Generation of Tailored and Confined Datasets for IDS Evaluation in Cyber-Physical Systems

Thomas Hutzelmann, Dominik Mauksch, Ana Petrovska, and Alexander Pretschner

Abstract—The state-of-the-art evaluation of an Intrusion Detection System (IDS) relies on benchmark datasets composed of the regular system's and potential attackers' behavior. The datasets are collected once and independently of the IDS under analysis. This paper questions this practice by introducing a methodology to elicit particularly challenging samples to benchmark a given IDS. In detail, we propose (1) six fitness functions quantifying the suitability of individual samples, particularly tailored for safety-critical cyber-physical systems, (2) a scenario-based methodology for attacks on networks to systematically deduce optimal samples in addition to previous datasets, and (3) a respective extension of the standard IDS evaluation methodology. We applied our methodology to two network-based IDSs defending an advanced driver assistance system. Our results indicate that different IDSs show strongly differing characteristics in their edge case classifications and that the original datasets used for evaluation do not include such challenging behavior. In the worst case, this causes a critical undetected attack, as we document for one IDS. Our findings highlight the need to tailor benchmark datasets to the individual IDS in a final evaluation step. Especially the manual investigation of selected samples from edge case classifications by domain experts is vital for assessing the IDSs.

Index Terms—Intrusion Detection Evaluation Problem, Benchmark Dataset, Methodology, Scenario-Based Optimization, Advanced Driver Assistance System, openpilot, Controller Area Network, Hardware in the Loop Simulation

1 Introduction

Assuring a high level of security in modern applications requires complementing measures during system design and operation. The last line of defense is intrusion detection, potentially mitigating previously unknown attacks. An intrusion detection system (IDS) is an additional security component that monitors a system during its runtime and yields an alarm once it observes suspicious behavior [1]. The plethora of detection approaches proposed and developed within the last decades [2] raises the need to compare the candidate IDSs and identify the most suitable approaches for a given use case and the available resources. This challenge is called the Intrusion Detection Evaluation Problem [3]. In general, all approaches to the evaluation of potential IDSs follow the same standard evaluation schema: All candidate IDSs analyze the same labeled dataset containing malicious and benign behavior recorded from the system. The performance of the IDS and all raised or missing alarms are recorded and summarized in metrics [4] to convey trade-offs and advantages of the determined winning candidate.

An essential requirement in the evaluation is having a dataset representative of the use case, which is also challenging for the analyzed IDSs. Especially for comparing various approaches in science, reusing such a dataset as a benchmark and baseline for future research makes creating sound datasets mandatory. Following the standard evaluation schema, domain experts collect the dataset first without considering the IDS they will evaluate later. This separation fosters a realistic evaluation and prevents biases. In science, researchers publish their dataset in a final and invariant form once they consider the collected traces sufficient. Other researchers then choose one or multiple of these datasets to showcase the performance of their newly proposed IDS relative to other competitors. However, this setup does not assess how suitable the chosen dataset is for evaluating the proposed IDS. Using an inadequate dataset endangers the validity of the obtained results.

In this paper, we work with the hypothesis that the quality of a benchmark dataset needs to be measured relative to the IDS under evaluation. In other words, the IDS under analysis determines the properties and samples constituting a "good" dataset for its evaluation. This hypothesis questions whether the standard evaluation schema to using a constant dataset is a sensible approach—or if a fair assessment of the IDS requires tailored and confined datasets that incorporate characteristics of the IDS and detection model itself. This new view requires a link between the IDS's classifications and the suitability of dataset samples for evaluation. If suitable data points are absent in current datasets, a sound evaluation requires a new methodology that fosters the inclusion of the most suitable samples for a given IDS.

To establish this missing link between the IDS and the dataset, we propose fitness functions that quantify the suitability of individual dataset samples regarding six different qualities for evaluating a given IDS. Furthermore, we propose a methodology based on scenario-based optimization to systematically deduce data points for the given IDS with the best rating according to our fitness function. This methodology extends the state-of-the-art evaluation with a final step, generating new sample points for the last assessment before deploying an IDS. In a case study, we investigated two state-of-the-art IDSs defending the same attack and analyzed the datasets used for their evaluation.

• *All authors are in the Chair of Software and Systems Engineering, Technical University of Munich, 85748 Garching b. München, Germany*
E-mail: {t.hutzelmann, d.mauksch, ana.petrovska, alexander.pretschner}@tum.de

Manuscript received April 19, 2005; revised August 26, 2015.

With our methodology, we found, in one case, valid attacker behavior that circumvents the IDS and causes critical damage to the system. The finding that both original evaluations do not consider behavior similar to the points we deduced supports our hypothesis. We suppose that the transition from signature-based detection to machine learning-based approaches and the increasing complexity in the detection models further impede the manual deduction of challenging data points. The small size of our investigated datasets might have contributed to their lack of critical points. But even massive datasets might only include such data points by chance and currently do not guarantee their inclusion. In the worst case, the dataset includes these critical points, but they get lost in the enormous number of other classifications of the IDS summarized in percentage scores. Hence, domain experts might not become aware of this critical behavior during the evaluation of the IDS.

Although we believe our hypothesis generalizes to the evaluation of all IDSs, this paper focuses on Intrusion detection on the Controller Area Network (CAN) bus in the automotive domain. There has been extensive research about the CAN bus in previous years, and various publications propose potential IDSs [5], [6]. Our deductions use concepts from the network domain and modify an attack by manipulations of the communication on the network. This choice of an attacker makes our methodology more intuitive for network-based-IDS. However, the evaluation of a host-based IDS that, for example, validates and monitors the internal models during computation does look identical as long as it mitigates the same type of attacker. Our fitness functions require a direct measure of the success of an attack or its attempt. While we also refer to approaches to generalize this concept, we tailored our methodology, for now, to safety-critical cyber-physical systems, as safety distances and violations provide this measure with high precision.

We use the following definitions and terminology to talk about datasets. A *trace* is a finite observation of the system's behavior during operation, potentially under the influence of an attacker, i.e., a time series of events recorded from the system. Together with a label about the nature of the recorded behavior, *benign* or *malicious*, a trace serves as a *data point* to benchmark the correct classification by a given IDS. A *(benchmark) dataset* is a fixed collection of various and diverse data points proposed by researchers or domain experts. Please note that other literature uses terms like sample, record, observation, item, or instance to describe elements of a dataset. Other terms in the literature reflect the concrete representation of a data point, e.g., log files, NetFlows, event streams, or raw package dumps. However, the standard evaluation schema and our methodology are agnostic to the nature of a data point as long as they provide the features for the IDS and sufficient description for the domain experts. We introduce the term *dataset space* to refer to the set containing all possible datasets.

To summarize, this paper makes the following *contributions*: (1) We propose an ensemble of six distinct fitness functions quantifying the suitability of data points in a dataset for evaluating a given IDS. We intentionally limit ourselves to safety-critical cyber-physical systems but sketch possible extension points for future work. (2) We propose a scenario-based optimization to systematically elicit edge case behavior of the system and network attacker to highlight strengths and weaknesses relative to the particular IDS under analysis. (3) We exemplify this methodology for a detailed security analysis of an open-source advanced driver assistance systems (ADAS) on automation level two (Adaptive Cruise Control and Lane Keeping Assist) [7] and two state-of-the-art IDSs. We provide a deduction of various realistic attack samples on an automated car usable as a benchmark for any CAN bus IDS. Overall, the divergence between the optimized data points and the previous benchmark dataset and between both IDSs leads us to conclude that optimal benchmark datasets must reflect the IDSs under evaluation. Therefore, (4) we elaborate on possibilities to establish our methodology within the development and evaluation process for IDSs in general and propose an adaptation in future benchmark datasets to ensure such critical data points.

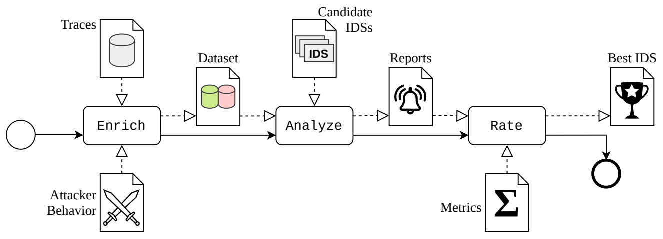

Fig. 1. Abstract schema of the three steps in the standard evaluation

The rest of this paper is structured as follows: Section 2 introduces related work. Section 3 elaborates our proposed methodology to generate optimized samples forming minimal benchmark datasets. Section 4 applies this methodology to a selected use case and lays the foundation for the experiments in Section 5, in which we investigate the selected scenarios in depth. Section 6 discusses the generated datasets and reflects on the impact of our work on future approaches for IDS evaluation. Finally, Section 7 concludes our work.

2 Related Work

2.1 Intrusion Detection Evaluation Problem

The commonly agreed way to evaluate potential IDSs for a concrete use case consists of three steps: Enrich, Analyze, and Rate, as depicted in Fig. 1. In the first step, domain experts gather a dataset from the protected system with representative normal behavior and attack samples. To generate malicious data points for the dataset, they either 1) include previously known attack samples, if available, or 2) simulate suspected attack effects. In the second step, they feed this labeled dataset into all candidate IDSs and record their performance and outputs. Finally, in the third step, they compare the yielded alarms of all IDS with the dataset labels. To handle numerous data points, they choose a metric [4] and accumulate the counts of correctly and wrongly classified data points to obtain a final ranking of the best candidates.

