



## **Application-Aware Intrusion Detection: A Systematic Literature Review, Implications for Automotive Systems, and Applicability of AutoML**

Downloaded from: <https://research.chalmers.se>, 2026-05-14 10:19 UTC

Citation for the original published paper (version of record):

Schubert, D., Eikerling, H., Holtmann, J. (2021). Application-Aware Intrusion Detection: A Systematic Literature Review, Implications for Automotive Systems, and Applicability of AutoML. *Frontiers in Computer Science*, 3. <http://dx.doi.org/10.3389/fcomp.2021.567873>

N.B. When citing this work, cite the original published paper.



# Application-Aware Intrusion Detection: A Systematic Literature Review, Implications for Automotive Systems, and Applicability of AutoML

David Schubert<sup>1\*</sup>, Hendrik Eikerling<sup>1</sup> and Jörg Holtmann<sup>2</sup>

<sup>1</sup>Software Engineering and IT Security, Fraunhofer IEM, Paderborn, Germany, <sup>2</sup>Software Engineering Division, Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden

## OPEN ACCESS

### Edited by:

Jan Pelzl,  
Hamm-Lippstadt University of Applied  
Sciences, Germany

### Reviewed by:

Kazuhiro Ogata,  
Japan Advanced Institute of Science  
and Technology, Japan  
Emilio Coppa,  
Sapienza University of Rome, Italy

### \*Correspondence:

David Schubert  
david.schubert@iem.fraunhofer.de

### Specialty section:

This article was submitted to  
Computer Security,  
a section of the journal  
Frontiers in Computer Science

**Received:** 30 May 2020

**Accepted:** 26 July 2021

**Published:** 24 August 2021

### Citation:

Schubert D, Eikerling H and  
Holtmann J (2021) Application-Aware  
Intrusion Detection: A Systematic  
Literature Review, Implications for  
Automotive Systems, and Applicability  
of AutoML.  
*Front. Comput. Sci.* 3:567873.  
doi: 10.3389/fcomp.2021.567873

Modern and flexible application-level software platforms increase the attack surface of connected vehicles and thereby require automotive engineers to adopt additional security control techniques. These techniques encompass host-based intrusion detection systems (HIDSs) that detect suspicious activities in application contexts. Such application-aware HIDSs originate in information and communications technology systems and have a great potential to deal with the flexible nature of application-level software platforms. However, the elementary characteristics of known application-aware HIDS approaches and thereby the implications for their transfer to the automotive sector are unclear. In previous work, we presented a systematic literature review (SLR) covering the state of the art of application-aware HIDS approaches. We synthesized our findings by means of a fine-grained classification for each approach specified through a feature model and corresponding variant models. These models represent the approaches' elementary characteristics. Furthermore, we summarized key findings and inferred implications for the transfer of application-aware HIDSs to the automotive sector. In this article, we extend the previous work by several aspects. We adjust the quality evaluation process within the SLR to be able to consider high quality conference publications, which results in an extended final pool of publications. For supporting HIDS developers on the task of configuring HIDS analysis techniques based on machine learning, we report on initial results on the applicability of AutoML. Furthermore, we present lessons learned regarding the application of the feature and variant model approach for SLRs. Finally, we more thoroughly describe the SLR study design.

**Keywords:** intrusion detection, security engineering, survey, AutoML, automotive

## 1 INTRODUCTION

Nowadays, the market demands of the automotive sector are more and more driven by digital natives. This induces a transition from traditional automobiles with deeply embedded electronic control units to connected, digital systems that involve concepts from information and communications technology (ICT) systems (KPMG, 2017). Such connected digital systems require more flexible platforms like AUTOSAR Adaptive or Automotive Grade Linux. Both have in common that they allow dynamic installation and update of user-level applications by

providing services and resources that can be requested dynamically. The functionality of the applications (infotainment involving user devices, V2X, etc.) inherently requires connectivity with remote and in-vehicle entities.

On the downside, the increasing connectivity extends the attack surface of the vehicle. This implies the need for additional security control techniques, which encompass intrusion detection systems (IDSs) known from the ICT system development. In order to detect suspicious activities in application contexts (e.g., code injections), certain host-based intrusion detection systems (HIDSs) (Scarfone and Mell, 2007) can be employed. This particular class of HIDSs monitors and analyzes events within applications or data that is relatable to applications or processes, focusing on software behavior. Extending the definition of Bace and Mell (2001), we call such HIDSs *application-aware HIDSs*. In contrast to system-wide IDSs, application-aware HIDSs have a great potential to deal with the flexible nature of adaptive automotive software platforms. This is due to the potential presence of multiple versions and configurations of different applications in such platforms, which application-aware HIDSs can cope with. For example, when installing or updating only one concrete application, a system-wide IDS potentially has to update its complete reference model of benign and malicious system-behavior. In contrast to this, an application-aware HIDSs only have to add or update that part of the reference model that relates to the application under consideration, thus being more modular than system-wide IDS. Furthermore, developers can tailor application-aware HIDSs to the particularities of the corresponding applications, which has the potential of a better detection performance compared with more general system-wide approaches. Please note that application-aware HIDSs and system-wide IDS can also be used in conjunction.

Today, the elementary characteristics of application-aware HIDSs are unclear, which impedes their development and application in practice. Current surveys focus on IDSs in general but not on application-aware ones. Moreover, these surveys cover almost no application-aware HIDSs in the automotive sector.

In order to provide an overview and thorough classification of state-of-the-art application-aware HIDSs approaches, we presented in previous work (Schubert et al., 2019) a systematic literature review (SLR) on such approaches as an introduction into the topic. The target audience are researchers and practitioners with an automotive background that are interested in current application-aware HIDS technology. We synthesized our findings using the formalism of a feature model and corresponding variant models resulting in a classification of each approach along 140 different features, which represent the approaches' elementary characteristics. In comparison with other surveys or SLRs on IDS approaches, our research questions focus on application-aware HIDSs. Furthermore, we summarize our key findings and infer implications for transferring application-aware HIDSs to the automotive sector.

In this follow-up article, we extend the previous work (Schubert et al., 2019) by several aspects. We adjust the quality evaluation process within the SLR to be able to consider high quality

conference publications, which results in an extended final pool of publications. Furthermore, one of the main findings of the previous work was that the configuration of application-aware HIDSs is mostly driven by expert knowledge. This is a manual and, hence, a cumbersome and time-consuming task. Particularly, this applies to the configuration of analysis techniques based on machine learning (ML). In order to provide information about possible frameworks for the automation of ML approaches to support HIDS developers, we conducted experiments on the applicability of automated machine learning (AutoML) (Hutter et al., 2019) and report on initial results. Furthermore, we present lessons learned regarding the application of the feature and variant model approach for SLRs and we round out the paper by more thoroughly describing the SLR study design. Finally, for readers who want to examine the topic more closely, we updated our supplementary material which provides summaries of all classified publications, the search documentation, an overview of the classification, and additional information regarding the AutoML experiments.

We introduce related work in the **Section 2** and explain our study design in **Section 3**. We present its detailed results in **Section 4** before summarizing the key findings and inferring implications for the automotive sector in **Section 5**. **Section 6** elaborates on our first experiments of using AutoML in the context of IDSs. Furthermore, **Section 7** discusses the usage of feature models in SLRs. Finally, we conclude this paper and outline future work in **Section 8**.

## 2 RELATED WORK

Research on IDSs can be dated back to the early 1980s (Bruneau, 2001). This resulted in a extensive body of publications that encompasses surveys, taxonomies, or general overviews, e.g., Lazarevic et al. (2005), Scarfone and Mell (2007), or Viljanen (2005). All of these publications differ from this paper in at least one of the following points. 1) None of the publications focuses on application-aware HIDSs, and particularly not on implications for the automotive sector. 2) Most of the publications do not follow clearly structured review procedures like the one of this SLR. 3) Some of the approaches are simply too old to capture the current state-of-the-art. In the following, we briefly recap recent publications that are complementary to our study.

Loukas et al. (2019) survey IDSs for vehicles. This publication also encompasses 36 IDSs for automobiles and automobile networks capturing the state-of-the-art of intrusion detection in this sector. With the exception of one publication, all of the reviewed publications are network-based approaches monitoring either the CAN or vehicular ad hoc networks.

Buczak and Guven (2016) elaborate on data mining and ML methods that are used in IDSs. The publication addresses high-level descriptions of the methods as well as particularities like their time complexity. Furthermore, the authors recap example papers that utilize the corresponding methods. We found many of the described methods in the publications of our SLR, which makes their publication a good source for more detailed descriptions.

Luh et al. (2017) review approaches that focus on the detection of targeted attacks. In the domain of HIDSs their survey covers seven approaches. The authors' research questions revolve around how semantics aspects are utilized by the approaches under consideration. In comparison to our classification, architectural aspects or aspects referring to the interplay of different techniques in different phases of the detection are not targeted.

### 3 STUDY DESIGN

In order to execute the SLR, we follow the process described by Kitchenham and Charters (2007) and Brereton et al. (2007). This process defines a general methodology for conducting SLRs in the software engineering domain. Since we consider the composition of application-aware HIDS to be a software engineering topic, we follow this process. The process may differ in certain ways, depending on the type of research questions. For example, data extracted from discovered publications may be evaluated using statistical or qualitative methods. In our case, the research questions, as defined in **Section 3.2** are of a qualitative nature, so we do not employ statistical methods for data analysis.

**Figure 1** depicts this SLR process with our particular characteristics (e.g., number of surveyed publications) specified in the Business Process Model and Notation (BPMN) [Object Management Group (OMG), 2014]. The SLR process is split into three coarse-grained phases: Planning, conducting, and reporting. In the planning phase, the research questions are defined. The research questions are part of a review protocol, laying out a basic search and filtering strategy in order to find appropriate sources. The review protocol should be approved by experts not directly involved in the study, which, in our case, was done by a review board made up of department colleagues and an external IDS expert. The SLR is then conducted according to the protocol, which entails a literature search, filtering, quality assessment, data extraction, and finally data synthesis or interpretation. Lastly, the results of the SLR should be published for other researchers (Kitchenham and Charters, 2007), which we do in this publication and the supplementary material.

In the following, we discuss each step as part of the coarse-grained SLR phases planning, conducting, and reporting. For this purpose, we describe each activity step of the overview in **Figure 1** in a dedicated section. After discussing each step, **Section 3.12** introduces the threats to validity relevant for this SLR and the measures taken to deal with these threats.

#### 3.1 Fundamentals

As visualized by the step Investigate Fundamentals as part of the Planning phase in **Figure 1**, the first step in any SLR is to familiarize oneself with the fundamentals of the examined topic to gain an overview of the used terminology and the current state in the specific field. This is needed to define the review protocol effectively, especially the search terms and filtering criteria. In this case, an informal initial literature

study preceded the definition of the review protocol, with the partial goal of defining the terminology used within the SLR. We generally follow the definitions of (Bace and Mell, 2001) and (Scarfone and Mell, 2007), but extended them by our understanding of application-aware HIDS as motivated in the introduction.

#### 3.2 Research Questions

The research questions form the core of the review protocol. In order to explore the current landscape in application-aware HIDS, we pose the following research questions (cf. Define Research Questions as part of the Planning phase in **Figure 1**):

RQ1: What are the *architectures* (e.g., distributed or centralized) of current application-aware HIDS approaches?

RQ2: What are the *techniques* of current application-aware HIDS approaches? The *technique* comprises the basic approach and the specific analysis technique used for behavior classification.

RQ3: What are *usage contexts* of current application-aware HIDS approaches? The usage context represents the context the HIDS is used in (i.e., targeted threats, programs, or operating systems, and the necessary monitored data).

RQ4: What are the relationships between the *techniques*, *usage contexts*, and *architectures*?

RQ5: How are the approaches evaluated? Are there practical case studies using public data sets?

We designed these research questions in such a way that they cover a range of approaches considered to work on the application level. They also capture important aspects of different intrusion detection approaches, such as the techniques that are used to classify behavior and the context in which a certain approach is to be used. Since the architecture of an IDS is important for the realization, it is targeted by the first research question. We also aim at determining relationships between certain aspects of intrusion detection approaches. The research questions form the basis of the following steps within the SLR process, in that they are used to derive the search terms as well as the in- and exclusion criteria.

#### 3.3 Search and Filtering Strategy

Following on the definition of research questions a systematic search procedure for publications as well as criteria for filtering the results have to be defined (cf. Define Search and Filtering Strategy as part of the Planning phase in **Figure 1**). In the following, we discuss these steps separately. **Section 3.3.1** presents the used search terms and libraries. Thereafter, **Section 3.3.2** discusses the overall filtering procedure and the used inclusion and exclusion criteria.

