A Systematic Approach to Automotive Security

Masoud Ebrahimi[1], Stefan Marksteiner[2,4], Dejan Ničković[3], Roderick Bloem[1],
David Schögler[2], Philipp Eisner[2], Samuel Sprung[2], Thomas Schober[2],
Sebastian Chlup[3], Christoph Schmittner[3], and Sandra König[3]

[1] Graz University of Technology, Graz, Austria
ebrahimi@tugraz.at
[2] AVL List GmbH, Graz, Austria
stefan.marksteiner@avl.com
[3] Austrian Institute of Technology, Vienna, Austria
dejan.nickovic@ait.ac.at
[4] Mälardalen University, Västerås, Sweden

Abstract. We propose a holistic methodology for designing automotive systems that consider security a central concern at every design stage. During the concept design, we model the system architecture and define the security attributes of its components. We perform threat analysis on the system model to identify structural security issues. From that analysis, we derive attack trees that define recipes describing steps to successfully attack the system's assets and propose threat prevention measures. The attack tree allows us to derive a verification and validation (V&V) plan, which prioritizes the testing effort. In particular, we advocate using learning for testing approaches for the black-box components. It consists of inferring a finite state model of the black-box component from its execution traces. This model can then be used to generate new relevant tests, model check it against requirements, and compare two different implementations of the same protocol. We illustrate the methodology with an automotive infotainment system example. Using the advocated approach, we could also document unexpected and potentially critical behavior in our example systems.

Keywords: Cybersecurity, Testing, Automotive, Threats

1 Introduction

The advent of *connected, cooperative automated mobility* provides a huge opportunity to increase mobility efficiency and road safety. However, the resulting connectivity creates new attack surfaces that affect the vehicle's safety, security, and integrity. With an estimated 100 million lines of embedded code, modern vehicles are highly complex systems that need to provide consistent cyber-security assurances. Indeed, there are an alarming spike in cyber-attacks targeting connected cars, their electronic control units (ECUs), and the original equipment manufacturer (OEM) back-end servers.

Therefore, making the right security decisions from the early design stages is crucial. The ad-hoc security measures done by domain experts are insufficient to

meet the requirements in the automotive domain. The standard ISO/SAE 21434 and the mandatory regulation UN R155 advocate for more systematic reasoning about system security. The United Nations Economic Commission for Europe (UNECE) has adopted new security regulations, such as UNECE R155 and R156, for the homologation of future vehicles that address the identified cyber-attack risks, for example, during software updates. Similarly, the cyber security standard ISO/SAE 21434, introduced in 2021, defines precise security requirements for vehicles during the entire product life cycle, from its development to its operation and maintenance. Hence, there is an urgent need for methods and tools that address multiple security-related aspects, from early vehicle design to deployment and operation phases.

This paper proposes a top-down methodology for systematically assessing automotive security at different stages of vehicle development. The proposed methodology follows the product cycle in several steps. During the early design phase, we use threat modeling, analysis, and repair to provide more systematic support for the concept design of secure (automotive) systems. These methods allow us to identify the system's weaknesses in security threats and develop structural measures to prevent and mitigate them. We then use the threat analysis results to capture the system's critical components concerning security properties and derive a verification and validation (V&V) plan. We apply established processes (fuzz testing, penetration testing, etc.) for testing the implemented system components. However, the source code of the component implementation is often unavailable to the V&V team, and they cannot efficiently use the classical testing methods and tools. In that case, we advocate using automata learning for testing that builds an explainable model of a black-box implementation of a component from a set of executed test cases that facilitates testing and other V&V activities. This methodology is a result of a joint research effort amongst the industrial and academic partners in TRUSTED[5], a project focusing on trust and security in autonomous vehicles. In implementing our proposed methodology, we were also supported by partners from the related LearnTwins[6] project, which focuses on learning-based testing methods for digital twins.