Classical metrics like Detection Rate or False Positive Rate combine four different counters during evaluation: false positives (FP), true negatives (TN), false negatives (FN), and true positives (TP). These counters are combinations of two properties: (1) the data point showing peaceful system behavior or system behavior under the manipulation of the attacker and (2) the IDS yielding an alarm or staying

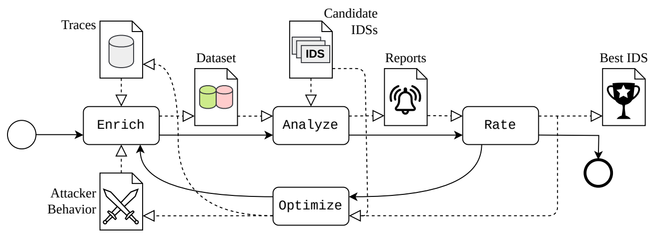

Fig. 2. Abstract Schema of our extension of the standard evaluation.

silent while analyzing this data point. We followed the same categories in defining our optimization criteria for deducing our fitness functions. In a classical evaluation, however, these metrics approximate the actual detection capabilities of the IDS that highly depend on the quality, quantity, and diversity of the analyzed data points. Our methodology aims to showcase the IDS's capabilities by focusing on particular data points for each category instead of an unprioritized accumulation of large datasets.

Noteworthy, the standard evaluation schema focuses only on the evaluation phase and assumes the existence of fully configured and executable candidate IDSs. Engineers generally differentiate between two datasets, one used for development (e.g., to train the detection algorithm) and one for evaluation to avoid overfitting and biases [8]. Although anomaly-based IDSs, unlike signature-based IDSs, do not require knowledge about the attacker and are trained on purely peaceful system behavior, the evaluation always requires realistic and diverse attack samples. Without attack samples in the dataset, it is impossible to determine the number of true positives (attacks spotted by the IDS) and false negatives (attacks remaining unnoticed by the IDS), and the evaluation remains partial. Since the first step in the standard evaluation schema relies on manual effort to collect and label the dataset, it might unnoticeably invalidate the evaluation. Wrong-labeled, oversimplified, incomplete, monotone, biased, or undersized datasets may result in good evaluation results in step three, even when the IDSs, in reality, do not provide sufficient protection against actual attackers. Therefore, collecting more advanced and subtle samples of attacker behavior from diverse attacks is vital for a sound evaluation and a core part of our methodology.

Our work addresses these limitations by extending the evaluation schema as follows (cf. Fig. 2): We introduce a feedback loop from the observed detection abilities on an initial dataset of the candidate IDS in the Rating step to the Enrich step to generate new, labeled data points. The essential artifact of this loop is the systematically spanned space of all potential datasets. To automate this process, we propose optimization using fitness functions to guide the generation of data points toward system behavior that showcases behavior specific to the IDS under analysis. These data points represent edge case behavior and are complementary to existing metrics for the detection rate and false positive rate of a candidate IDS and, in theory, make an evaluation without these metrics feasible. The methodology proposed in our work reduces the manual effort needed to create a challenging dataset. It supports the quality assessment with the worst edge case data point for each candidate IDSs.

2.2 Attack Generation

On an abstract level, we leverage automatic attack generation to improve the evaluation methodology. Szegedy et al. [9] first introduced the idea of automatically generating adversarial examples resulting in wrong classifications on a given deep neural network classifier. This idea transfers to IDS that internally use neural networks [10] or other machine learning classifiers [11], [12]. This technique can also analyze the classifiers in generic IDS without restrictions on their internal implementations [13]. These approaches follow a general pattern: The analyzed classifier processes a feature matrix of given or random points. In multiple iterations, the adversarial sample generator computes modifications of a selected entry in the feature matrix. The classifier repeats the analysis of this new point, hopefully resulting in a change in the certainty of the classification or the assigned class. Guided by this change, the generator mutates the modifications until a generated sample results in a wrong classification.

Compared to our work, these approaches focus only on the internal classifier and the feature representation but ignore the surrounding system and potential attacker behaviors. A generated modification of the feature matrix is artificial data and not actually behavior observable in the system or by an actually performed attack. This indirection questions the practical significance of the generated samples as they might constitute impossible behavior. Our optimization of data points focuses on the entire space of potential system behavior with gradual impacts of the attacks on the system's operation. In comparison, adversarial learning aims to tamper with an individual classification or flip a single decision. Overall, we investigate a broader system scope with comprehensible parametrizations ensuring realistic samples. Furthermore, this broader scope includes diverse sources of side effects for a wrong classification. Our fitness functions also provide a means to rank and prioritize all generated attacks to focus on the most critical oversights. Finally, our evasions equally analyze machine learning models, anomaly detection, or any black box intrusion detection method.

2.3 On Benchmark Datasets

Our efforts to identify challenging data points align with the general creation of benchmark datasets for IDSs. Some intuitive factors imply an update of a dataset, e.g., the emergence of new attacks or major changes in the protected system. However, these factors are not the main reason for creating new datasets. We found two repeating patterns: On the one hand, high-impact datasets as the most utilized and investigated dataset for IDS evaluation originated from the DARPA Intrusion Detection Evaluation [14]. After closer investigation, researchers spotted various flaws [15] and proposed improved variants [16] and re-recordings of the dataset in similar ways [17], [18], [19], [20], [21]. On the other hand, in domain-specific networks, like the CAN bus in our case study, the datasets are either 1) data traces directly published aside a security analysis as a documentation of the conducted attack (e.g., [22]), or 2) are collected from papers proposing new detection approaches (e.g., [23], [24]).

Despite all efforts to fix the spotted weaknesses, the successor dataset might still contain other or newly created flaws and biases. Without an objective metric or generation methodology, only further deep analysis will identify them and propose a better future dataset. This problem applies more to domain-specific datasets since they are relatively small and, in the beginning, contain merely a few variants of the same attack. Most importantly, no follow-up analysis has investigated their suitability for evaluation as profoundly as shown to be required by the experience from previous datasets. In our work, we investigate a methodology that actively tailors the term of a "good" dataset relative to the concrete protected system, the attacker behavior, and—crucially—to the IDSs under analysis. Thus, we aim to automate the dataset generation, focusing on objectivity and excluding implicit biases.

A general issue in the datasets used for IDS evaluation is a high imbalance among the types of data points represented in the dataset [25]. Benign behavior is more dominant than malicious data points, and datasets do not represent each type of attack with the same amount of data points. These imbalances impact the obtainable scores in the evaluation. Large datasets especially underrepresent critical attacks that remain unnoticed in the scores. Machine-learning algorithms, therefore, might tend to ignore these attacks completely [26]. Our methodology focuses the evaluation on a few data points that showcase the edge cases in classification. Each logical scenario (see Sect. 3.4) always results only in three benign and three malicious data points. The attacks always focus on the most impactful attack among the modeled attacks. Therefore, balance is no problem in our methodology.

In this context, Apruzzese et al. [27] promote the cross-evaluation of an IDS with data points from multiple datasets. While such evaluation provides additional insights compared to the individual dataset alone, our methodology fosters the challenge in the evaluation even more. First, only data points in the original datasets are in any combination. The combination compensates for oversights in a few datasets but does not improve on behavior that none of the datasets included. Considering our evaluation, this particularly applies to the edge case behavior we pinpoint with our methodology that was not present in the original dataset. Furthermore, the unfiltered combination of datasets further increases the number of classifications in the evaluation, increasing the problems of imbalance and oversights our methodology aims to mitigate. The systematic deduction of datasets offers a different perspective to the conceptual model for cross-evaluation. Datasets are samples from a spanned dataset space. Hence, instead of merging the sample sets, we hold combining the spanned spaces and selecting samples anew to be more beneficial. Our stepwise process can combine two sets of artifacts as a direct union before using the optimizer. Despite the larger space, the optimizer still converges towards the most relevant data points for domain experts and neglects irrelevant areas in the dataset space without further adjustments.

2.4 Scenario-Based Testing

Scenario-based testing is a methodology to systematically choose test cases that stress and verify the behavior of a system under test. On a high level, a structured set of scenarios describes all potential behavior of the system as a whole. As defined by Ulbrich et al. [28], a scenario is a temporal development between several scenes that describe a snapshot of the environment, including scenery, dynamic objects, and all actors' and observers' behavior. Menzel et al. [29] propose three different levels of abstraction for scenarios during the development and verification of a system: functional, logical, and concrete scenarios. Functional scenarios describe the relevant entities of a domain on a semantic level and their relationship using a consistent and domain-specific vocabulary. Logical scenarios refine entities and relations of the functional scenarios into parameter ranges within a state space. Finally, concrete scenarios depict the behavior through concrete values for each parameter within the state space. Each scenario type can be consistently refined or abstracted into the other types, thereby grouping a theoretically infinite amount of concrete scenarios.

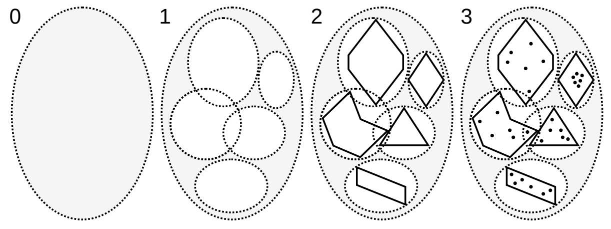

Fig. 3. Schematic models (from left to right): (0) the space of all datasets, (1) partitioned with functional scenarios, (2) further refined with logical scenarios, and (3) the concrete data points elicited through optimization.

