An Extended Hybrid Anomaly Detection System for Automotive Electronic Control Units Communicating via Ethernet

Efficient and Effective Analysis using a Specification- and Machine Learning-based Approach

Daniel Grimm[1], Marc Weber[2] and Eric Sax[1]

[1] *Institute for Information Processing Technologies, Karlsruhe Institute of Technology, Karlsruhe, Germany*

[2] *Vector Informatik GmbH, Stuttgart, Germany*

Keywords: Intrusion Detection System, Anomaly Detection, Machine Learning, Automotive, Ethernet, Feature Selection.

Abstract: Due to the increasing number of functions fulfilled by ECUs in a vehicle, there is a need for new networking technologies offering more bandwidth than e.g. Controller Area Network. Automotive Ethernet is one of the most promising candidates and already used in modern cars. However, currently there is the open issue of detecting and preventing cyber attacks via this well known networking technology. In this paper we present the extension of our hybrid anomaly detection system for ECUs to improve the security and safety of vehicles using Automotive Ethernet. The system combines specification- and machine learning-based anomaly detection methods. The features, necessary for the machine learning part, are selected to enable the detection of anomalies in real-time and with respect to the automotive specific communication scheme. Finally, the detection performance and the applicability of different machine learning algorithms is evaluated in a simulation environment based on synthetic and well defined anomalies.

1 INTRODUCTION

The evolution of vehicles is clearly drawn. Electronic control units (ECUs) have to realize an increasing number of functions. Currently, the more than 2000 functions of a modern car are realized with 10.000.000 lines of code running on more than 70 ECUs (Broy, 2006). Driving will be automated by the use of diverse technologies such as laser scanners, radar, cameras and ultrasonic sensors. The growth of functions and sensors lead to a massive increase of produced data that has to be transmitted between the different ECUs. Currently, Controller Area Network (CAN) is the predominant communication system for in-vehicle networks. Because CAN only provides a data rate of up to 1 Mbit/s other technologies have to be used in future. Ethernet, which is well-known from information technology, offers high bandwidth at low costs and great flexibility. For the in-vehicle use case a special physical layer, called 100BASE-T1 (IEEE Std 802.3bw-2015), was specified offering 100 Mbit/s full-duplex communication over a single unshielded twisted pair cable. The specification for the next generation, which offers 1 Gbit/s, is already ready (IEEE Std 802.3bp-2016) and there are first projects based on this technology. In future, a further increase of the provided bandwidth is expected, as the next standardizations are in the focus of interest already (IEEE-P802.3ch). Besides the increase of communication between ECUs, future vehicles will communicate with their environment through different wireless interfaces like WiFi or Bluetooth. Hence, vehicles are no longer a closed system but are exposed to (remote) cyber attacks. As ECUs are able to control the whole behavior of a vehicle, including steering, braking and acceleration, such attacks on the electrical and electronic system of a vehicle can have fatal effects. One possibility to improve the security of ECUs is the use of intrusion detection (IDS) or intrusion prevention systems (IPS) (Hoppe et al., 2009; Kleberger et al., 2011; Larson et al., 2008).

On this account, we presented a hybrid IDS combining specification-based detection mechanisms and machine learning algorithms to recognize irregularities in automotive CAN communication (Weber et al., 2018). Checking for irregularities instead of predefined attack patterns enables the detection of attacks unknown during IDS development time. However, as CAN is not the only in-vehicle networking technology, the introduced hybrid anomaly detection system

Grimm, D., Weber, M. and Sax, E.
An Extended Hybrid Anomaly Detection System for Automotive Electronic Control Units Communicating via Ethernet.
DOI: 10.5220/0006779204620473
In *Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2018)*, pages 462-473
ISBN: 978-989-758-293-6
Copyright © 2019 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

has to be extended to support other communication channels as well. Considering the increasing importance of Automotive Ethernet, we decided to include this in-vehicle networking technology in our research as a next step.

An irregularity in automotive communication systems can occur in different ways. One possibility is an irregular payload of messages, e.g. a signal value which is outside its specified minimum and maximum. Other potential irregularities concern the overall communication behavior of ECUs. Independent of the contained application data, a message can be irregular if, for example, the sending ECU is not allowed to transmit data to a given receiver. A second example for a behavior-related irregularity is a message that contains data of unknown protocols. The existing hybrid IDS for CAN is capable to recognize irregularities in communication signals. These mechanisms can be applied on Automotive Ethernet as well. For that reason, this paper focuses on the detection of behavior-related irregularities in Automotive Ethernet communication. The transported application data of exchanged messages can be analyzed further with the existing solution, independent of the used communication system.

The remaining paper is structured as follows: Section 2 introduces the term *anomaly* and its different types, followed by the discussion of related work in section 3. Afterwards, the extended hybrid anomaly detection system is presented in section 4 outlining its basic building blocks. Since machine learning is applied, section 5 gives an overview about the selected algorithms which are used to detect anomalies in Ethernet-based communication. Their implementation is discussed in section 6 together with the other extensions realized in context of the anomaly detection system. Section 7 presents first evaluation results based on a simulation before section 8 concludes the paper with a summary and a proposal for future research.