##### 3.3.1 Search Strategy

The search terms should be defined in such a way, that the largest amount of relevant sources is found. However, the searches bring up results that are not relevant to the specific topic or the research questions, which is dealt with by the filtering criteria. This section

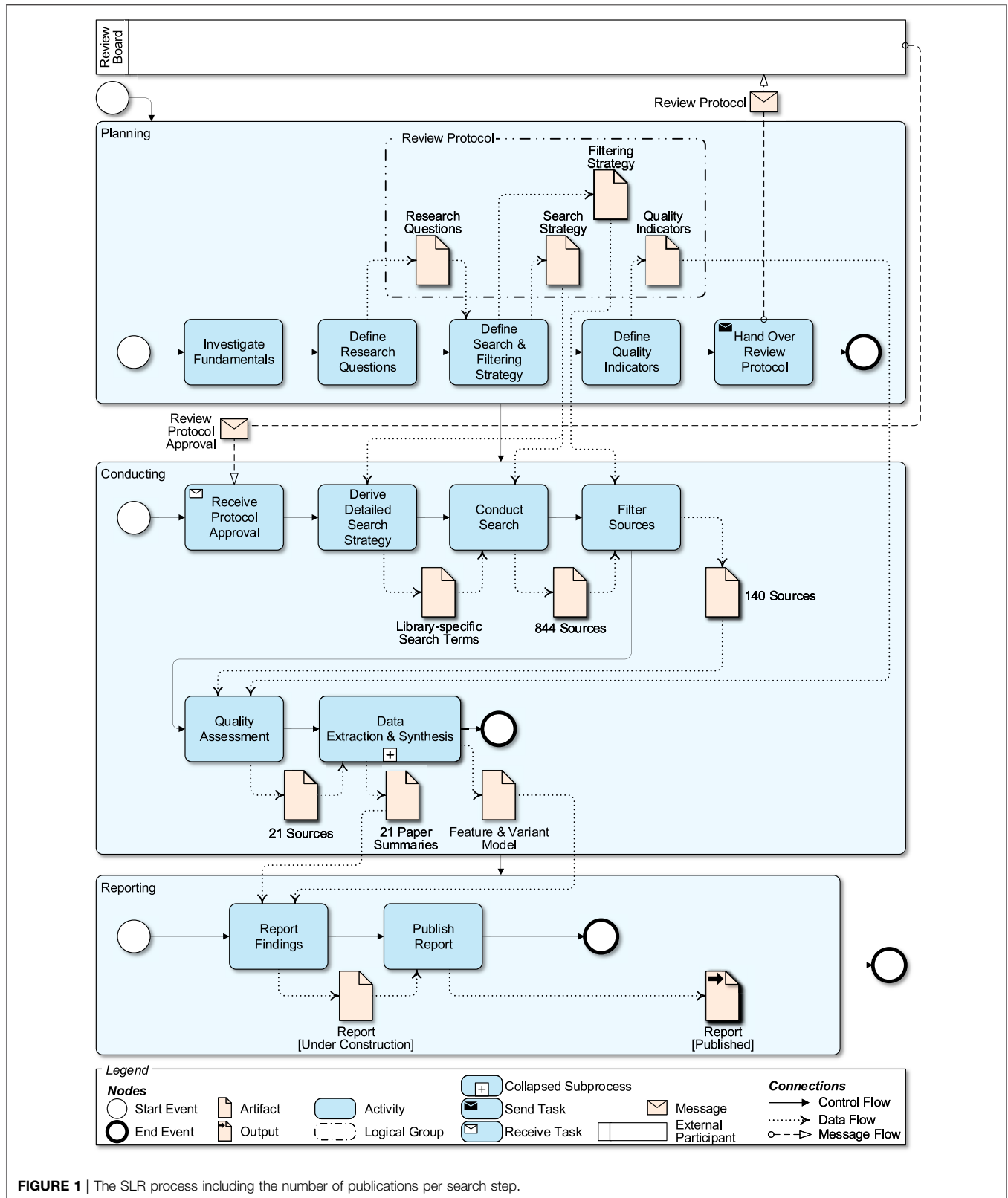


FIGURE 1 | The SLR process including the number of publications per search step.

discusses the basic search phrase and the filtering criteria employed in this study.

Generally, to keep the search terms targeted to the problem, they are derived from the research questions. Here, the basic search phrase consists of three main parts. The three parts contain separate synonyms for the appropriate search terms that are connected by a Boolean “OR”. In turn, the parts are connected by Boolean “AND”.

(“Intrusion Detection” OR “Anomaly Detection” OR “Application Intrusion Detection”) AND  
 (“Host-Based” OR “Application-Based” OR “Application Specific” OR “Application Aware” OR “Application Sensitive”) AND  
 (“Program Behavior” OR “Application Behavior” OR “Software Behavior”).

The first part targets the general topic of intrusion detection. Since “Anomaly Detection” is sometimes used synonymously, it is also part of this part. Additionally, we added the term “Application Intrusion Detection”, since it came up in the preliminary research.

Regarding the second part, the goal in this SLR is to examine IDS approaches that work on the application level. Since these approaches are usually host-based, or more specifically, application-based, these terms and their synonyms comprise the second part of the search phrase.

In terms of the last part, our research questions focus approaches that somehow analyze the behavior of software. Thus, the third part contains terms that target this behavioral aspect and were used in the literature that we examined in the preliminary research as part of the Investigate Fundamentals step in **Figure 1**.

In order to search efficiently and due to the limited selection in print libraries with regards to the domain covered in this review, all used libraries are digital. According to (Brereton et al., 2007), these are libraries relevant to the field of software engineering specifically. We limit ourselves to the digital libraries provided by the publishers ACM, Springer, IEEE, Elsevier. Whereas Springer’s digital library is not mentioned by (Brereton et al., 2007), it is recommended by Kitchenham and Charters (2007), so we chose to include it.

Due to preliminary searches showing results not published in the aforementioned libraries, we chose to specifically search for publications from Advanced Computing Systems Association (USENIX) and the Network and Distributed System Security Symposium (NDSS), which were recommended by the review board to be of relevance. In the case of NDSS publications, we employ Google Scholar and split the search phrase in order not to run into search term length restrictions, searching only for publications with NDSS named as the publisher. For USENIX, we searched the site using Google, filtering for PDF documents.

### 3.3.2 Filtering Strategy

The review protocol defines filtering criteria as to ensure that only the relevant sources are examined in the SLR. This also establishes

the basis for a process, in which sources are in- or excluded in a repeatable, unbiased, and transparent fashion. The review board approved the filtering criteria, which again combats potential bias in the researchers performing the filtering step. Since the process should be transparent, the reason for exclusion is recorded in the documentation, which can be found in the **Supplementary Material**.

Overall, we use two inclusion and ten exclusion criteria. If a source deals exclusively with the topic of application-aware HIDS or introduces a novel approach that follows the definition of application-aware HIDS, it is included in the pool of source to go to the next step.

We exclude sources dealing only with HIDS approaches relying on network data to detect intrusions or are not mainly concerned with a new application-aware approach. We also exclude sources if:

- The general topic of the source is not computer science;
- It is in the form of a survey, because only primary literature should be considered;
- It is in the form of a Bachelor’s, Master’s, Ph.D. Thesis, project/technical report, or book chapter since these are not peer reviewed sources;
- It is in the form of a patent or a standard description;
- It is in the form of a panel discussion, preface, tutorial, book review, poster, or presented slides, since these formats are of an informal nature;
- It is unclear where and when the source was published and if it has been peer reviewed;
- The source is not written in the English language, since it is the working language in this study;
- The full text is not available through the library of Paderborn University without payment, because no budget is allocated for this purpose;
- The source was published before 2012.

The decision to exclude sources that did not undergo a formal peer review process was made to ensure objective filtering of sources with regards to their originality, validity, and quality. While there are many excellent theses or technical reports with detailed information on application-aware HIDS, it would be difficult to find objective criteria for these aspects, which are covered by the peer review process of conferences and journals.

During the selection procedure, we first applied the inclusion parameters and subsequently the exclusion criteria. Once a source is included, it can no longer be excluded on the basis of the filtering criteria. Should a source not match any of the criteria explicitly, it is excluded. The selection procedure is then performed in stages: The title is examined first, then the abstract, and finally, the whole text is skimmed or read completely. If, at any stage, a criterion is met, the source is in- or excluded and the reason recorded.

## 3.4 Quality Indicators

To enable the evaluation of source in terms of their quality, corresponding indicators have to be defined (cf. Define Quality Indicators as part of the Planning phase in **Figure 1**). The quality

evaluation is recommended by (Kitchenham and Charters, 2007) to assess whether a subpar study is included, which may negatively impact the results of the data synthesis. Since we are performing a mostly qualitative data synthesis, we only consider sources that fulfill all quality criteria, so that the analysis and the resulting taxonomy reflects high quality publications. It also reduces the amount of studies in the final selection, which would otherwise have been too large to fit in the scope of our SLR.

We employ two quality indicators 1) the fact whether the source is a journal or high quality conference paper or not and 2) whether the authors perform a practical evaluation of their developed HIDS approach. We assume that journal publications are generally of higher quality than others, since such publications usually go through several review cycles and the practice of republishing extended versions of high quality conference publications. However, not all high quality conference papers are republished in journal articles and should still be considered in the SLR. Therefore, conference papers published at highly ranked conferences (A\* and A according to the CORE-Ranking<sup>1</sup>) that employ a journal-like rebuttal phase in the submission process fulfill the first quality criterion. We employed the CORE-Rankings from 2021 and checked, if the conferences currently employ a rebuttal phase, since checking these parameters for past conferences was not possible in all cases.

The second quality criterion is completely fulfilled, if the authors perform an evaluation with standardized or publicly available data. Otherwise, if the authors use data generated on their own and have not made it publicly available, the criterion is partially fulfilled. If no practical evaluation is performed, the criterion is not fulfilled. The final pool of selected sources is made up of those that fulfill both quality criteria at least partially.

### 3.5 Review Protocol Approval

This step relates to the Hand Over Review Protocol activity of the Planning phase in **Figure 1**. Generally, defining the review protocol is an iterative process, with several update cycles based on preliminary searches and feedback from the review board. In this case, we perform two rounds of feedback with the full Review Board. After taking the feedback from both rounds into account, the Review Board approved the protocol and the SLR can be executed under the guidance of the protocol. As such, the Review Protocol Approval indicates the transition from the Planning phase to the Conducting phase.

### 3.6 Detailed Search Strategy

Since the support for Boolean operators within search phrases differs for each of the digital libraries, a specific search string is devised for each of the databases (cf. Derive Detailed Search Strategy as part of the Conducting phase in **Figure 1**). Most of the databases used in this review support Boolean operators, so only minor adjustments are necessary, such as selecting advanced search and adding parameters defining that the full text

should be searched. All search phrases are available in the supplementary documentation, along with the dates of the search and all search results.

As already mentioned, NDSS USENIX do not provide their own digital libraries that allow searching with Boolean operators. Therefore, for USENIX we opt to search their website using Google, filtering for PDF results only. For USENIX, we search using Google Scholar, splitting the search phrase to avoid running into length restrictions. We use the advanced search feature and put one term from both the “Intrusion Detection” and “Observation Target” parts into the “All of the words” field. Then, the terms from the “Point of Implementation” part of the search phrase are places in the “Any of the words” field. In addition, the Publication/Journal field is set to NDSS.

### 3.7 Search Documentation

Having defined the specific search strategies for each digital library, we execute the searches (cf. Conduct Search in **Figure 1**). The documentation of the searches for each digital library is provided in the supplementary material to this article. After searching all libraries, we have 844 results in total.

### 3.8 Filtering Sources

In order to filter the sources, we start by excluding sources published after 2012 (cf. Filter Sources in **Figure 1**). Thus, we cover a period from 2012 to the time of our searches in mid-April 2019. This exclusion criterion was added since the purpose of the SLR is to reflect the state of the art, considering the review board advice. This leaves the rest of the publications to be filtered in the manner described in **Section 3.3**. After filtering, 140 publications are left to be reviewed for their quality. Please note again that we finalized the SLR for our original publication (Schubert et al., 2019). In particular, we did not extend the period for this publication.

### 3.9 Quality Assessment

The Quality Assessment (cf. Conducting phase in **Figure 1**) is conducted by two researchers, each working on their own subset of publications. A crosscheck was performed, in which each researcher reviewed some of the other’s assigned sources to ensure no bias or errors are part of the quality evaluation. The 21 sources containing a practical evaluation of the IDS approach and being published in journals and at highly ranked conferences with rebuttal phases make up the final pool of selected publications. **Table 1** shows these sources and the corresponding journals or conferences, respectively.