Each concrete scenario can form test cases if the expected behavior is encoded and checked during the execution. In our work, we utilize this idea to describe undesired behavior with the notion of a safety envelope [30] that enables the overall system to still react in time to prevent safety violations in the future. We used scenario-based optimization [31] to choose meaningful parameters for refining logical scenarios into concrete scenarios. This approach mutates the parameter set and automatically optimizes towards the Pareto Front according to a given fitness function. In the automotive domain, the fitness functions mainly encode safety envelopes and safety-critical edge case behavior, among other desired properties of the scenario, as a single objective. Our methodology follows the general concepts and the three abstraction levels known and approved for scenario-based testing. We put our focus on extending them to include attacker and system behavior in parallel and propose fitness functions similar to existing fitness function templates [32] that stress the advantages and disadvantages of a specific IDS under analysis. To our knowledge, we are the first to combine scenario-based optimization with IDS evaluation.

3 Methodology

This section presents our methodology at a high level and discusses the concepts without implementation details. To make these abstract ideas more tangible, Section 4 follows the same three steps to present an in-depth case study. Please note that the fitness functions are an independent contribution usable without the rest of our methodology. Nevertheless, for the text flow, we decided to introduce them in Section 3.5 within the scoping of the methodology.

3.1 General Overview

Our methodology provides a final validation before deciding whether to deploy an IDS implementation within the system.

In theory, any concrete system implementation and an attacker model define the set containing all possible system behavior and all possible interference of the attacker. We refer to this set as *dataset space* as it is a superset of all potential benchmark datasets. In other words, any benchmark dataset is a diverse and broad selection of data points from this dataset space. Due to the nature of the system environment and partial knowledge during engineering, this space is only a theoretical concept. Vague textual formulations without details describe this space, e.g., "an attacker inserts new messages into the system's network communication".

As depicted in Fig. 3, the first two steps systematically span and refine the dataset space. In the final third step, an optimizer elicits critical data points that particularly challenge the IDS in this system. In **step 1**, the deduced functional scenarios partition the dataset space into subspaces, each spanning a particular functionality of the system and a particular type of manipulation by the attacker, e.g., the attacker manipulating specific information on the network while the system is in a specific state. Depending on the use case, these subspaces might unavoidably overlap, but our methodology aims for broad coverage by systematically creating numerous partitions. In **step 2**, introducing parameters and their domains concretizes each subspace to form logical scenarios, e.g., the attack begins between n and m seconds after a specific event. With broad domains, these parameters may include the entire unrestricted subspace. Still, for more efficient evaluation, we recommend reducing the spaces to reasonable system operations, e.g., to conditions with guaranteed safety or expected system operation. Finally, in **step 3**, optimizations with specific fitness functions identify concrete scenarios from each spanned subspace. Each concrete scenario specifies a data point for the evaluation, e.g., the system's operation in a specific setup and situation with a concrete step-wise manipulation of the attacker. They show particularly relevant behavior within the subspace as they are optimized to stress the classification of the IDS towards edge cases. This results in a benchmark dataset containing only a few but particularly critical points. If needed, the fitness functions can further compare and rank points of different subspaces to a single data point for each fitness function. This dataset is too small to replace a traditional benchmark dataset but small enough to be analyzed in detail by domain experts. Their investigation might result in new insights and provide final confidence in deploying the IDS under analysis.

Please note that none of these steps depends on explicit modeling or knowledge of the IDS under analysis. The dataset space is spanned relative to the system and considered attackers but is invariant for any IDS mitigating these threats. In other words, domain experts never explicitly name edge cases in the IDSs' classifications. Only the optimization in step 3 investigates the IDS and its peculiarities. However, the fitness functions as objective measures ensure the optimizer converges to critical behavior without biases. Furthermore, the optimizer automatically identifies irrelevant parameters and focuses on the remaining relevant for the fitness of the concrete scenarios. Therefore, the generated data points are specific to one IDS, but the methodology presents equal challenges for analyzing competing IDSs.

We intend the steps of this methodology not as a one-time effort but as an iterative process that increases the quality and understanding of the IDS under analysis. The identified weaknesses and highlighted behavior of the IDS to the specific system and attacker behavior require investigation by domain experts who decide on the respective actions. If the identified weaknesses are critical for the use case, the detection mechanism within the IDS needs to be adjusted (e.g., retraining the model or adding a filter). This results in a new IDS prototype that must pass another evaluation, including our methodology. If the spotted points show irrelevant samples of behavior, i.e., should not be part of the dataset space, adjusting the parameter space in step 2 or fitness functions in step 3 can exclude or avoid this behavior in the analysis. Both outcomes and their corresponding changes imply a reiteration of the methodology, yielding different data points and a follow-up investigation.

3.2 Preparation: Find System and Attacker Models

Our methodology spans the space of all datasets using models of the system and the potential attacker. There is no requirement for a specific model notation or level of formalization in these models. A sound engineering process of the system or IDS under analysis should already provide this information and models. Every model is suitable as long as it answers these questions: • What states of operation can the system go through? • What information does the system process in what value ranges? • Which properties of the system need protection against an attacker? • What manipulations can the attacker conduct within the system and its data? • When can an attacker's influence on the system be tolerated, and when can it no longer be ignored?

3.3 Step 1: Partition the Dataset Space

Functional scenarios are the first fixation and partition within the dataset space. Each functional scenario defines the scope for potential system and attacker behaviors and focuses on one particular action within the system and by the attacker. They differ in the system's context, the set of signals processed by the system, or the different strategies of the attacker. We use the term signal to refer to the smallest logical unit of information packed into a message transmitted on the network, e.g., a Boolean flag, a counter, or value, but not individual bits representing them in parts.

A set of functional scenarios ideally covers the system and attacker behavior as extensively as possible and in all relevant aspects. Broad coverage is desirable, as it increases the chances of discovering unanticipated system and attacker behavior later during optimization. We recommend using many smaller subspaces, as they are easier to deduce, analyze, and optimize than a single abstract description of complex behaviors. Combining scenarios and optimizing for the best fitness values over multiple scenarios is possible. For example, our later experiments merge two logical scenarios that only differ in the sign of a single parameter. However, in general, we consider merging scenarios problematic as it may unintentionally add meaningless data points and complicates the exploration task for the optimizer. Only domain experts should explicitly decide after careful consideration about any reduction of the scenario space.

For our methodology, functional scenarios need to compose system and attacker behavior. We propose to independently define the behavior of the system and the attacker and combine them as a cross-product. The deduction of scenarios for the regular system behavior does not differ from the process of scenario-based system testing. Any requirements specification or existing test suites can be used as a foundation to deduce these scenarios.

We propose starting with a broad framing refined later to deduce the attacker behavior as functional scenarios. We recommend analyzing different sets of capabilities, locations, or goals of the attackers and combining them in different functional scenarios as reasonable in the use case under analysis. To refine and translate the attacker behavior into complete descriptions of actual attacks, we suggest using the notion of data or signal changes that cause reactions by the system or change the system state. Any signal, interpreted as binary or scalar, at any interface within the system can change in two directions: (1) increase or (2) decrease in its value. An attacker can use both of these signal changes in two ways: (A) suppressing the propagation of a legitimate change. Hence, the system's state remains and does not adjust to the changing environment. (B) faking a change that factually has not happened. Hence, the system transitions to a new state and does not behave according to the unchanged environment. Therefore, it is sufficient to enumerate all relevant signals in the system and analyze them for all four manipulations of an attacker.

Following these steps results in many functional scenarios, while not all are equally important. Methodologically, it is the best choice to further investigate all functional scenarios in step 2 and 3. Nevertheless, this might not be feasible within the available resources. In that case, we recommend conducting a risk analysis of all these functional scenarios and only continuing our methodology with the most critical scenarios. However, this reduces the size of the resulting benchmark data and threatens the completeness of the analysis. This reduction inhibits the risk of the generated data points barely identifying critical behavior. Hence, adjusting this risk analysis to be more inclusive within the following iteration might be necessary.

3.4 Step 2: Introduce Parameters

Logical scenarios formalize relevant parameters within each functional scenario and define domains for them. These parameters can parameterize the physical properties of the environment or the system, high-level goals, environment behavior, or timings of events. A standard optimizer supports integer and floating-point parameters. For our methodology, these parameters must cover the relevant system parts and variations of the attacker's behavior to enable a diverse dataset. We recommend modeling the attacker behavior as an addition to the system behavior and describing both in isolation. The best method for deducing system parameters depends on the concrete domain. In our view, the first three steps of the well-known category partition method [33] for test suite generation are the most generic approach for ensuring the quality of the coverage. Describing the system's behavior is a well-studied problem, and methodologies and templates exist in various domains, for example, for our use case from the automotive domain [34].

As a starting point for attacker parametrization, we suggest minimal proof-of-concept implementations of the attacks on the system and extending them with parameters on suitable positions. On a high level for attacks, we propose three different categories of parameters: Timing, Payload, and Volume. *The timing of events*, e.g., the time of the first modification of an attacker or the duration of the manipulation. This category aims to identify the most critical point in time for the attack relative to a specific manipulation. Also, the faking and suppression of signals happen related to different events and hence different parameters. Complex interferences require a global timer and might need to encode a temporal order for introducing multiple time parameters depending on each other. Overall, we found it sufficient to specify the absolute time since the beginning of the concrete scenario as a parameter. *The used payload for the manipulation*, e.g., the aberration from the manipulated signal to its original value. This category balances the trade-off between attacks with big and small aberrations. The latter have a smaller but potentially still critical impact and are more challenging to detect by an IDS. *The attack volume*, e.g., the number of manipulated messages per time or the pattern of the manipulation. This category encodes modifications that indirectly make the attack less intrusive in potentially monitored information. These parameters are the most individual to the used attacks and subsume everything that changes the shape of the attack within the observable features. In our case study, we inject or overwrite messages and use parameters to select parameterize patterns.

Like a broad scenario space, broad parameters potentially include more challenges for the analyzed IDS. Higher confidence in the evaluation requires such wide space, especially when it does not yield a critical attack or circumvent the IDS. Nevertheless, the investigated space grows exponentially with the number of parameters, so we again recommend prioritization.