2 IRREGULARITIES AND ANOMALIES

Irregular behavior is an undesirable deviation of the expected characteristics. In literature, this is referred to as anomaly or outlier in addition to other denotations. One definition that is frequently cited is given by D.M. Hawkins (Hawkins, 1980):

> *"An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism."*

Chandola et al. (2009) subdivided anomalies into three groups, which is helpful to develop a method for recognition of irregular behavior. The simplest form of an anomaly is the point anomaly, defined by Chandola et al. as follows (Chandola et al., 2009):

> *"If an individual data instance can be considered as anomalous with respect to the rest of data, then the instance is termed as a point anomaly."*

If e.g. an Ethernet network traffic log - also called trace - is the complete data set, a single recorded Ethernet message is one data instance. If that single message can be classified as anomaly without considering other data instances, e.g. because it contains the information that the destination address is unknown, it is a point anomaly. In contrast to the point anomaly, a contextual anomaly can only be classified as such, if additional contextual information is taken into account (Chandola et al., 2009). A destination address of a known ECU sent from an also known source address is not a point anomaly since it is realistic in the first place. However, if the message has to be sent only as an answer to a preceding request message and this request is absent, it can be declared as a contextual anomaly. The last type is the collective anomaly, referring to a sequence of data instances, which together form an anomaly (Chandola et al., 2009). This would be the case if two or more consecutive Ethernet messages indicate anomalous communication behavior, such as an increasing frequency of a specific message.

3 RELATED WORK

Most of the research concerning irregular in-vehicle communication was carried out on the detection of anomalies in CAN messages. For example, Marchetti et al. (2016) presented a method based on the Shannon entropy of the payload data. If the entropy of the payload data is outside of some pre-defined boundaries, a CAN message is considered anomalous. Taylor et al. (2015) compared different methods to detect inserted packets on the CAN bus, based on the occurrence frequency of different messages. A One-Class Support Vector Machine (OCSVM) and a statistical test were used for that purpose. In contrast, there are no known publications concerning anomaly detection in Automotive Ethernet communication.

Outside the automotive domain numerous methods have been published to address network IDS for communication based on the Internet Protocol (IP), which is mostly used on top of Ethernet networks.

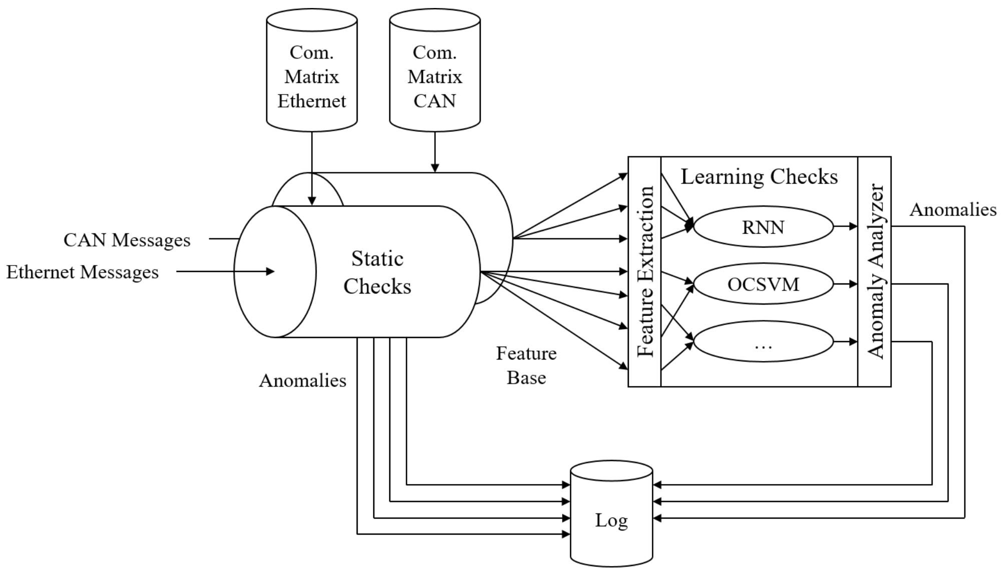

Figure 1: Block diagram of the extended hybrid anomaly detection system for CAN and Ethernet.

Chandola et al. (2009), Ahmed et al. (2016) and Aggarwal (2017) provide a detailed overview of the state of the art detection techniques, which include machine learning algorithms as well as statistical methods besides others. However, because of the different nature of the examined data the selection of applicable methods has to be conducted with care. In addition, one focus of an anomaly detection system for ECUs has to be the capability to perform online analysis, whereas most of the work concerning network IDS is based on offline analysis.

4 EXTENDED HYBRID ANOMALY DETECTION SYSTEM

In the following section an overview about the extended hybrid anomaly detection system for CAN and Ethernet is given. The main contribution of this work is the development of the parts necessary to extend our existing system to realize anomaly detection for Ethernet. These are the specific blocks *static checks*, *feature base* and *feature extraction* as well as the *learning checks* which are suited for the evaluation of Ethernet-based communication. The aforementioned terms are introduced below to give an outline of the overall function. Details on the developed blocks are presented further in sections 5 and 6.

The basic idea of the proposed system is to use specification-based anomaly detection and machine learning algorithms sequentially within the embedded software of an ECU. Figure 1 depicts the principle building blocks of the two-stage system. As this is an extension to the existing system described in previous research (Weber et al., 2018), Figure 1 is also based on the existing graphics.

Due to resource constraints on ECUs, machine learning should only be used if necessary. Therefore, specification-based techniques are used in the first stage as much as possible, re-using the knowledge provided by vehicle manufactures in terms of the communication matrix. This stage is further referred to as *static checks*. Static checks are very well suited to detect point anomalies. In addition, simple collective anomalies can be detected efficiently using static checks, as long as they can be derived from the communication matrix. When a message, either via CAN or Ethernet, is sent or received by an ECU, the corresponding static checks evaluate the protocol headers and the signals, contained in the payload section.

In the second stage of the system, *learning checks* extend the detection possibilities, especially for advanced contextual and collective anomalies by using machine learning. To apply the corresponding algorithms, at first relevant information has to be extracted from messages. The static checks forward selected information like protocol header fields to the learning checks, depicted in Figure 1 as *feature base*. The *feature extraction* block, pre-processes the information and generates features, which represent the input data for the following algorithms. Pre-processing can contain multiple aspects, like building time series and normalization.

