the needed functionality automatically: Only parts specific to SAM, its modeling concepts, rules, notation and integration

[a] https://www.in.th-nuernberg.de/professors/BerglerMa/SAM

with other tools, needed to be defined. This not only speeds up the implementation, but also enables easy evolution when the modeling needs change. MetaEdit+ provided also implementation of EAST-ADL MetaCase (2019) and other relevant functionality needed for automotive system development such as collaborative modeling, version control, integration with relevant tools applied in automotive (e.g., programming environments, Simulink, requirements management etc.) as well as having availability of supporting services. The only parts that deserved attention related to tooling were those parts not addressed by most metamodels, such as showing elements of SAM in the user interface (toolbar and browsers) based on their relevance, or decide how to indicate if constraints are not followed. In current definition, constraints that are considered mandatory are checked and reported at modeling time. Those constraints, not sensible to check like minimum cardinalities in the metamodel, are shown as recommendations in the live check pane at the bottom of the diagram. This way language users get immediate guidance to create security models. The original definition of SAM, similarly to the definition of EAST-ADL, focused on language concepts and on defining the exchange format via a metamodel. The definition of the whole language also needed to cover concrete syntax, all constraints, language usability topics as well as integration with other tools. These are detailed in the following subsections.

5.1. *Implementation of SAM concepts*

The implementation started by integrating SAM in the existing metamodel of EAST-ADL. First, the same conventions as applied in EAST-ADL and AUTOSAR were followed: Security models followed the same naming policies with short and optional longer names and all model elements had a globally unique identifier (UUID). Second, SAM was defined to follow the same package structure as EAST-ADL uses to organize specifications. Third, concepts of SAM were integrated with already existing EAST-ADL concepts, like Item from Dependability and ISO26262, VehicleFeature from variation models addressing product lines and Requirements from specifying and tracing with system requirements. Since MetaEdit+ allows to specify metamodels graphically (MetaCase (2018a)) similarly to UML, SysML, EAST-ADL or AUTOSAR as well as SAM we applied the form-based metamodeling tools of MetaEdit+. These allow the integration of all other related language constraints, notations as well as model checking and generators, which otherwise would be specified seperately, if at all, in addition to the metamodel. This tight integration of the whole language definition improves the quality of the language greatly. The typical problems from languages defined in an unintegrated manner, like inconsistencies and low quality Wilke and Demuth (2011); Bauerdick et al. (2004), can be more easily avoided. Figure 1 shows the concepts of SAM defined in MetaEdit+. The list of Objects shows the key modeling elements of SAM, the list of Relationships the connections between these elements, and the list of Roles how an object participates in the relationships, such as be directed or undirected, having constraints or detailed properties. To minimize the modeling effort the implementation defines one AttackMotivation and its concrete subtype is selected from a property. This way the type of AttackMotivation (Harm, Financial Gain etc.) can be changed without deleting the old one and creating and re-connecting a new one. This definition, compared to having a language construct for each subtype, is possible because all subtypes have the same properties and constraints. Also, for the reference from Attack to OperationalSituation, the role AttackSituation has a property to select if based on Traffic or Environment. Each element of SAM shown in Figure 1 are defined with further details. Figure 2 shows one such definition: The Vulnerability and its nine properties. The first three are obtained from EAST-ADL and AUTOSAR metamodel and the remaining from SAM. These properties have rules and constraints, such as 'Short name' being mandatory and starting with an alphabetical character followed by possible characters, numbers or underscores (defined as regular expression: `[a-zA-Z](_?[a-zA-Z0-9])*_?`. Another constraint of SAM is that Scope of Vulnerability may have only two possible values (unchanged or changed). To support usability, we slightly changed the ordering of the properties as set in the original metamodel to follow the order in which security engineers are expected to fill them and follow the same order as the vulnerability analysis tool Common Vulnerability Scoring System CVSS FIRST.Org (2019). The use

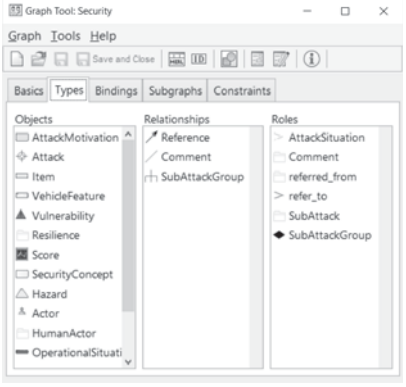

Fig. 1. Defined language concepts of SAM.

of Vulnerability concept is illustrated in Figure 3. The definition of the Vulnerability modeling concept was finalized by providing a description of this SAM concept (see Figure 2 the bottom of the window). Constraints of SAM are ensured in two different ways: either as part of the metamodel or applied via model checker. Examples of the former are rules on legal connections and uniqueness of element names. These constraints can be checked and ensured at modeling time, i.e., security models always follow them. As an example Vulnerability can refer to four model elements only (HumanActor, Item, Requirement and Score).