3.5 Step 3: Optimize with Fitness Functions

Any choice of concrete values for the parameters in a logical scenario forms a concrete scenario. It describes a data point in the dataset space that can assess the performance of an IDS. An optimizer with fitness functions enables automatic identification of the most relevant data points in the following way: The fitness function rates the suitability of a data point for a specific purpose and compares it with other data points. The optimizer iteratively samples the parameter space and compares the fitness values of the selected data points for choosing the following candidates. This process repeats until it converges toward the example with the highest rating.

The core of such optimizations is the fitness function, i.e., a measure of the suitability of a concrete scenario for the intended purpose. To create benchmarks for IDS evaluation, we propose six different fitness functions covering six distinct traits of data points: Two fitness functions, f_R (regular) and f_A (attack), measure the functionality of the overall setup. They establish a baseline for reference in the later investigation by domain experts. Four fitness functions, f_{FP}, f_{TN}, f_{FN}, and f_{TP}, reassemble the four classifications of data points in the confusion matrix during evaluation. Instead of approximating each class with percentage scores or accumulating metrics, the optimization yields concrete examples of extreme representatives in each classification. These samples are condensed descriptions of the worst and best performance of the analyzed IDS. Their small number makes a manual assessment by domain experts feasible.

The Optimization Problem: Formally, the optimizer's task is to solve the following problem:

$$\begin{aligned} X = P \times A,\ \ &sim : X \to S \\ F = \{f_R, f_A, f_{FP}, &f_{TN}, f_{FN}, f_{TP}\} \\ \forall f \in F : \min &f(sim(x)) \\ s.t.\ \ &x \in X, \end{aligned}$$

where X is the previously deduced parameters space composed of parameters P, denoting the peaceful system behavior, and parameters A, denoting the attacker behavior. sim describes the execution of the system or a simulation with the given parameters, returning a trace of a concrete scenario s from the corresponding space S. F is the set of our six fitness functions. Please note that this optimization problem includes no explicit constraints except for each parameter being an element of the corresponding domain. Properties observable at runtime (e.g., if the attack was successful) do not exist before the simulation sim. Hence, the fitness functions model these constraints on the observed concrete scenario, and the optimizer considers the constraints implicitly through the fitness value before choosing the subsequent parameters.

Instantiation: Our methodology relies on two functions to be defined individually for each domain and IDS under evaluation: the success of the attack $sam(t)$ and the detailed assessment of the IDS(t).

For the attack's success, we use impacted safety distances as continuous quantification after the attack has started. Observed time spans or distances are suitable candidates for such direct measures. Outside the safety context, as we discuss in detail in Sect. 6.2, alternatives like damage or costs of the attack are promising. $sam(t) = 0$ denotes a successful attack—the beginning of an enforced safety violation. Positive values describe the proximity to a successful attack—a higher safety margin. In other words, the smaller the number, the more significant the impact by the attacker.

The IDS analyses all data within a trace and continuously calculates a suspiciousness score IDS(t) based on the current observations. If desired, IDS(t) can consider a delay for including computational performance and overhead in the analysis. For the analysis, we norm this score such that a value $\geq$1 indicates a yield alarm and 0 denotes behavior that fully complies with the internal model. The range between 0 and 1 denotes more anomalous behavior that is still below the threshold of raising an alarm. If the internal model of the IDS provides higher certainty in an attack beyond the threshold, the score should also exceed 1 correspondingly. Although alarms are binary, it is critical to map the internal model of the IDS to this continuous anomaly score for guiding the optimization.

Building Blocks: Both custom functions describe properties at a given point in time. To describe a data point, we aggregate the entire timespan within the trace and quantify attack success, assessment of the IDS, and timings. Namely, to describe the system behavior, we measure the minimal observed safety margin sam in a concrete scenario s with $saf(s)$. Thus, $saf(s) \leq 0$ denotes a documented safety property violation in the scenario s.

$$saf(s) = \min_t (sam(t))$$

To compare successful attacks, $att(s)$ describes the time passing from the start of an attack to a safety violation in a concrete scenario s, where t_{attack} denotes the point in time of the first manipulation by the attacker.

$$att(s) = \operatorname*{argmin}_t (sam(t) = 0) - t_{attack}$$

To describe the classification of the IDS in more detail, we generalize the IDS assessment IDS(t) such that $sus(s)$ describes the maximum suspiciousness score observed within a concrete scenario s. Complementary to enable minimization by the optimizer, $san(s)$ denotes the sanguineness of the behavior observed in the concrete scenario.

$$\begin{aligned} sus(s) &= \underset{t}{max} (\mathrm{IDS}(t)) \\ san(s) &= 1 - sus(s) \end{aligned}$$

Finally, $rea(s)$ quantifies the reaction time after the IDS raises an alarm and before the attack results in a safety violation. We logically define a successful attack without an alarm as a reaction time 0.

$$rea(s) = \operatorname*{argmin}_t (\mathrm{IDS}(t) \geq 1) - \operatorname*{argmin}_t (sam(t) = 0)$$

Fitness Functions: As proposed by Hauer et al. [32], we build fitness functions following a template of 1) starting with form criteria that assure a desired behavior is part of the concrete scenario and 2) quality criteria that quantify the suitability for the desired purpose of this fitness function. The template manifests in one nested case statement for each criterion in our functions. We intentionally skip domain- and scenario-dependent outer criteria ensuring compliance to form constraints on the scenario first. All σ_n are suitably big constant integers (e.g., $\sigma_1 = 10^2$, $\sigma_2 = 10^4$ in our experiments) to nest the fitness criteria without overlaps from the value ranges of each basic block.

The fitness functions f_R and f_A for establishing the reference baseline are defined as follows:

f_R optimizes for attack impact in the running system without active attack or considering the IDS. This function is identical to classic scenario-based testing and ensures that the system, without interference from the attacker, does not violate the security properties. In our case, f_R determines the minimal safety margin in the logical scenario.

$$f_R(s) = saf(s)$$

f_A optimizes for security violations with an active attacker but without IDS monitoring. It confirms that the attacker's behavior successfully manipulates the system, in our case, violates the safety properties. A successful attacker should significantly reduce the minimal margin measured by optimizing f_R. When the optimized attack does not violate the security properties or is not considered relevant by the domain experts, the attack needs manual refinements in the previous steps of our methodology before continuing.

$$f_A(s) = \begin{cases} saf(s) + \sigma_1 & \text{if } saf(s) > 0 \\ att(s) & \text{else} \end{cases}$$

The fitness functions f_{FP}, f_{TN}, f_{FN} and f_{TP} describe different classifications by the IDS and are defined as follows: f_{FP} and f_{TN} optimize the sanguineness and suspiciousness of the observed system behavior. Most critically, the attack is disabled and not part of the concrete scenario.

$$f_{FP}(s) = san(s)$$
$$f_{TN}(s) = sus(s)$$

The requirement of a successful attack for f_{FN} and f_{TP} is accommodated in the nesting of multiple optimization criteria. First, both functions optimize and ensure a successful attack via $saf(s)$, then their structure differs. To model a critical attack that the IDS does not notice, f_{FN} in the second step optimizes for less suspicious attacker behavior until no alarm is triggered. Finally, it optimizes for the attack with the most immediate success.

$$f_{FN}(s) = \begin{cases} saf(s) + \sigma_2 & \text{if } saf(s) > 0 \\ \begin{cases} sus(s) + \sigma_1 & \text{if } sus(s) \geq 1 \\ att(s) & \text{else} \end{cases} \end{cases}$$

To model a critical attack that showcases the detection abilities of the IDS, f_{TP} in the second step optimizes for a more suspicious attack. Finally, it optimizes for an attack that is ideally immediately detected.

$$f_{TP}(s) = \begin{cases} saf(s) + \sigma_2 & \text{if } saf(s) > 0 \\ \begin{cases} san(s) + \sigma_1 & \text{if } san(s) > 0 \\ att(s) - rea(s) & \text{else} \end{cases} \end{cases}$$

With these fitness functions, an optimizer automatically navigates the generation of concrete scenarios toward particularly relevant data points. However, each elicited point requires a manual investigation in detail, as there are multiple possible outcomes: A) The concrete scenario reveals an undesired behavior of the IDS. The domain experts should report this wrong classification to the IDS developer to improve IDS. B) The observed behavior represents the desired purpose but is neglectable during operation. In this case, each step of our methodology requires adjustments to exclude such scenarios, e.g., by spanning the parameter space differently or adding a form criterion to the fitness function. C) The optimizer has not identified relevant behavior despite exhaustive iterations. This observation indicates that the IDS successfully operates correctly in the modeled context. Nevertheless, the certainty of this conclusion drastically depends on the overall coverage of the dataset space by the investigated scenarios. The first two outcomes result in another iteration of evaluation with a refined setup, and the third outcome results in the deployment of the IDS.

4 Exemplary Application

This section presents a case study analyzing two different IDSs for the CAN bus. It follows the same structure and enumerations as Sect. 3 introducing our methodology.

4.1 Intrusion Detection on the Automotive CAN bus

Building cars underlies high cost-pressure and strict safety requirements. The Controller Area Network (CAN) bus [35] is a cheap, reliable communication medium within cars developed in the early 90s. It is still broadly used in today's cars on the street, although the protocol has no authentication or tamper protection integrated. This lack of security measures opens an attack vector, and potential attacks have been demonstrated and documented in detail [36], [37], [38]. Furthermore, there is competition within academia [39] to identify an approach that increases the security level of existing cars without cryptography but minimal interference, implying an ideal but ambitious challenge for IDSs. Overall this network is well understood and enables our experiments with various existing tools and affordable hardware. In addition, the costs and impacts of attacks and missing or delayed alarms of an IDS are tangible and directly quantifiable due to the safety properties within the system. These properties make the CAN and its potential IDSs an ideal candidate for following our evaluation methodology.