Since the investigated input data differs between network technologies and anomalous behavior can be reflected in various specific properties, miscellaneous machine learning algorithms are applicable. Therefore, the proposed system allows using a variety of algorithms, working on the same features. Figure 1 shows two examples: Replicator Neural Networks (RNN) (Hawkins et al., 2002) and One-Class Support Vector Machines (OCSVM) (Schölkopf et al., 2001). Each algorithm produces as output either a binary value ('normal'/'anomaly') or a so-called anomaly score, which represents the probability of an anomaly. These produced outputs are evaluated within the *anomaly analyzer* block, which e.g. checks an anomaly score for a defined threshold value before it securely stores the anomaly in a log. The design of the learning checks also allows for ensemble-based methods, as proposed in recent research e.g. by Andreas Theissler (Theissler, 2017). In an ensemble, multiple algorithms run in parallel, checking for the same anomalies and each of them producing its own output. In such a setup, the anomaly analyzer performs a voting between the different outputs and finally decides whether an anomaly is logged or not. This kind of post-processing is not necessary for static checks, since they do not work with probabilities. Therefore, they directly log the detected anomalies together with the corresponding boundary conditions (e.g. which Ethernet message caused the anomaly) to enable an improved post-evaluation.

5 INVESTIGATED MACHINE LEARNING ALGORITHMS

Ethernet is the networking technology which will dominate the development of in-vehicle communication systems in future. Since e.g. autonomous driving is requiring higher data rates for the communication between ECUs, also safety-relevant data will be transmitted via Ethernet and not only via CAN or FlexRay. For that reason, Ethernet messages have to be analysed with respect to the allowed behavior. As outlined above, for example the destination address of a message can create a point anomaly. However, all possible and allowed destination addresses for a given sending ECU can be specified in the communication matrix. Hence, some of the possible anomalies concerning Automotive Ethernet communication can be detected with static checks. The development of these static checks is discussed in detail in section 6.1. On the other hand, only parts of the communication behavior is specified, since it is not possible or it would be too much effort to determine dynamic properties of the communication such as the order of messages beforehand. Though, anomalous communication behavior can also affect characteristics that are not specified. Resulting mostly in collective or contextual anomalies, this misbehavior can be detected with learning checks.

In a first step, algorithms have to be selected, which are suited to be applied in the proposed system as learning checks for anomaly detection in Ethernet communication. For this use case, potentially a lot of different methods can be used. Classification-based techniques use supervised learning and require the availability of so-called labeled data. In this case, training data instances are marked to belong to one defined class. Considering anomaly detection, there are the two classes *normal* and *anomaly*. With this pre-knowledge a classifier is learned, which is able to distinguish normal from anomalous data instances.

In contrast, clustering-based techniques use unsupervised learning and do not require labeled data. They group data instances and thereby try to find the classes *normal* and *anomaly* automatically. However, most of these algorithms require many data instances to be available during runtime in order to perform the grouping task.

Since the proposed anomaly detection system shall be implemented on an embedded ECU, considering the required resources is important. Computing power on an embedded device is limited, compared to office computers or server systems. Following this constraint, the computational complexity of an algorithm must be low when executed on the ECU to detect anomalies. On the other hand, the training of the algorithm may be computational complex as long as it can be performed offline, like on a back-end infrastructure. Afterwards, the trained algorithm has to be transferred to the ECU, e.g. in terms of updating a parameter set. Not only the computing power is limited on an ECU but also the available memory. This means that applied machine learning algorithms must not require much read-only memory (ROM) and, even more important, not much random access memory (RAM). Due to these constraints, clustering-based techniques are excluded from the decision process of finding an appropriate algorithm. There are unsupervised algorithms like Lightweight On-line Detector of Anomalies which also have a comparable low computational complexity and reduced memory consumption. The investigation whether one of these algorithms is also suited for detecting anomalies in Ethernet-based communication on ECUs has to be investigated in future research.

Another point concerning the selection of proper machine learning algorithms is the false alarm rate. Any identified anomaly has to be reported and should be further analyzed e.g. by a human in a Security Information and Event-Management (SIEM) center which causes a lot of additional effort. In future, if countermeasures may be applied within the vehicle itself, false alarms could cause serious effects like unintended and unnecessary emergency braking. Thus, false alarms have to be avoided as far as anyhow possible. As mentioned above, the remaining classification-based techniques require labeled data. However, to obtain labeled data for anomalies in Automotive Ethernet communication in a large scale is difficult or even impossible because of the data collection and labeling effort. Also, training with anomalous data would result in a classifier, which is probably only able to detect known anomalies. This contradicts the idea of anomaly detection, which tries to find any kind of deviation from normal behavior. Therefore, algorithms are used, which only require the presence of normal data instances during the training to perform a so-called one-class classification.

From the plenty of possible methods to use as learning check for Ethernet-based communication, which are described in literature (Chandola et al., 2009; Ahmed et al., 2016; Aggarwal, 2017), three are selected for further evaluation. These are the Mahalanobis Distance, the Principal Component Analysis and the One-Class Support Vector Machine. For each selected algorithm details on its working principle are given below.