2   TRUSTED Methodology

The TRUSTED methodology starts with the concept design with a *threat model* of the vehicle; see Stage ① in Figure 1. The threat model consists of two components: (i) a system model architecture and (ii) a threat database. The system model architecture provides a structural view of the vehicle. This view includes vehicle components and subsystems (e.g., sensors, actuators, ECUs) and describes their (wireless or wired) interconnections. We can assign security attributes (e.g., authentication, encryption) to system components and communication links. A system model can define security boundaries that enclose trusted subsystems and assets we need to protect from potential attacks. The

---

[5] `https://TRUSTED.iaik.tugraz.at/`

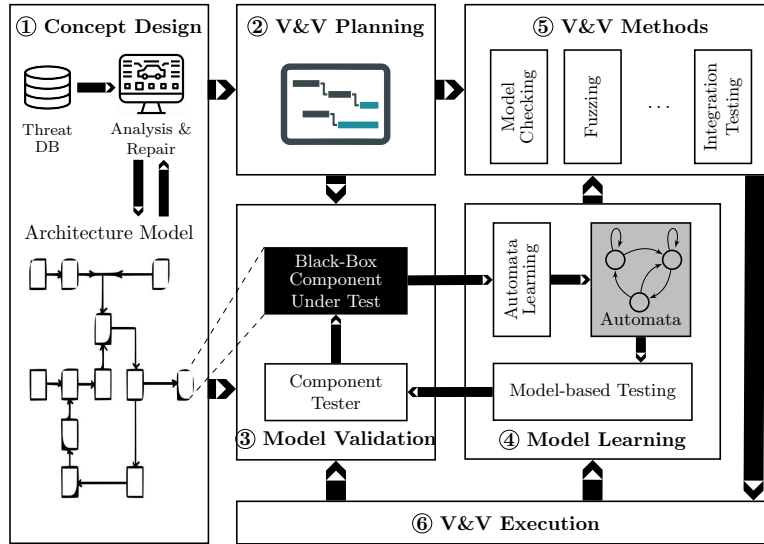[6] `https://learntwins.ist.tugraz.at/`

Fig. 1: Overview of the TRUSTED methodology

threat database contains a set of known threats—these threats from public domain sources, relevant standards, and previous experience. The threat model is an input to a threat analysis method allowing the detection of structural weaknesses in the system's architecture. We then combine the threat analysis with the repair activities to identify prevention and mitigation actions required to protect the system from identified threats.

The high-level threat analysis performed in the early stages of the design provides essential insights into the security-related weaknesses in the system architecture. We can take structural defense actions to improve the system's security based on threat repair outcomes (e.g., implementing authentication in a specific component). Yet, there is no guarantee that an attacker cannot break the resulting measures. Hence, it is imperative to have a solid verification and validation (V&V) plan. In the TRUSTED methodology, we use the insights gained by threat analysis and repair to identify risks and prepare an effective V&V plan corresponding to ② in Figure 1.

We use the system architecture model developed during the concept design phase to implement and integrate the components of the system. The implementation step is outside the scope of the TRUSTED methodology, but we assume the components are available as black boxes (see ③ in Figure 1). That is, we assume that we can execute components, but we cannot access their implementations.

During the development and integration of different components from the system architecture, verifying and testing safety and security functionalities becomes another critical aspect that we must address. Model validation (③ in Figure 1) tests the model for conformance against the component under test. This step provides either affirmation for the correctness (or completeness, re-

spectively) of the model or counterexamples to refine the latter in a loop until the model is considered good enough to be used for test case generation.

We propose a learning-for-testing approach using automata learning (④ in Figure 1) as the core method for generating tests during V&V. In automata learning (see Section 4.1), we construct a Finite State Machine (FSM) of the System Under Test (SUT). We use the inferred FSM to: (1) obtain potential attack data, and (2) identify critical inputs that might show differences between the FSM and the SUT. We must automatically perform the necessary tests during the development and especially the maintenance phase to guarantee a quick response in the event of a threat.