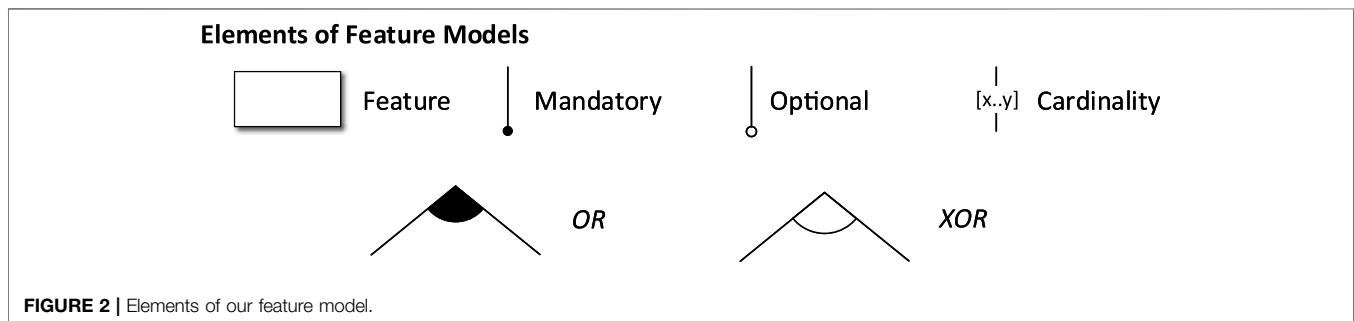
### 3.10 Data Extraction and Synthesis

To answer the research questions posed in the review protocol, relevant data needs to be extracted, consolidated and interpreted from the sources. The Data Extraction and Synthesis steps are closely linked in our case (cf. Conducting phase in **Figure 1**). The corresponding process is described in **Section 7** in more detail.

<sup>1</sup><http://portal.core.edu.au/conf-ranks/>

**TABLE 1** | Publications in the final pool.

Title and authors	Journal/Conference
Trust in IoT: Dynamic emote attestation through efficient behavior capture—Ali et al. (2017)	Cluster Computing
SAMADroid: A novel 3-level hybrid malware detection model for android operating system—Arshad et al. (2018)	IEEE Access
SHADuDT: Secure hypervisor-based anomaly detection using danger theory—Azmi and Pishgoo. (2013)	Computers & Security
PbMMD: A novel policy based multi-process malware detection—Bidoki et al. (2017)	Engineering Applications of Artificial Intelligence
On early detection of application-level resource exhaustion and starvation—Elsabagh et al. (2018)	Journal of Systems and Software
Inferring software component interaction dependencies for adaptation support—Esfahani et al. (2016)	ACM Transactions on Autonomous and Adaptive Systems
LEAPS: Detecting camouflaged attacks with statistical learning guided by program analysis—Gu et al. (2015)	IEEE/IFIP International Conference on Dependable Systems and Networks
Watch me, but don't touch me! contactless control flow monitoring via electromagnetic emanations—Han et al. (2017)	ACM SIGSAC Conference on Computer and Communications Security
Anomaly detection techniques based on kappa-pruned ensembles—Islam et al. (2018)	IEEE Transactions on Reliability
An anomaly detection system based on variable N-gram features and one-class SVM—Khreich et al. (2017)	Information and Software Technology
Combining heterogeneous anomaly detectors for improved software security—Khreich et al. (2018)	Journal of Systems and Software
Data-driven anomaly detection with timing features for embedded systems—Lu and Lysecky (2019)	ACM Transactions on Design Automation of Electronic Systems
Adaptive security monitoring for next-generation routers—Mansour and Chasaki (2019)	EURASIP Journal on Embedded Systems
Generating profile-based signatures for online intrusion and failure detection—Masri et al. (2014)	Information and Software Technology
Intrusion detection for resource-constrained embedded control systems in the power grid—Reeves et al. (2012)	International journal of critical infrastructure protection
Long-span program behavior modeling and attack detection—Shu et al. (2017)	ACM transactions on privacy and security
RAMD: Registry-based anomaly malware detection using one-class ensemble classifiers—Tajoddin and Abadi (2019)	Applied intelligence
Unifying intrusion detection and forensic analysis via provenance awareness—Xie et al. (2016)	Future generation computer systems
Detection of anomalies in behavior of the software with usage of markov chains—Zegzhda et al. (2015)	Automatic control and computer sciences
A defense framework against malware and vulnerability exploits—Zhang et al. (2014)	International journal of information security
Secloud: A cloud-based comprehensive and lightweight security solution for smartphones—Zonouz et al. (2013)	Computers and Security



In order to systematically interpret and present the findings, we develop a *feature model*, as first introduced by Kang et al. (1990) and later refined by Czarnecki et al. (2005). Feature models are meant to document, represent, and discover software system domains and their relationships. Here, we use the feature model to characterize the different approaches. To this end, extracted IDS characteristics are represented as features in a tree format, while allowing for the representation of relations between the characteristics. During the data extraction process, it is therefore necessary to extract these characteristics as well. In addition to providing an overview of all possible characteristics of application-aware HIDS approaches, the feature model can be used to classify single approaches. Hence, we *select* the corresponding features for each approach such that they

conform to the variability constraints expressed by the model. The result of this selection is a *Variant Model* as explained in **Section 7**.

**Figure 2** shows the elements of our feature model. *Features* represent IDS characteristics. They are hierarchically structured and may contain sub-features or feature groups. Features may be *mandatory*, meaning they must be selected in a classification, or *optional* when they can be left out. *Cardinalities* indicate that a feature can be selected more than once in an approach,  $x$  and  $y$  being the upper and lower bound, respectively. If a feature group is connected with the *OR* connector, at least one of the contained features has to be selected in each approach. For the *XOR* feature groups, only one of the connected features may be selected in an approach.

Starting with an initial version of the feature model, we added more features as more publications were examined and removed unused ones. Thus, only features found in the final pool of sources are present in the feature model. This way, the feature model accurately represents the characteristics of the selected publications. The final feature model and the SLR results follow in **Section 4**.

### 3.11 Reporting Findings

According to the guidelines of (Kitchenham and Charters, 2007), the results of any SLR should be reported and published, since any SLR can give valuable insight into the examined topic. In addition, since one of the goals of an SLR is to reduce bias, the documentation should be made available as well, ensuring repeatability and transparency. In the case of this SLR, the results were published in a conference paper along with the search documentation and the raw extracted data, as well as the feature model from the data synthesis (Schubert et al., 2019). This article presents a more in-depth description of the SLR and the continuation of our work on application-aware HIDS. Again, we publish the documentation and the raw data alongside this article.

### 3.12 Threats to Validity

An SLR seeks to eliminate bias through a well-defined procedure and an independent validation thereof by a review board. Nevertheless, bias can occur when only few people (i.e., two in this case) execute the search, data extraction, and interpretation steps. Here, we discuss the threats to validity of the SLR presented in this paper.

A general threat to meta-studies such as SLRs is the uncertainty about the validity of the surveyed publications. It may be, that the application-aware HIDS approaches in the literature do not lead to the expected results or were not evaluated in the described manner, for example. In order to mitigate this threat, a full replication study would have to be performed. Since the HIDS used in the discussed approaches are not generally openly available, such a replication study was not feasible for us to execute. Therefore, we assume that the descriptions and evaluations of the approaches are valid.

For the more specific types of threats, we follow the system laid out by Wohlin et al. for evaluating the validity of experiments in software engineering to guide our discussion (Wohlin et al., 2000). Each threat belongs to one of the following categories: Conclusion validity, internal validity, construct validity, and external validity. In the following, we present the applicable threats from these categories, according to Wohlin et al.'s checklist, and present our mitigation measures.

#### 3.12.1 Conclusion Validity

The risk of drawing biased or false conclusions from statistical information is covered by the conclusion validity threat. Generally, it is concerned with the conclusions drawn from observing the relationship between a treatment and an outcome. In this case, the extracted features and their frequency can be considered the "treatment" and our interpretation the "outcome". In the following, we discuss the relevant aspects leading to this threat.

#### *Low Statistical Power*

In an SLR, the statistical power of the data extracted from the search results depends on the SLR process, since errors during searching, filtering, and evaluating the results have a direct impact on the resulting pool of included sources. In the case of bad search phrases, for example, whole groups of relevant sources could be left out of the review, negatively impacting the amount of results and the topics covered in the results.

This threat is mitigated in this review by directly deriving the search terms from the research questions and expanding them to complete phrases using suggestions from the review board. Thus, potential bias is eliminated and the search phrases match the goal, that is, answering the research questions.

Another aspect of this threat is that the examined sources may not be representative of the whole field of research being addressed. In order to perform inferential statistics and generalize results to the whole population, i.e. make statements about the whole field of research addressed by the SLR, the sample size must be large enough to reach a certain statistical power. In this case, the 21 sources selected in the SLR process do not make up a large enough sample size. Therefore, we perform a descriptive analysis and do not claim that our findings hold for the whole research field of application-aware IDS. The findings should be seen as indications and are not to be generalized.

#### *Fishing*

"Fishing" refers to the selective alteration of research results by the researcher. It may threaten the validity of the searching, filtering, extraction, and evaluation steps, usually through a biased person executing these steps.

Here, two people executed the review with a third person and the review board giving feedback and serving as a controlling instance to eliminate bias. The review protocol was checked by the review board repeatedly and changed based on their input, mainly on the search terms, further reducing potential bias.

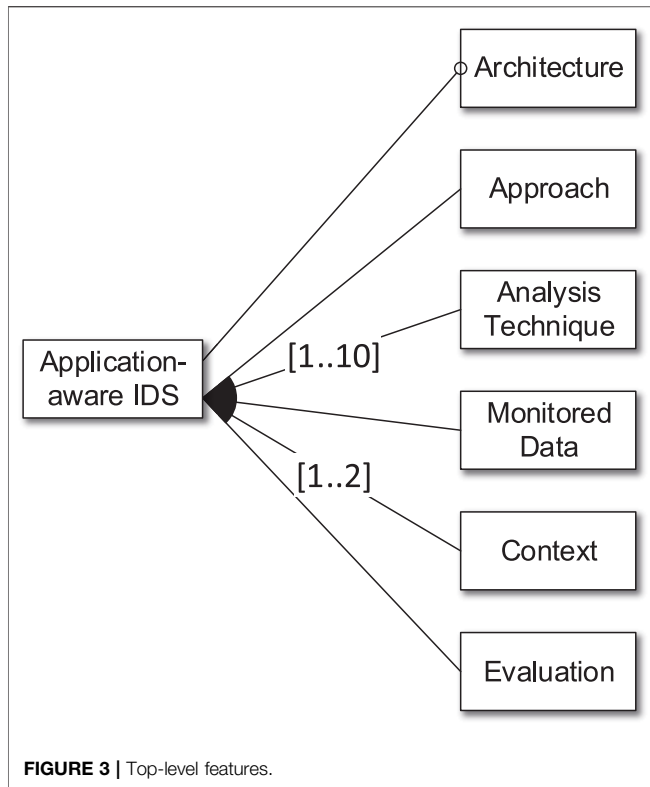
Another mitigating factor is the feature model used to represent the data in a structured manner. It defines a fixed syntax for the sources to be classified into and enables consistency checks, so that the data extraction is performed faithfully to the overall feature model. In the case of a source not fitting into the feature model, it was adapted and the consistency was checked with regards to previously classified sources.

#### *Reliability of Measures*

Since only two people perform the data extraction, filtering, and quality evaluation in this study, the reliability of measures, such as the pool of selected sources and the extracted data, is not guaranteed. However, the standardized process is followed by the researchers at each step, treating all sources in the same manner, reducing the error potential.

#### 3.12.2 Internal Validity

Internal validity refers to the causal relation between treatment and outcome. Therefore, in this SLR, it refers to the causal relation between extracted data and the conclusions drawn from interpreting



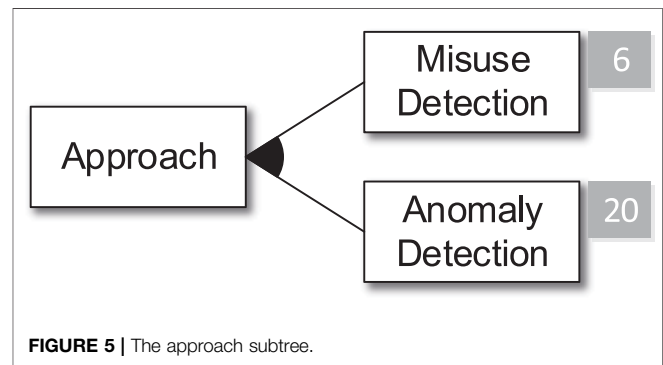
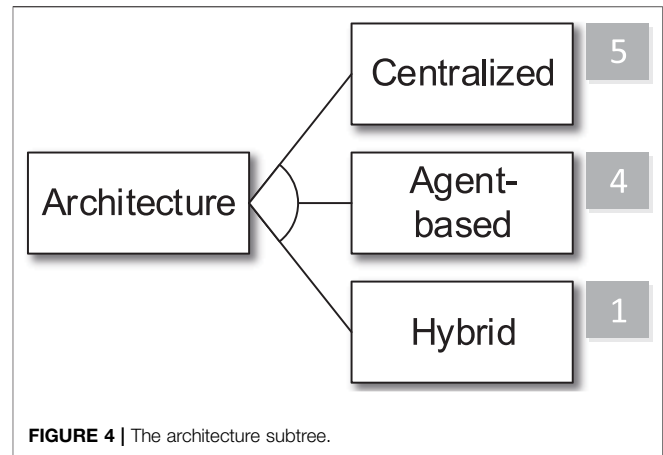
this data. Any effects that negatively impact the extraction and interpretation of data in an experiment may threaten the internal validity, since they invalidate the causality of data and conclusion.