5.1 Mahalanobis Distance

The basic assumption of the Mahalanobis Distance (Mahalanobis, 1936) is that the data originates from a multivariate gaussian distribution. Thereby the normal class, represented through the training data, is modeled by the use of one gaussian function. The learning process refers to the estimation of the underlying distribution, including the calculation of mean $\vec{\mu}$ and covariance matrix C of the training data. To define an anomaly score, the Mahalanobis Distance of a new data instance $\vec{x}$ to the mean of the training data is calculated, which is given by equation 1.

$$\mathrm{Mahal}(\vec{x},\vec{\mu},C_{\mathrm{XX}}) = \sqrt{\left((\vec{x}-\vec{\mu})^{\mathrm{T}} C_{\mathrm{XX}}^{-1} (\vec{x}-\vec{\mu})\right)} \quad (1)$$

Data instances with a larger distance to the mean are more likely to be generated by another distribution than the training data and hence, can be classified as anomalies. To use the Mahalanobis Distance as anomaly detection method, a threshold $\vec{\Theta}_{Mahal}$ has to be defined so that data instances with a larger or equal distance than the threshold are considered anomalous.

One big advantage of this method is the ease of use. Except for the specified threshold, there are no parameters which have to be defined.

5.2 Principal Component Analysis

With a Principal Component Analysis (PCA) it is possible to determine dependencies and correlations between dimensions of a feature space represented as the random variables X_1 to X_d. PCA describes the feature space using the principal components as new dimensions, which are a linear combination of the d original dimensions. It is used often to reduce the dimensionality of a given data set. Through an eigenvalue analysis of the covariance or the correlation matrix, the principal components are calculated as the eigenvectors e_1 to e_d. They are uncorrelated and ordered by their variance. The total variance in the new principal component space is the same as in the original space, where the variance of the new dimensions is given by the eigenvalues λ_1 to λ_d with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$.

Shyu et al. (2003) describe how PCA can be used to detect outliers and anomalies in an efficient way. The basic idea is that anomalous data is represented worse than normal data instances through the principal components. A higher deviation from the model of the training data, represented by the principal components and the corresponding eigenvalues, thus indicates anomalous data. They select the p principal components e_1 to e_p with the highest variances and q with the lowest variances, given by e_{d-q+1} to e_d, to get two sub-spaces of lower dimensionality. For any new data instance $\vec{x}$ the distance to the mean $\vec{\mu}$ of the training data is calculated in these two sub-spaces and weighted by the associated eigenvalues to obtain an anomaly score. For the p selected principal components, this is formally depicted by equation 2.

$$d_{PCA}^{p}(\vec{x}) = \sum_{i=1}^{p} \frac{\left(\vec{e}_i^{\mathrm{T}} \cdot \vec{z}\right)^2}{\lambda_i} \quad (2)$$

The vector $\vec{z} = (z_1, z_2, \ldots, z_d)$ is given by scaling of the new data instance according to equation 3, where μ_k and σ_k are the sample mean, respectively the sample variance of the variable X_k.

$$z_k = \frac{x_k - \mu_k}{\sigma_k} \quad (3)$$

Considering the q principal components with lowest variance, the distance is calculated accordingly.

Hence, the training process estimates the covariance or correlation matrix C, sample mean and variance of the training data, followed by the solution of the eigenvalue problem $(C - \lambda I) \cdot \vec{e} = 0$ to obtain the d principal components and eigenvalues. As suggested by Shyu et al. the correlation matrix is used instead of the covariance matrix for the calculation of the principal components since this has the advantage of weighting all features equally. That is especially important if the features are measured on very different scales, which cannot be excluded.

5.3 One-Class Support Vector Machine

The One-Class Support Vector Machine (OCSVM) was introduced by Schölkopf et al. (2001). It is a modification of the well-known Support Vector Machine (SVM) described by Vapnik (1995). A SVM mathematically separates a set of training data $x_i, i = 1 \dots N$ containing data instances of two classes $y \in \{-1, 1\}$ by a linear hyperplane. This classifier is calculated so that the smallest distance of a data instance to the hyperplane is maximized. This smallest distance ρ is denoted as *margin*. The learning process results in the *support vectors* (SVs), which are a particular subset of the training data depicting the hyperplane. Depending on the position of a new data instance x_t in the feature space with respect to the hyperplane it can be classified resulting in the class $y_t \in \{-1, 1\}$. Because a linear hyperplane is not sufficient to split non-linear distributed classes, usually a kernel function $\phi(\vec{x})$ is used. The function performs an implicit mapping to the feature space of the kernel function (kernel-space). Thus, the data instances are separated in a feature space of higher dimension, where a linear hyperplane is sufficient to separate the two classes.

Instead of being trained with data instances of two classes, an OCSVM is trained with data instances of one class only. Therefore, the algorithm is trained as SVM that separates the training data from the origin in the kernel-space to obtain a decision boundary. In the original feature space, this boundary encloses the training data and classifies between the areas of high and low data instance density, representing normal and anomalous data. To attain an anomaly score for a new data instance x_t, its position with respect to the decision boundary is determined. A data instance inside the boundary, which is hence normal, has a negative distance, whereas a data instance outside has a positive distance to the boundary. The learning process refers to the convex quadratic optimization problem, shown in equation 4, with the hyperplane $\vec{w}$ and margin ρ using N training data instances.

$$\min_{w \in F, \vec{\varepsilon} \in \Re^N, \rho \in \Re} \frac{1}{2}\|\vec{w}\|^2 - \rho + \frac{1}{\nu N}\sum_{i=1}^{N} \varepsilon_i \tag{4}$$

The variable $\vec{\varepsilon}$ depicts some small violations of the decision boundary to obtain a smoother classification, so that equation 5 holds true for the hyperplane.

$$\vec{w} \cdot \vec{x}_i \geq \rho - \varepsilon_i, \quad \forall_i x_i, i = 1 \dots N \tag{5}$$

The parameter ν is affecting the number of training data instances becoming support vectors. As the support vectors form the classifier, ν has to be chosen carefully to obtain a decision boundary that provides a sufficient generalization ability. To solve the constrained problem given by equations 4 and 5, Lagrange multipliers α_i are introduced resulting in the dual problem of equation 6 for which the constraints in equation 7 hold true.