We chose the learning-based testing approach due to its versatility and numerous V&V activities that we can undertake with the inferred FSM (⑤ in Figure 1). We can use the inferred FSM to: (1) visualize and understand the implementation, (2) model check it against its formalized requirements (possibly generating test cases on specification violations), (3) generate additional test cases by fuzz testing, and (4) Test for equivalence between implementation and a reference model or another implementation.

In the last phase (⑥ in Figure 1), we use various V&V strategies to verify the specified properties against the actual component under test. The test results are final verification outcomes; meanwhile, we can use them as counterexamples for the learning algorithms in ④ in Figure 1. This policy provides a feedback loop for refining the model in the learning-based testing approach. We execute and store tests using an automated test execution platform that augments generic test cases with additional information. This additional information comes from a test database or is provided in a grey box testing [12].

The threat model and the tests created during various design phases must be continuously maintained and updated throughout the vehicle lifecycle. We must incorporate new unknown threats and vulnerabilities into the model and re-evaluate the model to find new security issues. We must also integrate the changes to functions resulting from software updates into the system model and their impact on the vehicle's security analyzed and re-tested. This closely corresponds with the notions on testing in ISO 21434 and UNECE R155.

3   Automotive Security by Design

In this section, we demonstrate the use of THREATGET [17], a tool for threat modeling and analysis to improve the security of automotive applications during their early stages of design (step ① in Figure 1) and generate an appropriate V&V plan (step ② in Figure 1). We illustrate the approach with an automotive infotainment system developed by the industrial partner.

We first model the system using THREATGET (Section 3.1) and apply analysis to identify potential structural weaknesses in the system architecture (Section 3). We then use this analysis to derive a V&V plan (Section 3.3). Finally, we can augment it with threat repair to propose additional security measures [20].
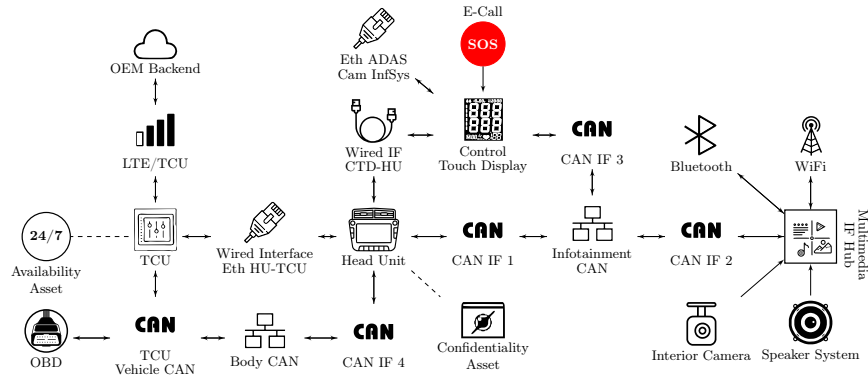
Fig. 2: Automotive infotainment system model.

3.1 System Architecture Model

We first create an accurate model of the automotive infotainment system (IS), shown in Figure 2. The IS is part of a larger ADAS reference model. It has several external interfaces that expose an attack surface of the vehicle. The external interfaces in Figure 2 are Bluetooth, WiFi, Interior Camera, and On-Board Diagnostics (OBD). The Multimedia Interface Hub (MIH) is an essential component of the infotainment system that (co-)implements core functionalities, including navigation, phone calls, and music playback. MIH also bridges external and internal interfaces. The Telematics Communication Unit (TCU) is the primary interface to the Internet. Many components in a modern vehicle depend on the TCU. For example, navigation systems use TCUs to access and update maps, and ECUs use them for over-the-air updates. Finally, all components except for TCU and Head Unit communicate through a CAN interface. We add two assets to the model – the confidentiality asset associated with the Head Unit and the availability asset associated with the TCU. The assets need to be protected, and their associated components are potential targets for attackers.