### Maturation

Executing an experiment may tire the subjects. This effect is referred to as maturation. In an SLR, there are no subjects, but such effects may still occur when the researchers are executing the review. Here, this threat was countered by following the process laid out by the review protocol and double-checking the results after potential maturation effects subsided.

### Instrumentation

Tools that are used in an experiment are referred to as instrumentation. They must be fit for the experiment's task, otherwise they negatively impact the results. In an SLR these tools are usually forms and databases to record the process. In this SLR, the instrumentation evolved during the course of the execution. Thus, we corrected inadequate tooling, mainly in terms of excel tables, and thereby reduced the threat. Additionally, to check the feature model for consistency we used a custom feature model editor that allows to check the variant models for conformance to the complete feature model. We employed the consistency check function when new features were added to the feature model, as described in **Section 7.1**. These automated checks were crosschecked manually on randomly chosen publications. Additionally, each determined inconsistency was validated manually as a starting point of the corresponding refactoring.



### 3.12.3 Construct Validity

The design of an experiment is subject to construct validity. An SLR is susceptible to this, since the search phrases and material filtering or evaluation parameters must be sound.

#### Experimenter Expectancies

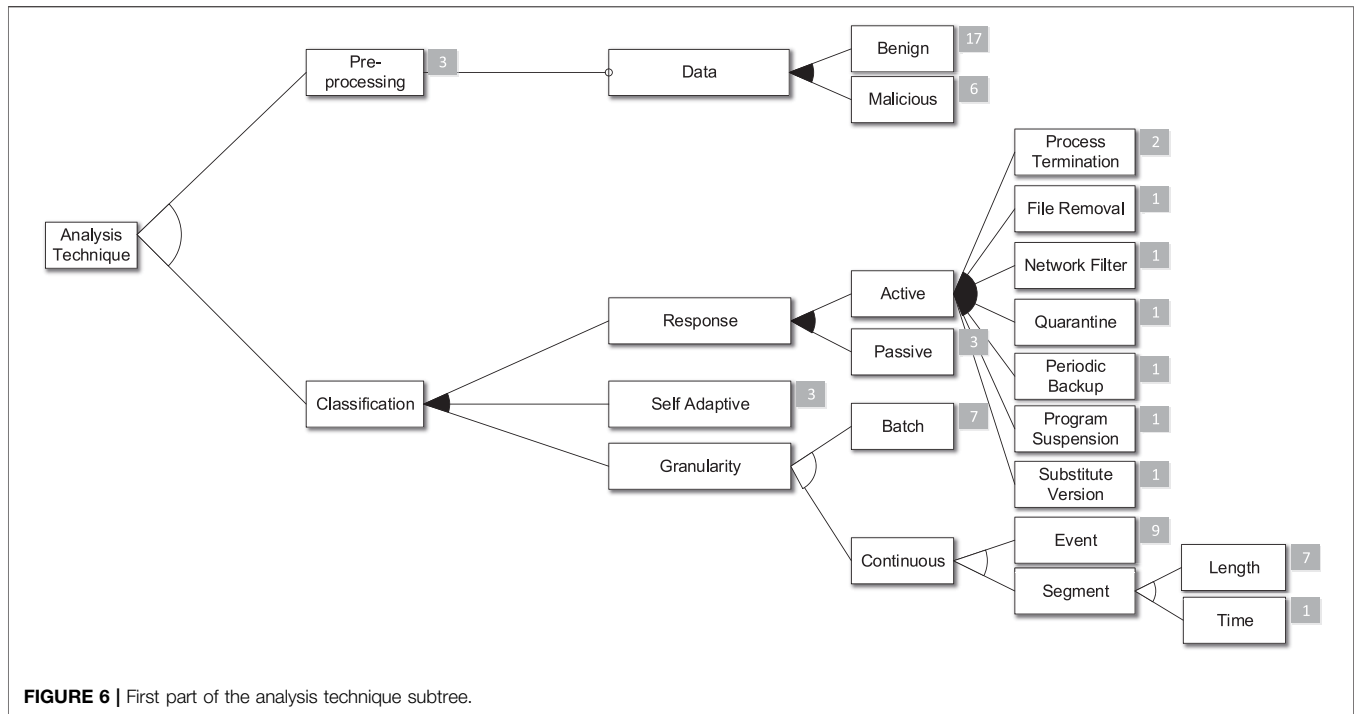
A potential threat to the validity of the construct of the experiment are the expectations of the experimenter, in this case the researcher performing the SLR. Similarly to the threat of “Fishing” discussed above, the researchers' expectancies may be influenced by personal biases. To eliminate bias, frequent feedback from the review board was taken into account and the review protocol was followed as closely as possible.

### 3.12.4 External Validity

The external validity refers to the generalizability of the experiment's results to practical applications. Here, this means the applicability of the found sources to HIDS solutions used in practice.

#### Interaction of Setting and Treatment

The setting of an experiment may be different to the practical setting in which the results are applied, threatening the external validity. In this case, the found HIDS approaches may perform well in a scientific



setting but not in actual computer systems during practical use. However, all of the examined HIDS approaches contain practical implementations, ensuring at least a basic level of practical applicability.

## 4 RESULTS

This section follows the research questions defined in **Section 3.2**. For any research question, we each introduce the corresponding subtrees of the synthesized feature model and discuss our conclusions. **Figure 3** shows the top-level features, which constitute the main categories of our taxonomy. The following **Figures 4–10** depict the subtrees of the top-level features. The numbers shown in the figures express how many publications select the corresponding feature. We only show these numbers for potential leaf selections to increase the readability of the figures while keeping the most important information.

### 4.1 RQ1: Architecture

Generally, only ten publications discuss the Architecture of their approach. We found three different types of architectures, i.e., Centralized, Agent-based, and Hybrid architectures (cf. **Figure 4**). We classify an architecture as being Centralized if the complete IDS is deployed on a single system. This type of IDS architecture is used in five publications and, thus, constitutes the most prominent architectural style. The main reason for this is that a centralized IDS is the easiest and fastest type to realize a research prototype and more elaborated architectures are not in the focus of the corresponding publications.

An Agent-based architecture uses agents for data collection and preprocessing. Please note that the term *agent* is common in

the HIDS domain and refers to remote monitoring entities or *probes*. It should not be confused with *agent* as used, e.g., in the domain of agent-based computing.

Any publication that does not strictly adhere to any of the aforementioned architectures is classified as being Hybrid. The only publication being classified like this is Zhang et al. (2014), which encompasses centralized monitoring and classification but explicitly refers to a remote entity to perform an initial examination of the application in scrutiny.

### 4.2 RQ2: Techniques

As depicted in **Figure 5**, we found approaches that realize Misuse and/or Anomaly Detection. Misuse Detection describes the process to detect attacks based on previously known data of malicious activities. In contrast to this, Anomaly Detection searches for deviations of normal system behavior to detect security incidents. Only one of our reviewed publications realizes exclusively Misuse Detection. The remaining 20 publications are either purely anomaly-based (15 approaches) or work with a combination (5 approaches).

The Analysis Technique subtree of our feature model is the most complex. This directly shows the complexity of the topic itself and that most of the reviewed publications focus on this topic. The Analysis Technique feature has a cardinality of (1 .. 10) (cf. **Figure 3**). Most approaches combine several techniques in terms of machine learning pipelines or ensemble classifiers, and several publications discuss a variety of alternative Analysis Techniques.

The subfeatures of Analysis Technique are structured in two groups. The group consisting of Preprocessing and Classification (cf. **Figure 6**) describes in which phase a technique is used. The

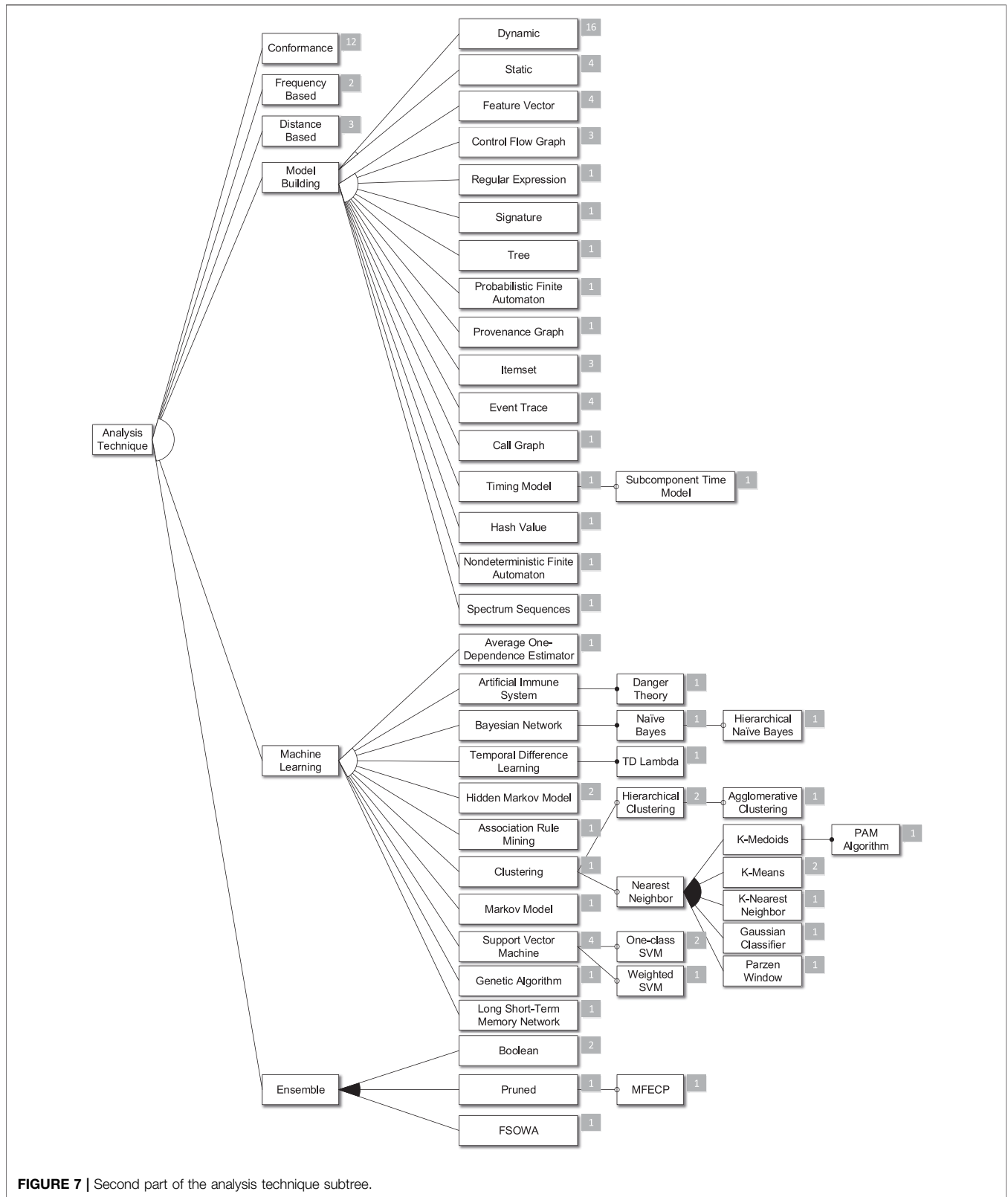
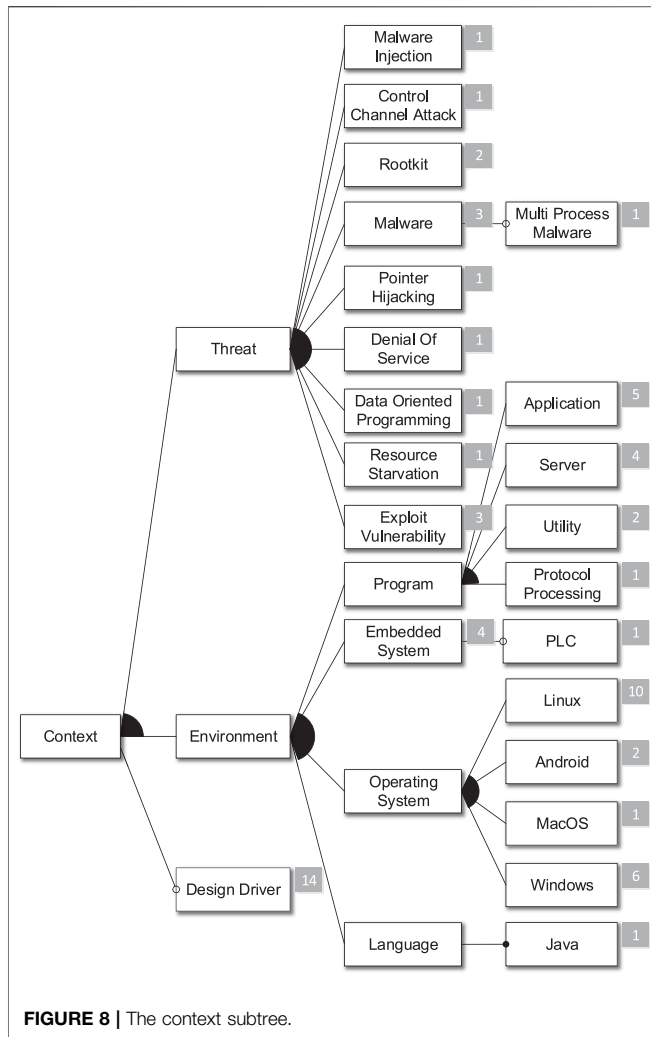


FIGURE 7 | Second part of the analysis technique subtree.