5.2. *Integration with Score Calculation*

Models made with SAM can be applied to analyze and calculate vulnerability scores. First we implemented a generator exporting data from models to CVSS to an online tool. CVSS provides a way to produce a numerical score reflecting the severity of vulnerability. The resulting numerical score can then be translated into a qualitative representation (such as low, medium, high, and critical) to help organizations properly assess and prioritize their vulnerability management processes. In the same way integration with other analysis tools can be done, or if they provide other mechanisms, like programmable APIs, the modeling tool can apply them. Depending on the capabilities of the analysis system, the results can then be included back to the model. For example, MetaEdit+ can show the score directly in the Vulnerability model element. This information would then be also available when tracing from security properties to requirements, features and system design in general. Since integration from the external web-based calculator was not possible, and even if available would lead to slower modeling and score calculation process, also requiring an online connection, we implemented the CVSS calculator into the SAM modeling tool. This was done using the same generator system applied to produce vulnerability vectors to the existing CVSS calculator. The benefit of this latter approach is showing vulnerability scores immediately. In other words, scores are calculated real-time during modeling. We applied the existing Score element to show the results and followed also the color schemes of CVSS for the notation of Score. Figure 3 shows running the calculation at modeling time and displaying the results directly in the model: Base metric of CVSS in case of Service Call Scam Attack is 8.8 (high) and CVSS for temporal score is 8.1 (high).

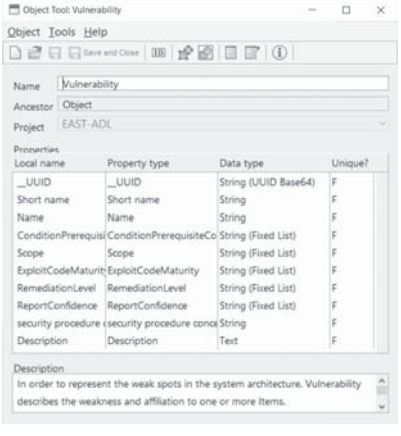

Fig. 2. Definition of Vulnerability.

6. Case Study: Service Call Scam

In this example, we show a social engineering attack with the aim of getting the vehicle owner to update the infotainment system with malicious software. This software is supposed to access the function of the hands-free system via a back door and record phone calls or conversations made in the vehicle and forward them to the attacker via the internet connection of the smart phone as seen in Costantino et al. (2018). For this purpose, as in Figure 3, a service call from the car manufacturer is faked and the victim is provided with a software update for the infotainment system. Since the identity of the service employee cannot be confirmed over a telephone call, the victim must have high values in cautiousness, experience and knowledge. What makes preventing such an attack on the part of vehicle manufacturers difficult and dangerous. Much more defective software could also be installed, which would result in a loss of control over the vehicle as seen in Greenberg (2016a). Since CVSS is also implemented into the modeling tool, the same metrics are shown inside the metric element connected to individual Vulnerabilities (both base and temporal metric values are 'high'). This way the resulting metric scores are not only seen at modeling time but also versioned, reported and documented along with the rest of the system design. Based on the specified security model, we can analyze the attack properties and try to derive SecurityConcept with requirements that must be satisfied to fix the vulnerabilities. In this case the requirements are different choices for update handling.

7. Conclusion

Security must be designed in at an early stage, be part of the rest of system designs, and be easily integrated with existing system development practices. We presented the extension of SAM for social engineering attacks and addressed the danger of social engineering attacks to create vulnerabilities. We also introduced a tool support