The IS is a weak security link in modern vehicles because it is more prone to successful cheap attacks than other components (e.g., *Body Control Unit* or the *Engine Control Unit*). This is due to versatile attack scenarios provided by the use of mainstream Unix-like operating systems, e.g., *Uconnect* and *Automotive Grade Linux*, the user requirements demanding functionalities like a built-in internet browser and installing third-party apps enabling remote code execution attacks, and the use of CAN bus that cannot guarantee communication integrity between the vehicle's external and internal interfaces.

3.2 Threat Analysis

We analyze the system model with THREATGET against its *threat database*, defining a set of possible threats formulated as *rules*. The threat descriptions are collected from multiple sources: automotive security standards and regulations

(e.g., ISO/SAE 21434, ETSI, UNECE WP29 R155, and UNECE R156), publicly documented threats identified in past incidents, and expert knowledge.

We illustrate threat rules with two examples used during the analysis of the infotainment system model: the rule named "Gain Control of Wireless Interface (e.g., WiFi, Bluetooth, or BLE)" and the rule named "Flood CAN Communication with Messages". Both threat rules originate from automotive security analyses performed by domain experts. The first threat's formalization is

```
ELEMENT  : "Wireless Interface"{
    "Authorization" NOT IN ["Yes", "Strong"] & "Input Sanitization" != "Yes" &
    "Authentication" NOT IN ["Yes", "Strong"] & "Input Validation" != "Yes" &
    PROVIDES CAPABILITY "Control" := "true". }
```

This rule specifies that a wireless interface (e.g., WiFi or Bluetooth) that neither implements authorization and authentication nor sanitizes or validates its inputs is susceptible to threats. The last line in the rule explicitly states that if this threat is exploited, the malicious user can control the wireless interface. The "Threat Flood CAN Communication with Messages" threat is formalized as

```
FLOW {
    SOURCE ELEMENT  : "ECU" { REQUIRES CAPABILITY "Control" >= "true" } &
    TARGET ELEMENT  : "ECU" {
        HOLDS ASSET {
            "Cybersecurity Attribute" = "Confidentiality" &
            PROVIDES CAPABILITY "Read" := "true" } } &
    INCLUDES ELEMENT  : "BUS Communication" &
    INCLUDES NO ELEMENT  : "ECU" { "Anomaly Detection" = "Yes". } }
```

This rule states that the threat is present if there is a path starting from an ECU that is under the control of a malicious user to another ECU that holds the confidentiality asset and that there is a bus between them and no ECU on the path has implemented anomaly detection.

When applied to the infotainment system model, THREATGET identifies multiple threats. One threat is "Spoof messages in the vehicle network because of the missing components". It describes a pattern that starts at an Interface with no Authentication and ends at an ECU with no Input Validation and holds an asset. It includes a wired Shared Medium representing a vehicle's CAN BUS. Moreover, no element (of type Firewall, Server, ECU, or Gateway) on the flow from the Interface to the ECU takes care of Anomaly Detection.

We can address the identified threats with appropriate security measures. Threat repair [20] consists of preventing concrete threats by proposing security measures that can be implemented during the system's design. THREATGET implements *attribute repair*, a method that proposes changes in the components' security attributes as locally deployed measures with a simple cost model.

In the case of the automotive infotainment system model, e.g., the proposed threat repair measures include enabling authorization and implementing authentication in the WiFi and Bluetooth components. We note that threat repair does not remove the need for the planned V&V activities. The fact that authentication is integrated into the WiFi device, following the outcomes of threat repair, does

not guarantee that the authentication algorithm's implementation is weakness free. On the contrary, systematic testing of the WiFi's authentication protocol is even more necessary to gain confidence that the WiFi device is not a possible entry point for malicious users.