4.2 Preparation: Manipulation of Bus Communication

Our methodology's foundation is eliciting models for the defended system and the attacker. For our exemplary analysis, we focus on aspects that describe manipulations of the CAN bus. Thus, the communication on this bus is the only information utilized by the IDSs within our analysis. We also assume that a vector for the attacker to access the bus exists without further discussion. As this network is not directly accessible from outside a vehicle, this assumption skips other required exploits for a multistep attack to finally gain access to the internal car network, e.g., through the car's infotainment system.

Our case study analyzes the security of openpilot[1], an open source level two [7] advanced driver-assistance system (ADAS), which is our concrete system under consideration. This platform is an upgrade kit for over 150 car models and runs on a smartphone, the EON, mounted under the windshield with an adapter to the internal communication bus of the vehicle. Openpilot provides 1) adaptive cruise control (ACC), keeping the car at a specific speed while slowing down in front of obstacles or vehicles in front, and 2) lane-keeping assistance (LKAS) holding the vehicle in the middle of the lane. To enable further developments and research, all relevant messages from the internal network of supported cars are published and available online[2]. All assistance systems comply with the same high-level architecture of a control loop depicted in Fig. 4. The ADAS receives information about the environment from sensors inside the car and calculates desired maneuvers accordingly. The ADAS transmits these maneuvers as control commands into the network of the car, which realizes the desired steering movements. The new movements result in differing sensor readings, which closes the control loop.

Our attacks on the ADAS tamper with this control loop in the following way: The attacker has gained access to the CAN bus and manipulates the channel transmitting information from the sensors to the ADAS. Thereby, the ADAS receives false signals that deviate from the actual sensing of reality. This wrong information results in a wrong internal model aberrating from reality. Based on this wrong model, the ADAS deduces steering commands and sends them to various actuators via the network. However, due to

1. Available at: https://github.com/commaai/openpilot
2. Available at: https://github.com/commaai/opendbc

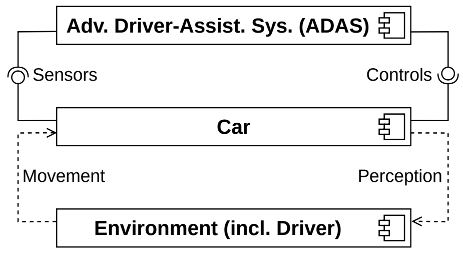

Fig. 4. Control Loop of Advanced Driver-Assistance Systems

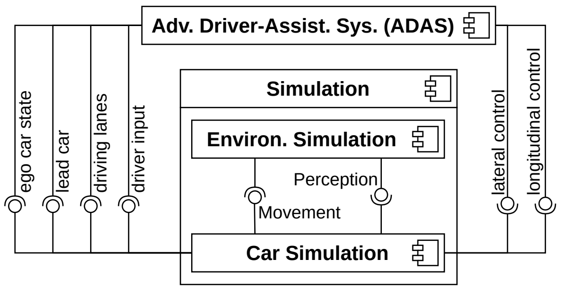

Fig. 5. Detailed Control Loop with the Data Flow of the ADAS plus complementary inputs from the closed loop simulation.

the enforced aberrations of model and reality, these steering commands only constitute safe driving maneuvers relative to the internal model but no longer to reality. Ultimately, this enables the attacker to craft a manipulated model that causes the ADAS to actively violate its safety properties.

To mitigate such attacks, the IDS is attached to the network and monitors the communication. The IDS should raise an alarm for each manipulation by an attacker. As we will discuss in Sect. 5, the time frame to react to these alarms is tight and only within a few seconds. We see two potential reactions: 1) immediately warn the driver to take over the steering while the ADAS turns off gracefully or 2) try to suppress manipulated signals with the car falling back to a fail-safe state. Nevertheless, the elicitation and evaluation of an appropriate reaction is out of the scope of this work.

4.3 Step 1: Deduction of Functional Scenarios

For the scenarios without attack, we use existing work about the test-suite generation for testing the regular behavior of autonomous vehicles [34]. According to the functionality of the ADAS under analysis, the functional scenarios are driving on a track without traffic, a vehicle appearing in front of the ego vehicle, and the user changing the operation mode of the ADAS. In each scenario, the ADAS is engaged. Thus, the ACC keeps a specific speed, and the LKAS keeps the car on the lane. This selection of scenarios is simplified for our case study but could be further refined and extended. For the attacks, we use the attacker model also used to develop the IDS in our case study. An attacker has gained access to the internal CAN bus through an additional device attached to the bus (e.g., via the OBD connection) or hijacking an ECU (e.g., via a bug in the infotainment system). Manipulation is possible by injecting new messages or as a man-in-the-middle overriding existing information. In the system model, the attacker manipulates the signals sent from the sensors to the ADAS. In our methodology, we consider each signal twice: 1) with the suppression of legitimate changes and 2) with the injection of fake changes. Figure 5 depicts the entire data flow of our exemplary ADAS, and we selected the three sensor inputs realized on the CAN bus for our further analyses. The lane detection uses a separate camera.

Following our deduction of attack scenarios, an attacker can manipulate each signal in four different ways. These ports transmitting six signals results in 24 different attacker scenarios depicted in Tab. 1. Each field of the table names the direct impact of the manipulation and classifies its overall consequence in isolation. For example, the ADAS driving at a higher speed than desired might confuse the driver. However, the operation at this higher speed is still within the operational safety of the ADAS. An ADAS not engaging or turning off might irritate and annoy the driver. However, it is not a safety problem on autonomous driving level two, as in this level, the driver still needs to keep his hand at the steering wheel and monitor the driving all the time. Hence, these attacks have lower priority than others, which might lead to collisions or the car leaving the lane.

Finally, our methodology forms a catalog of all scenarios as the cross-product of (1) all scenarios of regular system operation and (2) all manipulations of an attacker. To showcase regular system behavior, we considered a simplistic set of variations of a) straight or single curve tracks and b) a car in front or behind—resulting in four distinct functional scenarios without attack. Overall this yields 24 times 4 equals 96 functional scenarios to analyze further in the following step. Comprehensively analyzing all these scenarios requires years of total system runtime in the optimizer and is, therefore, not feasible for automotive manufacturers and is out of this paper's scope. We needed detailed domain knowledge about each scenario to prioritize them by the relevance for the IDS usage. Therefore, we conducted experiments about the functional scenarios marked with α and β in Tab. 1. We used a real car and documented our setup in Appendix B.

4.4 Step 2: Logical Scenarios and Parameterizing

In the following steps, we only focus on and discuss the manipulation of the steering angle (the scenarios marked with α in Tab. 1). We found this manipulation to be gradual and the least noticeable by a human driver, i.e., to have the highest need for an alarm by an IDS. The driver or ADAS moving the steering wheel to the left or right results in an increased or decreased angle broadcasted by the steering wheel sensors. The ADAS uses this signal in a feedback loop: For keeping the lane or driving a curve, it calculates a suitable angle and triggers motors to apply slight torque on the wheel for turning it. If the sensor messages report the approximation of the desired angle, the ADAS lowers this torque; if the divergence does not shrink, the torque is increased based on an internal model. The corresponding attack is the following: An attacker fakes the sensor signal, reporting that the steering wheel is off to the left of its actual angle. This results in the ADAS requesting a movement of the steering wheel to the opposite, right side, supposedly to keep the car on the track. In reality, this movement pushes the steering wheel to the left and out of the angle that matches the desired trajectory. Consequently, this manipulation causes the car to unintentionally leave the lane on the left. An identical manipulation to the right causes mirrored behavior.

TABLE 1
Each cell denotes a functional attack scenario that is the result of a manipulation of the signal (row) in the given way (column) and holding all other factors fixed. Information in brackets refers to the potential consequences of this attack.

		suppress legitimate changes		inject faked changes	
		increase	**decrease**	**increase**	**decrease**
ego car state	**current speed**	car drives too fast (confusion)	car stops (rear collision)	car stops (rear collision)	car drives too fast (confusion)
	steering angle	increased steering angle (leave lane)$^{\alpha}$	counter-steering (leave lane)$^{\alpha}$	counter-steering (leave lane)$^{\alpha'}$	increased steering angle (leave lane)$^{\alpha'}$
lead car information	**relative velocity**	car stops (rear collision)	crash into slower car (frontal collision)$^{\beta}$	crash into slower car (frontal collision)$^{\beta}$	car stops (rear collision)
	distance to lead car	car stops or drives too slow (rear collision)	crash into slower car (frontal collision)$^{\beta}$	crash into slower car (frontal collision)$^{\beta}$	car stops or drives too slow (rear collision)
user input	**engaged**	ADAS stays inactive (irritation)	ADAS stays active (confusion)	ADAS activates unexpectedly (confusion)	ADAS deactivates unnoticed (confus., leave lan.)
	target speed	car will not accelerate (irritation)	car will not decelerate (confusion)	car suddenly accelerates (confusion)	car suddenly decelerates (rear collision)

We elicited reasonable variables and domains based on our experiments within the real car. For this attack, we describe the logical scenario with the parameter space $X = P \times A$ as follows:

$$\begin{aligned} P =& [32, 33, \ldots, 96]\, km/h \times \{\text{straight}\} \cup \\ & [32, 33, \ldots, 72]\, km/h \times \{\text{curve}\} \\ A =& [0, 30]\, s \ \times [0, 1, \ldots, 2095] \cdot 0.0573^{\circ} \times \\ & [0, 1, \ldots, 5]\,\text{messages} \ \times (\{0\} \times \{\text{replace}\} \cup \\ & [1, 2, \ldots, 19] \cdot 0.05\ \text{T} \times \{\text{inject}\}) \end{aligned}$$