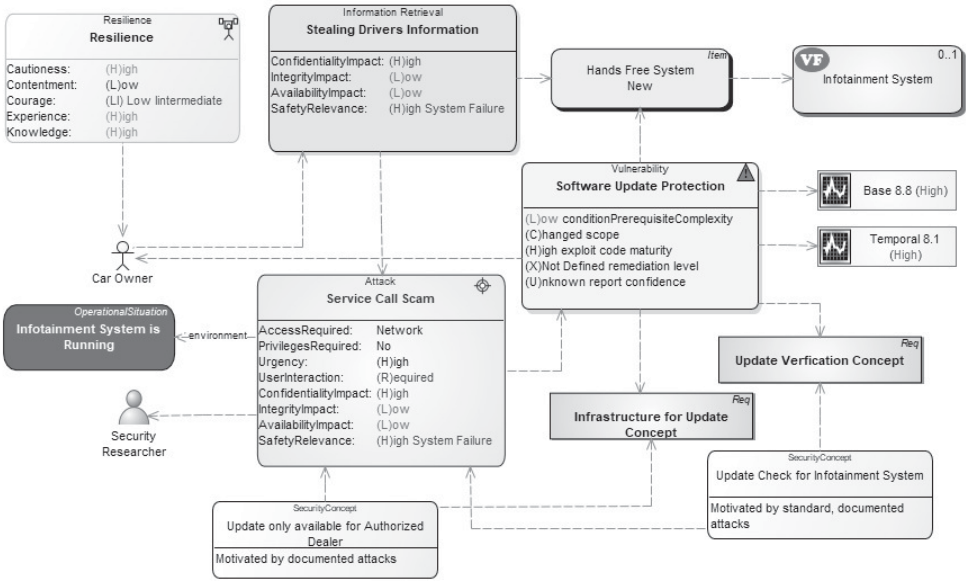

Fig. 3. SAM model of Service Call Scam.

for developing secure automotive systems. The unique part of the proposed approach is its integration in existing architecture modeling languages applied in the automotive industry. In our case we demonstrated the implementation with EAST-ADL tooling. The developed modeling support provides several benefits for development teams:

- Security issues can be considered in relation to the system design—starting when the first customer visible features are identified.
- The proposed security language and tool support, guides developers by considering relevant aspects such as attacks, their motivation, vulnerabilities and relation to vehicle features.
- The security models are related with the rest of the system designs, with traceability.
- Vulnerability scores can be calculated immediately and be part of security models.
- Support communication and collaboration within a team.

We also presented the language implementation process covering the metamodel with rules, visual notation and integration with model analyzer. Because the investment on implementing tool support is modest, it pays off quickly as all the other developers can then model with the language, link with existing designs and produce vulnerability metrics on the identified attacks. This also indicates that the effort in implementing tool support for SAM with other modeling languages is modest—at least with current tooling and tools having access to the full metamodel. As both the modeling language and generators are fully open for modifications, the presented approach also gives full control for the company enabling future updates, like targeting other vulnerability analysis tools. Decision makers such as managers who allocate resources to security and engineers to implement countermeasures, or people without a security background who are not interested in all the details of an attack, may be overwhelmed by the level of detail of SAM, so due to collaborative nature of design work, the main users are both security engineers and system developers. At the same time, however, it is essential, especially for decision-makers, to understand and correctly assess the threat posed by attacks. This is why an additional view is needed that does not show all the details that a fully blown SAM model offers but can provide a quick overview of the mechanisms of operation and the severity of an attack. This provides a reliable initial basis for deciding on the allocation of resources for the creation of countermeasures. Our future work also focuses on process: Provide a methodology with a step-by-step refinement of the available information, whereby the refinement results on the one hand in the security expert gaining knowledge about details of the attack (over time) and on the other hand in extending the CVSS to consider resilience for social engineering attacks.

References

Amendola, S. (2004). Improving automotive security by evaluation—from security health check to common criteria. *White paper, Security Research & Consulting GmbH 176*.

Bauerdick, H., M. Gogolla, and F. Gutsche (2004). Detecting ocl traps in the uml 2.0 superstructure: An experience report. In *International Conference on the Unified Modeling Language*, pp. 188–196. Springer.

Blom, H., H. Lönn, F. Hagl, Y. Papadopoulos, M.-O. Reiser, C.-J. Sjöstedt, D.-J. Chen, F. Tagliabo, S. Torchiaro, S. Tucci, et al. (2013). East-adl: An architecture description language for automotive software-intensive systems. In *Embedded Computing Systems: Applications, Optimization, and Advanced Design*, pp. 456–470. IGI Global.

Cheah, M., H. N. Nguyen, J. Bryans, and S. A. Shaikh (2017). Formalising systematic security evaluations using attack trees for automotive applications. In *IFIP International Conference on Information Security Theory and Practice*, pp. 113–129. Springer.