3.3   V&V Planning

In addition to threat analysis, there is support for identifying and modeling more sophisticated threats using attack trees; c.f. [8]. This results in more knowledge about potential attackers' steps when intruding into a system. Simple rules can be assigned attributes called capabilities that are either required for an intrusion or can be gained through the intrusion of a system component. Moreover, we can define the different access levels to a component (e.g., $Access < Read < Modify < Control$). Depending on previously acquired capabilities, different attack tree rules trigger, yielding distinct attack trees. An example of such a generated attack tree is illustrated in Figure 3.

The attack tree depicted in Figure 3 shows how a malicious user can access the *confidentiality* asset associated with the Head Unit via external interfaces such as WiFi and Bluetooth. For instance, control of the Bluetooth interface can be gained if its security attributes (input validation and sanitization, authorization and authentication) are not implemented or have weaknesses. From there, the user can gain control of the Multimedia Interface Hub, which is not sufficiently secure, and then get control of the Head Unit and hence the access to the asset. The attack tree exposes the most critical components that need to be protected. We note that the attack tree from Figure 3 is not maximal nor unique – while THREATGET generates multiple trees for each asset in the model, including the maximal attack trees, we use a simpler tree for illustration purposes.
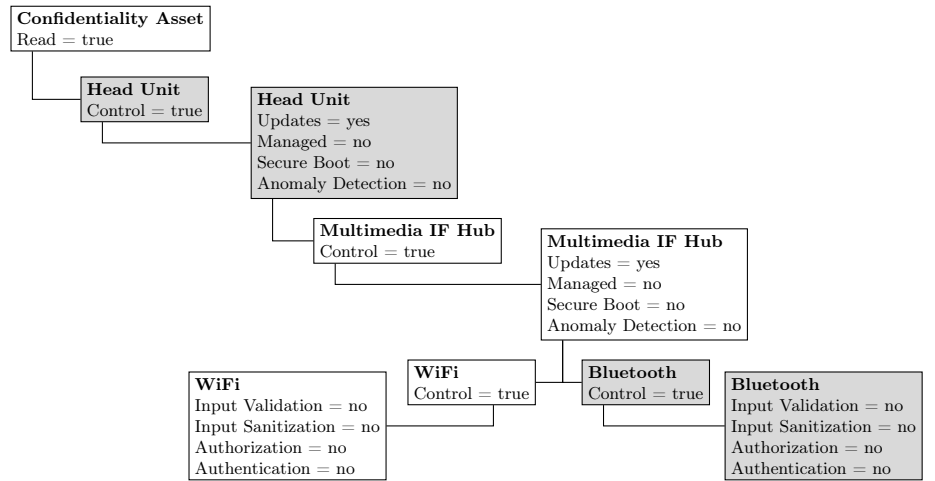

Fig. 3: Attack tree derived from THREATGET. Multiple children from the same node are implicitly interpreted with an OR operation.

4   Automotive Security Testing

In this section, we advocate an approach based on learning to test critical components identified by the threat analysis methods during concept design, when these components are assumed to be black-box to the tester.

4.1   Automata Learning for Correctness

Many cyber-physical components in the automotive domain implement one or multiple finite state machines (FSMs). Implementing larger automotive FSMs becomes cumbersome mainly because: (1) ensuring FSM's correctness w.r.t. its specification is expensive, (2) correctly coding the structure of a large FSM is difficult, and (3) correct integration of FSMs in complex software is hard. Unfortunately, many software-driven components in the automotive industry are black boxes from different manufacturers, hence are hard to verify and thus do not provide functional or non-functional guarantees.

Given an FSM of a black-box automotive component, we can test and verify it to increase our confidence in its correctness. Automata learning has proven to be a successful method for learning-based testing of communication protocols that are also used in the automotive domain, e.g., MQTT [19] or Bluetooth Low Energy [16]. We use automata learning [3] to infer an FSM model (concretely a Mealy machine) of the the SUT. In the learning context we refer to the SUT by system-under-learning (SUL). In automata learning, a *learner* asks an *oracle* two types of queries. First, *membership queries* to determine the SUL's output for a given input word. Second, *equivalence queries* check whether a learned model conforms to the SUL, to which the oracle returns positive answer or a counterexample. A counterexample is an input-output word distinguishing SUL from hypothesis. In practice, oracles for black box systems work with conformance testing.