Set P describes the peaceful operation of the car. The parameters are used the same as for testing of the ADAS: *the target driving speed* as an integer of km/h (transmitted over the bus) and *the curvature of the street* (set in the simulation software). For simplicity, we chose two setups: driving autonomously, with speeds from 32 km/h to 96 km/h on a straight street and speeds from 32 km/h to 72 km/h in a slight curve. Set A is the parametrization of the attacker and follows our methodology on timing, payload, and volume. The time is a rational number in seconds after the concrete scenario starts denoting *the point when the attacker manipulates the first message*. This single time generically specifies the attack's timing without considering any system behavior within the scenario. However, during space exploration, the optimizer will adjust this parameter without manual modeling to the most critical point, e.g., when the car reaches the curve for all driving speeds and road setups. The payload for this attack is *the aberration from the actual steering signal* from 0° to 120° in steps of 0.0573°. This step size is the equivalent of changing one bit in the fixed-point representation of the steering angle signal. The attack volume is represented by one parameter for replacement attacks and an additional parameter for injection attacks. The first and common parameter is an integer that denotes *the manipulation of only every n-th original message*. The additional parameter is the time as a floating-point number denoting *the delay of the injected message from the original message* transmitting the signal. Such manipulation results in an alternating sequence of values, mixing actual and manipulated signals. For injections, any fake message received after an actual message overrides the internal state until the next actual message is received. We split the regular time interval T between two messages into 20 spans, and the parameter selects a multiple. A short delay causes the fake information to be dominant most of the time for decisions of the ADAS. In particular, this dominance unfolds if the attacker injects directly after consecutive messages. Nevertheless, this type of manipulation is more obvious to detect by potential IDSs.

4.5 Step 3: Instantiating the Fitness Functions

In the final step, we need concrete implementations for $sam(t)$ for each logical scenario and IDS(t) for each IDS under analysis. Please note that this is the only time our methodology considers the IDSs under analysis. In our use case, $sam(t)$ is defined via the physical properties of the car and corresponding safety margins. In scenarios with the risk of front and rear collisions, we define $sam(t)$ as the distance to the safety margin where a human driver can still avoid a collision. In scenarios with the risk of leaving the lane, we define $sam(t)$ either as (1) the distance of the lane marking to the wheels to include weaker attacks or (2) the distance of the lane marking to the center of the car to narrow the analysis to critical attacks. Scenarios involving the attacker's impact on the driver are out of the scope of this paper, but, for example, approximations of human reaction times and awareness levels are promising definitions of $sam(t)$.

For our experiments we analyzed two state-of-the-art IDSs, IDS$_T$ proposed by Taylor et al. [40] and an IDS$_M$ by Marchetti et al. [41]. IDS$_T$ uses Long Short-Term Memory (LSTM) networks processing blocks of 20 consecutive messages on the network of the same signal to predict the following message to be transmitted. It calculates a loss and anomaly signal based on the difference to the actual message received next. An anomaly signal equal to 0 denotes a total match with the prediction and, thereby, normal behavior. In contrast, higher numbers denote more unanticipated, anomalous behavior and, thereby, a higher likelihood of an ongoing attack. For IDS$_T$, we normalized the anomaly signal's value at the threshold for an alarm. IDS$_M$ monitors the header entropy yielded from all arbitration IDs within non-overlapping time windows. The model describes an attack with the observed entropy being outside an interval of plus/minus a fixed number of standard deviations around the mean observed entropy in the training set. For IDS$_M$, we normalized this interval through the distance from the middle of the interval and the closest side.

5 Experiments and Analysis

5.1 Setup and Implementation

Enumeration of Concrete Scenarios: For our experiments, we modeled the two similar logical scenarios (see Tab. 1 marked with α') about fake changes in the steering angle with parameters elaborated in Sect. 4.4. For a more comprehensive discussion, we optimized both tracks (straight and curve) and attackers (replace and inject) individually. Combining both tracks and attackers in a single optimization would be possible and would quarter the execution time and manual analyses. However, a combined analysis would conclude only in the single data point with the highest fitness value that is challenging to comprehend in isolation and without considering other scenarios.

In total, we ran the following optimizations: F_R only depends on the system and is optimized twice - once for both tracks. F_A depends on the system and attacker but not on the IDS. Therefore, we optimized it four times - once for both attacks on both tracks. The other four fitness functions depend on the IDS. Consequently, we optimized once on both tracks and with the attack corresponding to the IDS. This setup totals 22 optimizations. To conduct the experiments, we built a Hardware-In-The-Loop (HIL) simulator (see Appendix A). While aiming for approximately a day of execution time per optimization, we iterated over 1000 concrete scenarios in each optimization. Each concrete scenario, on average, constitutes 30 seconds of driving. This schedule roughly accumulates to eight days of specific driving behavior and requires a month of execution time.

Prepartion of the IDSs: The IDS configurations and internal models are constant in all optimizations. We trained both candidate IDSs with the same traces of regular behavior recorded from our HIL. We collected them manually and separately from all optimization. The training dataset contains traces from various speeds and six hours of driving on both tracks in our setup. We followed the instructions for parameter-tuning and optimization listed in the publication introducing these IDSs.

Optimizer: The optimizer chooses and schedules the next concrete scenario automatically based on ratings of the fitness function of previous populations and an initial population obtained with randomly selected parameters. Our implementation is based on the Optuna optimizer framework [42] and used the Tree-structured Parzen Estimator algorithm in combination with a median pruning algorithm [43], [44]. The optimizer grows an independent population for each logical scenario, each IDS, and each fitness function without considering the other optimizations. The optimizer internally calculates the correlation of each parameter with the obtained fitness values. After each optimization, we manually investigate the traces with the highest fitness value and elaborate our findings in the following.

5.2 Experimental Results

Since the system's behavior and the IDS's classifications are highly multidimensional, we could not find any beneficial way to visualize them to comprehend the manual interpretation of individual data points. Concretely, each parameter in each scenario forms one dimension, and multiple parameters might correlate with specific classifications of the IDS and particular events within the scenarios. We directly tailored our methodology for deducing scenarios and optimization with different fitness functions to a small set of data points that can be manually investigated and interpreted at low costs to approach this complexity. Therefore, in the following, we analyze the recordings of the concrete scenarios with domain knowledge.

Baseline: f_R: All trials confirm that the operation of the ADAS on our tracks within our chosen parameters is safe. The car does not leave the lane, and the car's center stays at least 1.6 meters away from the lane marking. f_A: The optimization yielded numerous traces with safety violations forced by the attacker. Every angle above 46.0° malicious aberration leads to the fastest discovered attacks, needing around 4.0 seconds and 79 manipulated messages before forcing leaving the lane. The most important hyperparameter is the manipulated angle, and the parameters driving speed and timing of the attack are only of minor importance. Injection attacks are the more effective, the shorter after the original the attacker injects the manipulated message. With minimal delays, the manipulated information stays dominant most of the time.

IDS$_T$: Regular system behavior is classified correctly. However, the value of f_{FP} drops by 0.2 points in the classification certainty when the car drives below 44 km/h. We see a link to some minor swinging within the lane in our simulation at these lower speeds. Optimizing f_{TN} confirms this since an operation with higher speeds leads to more certain classification, at best at the top speed of 96 km/h. The analysis with f_{TP} shows that the classification of IDS$_T$ depends on the steering aberration, and an angle between 97.0° and 106.8° has the fastest reaction time of 0.36 seconds. Nevertheless, f_{FN} reveals undetected attacks with smaller aberrations, the worst attack with an angle of 69.0°.

IDS$_M$: Regular system behavior is classified correctly. f_{FP} and f_{TN} show no impact between any of the available parameters changing the system behavior and the detection ability of IDS$_M$. All generated scenarios were equally not considered suspicious, with an almost identical rating. An optimization of f_{FN} does not identify an undetected attack, although the optimizer tried various injection timings and patterns. The optimized f_{TP} converges closely to 0.5 seconds and documents that the window size (configured to 0.5 seconds) is a lower boundary for the detection.

5.3 Into the Next Iteration

After the domain experts have analyzed the optimized traces, our methodology supports the deployment decision or guides the next iteration in the development process. In general and as discussed in Sect. 3.5, there are three possible outcomes (we identify them as A, B, and C):

IDS$_T$ is an example of outcome A, an IDS suffering from a critical and relevant vulnerability revealed by the optimization. Depending on the internal detection model and the determined threshold, manipulations with a particular steering angle exist that do not yield an alarm but cause a safety violation. The optimization with f_{FN} extends towards that edge point and converges at this particular angle. However, a different configuration of IDS$_T$ classifies this particular generated data point correctly. Yet, a new optimization of f_{FN} for this IDS configuration results in a different data point that documents this weakness again with a different steering angle. In addition, the optimization with f_{FP} guided the optimization to a set of parameters that correlate with stronger steering actions by the ADAS, most likely caused by a small aberration from the internal car model and the actual steering behavior. Although none of the generated samples results in a false positive classification, we see them as an indicator for another problem with an adjusted IDS model or in further experiments in different scenarios. Our simple track is insufficient to challenge correct IDS classifications during more complex steering maneuvers potentially described by refinements of the parametrization. These findings indicate that IDS$_T$ needs adjustments or extensions to fully protect an autonomous car.