$$\min_{\vec{\alpha}} \frac{1}{2}\sum_{i,j=1}^{N} \alpha_i \alpha_j \vec{x}_i \cdot \vec{x}_j \tag{6}$$

$$0 \leq \alpha_i \leq \frac{1}{\nu N} \text{ and } \sum_{i=1}^{N} \alpha_i = 1 \tag{7}$$

Inserting the kernel function $\phi(\vec{x})$ for $\vec{x}$, specified by its cross-product $K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i)\phi(\vec{x}_j)$, this results in equation 8.

$$\min_{\vec{\alpha}} \frac{1}{2}\sum_{i,j=1}^{N} \alpha_i \alpha_j K(\vec{x}_i, \vec{x}_j) \tag{8}$$

Kernel functions are for example polynomial or sigmoid functions, though the gaussian kernel or radial basis function (RBF-kernel) is the most popular given by equation 9.

$$K(\vec{x}_i, \vec{x}_j) = \exp\left(-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{2 \cdot \sigma^2}\right) \tag{9}$$

For the following evaluations, the RBF-kernel is used. Therefore, a suitable value for σ has to be found. Conducting the learning process by minimizing equation 8 the margin ρ and the N α_is are obtained. All $\vec{x}_i$ for which α_i is not equal to zero become support vectors. The anomaly score can be calculated according to equation 10. In contrast to the original formulation of Schölkopf et al. (2001) the minus sign is introduced to ensure that a higher anomaly score is indicating a more anomalous data instance $\vec{x}_t$.

$$f(\vec{x}_t) = -\left(\sum_{i=1}^{N} \alpha_i \cdot K(\vec{x}_i, \vec{x}_t) - \rho\right) \tag{10}$$

An important point when using an OCSVM for anomaly detection is the selection of proper values for the parameters ν and σ. In literature, there exist different ideas how to choose these parameters. They focus mostly on high detection rates, ignoring possible higher false alarm rates. But as outlined above, low false alarm rates are indispensable in automotive context and thus another method for the parameter selection is used. By a grid search with k-fold cross-validation and a stepwise refinement of the parameter range an appropriate value for σ is found, whereas ν is set beforehand. As optimization criterion for the grid search the minimum mean validation error across all folds is used in order to receive a good generalization ability at low false alarm rates.

6 ANOMALY DETECTION SYSTEM FOR ETHERNET-BASED COMMUNICATION

This section provides a detailed view on the design of the basic anomaly detection system elements concerning Ethernet-based communication. At first, the static checks are outlined in subsection 6.1. In accordance with the processing sequence of messages, subsection 6.2 introduces the extracted features used by the following learning checks. Finally, subsection 6.3 discusses the configuration of the presented machine learning algorithms with respect to the developed system.

6.1 Static Checks

For the implementation of static checks a slim model of the underlying system specification (communication matrix) is beneficial. Normally, vehicle manufacturers specify the desired communication on their in-vehicle Ethernet networks in a standardized manner. One example for such a semi-formal specification is the AUTomotive Open System ARchitecture (AUTOSAR) System Description, which is a Extensible Markup Language (XML)-based format. It is difficult to use such files directly to check messages against violations of the specification since these files are usually very large and of complex structure. Thus, another model was developed being able to represent the normal Ethernet communication in a slim form based on a Unified Modeling Language (UML) class diagram. The basic idea of the model is to concentrate all parameters given by specification and which can represent a restriction of allowed Ethernet messages within a network. To allow for efficient anomaly detection it is used as a whitelist model containing only the allowed message parameter values. Because of the whitelist approach implicitly every parameter value that is not included in the model is prohibited. For example, it could be specified that an ECU is allowed to send data with one specific source IP address. In that case, the allowed parameter value is only this single address, whereas every other address is forbidden to use.

As anomaly detection is conducted on a specific ECU, the system is configured especially for it. Therefore, the model describes the communication of the selected ECU with its partners. Since the root element holds the parameters relating to the Ethernet protocol, one instance of the model represents one communication path in the network. In other words, the allowed communication between the selected and one other ECU is modeled. Hence, if there exists communication with e.g. six other ECUs, six different instances of the model are necessary to include all information for the static checks. During normal operation of the anomaly detection system any received and sent Ethernet message is compared to all model instances. If any model is matching to the message regarding all defined parameters such as addresses and packet length, the message is considered normal. Otherwise, if no model instance matches, the message is an anomaly.

6.2 Feature Selection

One very important but mostly ignored challenge is the selection of proper features for the following machine learning algorithms. Suitable features contain all information necessary to distinguish between normal and anomalous data instances. Considering only the payload data of messages in automotive communication, the initial feature selection corresponds to the extraction of single signals from the payload, which can be done based on the communication matrix. Physical signals like the speed of the vehicle can be interpreted as elements of a time series. Hence, the time series itself and further derived features form the dimensions of the input data. On the other hand, the definition of features, which model the overall behavior of a communication, is more challenging and has to be conducted with care. Evaluations of network IDS are frequently based on the KDD Cup Dataset (Stolfo, 1999), which contains data instances as 35-dimensional feature vectors. Unfortunately, the features are designated to be evaluated offline and are specific for Internet applications as http or email.

Therefore, they are not applicable for real-time anomaly detection in automotive networks and it is necessary to define specific adequate features for in-vehicle Ethernet.

As the anomaly detection system is executed on one specific ECU, the features must be defined accordingly. For that purpose the concept of *global* and *local* features is introduced. A categorization into these domains allows for a better analysis of anomalous communication behavior. On the one hand, an anomaly can influence a specific connection between

Table 1: Selected features to describe Automotive Ethernet communication.