Costantino, G., A. La Marra, F. Martinelli, and I. Matteucci (2018). Candy: A social engineering attack to leak information from infotainment system. In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pp. 1–5.

FIRST.Org, I. (2019). First, common vulnerability scoring system, version 3.1.

Foster, I., A. Prudhomme, K. Koscher, and S. Savage (2015). Fast and vulnerable: A story of telematic failures. In *9th {USENIX} Workshop on Offensive Technologies ({WOOT} 15)*.

Greenberg, A. (2016a, January). *The Jeep Hackers Are Back to Prove Car Hacking Can Get Much Worse*. Wired.

Greenberg, A. (2016b, March). *Radio Attack Lets Hackers Steal 24 Different Car Models*. Wired.

Greenberg, A. (2020). *Hackers can clone millions of Toyota, Hyundai, and Kia keys*. Wired.

Hubaux, J.-P., S. Capkun, and J. Luo (2004). The security and privacy of smart vehicles. *IEEE Security & Privacy 2*(3), 49–55.

ISO 26262-1:2018 (2018, December). Road vehicles — Functional safety. Standard, International Organization for Standardization, Geneva, CH.

Lab, T. K. S. (2018). Experimental security assessment of bmw cars: A summary report.

Lab, T. K. S. (2019). Experimental security research of tesla autopilot.

Macher, G., E. Armengaud, E. Brenner, and C. Kreiner (2016). A review of threat analysis and risk assessment methods in the automotive context. In *International Conference on Computer Safety, Reliability, and Security*, pp. 130–141. Springer.

Matulevičius, R. (2017). Security risk-oriented misuse cases. In *Fundamentals of Secure System Modelling*, pp. 93–105. Springer.

MetaCase (2018a). The graphical metamodeling example.

MetaCase (2018b). Metaedit+ user's guide.

MetaCase (2019). East-adl tutorial.

Microsoft-Corporation (2005). The stride thread model.

Mitnick, K. D. and W. L. Simon (2003). *The art of deception: Controlling the human element of security*. John Wiley & Sons.

Mouton, F., L. Leenen, and H. Venter (2016). Social engineering attack examples, templates and scenarios. *Computers & Security 59*, 186–209.

Nie, S., L. Liu, and Y. Du (2017). Free-fall: Hacking tesla from wireless to can bus. *Briefing, Black Hat USA 25*, 1–16.

Nie, S., L. Liu, Y. Du, and W. Zhang (2018). Over-the-air: How we remotely compromised the gateway, bcm, and autopilot ecus of tesla cars. *Briefing, Black Hat USA*.

Pattaranantakul, M., R. He, Q. Song, Z. Zhang, and A. Meddahi (2018). Nfv security survey: From use case driven threat analysis to state-of-the-art countermeasures. *IEEE Communications Surveys & Tutorials 20*(4), 3330–3368.

PurpleSec (2021). 2021 cyber security statistics the ultimate list of stats, data & trends.

Ring, T. (2015). Connected cars - the next target for hackers. *NetworkSecurity 11*, 11–16.

SAE, S. (2016). j3061, cybersecurity guidebook for cyber-physical vehicle systems. *Nr 1*, 52.

Timberg, C. (2015, July). *Hacks on the Highway*. Washington Post.

Van den Herrewegen, J. and F. D. Garcia (2018). Beneath the bonnet: A breakdown of diagnostic security. In *European Symposium on Research in Computer Security*, pp. 305–324. Springer.

Wilke, C. and B. Demuth (2011). Uml is still inconsistent! how to improve ocl constraints in the uml 2.3 superstructure. *Electronic Communications of the EASST 44*.

Wolf, M., A. Weimerskirch, and C. Paar (2004). Security in automotive bus systems. In *Workshop on Embedded Security in Cars*, pp. 1–13. Citeseer.

Xiangyu, L., L. Qiuyang, and S. Chandel (2017). Social engineering and insider threats. In *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pp. 25–34.

Zhang, Y., B. Ge, X. Li, B. Shi, and B. Li (2016). Controlling a car through obd injection. In *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 26–29. IEEE.

Zoppelt, M. and R. T. Kolagari (2018). Sam: a security abstraction model for automotive software systems. In *Security and Safety Interplay of Intelligent Software Systems*, pp. 59–74. Springer.

Zoppelt, M. and R. T. Kolagari (2019). What today's serious cyber attacks on cars tell us: consequences for automotive security and dependability. In *International Symposium on Model-Based Safety and Assessment*, pp. 270–285. Springer.