IDS$_M$ is an example of outcome C, an IDS passing all tests generated by the optimization in our parameter space. However, this finding alone is no proof for the security of the IDS and the absence of any false classification, but only relative to the spanned and examined parameter space. The comparison of the four IDS-specific fitness functions with the baseline reveals little impact of all parameters provided to the optimizer despite a broad probing of various combinations. Such evaluation yields two questions for the domain experts: (1) Is this entire space parameter space realistic and representative, and (2) is the gained safety through the IDS sufficient? A real-world setup would require extending the parameter space and analyzing more functional scenarios. Different parametrizations describe more complex patterns of attacker timing, and the regular system behavior in our scenarios always has a perfectly deterministic communication pattern with a constant busload. Nevertheless, our optimizations already provide an approximation for the remaining safety margin. Especially, f_{TP} showed that the window size within the detection model of IDS$_M$ is the minimal delay of a potential alarm. Hence, small window sizes are preferable if the detection abilities remain high. Suppose domain experts consider both properties in question as sufficiently fulfilled. In that case, our evaluation provides a witness in support of the deployment decision of an IDS configuration.

The deductions only give the idea of further steps, but we see investigating them more closely out of the scope of this publication. Refining or extending IDS$_T$, suggesting a new idea with an evaluation showing the improved security, is a new distinct paper. In the case of IDS$_M$, we expect such deeper analysis to overall support the functionality of the IDS. But more importantly, such investigation would produce valuable insights about handling the dataset space in our evaluation framework. We elaborate on possible aspects to investigate the dataset space more deeply in Sect. 6.2.

5.4 Comparison with Benchmark Datasets

The datasets generated using our methodology drastically differ from those used for the original IDS validation. Our insights are not elaborated nor mentioned in the original papers. Similar data points were not present or not noticed in the dataset. All our optimized data points are not edge cases within our parameter space that could be determined or anticipated upfront, especially if the traits and quality of the evaluated IDS are not known upfront. All this combined showcases the need for an additional evaluation considering the concrete IDSs under analysis.

Moreover, our generated datasets document that a dataset optimized for challenging an IDS cannot evaluate other IDSs than that for which it has been generated. In particular, on the one hand, the dataset generated for IDS$_M$ does contain a broad set, probing all parameter combinations. Nevertheless, the critical points for the classification behavior for IDS$_T$ were omitted in all explored concrete scenarios. Thus, an evaluation of IDS$_T$ with the dataset of IDS$_M$ results in a wrong evaluation that misses the false negatives completely. On the other hand, the dataset optimized for IDS$_T$ only depicts focused behavior, mostly varying the payload of the attack. Furthermore, the data points with the strongest misclassification depend on the concrete internal detection model. Moreover, various other parameter combinations documenting an accurate functionality of a different detection feature, e.g., different timings as for IDS$_M$, are not provided by a dataset generated with a specific set of attacks in mind. Thus, an evaluation of IDS$_M$ with the dataset optimized for IDS$_T$ results in an evaluation with neither false negatives nor false positives but with much less actual confidence as desirable in this assessment.

5.5 Threats on Implementation

We require executable and functional implementations of the IDS under analysis for our optimizations. As the original authors did not publish their code or models, we reimplemented their algorithms and trained them on data collected from our setup. We cannot verify if and how much our IDSs differ from the original publications. As our research focuses on a methodology for the IDS evaluation in general, using slightly diverging implementations of the IDS is not problematic. However, we want to point out that a different implementation or training might not result in the identical weaknesses of the IDSs we documented.

In addition, using a HIL for our simulation reduces the realism of the observable traces. However, due to safety and monetary restrictions, repeatedly attacking a real car during operations at high speed is not feasible. Therefore, we conducted selected concrete scenarios at low speed in a real car, validated our attacks and simulation, and compared the collected traces to ensure high similarity (see Appendix B). Another factor is the imperfect determinism of our simulation. Although the car operates in an identical environment among different runs, the network communication differs in timing and, thereby, in minorly different payloads. We believe hitting identical timings is impossible without a full simulation of all hardware, and we intentionally decided against that. Nevertheless, in multiple executions with the same parameters, we observed only differences within milliseconds of the otherwise identical behavior. We use an optimizer that can work with noisy fitness values and repeats identical parameters if they seem promising to show the optimal behavior.

6 DISCUSSION

6.1 Implications on IDS Evaluation

The evaluation of IDSs requires the analysis of an exemplary dataset as a benchmark for their performance. However, our results indicate that upfront generated datasets do not contain the data points that especially showcase all strengths and weaknesses of all IDSs they will evaluate. This shortcoming emerges from two aspects inherently missing in these datasets: (1) normal behavior of the system that is not edge case behavior for the system itself but might still be misleading for the classification by the specific detection model in the IDS, resulting in undocumented false positives, and (2) attacks with modifications within the attacker's abilities actively trying to avoid and hide from the concrete detection model in IDS while an unaltered attack is detected, resulting in undocumented false negatives. Both aspects are relative to the IDSs under analysis and, in our view, must be reflected in the evaluation. Massively increasing the size and diversity of a fixed dataset reduces the risk of missing these critical points, as they might be part of the dataset by chance. But most severely, even if the meaningful traces would all exist within a large benchmark dataset, domain experts might not notice them in the multitude of investigated classifications and the finally summed scores. Considering the exponential growth of the dataset space with the number of parameters, universal, fixed datasets are not feasible and waste resources by evaluating data points without particular meaning for the IDS under analysis.

Therefore, we argue for a cultural change in creating, publishing, and sharing benchmark datasets. A good benchmark dataset must provide means to adjust directly to the IDS under evaluation and structured information to further extend them with data points after publication. For our methodology, this information is the deduction that spans the dataset space corresponding to the dataset, including the parameter domains and a simulation environment. Ideally, a standardized and reusable format or universal catalog of guided generation methods scope these confinements .

Finally, we highlight that domain experts are central to the evaluation process. In our experiments, we spotted attacks not detected by both IDSs under evaluation and still cause safety-relevant effects on the overall system. An example of such an attack is a MitM-attacker transmitting the messages regularly but freezing the current steering angle. In sophisticated evaluations covering large dataset spaces, ideally fulfilling some completeness criteria, undesired behavior might always exist. Nevertheless, not the entire space or shown edge case behavior is critical for the system's daily operation and requires a correct classification by an IDS. After a deeper analysis of the generated system data points, domain experts might conclude that all these remaining flaws in the IDS are acceptable and form a reasonable trade-off between security and efforts for mitigation. Our optimization-based methodology, or more specifically, the fitness functions, can be adjusted to exclude data points that are not considered relevant and explore the remaining parameter space for other critical behavior. In our view, it is impossible to fully investigate the whole dataset space and demonstrate an IDS is secure. Hence, the goal should be to support the trade-off for a reasonably secure system with sufficient evidence. With our methodology, we aim to focus the attention of the domain experts on the most relevant data points.

6.2 Future Work

We see a direct potential to extend our methodology by including the reaction to alarms. Our evaluation approach analyses the functionality of the system and IDS as a whole and could cover automatic reactions to all alarms seamlessly. Suitable reactions, especially in safety-critical environments, are a different field of research and, therefore, out of the scope of this work.

As an important next step, we consider extending our methodology for space exploration of potential datasets beyond network IDSs and the domain of safety-critical cyber-physical systems. Namely, two core points need to be adjusted to generalize our methodology: (1) The schema we used for scenario deduction needs to be transferred to other domains, and (2) The building block $sam(t)$ needs to be generalized. About (1): For deducing our methodology, we followed the steps of scenario-based testing, which has been developed for the automotive domain and adopted for other use cases of cyber-physical systems. However, scenario-based testing is a specific functional testing strategy and a refinement of the equivalence class testing concept. The choice of the optimal test strategy depends on the system under analysis. Especially, concrete attack implementations are domain-specific and thus require further analysis and other technical implementations that we see out of scope from this paper. The scope of penetration testing approaches and tools is broad [45], and we see potential instrumentation in all of them. About (2): We use $sam(t)$ to quantify the impact and success of an attack. We rely on quantifying safety violations as they are measurable directly and provide a fine-grained differentiation between each manipulation. Although this, for the moment, is restricting the number of use cases for IDSs, we are confident that an adaption of the fitness functions for general application is possible in the future. Estimating costs caused by a successful attack or a false positive has been proposed for evaluating IDS for over two decades [46]. Replacing the safety margin with costs in our fitness function seems the most promising. Still, it requires an additional step of indirection to estimate the costs and should be evaluated further on another use case.

Furthermore, we see the need to broaden the knowledge of the space of potential benchmark datasets. With the setup of our experiments, we focused on maximal realism and validity using the realistic simulation setup and the confirmations from inside the car. Future research can implement our methodology with less realistic but faster simulation tools. Such faster tools enable more iterations and a deeper dataset space analysis than we have done.

We focused solely on the generation and augmentation of datasets to evaluate IDSs and separated these efforts strictly from IDS development and tuning. However, these two activities cannot be separated, as the wrong-classified samples yielded by our methodology are valuable inputs for refining IDSs. Future work can explore these research challenges. For example, the IDS developers could simply add these data points to their training dataset for a machine learning-based IDS.

6.3 Threats to Validity

The main threat to the validity of our methodology and, overall, the generated data points lies within the dependency on the appropriate deduction and modeling of the dataset space. Describing and parameterizing the regular system operation in isolation is as challenging as deducing complete test suites for the system. In addition, identifying and describing potential attacks on this system behavior is the core of penetration testing. After documenting such an attack, additional parametrization of the attacker's behavior is necessary. Too complex attacks or attacks without functional variations will result in much longer optimization times and the optimizer only spotting exactly the prepared attack. Conversely, too simplified scenario spaces will not contain exploits to an IDS and result in wrong confidence. Despite the weaknesses we spotted, we know little about the relation between our data points and the overall dataset space.