Layer	**Scope**	
	Global	Local
MAC	num. of communicating MAC addresses mean packet length in bytes data rate in bytes per second mean time interval between packets entropy of local MAC addresses entropy of remote MAC addresses	mean packet length in bytes mean time interval between packets received bytes per second sent bytes per second packet length of each packet time interval since last packet
IP	entropy of local IP addresses entropy of remote IP addresses num. of communicating IP addresses entropy of the VLAN-ID num. of VLANs on which is communicated	mean packet length in bytes mean time interval between packets received bytes per second sent bytes per second packet length of each packet time interval since last packet
UDP/TCP	entropy of local ports entropy of remote ports num. of used local ports num. of used remote ports	mean packet length in bytes mean time interval between packets received bytes per second sent bytes per second packet length of each packet time interval since last packet additionally for TCP communication: % URG, PUSH, RST, SYN, FIN packets TCP window size

two ECUs, for example a higher frequency of messages, which is therefore a local anomaly on this single connection. On the other hand, anomalous behavior can influence all connections of an ECU, for example if this ECU is sending no messages at all. Thus, this anomaly has a global effect. Considering these two different domains, local and global features can be defined accordingly, describing either one or all connections of an ECU. Another possible categorization is based on the used communication protocols on top of Ethernet like IP, UDP (User Datagram Protocol) and TCP (Transmission Control Protocol). Features can be defined describing the behavior with respect to one specific communication protocol.

Having a look at the KDD Cup features and other recent research (Lakhina et al., 2005; Lazarevic et al., 2003; Mantere et al., 2012) there are some basic communication properties represented in all used feature sets. These are the temporal behavior such as latency and connection duration, packet sizes, flow direction and the number of communication partners. On the basis of the discussed feature categorization and the main properties which are frequently used in other research, the features in table 1 are defined and used within the anomaly detection system. As depicted in Table 1, the entropy of different variables is used as feature.

$$\hat{H} = -\sum_{i=1}^{M} p_i \log(p_i) \tag{11}$$

The entropy is calculated according to equation 11, where $p_i = P(X = x_i)$ denotes the probability of variable X having the value $x_i \in \{x_1, \ldots, x_M\}$. It is a measure of distribution, because higher entropy values indicate a variable that is more likely to be uniform distributed. Since e.g. the probability of an address occurring at a certain frequency is not given beforehand, it has to be estimated online. For that reason, a sequence of occurred variable values has to be recorded. Using a recording window of length M the probability of every recorded value can be calculated. Alongside, the mean calculation is also based on the last M variable values. For this work, M is set to 15.

6.3 Learning Checks

After the definition of proper features to describe in-vehicle Ethernet communication, learning checks can be applied on them. Following the feature categorization, it is necessary to use more than one learning check to analyze the communication. Not all features are valid for every message. For example, the local features describing one specific communication path on MAC layer are not valid for messages of other paths. Thus, a particular learning check is necessary for every communication path on every layer. In addition, the global features can be analyzed with respect to one layer only, or across layers. Therefore, one learning check is applied on the global features of one layer and one on the MAC-IP, MAC-IP-UDP/TCP and IP-UDP/TCP global features, respectively.

One important aspect concerning the use of the PCA-based method for anomaly detection is the selection of the parameters p and q. Following Shyu et al., the principal components that account for more than 50 percent of the total variance are used as the p components, and the ones with eigenvalues smaller than 0.2 as the q components. Regarding the OCSVM, the parameters ν and σ are chosen before the training process as outlined above. Based on our research results ν was set to 0.01. This value yields the lowest false positive rates during the examination of the evaluation data set used below in section 7. Other possibilities to optimize ν and σ, such as choosing both values based on the grid search with k-fold cross validation process or the DFN method suggested by Xiao et al. (2014), caused more false alarms. Thus, they were not investigated further.

The training process is carried out by a special program using recorded Ethernet messages. These messages are passed as input to the anomaly detection system. Using the static checks, the base features of every message are extracted and forwarded to the feature extraction block. Inside of this block, further calculations are conducted to generate the feature values for training, such as the calculation of entropies. The learning checks used in this work buffer all input feature values as they are batch learning algorithms. After the processing of all messages by the anomaly detection system, the actual training process can be executed. For all learning checks the maximum value of the anomaly score, generated during validation, is used as threshold for the detection of anomalies during runtime.

7 EVALUATION

The evaluation of the proposed system is performed using a simulated environment. Ethernet communication is generated with the tool CANoe from Vector Informatik. Several simulated ECUs exchange messages providing a behavior close to a real in-vehicle Ethernet network. For simplicity reasons, the static checks are not generated directly from the communication matrix but built on basis of the observed messages during the training process. Hence, every message which would arouse an anomaly within the static checks is used to extend the whitelist model so that all training messages are considered normal by means of the static checks.

For the training process, 50.000 Ethernet messages were collected, which are intended for the selected ECU. Thus, the messages are either sent or received from it. The evaluation data set consists of 100.000 messages and is modified in several ways to generate synthetic anomalous behavior.

- Replaying previously sent normal packets: Some packets are randomly selected and replayed once, five or 20 times. The replayed packets are inserted starting 0.1 ms after the original one and with a period of 1 ms, if applicable. This attack can be used to gain information about the behavior of ECUs as the answers can be recorded.
- Modification of the message timestamps: The original timestamp of randomly selected messages is modified by adding a gaussian noise with a mean of zero and a variance of 1 ms or 10 ms. This shall simulate messages sent too early or too late, which could be caused by either a malfunction or an ECU that was taken over by an intruder.

Additionally, a trace of 220 s duration containing a third type of anomalous behavior is evaluated, using the modification of message cycle intervals: A part of the simulated Ethernet traffic is sent in a cyclic manner with defined intervals. The modification of the cycle intervals can be used to create anomalous behavior as the corresponding messages are sent more often or rarely if the cycle time is modified with respect to the training data. Six different modifications were taken into account.