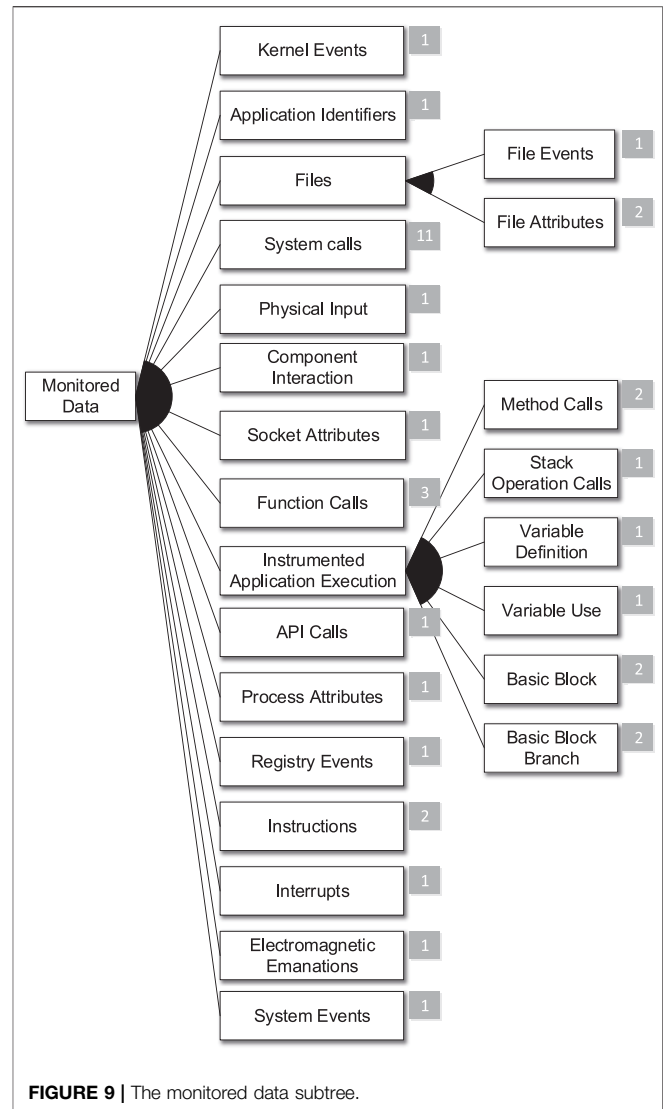
second group (Conformance .. Machine Learning) (cf. Figure 7) provides information regarding the used algorithms or the produced models.

The majority of the approaches uses benign data in Preprocessing steps. This again reflects the predominance of anomaly-based approaches. Furthermore, seven publications



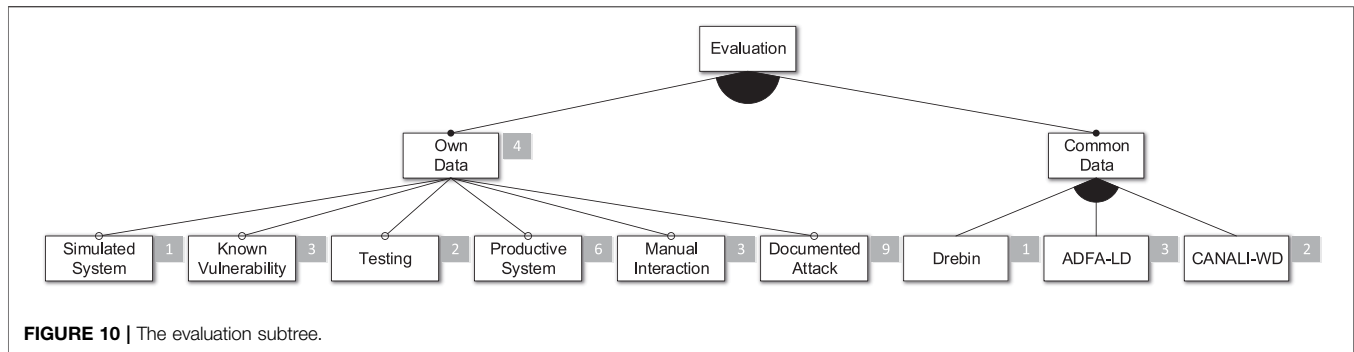
discuss automatic Responses that follow on detected security incidents. Here, we distinguish between Active and Passive responses. In the former case, the system automatically initiates countermeasures. In the latter case, the system informs persons responsible, e.g., system administrators, and relies on them to counteract the incident. For the Classification, we can additionally distinguish between Batch and Continuous techniques, where the latter classify single Events or Segments constructed using fixed Length or Time intervals. Furthermore, we select the Self-adaptive feature whenever an anomaly that is not classified as being malicious is used to extend the underlying classification model.

The Conformance, Frequency Based, and Distance Based features generally describe classification techniques. We select the Conformance feature whenever an approach checks strict conformance to a specification, without using statistical means. An example could be a sequence of system calls that is checked against a model of normal system call sequences. 12 publications encompass at least one analysis step that uses this type of check. A Frequency Based analysis compares the number of occurrences of events on segments. Furthermore, a Distance Based analysis compares the similarity of feature vectors.



The Model Building feature subsumes the different types of artifacts that are constructed during a Preprocessing phase. 17 publications mention such an analysis step. In addition to the representation of these artifacts, we distinguish between the construction via a Static analysis or by analyzing Dynamic data acquired by monitoring. The fact that the Dynamic feature is selected the most is not surprising. IDSs are dynamic analyses by definition. Therefore, the benefit of having a preprocessed representation of dynamic system information is evident. However, the concept of additionally utilizing information obtained by static analyses is common in the IDS domain.

16 of the reviewed publication encompass some sort of Machine Learning technique. The most commonly employed technique are support vector machines (SVMs) with seven selections in total. SVMs are studied for years and known for their good performance in the context of multiple application



domains and datasets (Meyer et al., 2003). Moreover, six publications use Clustering techniques to refine previously inferred models that again will be used for classification in later steps. This observation is in line with similar observations of Buczak and Guven (2016).

The Ensemble feature comprises algorithms that the authors use to combine several classifiers to form an ensemble classifier. We found three publications that utilize this technique. The corresponding authors aim for an increased precision of their combined approach in comparison to the individual classifiers.

### 4.3 RQ3: Context

As depicted in Figure 8, we mainly distinguish between targeted Threats of the approaches and the Environment of the approach in this research question. The cardinality of the context feature allows to differentiate between several contexts (cf. Figure 3). This is used to mark certain contexts as being directly respected in the conceptual design of an approach. To this end, we utilize the Design Driver feature. We select this feature if the authors identify a certain characteristic in the context of their approach and argue how they respect this characteristic. We do not select the feature if, e.g., the authors use a Context in their evaluation due to convenience. This typically happens with Operating Systems or Programs that are widely available.

14 out of our 21 publications mention Design Drivers for their approach. As some publications mention several Design Drivers, we end up with 20 selections of context features being classified as Design Drivers. We can conclude that most of the reviewed approaches are driven by a Context, without a strong tendency towards the Environment (ten selections) or certain Threats (nine selections). Moreover, the generality of the most prominent Threats (Exploit Vulnerability and Malware) shows that this does not necessarily mean that a certain IDS is only capable of detecting a specific type of attack.

Another dimension of this research question refers to the Monitored Data of the approaches (cf. Figure 9). Here, we see a predominance of system-call-based approaches (11 publications). With the exception of one publication, all publications that select the Linux feature also monitor System Calls. Especially for Linux, there exists a number of tracing tools that can be utilized in this context. Furthermore, there is a long history of such approaches in the IDS community encompassing the prominent work of Forrest et al. (1996).

### 4.4 RQ4: Relationships

This section provides a more general view on relationships between features that we discuss in Section 4.1, Section 4.2, or Section 4.3 separately. In particular, we refer to inter-tree dependencies spanning the subtrees discussed there. The relatively small sample size of 21 publications makes statistical analysis infeasible. Thus, we report on relationships that we noted during the reviewing and the data synthesis phases.

First, we discuss resource-constrained environments (Android and Embedded Systems). Here, the authors use either Agent-based architectures or elaborate on the realization of their approach as a Centralized hardware module. Furthermore, all approaches targeting Embedded Systems perform exclusively Anomaly Detection. One of the reasons for this, although not always mentioned by the authors, is the very restricted availability of malicious data for this type of systems. Additionally, the only publication that directly targets timing anomalies as an indicator for security incidents also targets Embedded Systems (Lu and Lysecky, 2019). The timing in these systems, particularly in the subdomain of cyber-physical systems (CPSs), is typically more predictable than in general-purpose systems. This makes deviations in the timing behavior a strong indicator for security incidents (Zimmer et al., 2010).

Second, several authors mention the risk of mimicry attacks (Wagner and Soto, 2002) in the context of anomaly-based systems. These attacks mimic normal system behavior to evade detection by an IDS. We did not add these attacks to the feature model as the authors use the term to describe the concept of evasion and not the type of attack the IDS tries to detect. The authors comment on two closely related possibilities to counteract mimicry attacks. The first possibility is to extend the monitoring capabilities of the approach, e.g., by not exclusively relying on System Calls but also analyzing Function Calls or other artifacts (Khreich et al., 2017). The second possibility is to respect more information in the analysis, e.g., in addition to a sequence-based analysis they also analyze the timing behavior of the events (Lu and Lysecky, 2019).

Lastly, the selection of the Design Driver feature in 14 approaches indicates that the Context has an impact on the design of the IDS. However, these relationships span a wide range from having a motivational nature to triggering fundamental, sophisticated design decisions. Thus, many relationships mentioned in the publications are hard to generalize.

## 4.5 RQ5: Evaluation

As a practical evaluation is a prerequisite for the inclusion of a publication in this SLR, all of the publications include case studies of an implementation of their approach (cf. **Figure 10**). Unfortunately, the authors do not publish these implementations.

Only four of the publications utilize commonly available data sets for their evaluation. This is a well-known issue in the IDS community and, as such, observed in several publications. For example, Buczak and Guven (2016) observe that even if studies utilize common data sets they do it in a nonuniform fashion, e.g., by only using a subset of the data. The lack of evaluations that rely on common data sets hinders the comparability of IDS approaches and the reproducibility of evaluation results. This is also the reason why we do not include results of the evaluations, like the approaches' precision or recall, in our feature model.

However, the usage of common data sets is in many cases simply not possible because appropriate data sets do not exist. For example, none of the approaches that target Embedded Systems is evaluated using such a data set. Besides privacy and intellectual property protection issues, this domain is characterized by a high heterogeneity, which is hard to represent in such data sets.

## 5 KEY FINDINGS AND AUTOMOTIVE IMPLICATIONS

This section summarizes the key findings of our SLR and discusses the implications they have for the automotive sector. For this purpose, we follow the structure of our research questions (cf. **Section 3.2**).

*Architecture:* Architectural aspects are underrepresented in the publications reviewed in the context of our SLR (cf. **Section 4.1**). In particular, we could not find any approach that we would classify as being distributed. In our understanding, a distributed IDS consists of minimal self-contained components and encompasses a modularization of the analysis itself. Furthermore, a distributed architecture has to enable the reuse of components for different IDS configurations and their deployment across different platforms.

This is particularly meaningful in resource-constrained and highly connected environments, such as the next generation of automotive systems. The argumentation is analogous to the usage of agent-based architectures in these environments as computational expensive tasks can be outsourced to remote entities. However, as connectivity cannot always be guaranteed, a layer of resource-efficient analyses can directly be deployed on the system.