Ordinarily, real-world systems' alphabets are not manageable for learning algorithms. Abstraction helps to both cope with this fact and to make inferred models more human-readable. Too much abstraction, however, might induce non-deterministic behavior and hide problems we intend to find. There are also automatic abstraction refinement approaches for an optimum of abstraction in a mapper [1, 10]. An abstraction mapper consists of a mapping function that converts a concrete input into an abstract symbol. It also observes the SUL's concrete outputs and sends an abstraction to the learner. To send a concrete input to the SUL, the mapper inverses the abstraction. There are multiple methods to assess the behavioral correctness of the learned FSMs, including (1) black-box checking [15], adaptive model checking [9], a combination of learning-based testing and machine learning [13] and symbolic execution [2].

4.2   Use-Case Scenarios

The attack tree (see Figure 3) poses the critical components that need to be tested for security. In this section, we illustrate our learning-based testing approach on the two components highlighted in gray color in Figure 3 - the Bluetooth interface (as an entry vector) and the Head Unit ECU.

Bluetooth and Bluetooth Low Energy Bluetooth is a well-established standard for wireless audio used in most infotainment systems. Bluetooth Low Energy (BLE) grows in popularity for car access and sensor data transmission. The protocols have a variety of known vulnerabilities [4–7, 18], some also specifically for automotive systems[7].
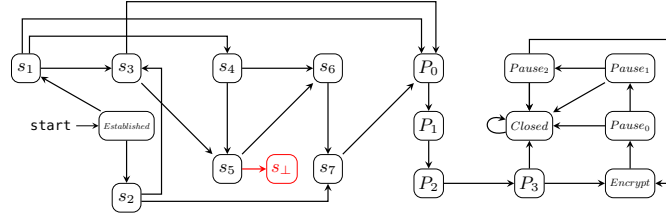

Fig. 4: Inferred FSM structure for Bluetooth pairing.

Learning Setup we use Intel Wireless Controllers (AC 8265 and AX200) implementing Bluetooth and BLE. The learning setups are similar, the difference is in the radio hardware and the physical layer, requiring three entities: (1) Radio Device, (2) Learner, and (3) Interface between the two with a mapper. The learner was implemented using the LearnLib framework [11].

Learned Model and Findings We inferred the pairing process models, which are used for encryption and therefore security-critical in the SULs. As a tangible result, we discovered a BLE deadlock state (red state in Figure 4) in the Linux BLE host software. With repeated out-of-order transmission of pairing requests of different types, we force the respective BLE stack into a state that limits the device to respond to basic link-layer control packets. After the state is reached, each following connection will start in this state until the controller is reset.

Unified Diagnostic Services Each ECU has a secure access mode reachable through its UDS implementation, available via vehicle's OBD connector. An attacker able to exploit UDS security features would be also able to manipulate data or even flash the ECU with a malicious firmware.

Learning Setup To communicate with the ECU we used a CAN interface. To learn a different ECU we only need to adapt the interface. We started by implementing a reduced UDS interface, consisting of instructions to put an ECU into secure access mode. Communications occures via a CAN bus interface. The learner was implemented using the AALpy framework [14].

Learned Model and Findings The learning experiment resulted in a reduced FSM of the UDS shown in Figure 5. An analysis of the results shows that once being

---

[7] https://research.nccgroup.com/2022/05/15/technical-advisory-tesla-ble-phone-as-a-key-passive-entry-vulnerable-to-relay-attacks/

successfully authenticated (state s_4), an incorrect authentication key will still result in the same state. This is unexpected and allows for prolonging a session without authentication. When requesting a new seed for re-authentication (s_5) this behavior persists. Moreover, on re-entering a secure session afterwards (from s_6), the ECU accepts an old key as well; an unexpected behavior after re-initiating the key authentication. Figure 5 marks all unexpected behaviors in red.