The following evaluation of the selected algorithms is based on their true positive (TP) and false positive (FP) rate. A normal message that is classified as anomaly is a false positive, whereas an anomalous message which is classified as anomaly indicates a true positive detection. In other words, anomalies represent the positive, normal data the negative class. As the anomaly detection system works in an online manner, every processed message will be classified as normal or anomaly. Since the features used to observe Ethernet communication include calculations based on a certain time interval, one anomalous message can influence the classification result for a complete interval. Hence, not only the anomalous message itself will be classified as anomaly but possibly also the following ones. On this account, it is allowed to detect an anomaly within a time interval of two seconds after its occurrence. Within this interval, multiple messages classified as anomalies will result in only one true positive. Outside the detection interval, every message classified as anomalous is a false positive.

Results of Static Check. First of all, it has to be noted that the static checks do not have to be evaluated with synthetic anomalies. As a concept that is not based on machine learning it is predetermined how the static checks will react on every message. Thus, it is not necessary to examine which messages are detected as anomalies. Because the static checks for Ethernet are performed on basis of a whitelist, every message that is not covered by the model will be classified as an anomaly. The comparison of a new message against the model can be implemented efficiently in terms of memory consumption and computing power by using a series of classical *if...then* statements. In summary it is an effective and efficient way to check Ethernet communication against the specification.

Results of Learning Checks. Since the performance of machine learning-based methods is not known in advance, the detection of anomalies by learning checks has to be analyzed in detail. The results of the evaluation with different synthetic anomalies are depicted in Tables 2, 3 and 4. Examining the replay scenario shows that it is easier for all investigated methods to detect five or 20 inserted messages than just one additional message. All investigated algorithms detect a noticeable higher amount of anomalies with an increasing n, where n is the number of replayed messages, see the true positive rates in Table 2. However, the OCSVM is capable of detecting more anomalies than the other methods independent of the number of replayed messages, whereas the Mahalanobis Distance recognizes least.

As depicted in Table 3, it seems to be a more difficult task to detect messages with an anomalous timestamp. The detection rates are below the results in the replay scenario, see Table 2. Another interesting fact is that a higher variance σ^2 of the added gaussian noise is not necessarily leading to higher detection rates. For the Mahalanobis Distance, the number of detected anomalies is even lower with $\sigma^2 = 10\,\text{ms}$. The PCA-based method is the best option for this type of anomaly in terms of true positives, but at the cost of a slightly higher false positive rate than the OCSVM.
Concerning the last type of synthetic anomalies the results are depicted in Table 4 and in figure 2. The dashed lines in figure 2 indicate the actual class of the analyzed messages, whereas the solid lines show the classification results of the particular learning check. The Mahalanobis Distance identifies only one out of six inserted anomalies and thus provides the worst detection performance. It is possible to detect three anomalies by PCA, whereas the OCSVM detects only two. Therefore, the PCA-based detection yields the best results in this scenario. PCA classifies some messages directly after the second detection interval as anomaly, which can be interpreted as reaction to the second anomaly. Besides, the OCSVM was able to detect the second anomaly. But since the OCSVM issues alarms in the time interval between first and second anomaly, it is not possible to distinguish between them. Thus, the detection of the second anomaly in the time interval of 55 to 70 s is denoted in parentheses in Table 4 for both the PCA and the OCSVM. It should be noted that the used features contain information calculated by a sliding window approach, such as the mean time interval between packets. Thereby, after returning to normal message cycle times and hence normal communication behavior, there is still the possibility to detect anomalies. This can be one explanation for the OCSVM characteristic of issuing false positives between first and second as well as second and third anomaly.

Table 2: Detection- and false alarm rates in the scenario of replaying n normal messages.

	Mahalanobis	PCA	OCSVM
		$n = 1$	
FP-Rate	0.0019	0.0026	0.0026
TP-Rate	0.2750	0.3250	0.7250
		$n = 5$	
FP-Rate	0.0015	0.0021	0.0019
TP-Rate	0.4872	0.7179	0.7692
		$n = 20$	
FP-Rate	0.0018	0.0024	0.0022
TP-Rate	0.9750	0.9750	1.0000

Table 3: Detection- and false alarm rates in the scenario of adding gaussian noise with variance σ^2 to message timestamps.

	Mahalanobis	PCA	OCSVM
		$\sigma^2 = 1\,\text{ms}$	
FP-Rate	0.0021	0.0028	0.0023
TP-Rate	0.1212	0.3030	0.2121
		$\sigma^2 = 10\,\text{ms}$	
FP-Rate	0.0016	0.0022	0.0017
TP-Rate	0.0909	0.3030	0.2424

Table 4: Detected anomalies in the scenario of modified message cycle intervals.

Anomalous time interval	Mahalanobis	PCA	OCSVM
30 s - 45 s	✗	✗	✓
55 s - 70 s	✗	(✓)	(✓)
85 s - 100 s	✗	✓	✗
120 s - 135 s	✓	✓	✓
155 s - 170 s	✗	✗	✗
185 s - 200 s	✗	✓	✗