*Techniques:* Furthermore, none of the reviewed publications utilizes behavior-specification-based techniques. These techniques rely on human experts to specify benign system behavior. Thus, they constitute a subclass of anomaly detection-based approaches. Mitchell and Chen (2014) state the high potential of these techniques in the context of CPSs but also the disadvantage of having a high effort for the creation of the formal specification.

However, automotive systems already tend to provide thorough specifications due to the safety-critical nature and

legislative regulations. Thus, utilizing this information for anomaly detection is a promising research opportunity.

Furthermore, 16 out of the 21 publication utilize some kind of ML algorithm. Out of the 16 publications making use of ML, seven employ an SVM in their anomaly detection process, two of which under the same lead author [(Khreich et al., 2017; Khreich et al., 2018)]. Therefore, SVMs make up the largest proportion of any 1 ML technique in the examined publications.

ML is one approach to automate the task of specifying benign and malicious behavior, respectively. On the one hand, this releases the IDS developer from the cumbersome task of a manual specification. On the other hand, the configuration of the ML approach itself is not trivial and requires expert knowledge. **Section 6** elaborates on this in more detail.

*Context and Relationships:* As already mentioned in **Section 4.4**, we could not identify an overarching structured process for the configuration of application-aware HIDS, presumably due to the complexity of the domain. The argumentation for a configuration is mostly driven by expert knowledge of the authors and *trial-and-error* approaches for the selection.

This is hardly compatible with the automotive sector that historically strives for standardization and zero-error ability due to extensive supply chains, regulations, and safety concerns. A partial configuration support is still possible. For example, Buczak and Guven (2016) elaborate on decision criteria for the selection of ML techniques. Additionally, the selection of monitoring points is restricted by the target platform. Moreover, the general problem of balancing different (competing) constraints in software design is in the focus of search-based software engineering (Harman and Jones, 2001). Thus, techniques that originate from this research area could help to systematize the *trial-and-error*-like approaches.

*Evaluation:* **Section 4.5** already discusses the problems that originate from the nonuniform evaluation based on public data sets and the issues regarding their availability. In combination with design processes that follow a *trial-and-error* paradigm, the lack of comparability of the resulting configurations is obvious.

For the automotive sector, this is particularly critical as the high standards for intellectual property protection collide with the need for representable data sets. One way to cope with this is to establish methods for data set generation that guarantee a high quality. Furthermore, a common framework for the prototypical realization of approaches can strengthen their comparability in terms of their runtime performance.

## 6 PRELIMINARY RESULTS ON USING AUTOML

As stated in the previous section, we could not identify approaches that support the developer in the configuration of application-aware HIDSs. This finding is particularly critical when it comes to the configuration of the actual Analysis Technique since this task stands out in terms of its complexity (cf. **Section 4.2**). Thus, our first efforts on improving the state of

the art of application-aware HIDSs focus on this Analysis Technique area.

The main challenge in the configuration of analysis techniques for application-aware HIDS is the extraction of specification of knowledge about the benign and malicious behavior of the system under development. As explained in **Section 4.2**, application-aware HIDSs typically apply combinations of misuse and anomaly detection for this purpose and represent the knowledge in terms of rules or other models (cf. Model Building in **Figure 7**). In the case of a manual specification, the HIDS developer specifies models for the attack detection based on knowledge about malicious behavior or based on deviations from the intended system behavior, respectively. The complexity of this task mainly results from the sheer extent of possible benign and malicious behavior. Approaches based on ML aim at learning these models and, thus, automate the tedious task of specification. Our SLR results underline this, as 16 of the 21 publications that we surveyed utilize ML algorithms in their approaches (cf. **Section 4.2**).

However, the construction of ML pipelines similarly is a tedious and time-consuming task, because it encompasses data cleaning, data preprocessing, feature construction, selection of a model family, optimization of hyperparameters, postprocessing of the models, and the analysis of results. In the case of application-aware HIDSs, this is even more problematic as those steps have to be conducted for each application and potentially each version of those applications.

This section introduces the results of our first experiments on the usage of AutoML for the configuration of machine learning pipelines for application-aware HIDS at design time. The general goal of AutoML approaches is to make machine learning systems accessible and effectively usable by domain scientists that are not experts in machine learning. Thus, AutoML frameworks and tools focus on configuring machine learning pipelines. Fundamental capabilities of AutoML are to optimize the choice of preprocessors, the choice of machine learning algorithms, and the values of hyperparameters (Hutter et al., 2019). In the context of application-aware HIDSs, the frameworks can be used to configure tailored pipelines for each version of an application. The AutoML frameworks can be executed at design time based on runtime data of the application under consideration. Thus, the AutoML frameworks do not have to be executed on the resource-constrained automotive systems, i.e., they do not introduce any runtime overhead.

We use two different AutoML frameworks for our experiments, namely, *auto-sklearn* and *TPOT*. *Auto-sklearn* (Feurer et al., 2015) utilizes bayesian optimization with a fixed number of variables to solve the underlying configuration problem. *TPOT* (Le et al., 2020) utilizes genetic programming for this task and allows a higher complexity of the ML pipeline than *auto-sklearn*. Both frameworks are recommended by a recent evaluation of AutoML Frameworks (Balaji and Allen, 2018), making them a good fit for our first experiments.

The following section elaborates on the dataset that we used in our experiments and on the data preprocessing procedure. Thereafter, **Section 6.2** introduces the execution environment,

the configuration of the AutoML frameworks, and the results of the experiments. Finally, **Section 6.3** discusses these results.

## 6.1 Data and Preprocessing

Due to the lack of an host-based automotive dataset, we opt for using ADFA-LD (Creech and Hu, 2013), which is also the most used common dataset we found in our systematic literature review (cf. **Section 4.5**). The traces provided by ADFA-LD comprise sequences of System Calls and exclude their parameters. The dataset was created on a system running with Ubuntu 11.04, prepared to be exploitable by means of six attack vectors:

- 1) Brute force password guessing attempts *via* SSH
- 2) Brute force password guessing attempts *via* TCP
- 3) The creation of a new superuser by a malicious payload encoded into a normal executable
- 4) Remote injection of Metasploit's Meterpreter Java payload and the subsequent execution of various malicious actions
- 5) Linux executable Meterpreter upload *via* social engineering and subsequent execution of various malicious actions
- 6) Injection of C100 Webshell by means of a PHP-based vulnerability and subsequent privilege escalation

**Table 2** shows the general structure of the dataset. Creech and Hu (2013) created the traces of the training and validation datasets during normal operation and filtered them depending on their size. Analogously, they created the traces for the attack dataset by executing each of the aforementioned attacks ten times. This results in a number of text files that contain sequences of integer identifiers, which represent the corresponding system calls.

We conduct some basic preprocessing to make this data processable by the AutoML frameworks, which by default do not yet support the kind of data we are facing, i.e., sequential and categorical data. Wunderlich et al. (2019) evaluated different preprocessing methods and the corresponding system call representations in the context of ADFA-LD. Since their setting is similar to our experiments in terms of preprocessing, we opt for following closely on their approach.

Firstly, we construct a training and a testing dataset that encompasses benign and malicious behavior. For this purpose, we split the attack dataset of ADFA-LD. We add the traces obtained by the first five attack iterations to the training dataset of ADFA-LD. Analogously, we add the traces of the last five iterations to the validation dataset.

Secondly, our setting is an IDS that is used to classify segments of a fixed Length (cf. **Section 4.2**). This enables the IDS to be used in an online fashion. Here, we use the technique of sliding windows introduced by Forrest et al. (1996) or more precisely its full sequence variant (Inoue and Somayaji, 2007). Following Wunderlich et al. (2019) we choose a window size of twenty. This splits the sequences of system calls in overlapping segments of the corresponding size. We label these segments with respect to their origin as benign or malicious.

Thirdly, we balance the training dataset. Since the training dataset contains more benign segments than malicious ones, an

**TABLE 2** | Structure of ADFA-LD (cf (Creech and Hu, 2013; Wunderlich et al., 2019)).

	Type of data	Nbr. of traces	Nbr. of system calls
Training data	benign	833	308,077
Validation data	benign	4,373	2,122,085
Attack data	malicious	746	317,388
—	—	10 Attacks per vector	—

**TABLE 3** | Results of the two AutoML experiments conducted by us and of the experiment of Wunderlich et al. (2019) using a conventional ML approach.

Approach	One-hot encoded	TPR/FPR/Balanced accuracy (accuracy)
Auto-sklearn	yes	0.90/0.13/0.88 (0.87)
—	no	0.88/0.15/0.87 (0.85)
TPOT	yes	0.90/0.16/0.87 (0.84)
—	no	0.88/0.15/0.87 (0.86)
Wunderlich et al. (2019)	yes	0.95/0.16/NA (0.85)

AutoML algorithm could produce pipelines that tend towards classifying new segments as benign. There are a number of approaches to cope with imbalanced data. For example, in the context of AutoML we could choose a metric that is capable of expressing the performance of a pipeline on imbalanced data. However, to be as comparable as possible to Wunderlich et al. (2019), we opt for the additional preprocessing step and use random oversampling of the underrepresented class. Thus, we duplicate malicious segments randomly until the dataset is balanced.

Lastly, some of our experiments apply one-hot encoding (OHE) (cf. Zheng and Casari (2018)) to respect the categorical nature of system calls. The representation of system calls in terms of integer identifiers implies an order on those system calls. However, there is no obvious order on the set of system calls supported by a kernel. OHE is a simple, state-of-the-art approach to cope with this and it produced the best results of the approaches evaluated by Wunderlich et al. (2019). OHE substitutes each system call of the segments with a vector of the size of all possible system calls, where each position represents a system call. Here, all entries of this vector are zero except the one at the position dedicated to the corresponding system call. The resulting vectors are not ordered. As such, this step potentially increases the performance of a classifier.

## 6.2 Settings and Results

All of our experiments were executed on a virtual machine with Ubuntu 18.04.4, eight cores (Intel Xeon E5-2,695 v3), and 128 GB RAM. Furthermore, we use version 0.6.0 of auto-sklearn and version 0.11.1 of TPOT. We try to keep the configuration of the frameworks as simple as possible. We set the `n_jobs` parameter of both frameworks to eight, hereby, enabling the utilization of all cores. Furthermore, we set the `per_run_time_limit` parameter of auto-sklearn and the comparable `max_eval_time_mins` parameter of TPOT to 30 minutes, limiting the time spend for the evaluation of a certain configuration. The parameters limiting the overall runtime of the frameworks are set to 24 h.

We conducted the experiments on encoded and non-encoded data (cf. **Section 6.1**). Generally, both frameworks have the capabilities to optimize the data preprocessing. By default, TPOT uses OHE only for features that have less than eleven unique values, rendering it inapplicable in our case. Thus, providing encoded and non-encoded datasets is a straightforward way to increase the comparability of the frameworks. To account for the usage of sparse matrices in the case of encoded datasets, we have to set the `config_dict` parameter of TPOT to TPOT sparse.

**Table 3** presents the results of our experiments. We report on two runs of each AutoML Framework. One run on encoded and one run on non-encoded data. Furthermore, we show the best results of the experiments presented by Wunderlich et al. (2019) to give the reader a comparison with a manually configured classifier. Please note that these results are not obtained with the goal of constructing the best possible classifier but with the objective of comparing different representations of system calls. Thus, they may not accurately represent the best classifier a human machine learning expert would configure. Although the authors also use ADFA-LD as a basis, their dataset is not strictly the same as used in this publication. Wunderlich et al. (2019) perform a randomized split of the attack traces where half of these traces are added to the training and the other half to the validation dataset<sup>2</sup>. As such, the dataset construction is not completely reproducible for us.

For each of these runs, we show the true positive rate (TPR), false positive rate (FPR), balanced accuracy, and the accuracy. For the runs of the AutoML frameworks these values represent the performance of the constructed pipeline. We present these pipelines in the supplementary material of this publication. In our setting, a higher TPR relates to a better detection of attacks and a lower FPR relates to fewer false alarms. Furthermore, in contrast to balanced accuracy, accuracy is not a valid metric due to the unbalanced validation dataset. However, we include it because the balanced accuracy of the experiments of Wunderlich et al. (2019) is not available (marked as NA in **Table 3**).

Generally, the results produced by utilizing the AutoML frameworks are similar to those produced by the manual configuration of the ML pipeline. Our best run uses auto-sklearn on encoded data. Compared to Wunderlich et al. (2019) it performs slightly worse in terms of the TPR but also slightly better in terms of the FPR. Furthermore, the results of the AutoML runs do not defer to a large margin from each other. However, they indicate that the preceding encoding of the dataset

<sup>2</sup>As stated in personal communication

had positive effects on the performance. This is not surprising as we essentially add a meaningful preprocessing step that the AutoML framework does not have to find by itself.