5   Conclusion

We introduced the TRUSTED methodology for designing and assessing trusted and secure automotive systems. The main novelty of the proposed methodology is its holistic and systematic approach to security, which starts at concept design and is carried down to the implementation and assessment of individual components. We instantiated the different parts of the methodology using the state-of-the-art methods and tools for threat modelling and analysis, automata learning and testing. We illustrated the use of the methodology by applying it step-by-step an automotive infotainment system. Using the learning-based testing approach we could document previously unpublished denial-of-service conditions in the examined BLE setups, as well as unexpected behavior allowing for extending secure UDS programming sessions on the scrutinized ECU.

Future Work We plan to further automate the transition from the concept design and V&V planning on one side, to the actual testing activities done on the level of components by devising a domain-specific test description language that can define abstract V&V plans derived from the attack trees, and be refined in a way so that eventually it can be executed on a platform (e.g., as in [21]). Second, the TRUSTED methodology mainly focuses on the transition from concept design to testing the implementation. We plan to also study the opposite direction – how to use the component testing results to update the system model and have a more refined threat analysis and a more realistic threat assessment.

Acknowledgements

Fig. 5: Inferred UDS FSM.

References

1. Aarts, F., Heidarian, F., Kuppens, H., Olsen, P., Vaandrager, F.W.: Automata learning through counterexample guided abstraction refinement. In: FM 2012. pp. 10–27. Springer, Berlin (2012)

2. Aichernig, B.K., Bloem, R., Ebrahimi, M., Tappler, M., Winter, J.: Automata Learning for Symbolic Execution. In: 2018 Formal Methods in Computer Aided Design (FMCAD). pp. 1–9. IEEE, Austin, Texas, USA (2018). https://doi.org/10.23919/FMCAD.2018.8602991

3. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation **75**(2), 87–106 (Nov 1987). https://doi.org/10.1016/0890-5401(87)90052-6

4. Antonioli, D., Tippenhauer, N.O., Rasmussen, K.: BIAS: Bluetooth Impersonation AttackS. In: 2020 IEEE Symposium on Security and Privacy (SP). pp. 549–562. IEEE, San Francisco, CA, USA (May 2020). https://doi.org/10.1109/SP40000.2020.00093

5. Antonioli, D., Tippenhauer, N.O., Rasmussen, K.: Key Negotiation Downgrade Attacks on Bluetooth and Bluetooth Low Energy. ACM Trans. Priv. Secur. **23**(3), 14:1–14:28 (Jun 2020). https://doi.org/10.1145/3394497

6. Antonioli, D., Tippenhauer, N.O., Rasmussen, K., Payer, M.: BLURtooth: Exploiting Cross-Transport Key Derivation in Bluetooth Classic and Bluetooth Low Energy. In: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security. pp. 196–207. ASIA CCS '22, Association for Computing Machinery, New York, NY, USA (May 2022). https://doi.org/10.1145/3488932.3523258

7. Antonioli, D., Tippenhauer, N.O., Rasmussen, K.B.: The KNOB is broken: Exploiting low entropy in the encryption key negotiation of bluetooth BR/EDR. In: Heninger, N., Traynor, P. (eds.) 28th USENIX Security Symposium, USENIX Security 2019. pp. 1047–1061. USENIX Association, Santa Clara, CA, USA (2019)

8. Ebrahimi, M., Striessnig, C., Triginer, J.C., Schmittner, C.: Identification and verification of attack-tree threat models in connected vehicles. In: SAE Technical Paper 2022-01-7087 (2022). https://doi.org/10.4271/2022-01-7087

9. Groce, A., Peled, D., Yannakakis, M.: Adaptive Model Checking. In: Katoen, J.P., Stevens, P. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 357–370. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2002). https://doi.org/10.1007/3-540-46002-0_25