In our analysis, we chose a scenario space that, based on our experience, is around the ideal area between simple and complex. Our methodology can encode more complex attacks and combinations than we did, but our scenarios already discovered successful attacks. Furthermore, the sophisticated generation of attacker behavior is not the focal point of this paper. Choosing a more straightforward attacker behavior simplifies the optimization and reduces the required runtime. Hence, future research should investigate more sophisticated attack generation approaches while spanning the dataset space and repeat our dataset comparison on the new attack samples. However, we expect these data points to show more extreme behavior and support our findings even more.

Finally, we only conducted the experiments within a single use case related to network intrusions. However, there are generally various use cases and approaches for IDSs. We expect the evaluation of host-based IDSs to show the most extensive divergence from our findings. Although we aim to better understand the ground nature of evaluating all IDSs, all experiments can only consider specific IDSs for specific use cases. Therefore, our findings yield the necessity of further investigations. Nonetheless, it will remain impossible to generally prove the properties of all the potential spaces for datasets for all IDSs universally.

7 Conclusion

This paper critically reflects the process of creating benchmark datasets independent of the IDSs in the evaluation. We chose a previously well-studied use case for intrusion detection on automotive networks for our investigation. Based on successful attacks against an open-source ADAS, we used scenario-based optimization to systematically explore the space of all possible data points in benchmark datasets. Using six fitness functions that quantify the suitability of a data point for the evaluation, we generated system and attacker behavior that especially showcase edge cases for the classifications of the IDS under analysis.

Analyzing the so-generated confined dataset, we found that only a tiny part of the edge case classifications are covered when the system and attacker are considered in isolation of the IDS while collecting a benchmark dataset. Utilizing our methodology, we identified scenarios demonstrating classification behavior for each IDS that previous datasets have not shown and that the original evaluations of the two IDSs have not touched. Most importantly, these data samples are specific for each IDS and show almost no intersection. In other words, a dataset suitable for a sound benchmark of one of the IDSs is not beneficial for analyzing another.

Our finding questions the practice of using static, universal benchmark datasets for evaluating an IDS suggested afterward. Even in identical systems and attacker models, all IDSs need to be evaluated with data points specifically reflecting the properties of the IDS' detection model. Only these specific data points can emphasize edge case behavior and unique strengths of each candidate IDS under analysis missing in static datasets generated upfront. Consequently, invariant benchmark datasets are not sufficient to compare IDSs. Instead, the evaluation should follow a fixed methodology, considering each IDS individually to create tailored dataset points. Our blueprint for scenario-based dataset generation is a step toward a new culture for benchmark datasets and a well-founded evaluation methodology providing final confidence in deploying an IDS.

References

[1] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," *NIST special publication*, 2007.

[2] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, 2013.

[3] A. Cárdenas, J. Baras, and K. Seamon, "A framework for the evaluation of intrusion detection systems," in *IEEE Symposium on Security and Privacy*, 2006.

[4] N. Stakhanova and A. A. Cardenas, "Analysis of Metrics for Classification Accuracy in Intrusion Detection," in *Empirical Research for Software Security: Foundations and Experience*, 2017.

[5] E. Aliwa, O. Rana, C. Perera, and P. Burnap, "Cyberattacks and countermeasures for in-vehicle networks," *ACM Comput. Surv.*, 2021.

[6] O. Y. Al-Jarrah, C. Maple, M. Dianati, D. Oxtoby, and A. Mouzakitis, "Intrusion detection systems for intra-vehicle networks: A review," *IEEE Access*, 2019.

[7] SAE International, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," 2021.

[8] M. Kuhn and K. Johnson, *Applied predictive modeling*. Springer New York, 2013.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *International Conference on Learning Representations*, 2014.

[10] A. Warzyński and G. Kołaczek, "Intrusion detection systems vulnerability on adversarial examples," in *Innovations in Intelligent Systems and Applications*, 2018.

[11] M. Pujari, Y. Pacheco, B. Cherukuri, and W. Sun, "A comparative study on the impact of adversarial machine learning attacks on contemporary intrusion detection datasets," *SN Computer Science*, 2022.

[12] H. Qiu, T. Dong, T. Zhang, J. Lu, G. Memmi, and M. Qiu, "Adversarial attacks against network intrusion detection in iot systems," *IEEE Internet of Things Journal*, 2021.

[13] M. J. Hashemi, G. Cusack, and E. Keller, "Towards evaluation of NIDSs in adversarial setting," in *CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, 2019.

[14] R. Lippmann, R. K. Cunningham, D. J. Fried, I. Graf, K. R. Kendall, S. E. Webster, and M. A. Zissman, "Results of the DARPA 1998 offline intrusion detection evaluation," in *Recent Advances in Intrusion Detection*, 1999.

[15] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Trans. Inf. Syst. Secur.*, 2000.

[16] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for network anomaly detection," in *Recent Advances in Intrusion Detection*, 2003.

[17] R. Singh, H. Kumar, and R. Singla, "A reference dataset for network traffic activity based intrusion detection system," *International Journal Of Computer Communications & Control*, 2015.

[18] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Military Communications and Information Systems Conference*, 2015.

[19] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón, "UGR'16: A new dataset for the evaluation of cyclostationarity-based network idss," *Computers & Security*, 2018.

[20] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Flow-based benchmark data sets for intrusion detection," in *European Conference on Cyber Warfare and Security*, 2017.
[21] I. Sharafaldin, A. Habibi Lashkari., and A. A. Ghorbani., "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *International Conference on Information Systems Security and Privacy*, 2018.
[22] C. Miller and C. Valasek, "Adventures in automotive networks and control units," *Def Con*, 2013.
[23] H. Lee, S. H. Jeong, and H. K. Kim, "OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame," in *Privacy, Security and Trust*, 2017.
[24] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "CANet: An unsupervised intrusion detection system for high dimensional CAN bus data," *IEEE Access*, 2020.
[25] S. S. Gopalan, D. Ravikumar, D. Linekar, A. Raza, and M. Hasib, "Balancing approaches towards ml for ids: A survey for the cse-cic ids dataset," in *International Conference on Communications, Signal Processing, and their Applications*, 2021.
[26] S. Sapre, K. Islam, and P. Ahmadi, "A comprehensive data sampling analysis applied to the classification of rare iot network intrusion types," in *IEEE 18th Annual Consumer Communications & Networking Conference*, 2021.
[27] G. Apruzzese, L. Pajola, and M. Conti, "The cross-evaluation of machine learning-based network intrusion detection systems," *IEEE Transactions on Network and Service Management*, 2022.
[28] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving," in *IEEE 18th International Conference on Intelligent Transportation Systems*, 2015.
[29] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *IEEE Intelligent Vehicles Symposium*, 2018.
[30] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, 2016.
[31] G. E. Mullins, P. G. Stankiewicz, and S. K. Gupta, "Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles," in *IEEE International Conference on Robotics and Automation*, 2017.
[32] F. Hauer, A. Pretschner, and B. Holzmüller, "Fitness functions for testing automated and autonomous driving systems," in *Computer Safety, Reliability, and Security*, 2019.
[33] T. J. Ostrand and M. J. Balcer, "The category-partition method for specifying and generating fuctional tests," *Commun. ACM*, 1988.
[34] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, 2020.
[35] Robert Bosch GmbH, "CAN specification," 1991.
[36] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Snachám, and S. Savage, "Experimental security analysis of a modern automobile," in *IEEE Symposium on Security and Privacy*, 2010.
[37] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *USENIX Conference on Security*, 2011.
[38] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, 2015.
[39] M. Bozdal, M. Samie, S. Aslam, and I. Jennions, "Evaluation of can bus security challenges," *Sensors*, 2020.
[40] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *IEEE International Conference on Data Science and Advanced Analytics*, 2016.
[41] M. Marchetti, D. Stabili, A. Guido, and M. Colajanni, "Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms," in *IEEE International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow*, 2016.
[42] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *International Conference on Knowledge Discovery and Data Mining*, 2019.
[43] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *International Conference on Neural Information Processing Systems*, 2011.
[44] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *International Conference on International Conference on Machine Learning*, 2013.
[45] D. D. Bertoglio and A. F. Zorzo, "Overview and open issues on penetration test," *Journal of the Brazilian Computer Society*, 2017.
[46] J. Gaffney and J. Ulvila, "Evaluation of intrusion detectors: a decision theory approach," in *IEEE Symposium on Security and Privacy*, 2001.
[47] T. Hutzelmann, D. Mauksch, and A. Pretschner, "How to conduct experiments with a real car? experiences and practical guidelines," in *Communications in Computer and Information Science*, 2020.

Thomas Hutzelmann is currently a research associate at the Chair of Software and Systems Engineering at the Technical University of Munich at the School of Computation, Information and Technology. His current research focuses on making security properties quantifiable and measurable, especially for intrusion detection systems.

Dominik Mauksch is currently working as a penetration tester and security engineer in the automotive industry. This work was done during his stay at the Technical University of Munich, where he received the M.Sc. degree in Automotive Software Engineering. His research interests include the security testing of embedded, safety-critical systems and test automation techniques for CPSs.

Ana Petrovska is a research associate at the Chair of Software and Systems Engineering at the Technical University of Munich at the School of Computation, Information and Technology. Her research focuses on the foundations of system adaptation and self-adaptive systems, and knowledge representation and reasoning under uncertainties in CPSs.

Alexander Pretschner is head of the Chair of Software and Systems Engineering at the School of Computation, Information and Technology at the Technical University Munich; the scientific director of fortiss, the research institute of the Free State of Bavaria for Software-Intensive Systems; and the speaker of the board of directors of bidt, the Bavarian Research Institute for Digital Transformation.