### 6.3 Discussion

AutoML frameworks do not yet support our problem setting by default. Thus, we conducted basic steps in the data preprocessing to tailor the dataset to the needs of the frameworks. Given these circumstances, the results absolutely encourage to further investigate the utilization of AutoML in the context of application-aware HIDS. In terms of their performance, the ML pipelines configured by the frameworks are comparable to the manually configured one of Wunderlich et al. (2019). However, they are useable by domain experts that are not experts in ML. Furthermore, they automate cumbersome and time-consuming tasks, which is a benefit even for ML experts. This is particularly important for application-aware HIDS approaches, as the effort for the manual configuration of ML pipelines for each version of an application will most likely hinder the employment of these approaches in practice.

Please note that this section reports on our first experiments and does neither represent 1) a final solution for the construction of ML pipelines for application-aware HIDSs for the automotive sector nor 2) a thorough evaluation of the AutoML frameworks. This is due to several reasons.

Regarding the former point, we do not utilize a dataset collected from an automotive system but ADFA-LD (Creech and Hu, 2013). As mentioned before, this is due to the simple fact that public automotive datasets for the evaluation of HIDSs do not exist and are also not easily constructible. However, ADFA-LD comprises sequences of System Calls. This data source is also available on the modern and flexible automotive systems mentioned in **Section 1**. Nevertheless, we do not assume that ADFA-LD and the covered attacks are representable for automotive systems. Additionally, our setup does not consider the resource consumption of the constructed pipeline, which can be a crucial factor particularly in the context of algorithm selection for resource-constrained systems. Moreover, the steps described in this section require a training dataset that encompasses benign and malicious behavior. These are not easily available in many if not most productive settings. Additionally, the system calls of ADFA-LD cannot be associated to particular processes (Xie and Hu, 2013). As such, ADFA-LD is designed to evaluate system-wide, host-based IDS approaches and does not target application-aware approaches. We expect that the AutoML frameworks produce better results on application-aware datasets due to the expected lower variance in the benign behavior.

Regarding the latter point, we do not follow best practices for comparing AutoML tools (cf. Lindauer and Hutter (2019)) but want to give an intuition on their applicability in the context of intrusion detection. Furthermore, the comparison with the manually configured classifier of Wunderlich et al. (2019) is not meant to be used in terms of a thorough evaluation due to the reasons mentioned in **Section 6.2**.

## 7 FEATURE MODELS IN SYSTEMATIC LITERATURE REVIEWS

This section describes how we used feature models to represent and relate the different IDS approaches found in the SLR. As already mentioned, feature models were first introduced by Kang et al. (1990) and developed further by Czarnecki et al. (2005). Originally intended to visualize feature trees for software systems, feature models can be used to systematically represent different entities according to their differentiating features in any domain. By representing features in this way, commonalities and differentiations between entities can be identified.

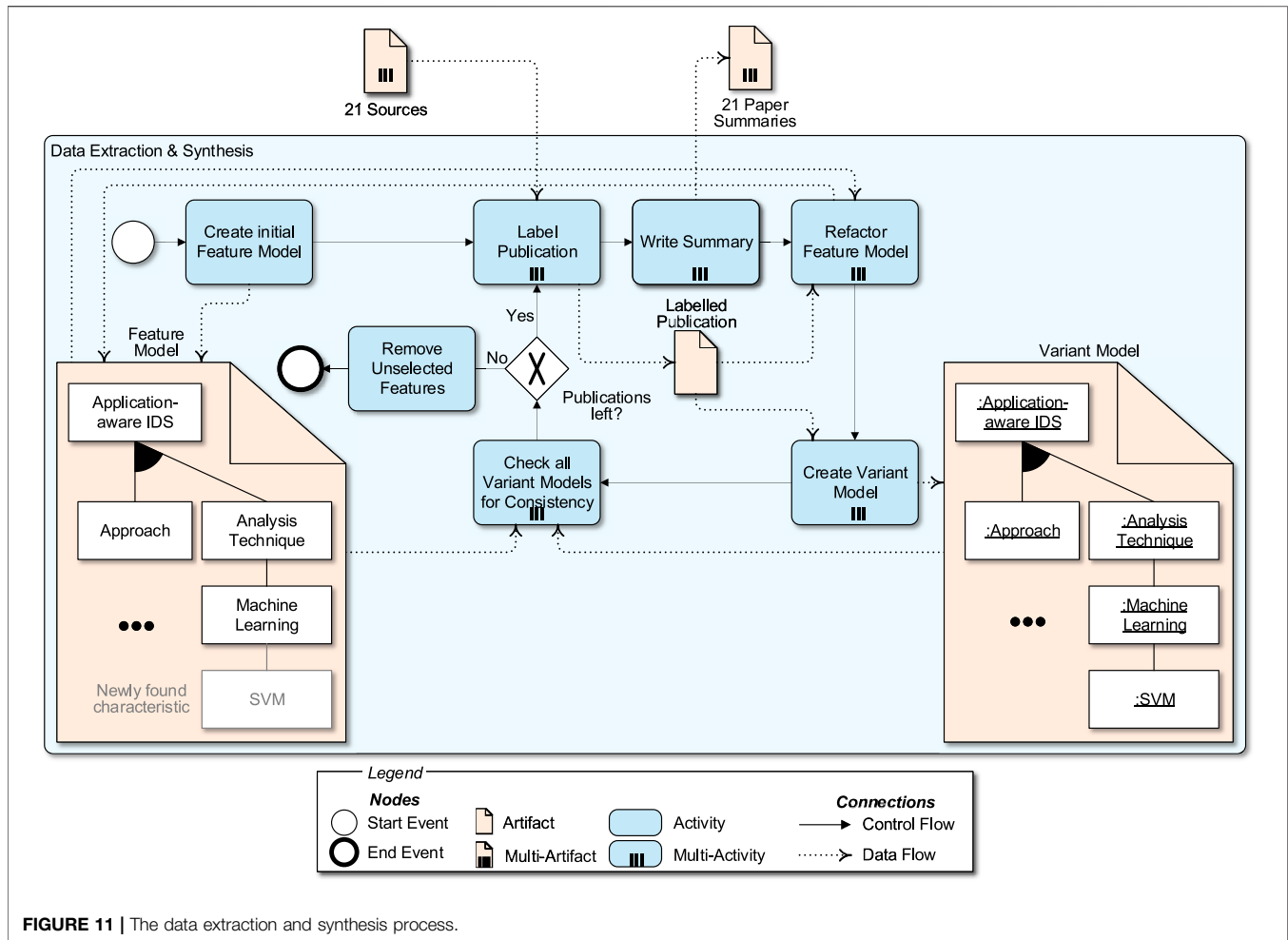
The different syntactical elements of feature models are presented in **Figure 2** and described in the data extraction and synthesis description in **Section 3.10**. In short, feature models describe the features an entity can have. This entity is represented by the root node of the feature model, which is in the form of a tree. A node in a feature model represents a certain feature, which can have an arbitrary number of child nodes connected in groups. Within these groups, the logic of Boolean operators can be used between the features contained in the group. The exemplary partial feature model in 11, for example, shows a grouping using the Boolean “OR”, requiring one or more features within the group to be fulfilled.

**Section 7.1** presents the process employed to create and refine the feature model as more sources are added. In **Section 7.2**, we discuss the experience of working with feature models to facilitate the data synthesis in an SLR.

### 7.1 Application

The goal of creating the feature model is the creation of a taxonomy and to simplify reasoning over the different IDS approaches by visualizing and clearly structuring the characteristics and the relations between them. This facilitates the understanding of the data by fitting it into a model with defined syntax and semantics. The complete feature model contains all possible features, that is, in this case all important differentiating characteristics of application-aware HIDS. Thus, it is a representation of all possible application-aware HIDS approaches found in the SLR. When the features of one entity, in this case an HIDS approach, are represented as specified by the feature model in a tree format, this tree represents a *configuration* of features, also called *variant model*. The relationship between feature model and variant model is analogous to classes and objects in the unified modeling language used in software engineering. The variant model can be understood as being an instance of the feature model, containing the features of a specific entity, in this case an application-aware HIDS approach, fulfills. The variant models must be consistent to the variability constraints of the feature model. For example, optional features may be missing in the variant model, whereas mandatory features must be contained. Comparing two variant models allows to quickly identify the distinguishing features between two classified entities.

**Figure 11** depicts our iterative process of creating feature and variant models. As such, it shows the internals of the Data Extraction and Synthesis subprocess of **Figure 1**. Firstly, we created the initial version of the feature model based on the



existing taxonomies discovered in our preliminary research, by Lazarevic et al. (2005); Luh et al. (2017); Scarfone and Mell (2007).

Secondly, we examine the publication that is to be added and label it by the characteristics of the presented IDS approach. In the case of characteristics that we encountered in a previously examined publication, we label them with the feature from the feature model representing the characteristic. We introduce new labels for characteristics that we encounter for the first time. These labels correspond to features that have to be added to the feature model eventually. Thirdly, we write the summary of the approach and focus on the characteristics that we labeled.

In the next step, the new features are added to the overall feature model. This may lead to a simple extension of the feature model by a feature in a certain branch or trigger the need for a refactoring, in which sub-groups are moved or introduced for example. In a fifth step, we create a variant model for the examined publication, including the selection of the newly found features.

Lastly, a check for consistency is performed for each already existing variant model. Since changes were made to the feature model by adding new features, publications that were labeled at the beginning may have a variant model no longer consistent with

the feature model. In our case, the feature model tooling is able to check automatically the consistency of variant models to the feature model.

Found inconsistencies are removed and the inconsistent publications are relabeled. If this leads to changes in the overall feature model again, more consistency checks are performed until all publications are labeled correctly and the corresponding variant models are consistent with the feature model. In a last step, features that are still present in the feature model but not selected by any variant model are removed.

## 7.2 Lessons Learned

During the creation of the feature model used in this article and our previous work (Schubert et al., 2019), a few lessons became apparent. In general, we perceive feature models as a good way of representing the data extracted during the SLR, especially in cases where qualitative data is to be examined. Having the feature model helps in having consistent data extraction and, thus, consistent reasoning on the extracted data. In our case, the consistency check during the labeling of publications found inconsistencies quite often, which may have gone unnoticed in a manual labeling approach. Such inconsistencies would have had a direct result

on the validity of the SLR results, since the number of times a certain inconsistently labeled characteristic appears would have been incorrect or features could have been selected in an inconsistent manner. Therefore, the feature model helped ensure the validity of the interpretation of the SLR. The feature model can also serve as a guide when examining a publication, since it points to details that are especially noteworthy. It also becomes immediately obvious when an approach is unusual, since it does not fit the feature model, requiring adaptation.

However, feature models are not without drawbacks. Firstly, the iterative process of creating the model can be difficult, since adding features can result in a complete refactoring, requiring large changes to the model. It is also not advisable to create the model without tooling that supports feature models and can check for consistency between the feature model and variants. With manual feature model creation, it is very difficult to ensure consistency and the results would be prone to errors, making the use of the feature model useless.

## CONCLUSION AND FUTURE WORK

This article presents the results of our SLR concerning application-aware HIDS as introduced in our previous work (Schubert et al., 2019) and adds further aspects. We filtered the initial 844 publications and conceived a detailed taxonomy in terms of a feature model with 148 features that classifies 21 current publications. We use this taxonomy to answer our research questions, summarize the key findings, and infer implications toward the realization of productive application-aware HIDSs in an automotive context. In this article, we document the underlying review procedure in more detail than in our previous work (Schubert et al., 2019). Furthermore, we introduce our lessons learned about the application of feature models for SLRs. Finally, we conducted experiments and report on preliminary results on using AutoML for supporting HIDS developers on the task of configuring analysis techniques based on machine learning.

The readers get detailed answers to our research questions, and they can use our taxonomy, the supplementary material, and the classified publications as a basis for further information. Particularly, the taxonomy represents the structure and the variation points of current application-aware HIDSs approaches and thus helps in understanding new approaches. Furthermore, the underlying variant models are detailed classifications of the publications in the final pool. The reader interested in a certain feature can therefore inspect the supplementary material and find all publications that select this feature, as well as summaries of these publications. These publications can serve as a starting point to dive deeper into the topic by, e.g., snowballing.

Moreover, our key findings and implications for the automotive sector provide a management summary of the main aspects of application-aware HIDSs as well as an outline for their automotive utilization. Researchers who focus on application-aware HIDS technology in general can benefit from our results due to the currency of the reviewed approaches. Our approach of using feature models for the classification enables adapting and extending our taxonomy for the purpose of elaborate SLRs, and

the lessons learned provide guidance for the application. In addition, our experiments regarding the applicability of AutoML yield that its performance is nearly as good as the application of conventional ML approaches. However, the HIDS developer gets a systematic approach with much more automation and usable by non-ML experts.