10. Howar, F., Steffen, B., Merten, M.: Automata Learning with Automated Alphabet Abstraction Refinement. In: Jhala, R., Schmidt, D. (eds.) Verification, Model Checking, and Abstract Interpretation. pp. 263–277. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-18275-4_19

11. Isberner, M., Howar, F., Steffen, B.: The Open-Source LearnLib. In: Kroening, D., Păsăreanu, C.S. (eds.) Computer Aided Verification. pp. 487–495. Lecture Notes in Computer Science, Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-21690-4_32

12. Marksteiner, S., Marko, N., Smulders, A., Karagiannis, S., Stahl, F., Hamazaryan, H., Schlick, R., Kraxberger, S., Vasenev, A.: A Process to Facilitate Automated Automotive Cybersecurity Testing. In: 2021 IEEE 93rd Vehicular Technology Conference (VTC Spring). pp. 1–7. IEEE, New York, NY, USA (2021)

13. Meinke, K.: Learning-Based Testing of Cyber-Physical Systems-of-Systems: A Platooning Study. In: Reinecke, P., Di Marco, A. (eds.) Computer Performance Engineering. pp. 135–151. Lecture Notes in Computer Science, Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-66583-2_9

14. Muškardin, E., Aichernig, B.K., Pill, I., Pferscher, A., Tappler, M.: AALpy: An active automata learning library. Innovations in Systems and Software Engineering **18**(3), 417–426 (Sep 2022). https://doi.org/10.1007/s11334-022-00449-3

15. Peled, D., Vardi, M.Y., Yannakakis, M.: Black Box Checking. In: Wu, J., Chanson, S.T., Gao, Q. (eds.) Formal Methods for Protocol Engineering and Distributed Systems: FORTE XII / PSTV XIX'99 IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XII) and Protocol Specification, Testing and Verification (PSTV XIX) October 5–8, 1999, Beijing, China, pp. 225–240. IFIP Advances in Information and Communication Technology, Springer US, Boston, MA (1999). https://doi.org/10.1007/978-0-387-35578-8_13

16. Pferscher, A., Aichernig, B.K.: Fingerprinting Bluetooth Low Energy devices via active automata learning. In: Huisman, M., Pasareanu, C.S., Zhan, N. (eds.) Formal Methods - 24th International Symposium, FM 2021, Virtual Event, November 20-26, 2021, Proceedings. Lecture Notes in Computer Science, vol. 13047, pp. 524–542. Springer (2021). https://doi.org/10.1007/978-3-030-90870-6_28

17. Schmittner, C., Chlup, S., Fellner, A., Macher, G., Brenner, E.: Threatget: Threat modeling based approach for automated and connected vehicle systems. In: AmE 2020 - Automotive meets Electronics; 11th GMM-Symposium. pp. 1–3. VDE Verlag, Berlin (2020)

18. Seri, B., Vishnepolsky, G.: The dangers of Bluetooth implementations: Unveiling zero day vulnerabilities and security flaws in modern Bluetooth stacks. Tech. rep., Armis Inc. (2017)

19. Tappler, M., Aichernig, B.K., Bloem, R.: Model-based testing IoT communication via active automata learning. In: 2017 IEEE International Conference on Software Testing, Verification and Validation, ICST 2017, Tokyo, Japan, March 13-17, 2017. pp. 276–287. IEEE Computer Society (2017). https://doi.org/10.1109/ICST.2017.32

20. Tarrach, T., Ebrahimi, M., König, S., Schmittner, C., Bloem, R., Nickovic, D.: Threat repair with optimization modulo theories. CoRR (2022)

21. Wolschke, C., Marksteiner, S., Braun, T., Wolf, M.: An Agnostic Domain Specific Language for Implementing Attacks in an Automotive Use Case. In: The 16th International Conference on Availability, Reliability and Security. pp. 1–9. ARES 2021, Association for Computing Machinery, New York, NY, USA (Aug 2021). https://doi.org/10.1145/3465481.3470070