Our inferred implications do not yet form a complete research agenda towards HIDS technology for automotive systems. Thus, the most fundamental future work is to concertize domain-specific requirements on this technology. Furthermore, we will focus on several aspects mentioned in the inferred implications. In particular, we see a great potential in the utilization of AutoML in the context of application-aware HIDS. Thus, we will work on the limitations discussed in **Section 6.3**. Additionally, we will assess behavior-specification-based techniques, which we consider as being a great research opportunity in the automotive context.

## DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/**Supplementary Material**, further inquiries can be directed to the corresponding author.

## AUTHOR CONTRIBUTIONS

All authors worked collaboratively on this publication. This involves many discussions and mutual reviews. Other than that, each author was responsible for certain sections. DS was responsible for **Section 2**, **Section 4**, **Section 5**, and **Section 6**. HE was responsible for **Section 3** and **Section 7**. JH was responsible for the abstract, **Section 1**, **Section 8**, and the creation of **Figure 1**.

## FUNDING

This work has been partially supported by the ITEA 3 APPSTACLE project funded by the German Federal Ministry of Education and Research (No. 01IS16047I) and the SecureIoT project funded by the European Union's Horizon 2020 research and innovation program (No. 779899).

## ACKNOWLEDGMENTS

We want to thank the review board consisting of Ben Hermann, Byron Hawkins, Johannes Geismann, and Matthias Becker. Furthermore, we want to thank Markus Fockel and Christopher Gerking for their feedback and reviews of drafts of this publication.

## SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: <https://www.frontiersin.org/articles/10.3389/fcomp.2021.567873/full#supplementary-material>

## REFERENCES

- Ali, T., Nauman, M., and Jan, S. (2017). Trust in Iot: Dynamic Remote Attestation through Efficient Behavior Capture. *Cluster Comput.* doi:10.1007/s10586-017-0877-5
- Arshad, S., Shah, M. A., Wahid, A., Mehmood, A., Song, H., and Yu, H. (2018). SAMADroid: a Novel 3-level Hybrid Malware Detection Model for Android Operating System. *IEEE Access* 6. doi:10.1109/access.2018.2792941
- Azmi, R., and Pishgoo, B. (2013). SHADuDT: Secure Hypervisor-Based Anomaly Detection Using Danger Theory. *Comput. Security* 39. doi:10.1016/j.cose.2013.08.005
- Bace, R., and Mell, P. (2001). *Intrusion Detection Systems*. Gaithersburg, MD: NIST Special Publication.
- Balaji, A., and Allen, A. (2018). Benchmarking Automatic Machine Learning Frameworks. arXiv preprint arXiv:1808.06492.
- Bidoki, S. M., Jalili, S., and Tajoddin, A. (2017). PbMMD: a Novel Policy Based Multi-Process Malware Detection. *Eng. Appl. Artif. Intelligence* 60. doi:10.1016/j.engappai.2016.12.008
- Brereton, P., Kitchenham, B. a., Budgen, D., Turner, M., and Khalil, M. (2007). Lessons from Applying the Systematic Literature Review Process within the Software Engineering Domain. *J. Syst. Softw.* 80. doi:10.1016/j.jss.2006.07.009
- Bruneau, G. (2001). The History and Evolution of Intrusion Detection. *SANS Inst.* 1.
- Buczak, A. L., and Guven, E. (2016). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Commun. Surv. Tutorials* 18. doi:10.1109/comst.2015.2494502
- Creech, G., and Hu, J. (2013). Generation of a New Ids Test Dataset: Time to Retire the Kdd Collection. In 2013 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 4487–4492. doi:10.1109/wcnc.2013.6555301
- Czarnecki, K., Helsen, S., and Eisenecker, U. (2005). Formalizing Cardinality-Based Feature Models and Their Specialization. *Softw. Process: Improvement Pract.* 10. doi:10.1002/spip.213
- Elsabagh, M., Barabá, D., Fleck, D., and Stavrou, A. (2018). On Early Detection of Application-Level Resource Exhaustion and Starvation. *J. Syst. Softw.* 9404. doi:10.1016/j.jss.2017.02.043
- Esfahani, N., Yuan, E., Canavera, K. R., and Malek, S. (2016). Inferring Software Component Interaction Dependencies for Adaptation Support. *ACM Trans. Autonomous Adaptive Syst.* 10. doi:10.1145/2856035
- Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and Robust Automated Machine Learning. In Advances in neural information processing systems. 2962–2970.
- Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A. (1996). A Sense of Self for Unix Processes. In Proceedings IEEE Symposium on Security and Privacy. IEEE.
- Gu, Z., Pei, K., Wang, Q., Si, L., Zhang, X., and Xu, D. (2015). Leaps: Detecting Camouflaged Attacks with Statistical Learning Guided by Program Analysis. In Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. IEEE. doi:10.1109/dsn.2015.34
- Han, Y., Etigowni, S., Liu, H., Zonouz, S., and Petropulu, A. (2017). Watch Me, but Don't Touch Me! Contactless Control Flow Monitoring via Electromagnetic Emanations. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (ACM), CCS '17 doi:10.1145/3133956.3134081
- Harman, M., and Jones, B. F. (2001). Search-based Software Engineering. *Inf. Softw. Tech.* 43. doi:10.1016/s0950-5849(01)00189-6
- Hutter, F., Kotthoff, L., and Vanschoren, J. (2019). *Automated Machine Learning*. Springer.
- Inoue, H., and Somayaji, A. (2007). Lookahead Pairs and Full Sequences: a Tale of Two Anomaly Detection Methods. In Proceedings of the 2nd Annual Symposium on Information Assurance. 9–19.
- Islam, S., Khreich, W., and Hamou-Lhadj, A. (2018). Anomaly Detection Techniques Based on Kappa-Pruned Ensembles. *IEEE Trans. Reliability* 67. doi:10.1109/tr.2017.2787138
- Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., and Peterson, A. S. (1990). *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Technical report. Carnegie-Mellon University Software Engineering Institute.
- Khreich, W., Khosravifar, B., Hamou-Lhadj, A., and Talhi, C. (2017). An Anomaly Detection System Based on Variable N-Gram Features and One-Class Svm. *Inf. Softw. Tech.* 91. doi:10.1016/j.infsof.2017.07.009
- Khreich, W., Murtaza, S. S., Hamou-Lhadj, A., and Talhi, C. (2018). Combining Heterogeneous Anomaly Detectors for Improved Software Security. *J. Syst. Softw.* 137. doi:10.1016/j.jss.2017.02.050
- Kitchenham, B., and Charters, S. (2007). *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. Tech. rep. Durham, England: Keele University and Durham University.
- KPMG (2017). KPMG's 18<sup>th</sup> Consecutive Global Automotive Executive Survey.
- Lazarevic, A., Kumar, V., and Srivastava, J. (2005). Intrusion Detection: A Survey. *Managing Cyber Threats*, 19–78.
- Le, T. T., Fu, W., and Moore, J. H. (2020). Scaling Tree-Based Automated Machine Learning to Biomedical Big Data with a Feature Set Selector. *Bioinformatics* 36, 250–256. doi:10.1093/bioinformatics/btz470
- Lindauer, M., and Hutter, F. (2019). Best Practices for Scientific Research on Neural Architecture Search.
- Loukas, G., Karapistoli, E., Panaousis, E., Sarigiannidis, P., Bezemskij, A., and Vuong, T. (2019). A Taxonomy and Survey of Cyber-Physical Intrusion Detection Approaches for Vehicles. *Ad Hoc Networks* 84. doi:10.1016/j.adhoc.2018.10.002
- Lu, S., and Lysecky, R. (2019). Data-driven Anomaly Detection with Timing Features. *ACM Trans. Des. Automation Electron. Syst. (Todaes)* 24. doi:10.1145/3279949
- Luh, R., Marschalek, S., Kaiser, M., Janicke, H., and Schrittwieser, S. (2017). Semantics-aware Detection of Targeted Attacks: a Survey. *J. Comp. Virol. Hacking Tech.* 13. doi:10.1007/s11416-016-0273-3
- Mansour, C., and Chasaki, D. (2019). Adaptive Security Monitoring for Next-Generation Routers. *EURASIP J. Embedded Syst.* doi:10.1186/s13639-018-0087-0
- Masri, W., Abou Assi, R., and El-Ghali, M. (2014). Generating Profile-Based Signatures for Online Intrusion and Failure Detection. *Inf. Softw. Tech.* 56. doi:10.1016/j.infsof.2013.09.004
- Meyer, D., Leisch, F., and Hornik, K. (2003). The Support Vector Machine under Test. *Neurocomputing* 55. doi:10.1016/s0925-2312(03)00431-4
- Mitchell, R., and Chen, I.-R. (2014). A Survey of Intrusion Detection Techniques for Cyber-Physical Systems. *ACM Comput. Surv.* 46. doi:10.1145/2542049
- Object Management Group (2014). *Business Process Model and Notation (BPMN), Version 2.0.2*. OMG Document Number: formal/2013-12-09.
- Reeves, J., Ramaswamy, A., Locasto, M., Bratus, S., and Smith, S. (2012). Intrusion Detection for Resource-Constrained Embedded Control Systems in the Power Grid. *Int. J. Crit. Infrastructure Prot.* 5. doi:10.1016/j.ijcip.2012.02.002
- Scarfone, K., and Mell, P. (2007). Guide to Intrusion Detection and Prevention Systems (IDPS). *Tech. rep., NIST Spec. Publ.*, 800–894. doi:10.6028/nist.sp.800-94
- Schubert, D., Eikerling, H., and Holtmann, J. (2019). Application-aware Intrusion Detection: A Systematic Literature Review and Implications for Automotive Systems. In Proceedings of the 17th European Conference on Embedded Security in Cars (escar Europe). Bochum, Germany: Ruhr-University Bochum.
- Shu, X., Yao, D. D., Ramakrishnan, N., and Jaeger, T. (2017). Long-span Program Behavior Modeling and Attack Detection. *ACM Trans. Privacy Security (Tops)* 20. doi:10.1145/3105761
- Tajoddin, A., and Abadi, M. (2019). Berlin, Germany: Applied Intelligence. doi:10.1007/s10489-018-01405-0RAMD: Registry-Based Anomaly Malware Detection Using One-Class Ensemble Classifiers.
- Viljanen, L. (2005). *A Survey of Application Level Intrusion Detection*. Tech. Rep. C-2004-61. Helsinki, Finland: University of Helsinki, Department of Computer Science.
- Wagner, D., and Soto, P. (2002). Mimicry Attacks on Host-Based Intrusion Detection Systems. In Proceedings of the 9th ACM Conference on Computer and Communications Security. New York, USA: ACM. CCS '02. doi:10.1145/586110.586145
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2000). *Experimentation in Software Engineering: An Introduction*. Springer US.
- Wunderlich, S., Ring, M., Landes, D., and Hotho, A. (2019). Comparison of System Call Representations for Intrusion Detection. In International Joint Conference: 12th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2019) and 10th International Conference on European Transnational Education (ICEUTE 2019). Springer, 14–24. doi:10.1007/978-3-030-20005-3\_2

- Xie, M., and Hu, J. (2013). Evaluating Host-Based Anomaly Detection Systems: A Preliminary Analysis of Adfa-Ld. In 2013 6th International Congress on Image and Signal Processing (CISP), 3. IEEE, 1711–1716. doi:10.1109/cisp.2013.6743952
- Xie, Y., Feng, D., Tan, Z., and Zhou, J. (2016). Unifying Intrusion Detection and Forensic Analysis via Provenance Awareness. *Future Generation Comp. Syst.* 61. doi:10.1016/j.future.2016.02.005
- Zegzhda, P. D., Kort, S. S., and Suprun, A. F. (2015). Detection of Anomalies in Behavior of the Software with Usage of Markov Chains. *Automatic Control. Comp. Sci.* 49. doi:10.3103/s0146411615080386
- Zhang, M., Raghunathan, A., and Jha, N. K. (2014). A Defense Framework against Malware and Vulnerability Exploits. *Int. J. Inf. Security* 13. doi:10.1007/s10207-014-0233-1
- Zheng, A., and Casari, A. (2018). *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. Newton, MA: O'Reilly Media, Inc.
- Zimmer, C., Bhat, B., Mueller, F., and Mohan, S. (2010). Time-based Intrusion Detection in Cyber-Physical Systems. In Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems. New York, USA: ACM. doi:10.1145/1795194.1795210
- Zonouz, S., Houmansadr, A., Berthier, R., Borisov, N., and Sanders, W. (2013). Secloud: A Cloud-Based Comprehensive and Lightweight Security Solution for Smartphones. *Comput. Security* 37. doi:10.1016/j.cose.2013.02.002

**Conflict of Interest:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

**Publisher's Note:** All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Copyright © 2021 Schubert, Eikerling and Holtmann. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.