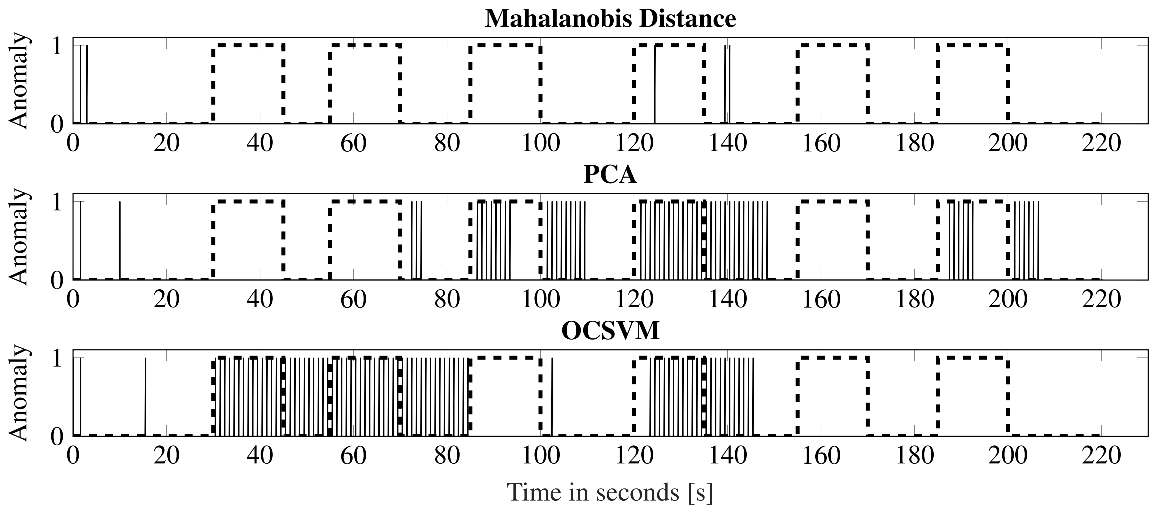

Figure 2: Evaluation of modified message cycle intervals.

8 CONCLUSION AND FUTURE WORK

This paper extended our existing hybrid anomaly detection system for ECUs (Weber et al., 2018) by the support for Ethernet-based communication. The system is designed for in-vehicle ECUs but it can be adapted to other application domains. Specification-based detection is implemented by the so-called static checks and used in the first stage of the system. In contrast to IT systems, there is usually a semi-formal network and data specification available for in-vehicle networks, which is used to implement an effective and efficient anomaly detection. A simplified model was created enabling a slim definition of normal Ethernet communication. It is applied within the static checks as a communication whitelist similar to the specification of a firewall. Every message which is not conform to the whitelist model is considered anomalous.

The overall behavior of the Ethernet-based communication, like the number of messages in a certain time interval, is usually not specified and therefore not represented in the model. Thus, machine learning methods are used in the second stage of the hybrid anomaly detection system capable to extract the normal behavior from training data. The paper at hand evaluated the performance of the statistical methods Mahalanobis Distance and Principal Component Analysis as well as the classical machine learning method One-Class Support Vector Machine. Before applying these algorithms, an appropriate feature set was defined focussing on a good description of in-vehicle Ethernet communication and enabling online detection of anomalies. First results based on the evaluation of different synthetic anomalies are promising. Especially the PCA-based method showed good detection rates at few false alarms combined with low expenses related to memory consumption and computing power. Therefore, this technique is an interesting candidate for future research. Apart from that, the Mahalanobis method did not detect a considerable number of anomalies. The OCSVM should be considered in future as well, because it detected different types of anomalies than the PCA. This indicates that it might be beneficial to use a variety of different algorithms to detect a large portion of possible anomalies. For that reason, it could be interesting to evaluate more methods in future such as neural network based approaches. Another recent candidate for investigation in future, LODA, is strongly related to the PCA and was introduced by Pevný (2016).

One major point for future research is the defined set of features used to model the Ethernet-based communication. As outlined above, with this feature set it is possible to detect different types of anomalies with various methods. However, the exact influence of the individual features on the detection performance was not investigated yet. An analysis on a reduced feature set to minimize the required resources or on an extended feature set to improve the detection performance could be conducted.

The evaluation was performed on the basis of synthesized communication within a simulation environment on an office computer. Hence, further research includes the test of the system with real communication data and the implementation of the system on a real ECU with strict resource limitations and real-time requirements.

REFERENCES

Aggarwal, C. C. (2017). *Outlier Analysis*. Springer International Publishing.

Ahmed, M., Mahmood, A. N., and Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31.

Broy, M. (2006). Challenges in automotive software engineering. In *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06, pages 33–42, New York, NY, USA. ACM.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15.

Hawkins, D. M. (1980). *Identification of outliers*, volume 11. Springer.

Hawkins, S., He, H., Williams, G., and Baxter, R. (2002). Outlier detection using replicator neural networks. In *DaWaK*, volume 2454, pages 170–180. Springer.

Hoppe, T., Kiltz, S., and Dittmann, J. (2009). Applying intrusion detection to automotive IT-early insights and remaining challenges. *Journal of Information Assurance and Security (JIAS)*, 4:226–235.

IEEE-P802.3ch (2017). IEEE 802.3 Multi-Gig Automotive Ethernet PHY Task Force. http://www.ieee802.org/3/ch/index.html. Accessed: 2017-11-28.

IEEE Std 802.3bp-2016 (2016). IEEE Standard for Ethernet Amendment 4: Physical Layer Specifications and Management Parameters for 1 Gb/s Operation over a Single Twisted-Pair Copper Cable.

IEEE Std 802.3bw-2015 (2015). IEEE Standard for Ethernet Amendment 1: Physical Layer Specifications and Management Parameters for 100 Mb/s Operation over a Single Balanced Twisted Pair Cable (100BASE-T1).

Kleberger, P., Olovsson, T., and Jonsson, E. (2011). Security aspects of the in-vehicle network in the connected car. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 528–533.

Lakhina, A., Crovella, M., and Diot, C. (2005). Mining anomalies using traffic feature distributions. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 217–228. ACM.

Larson, U. E., Nilsson, D. K., and Jonsson, E. (2008). An approach to specification-based attack detection for in-vehicle networks. In *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 220–225.

Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., and Srivastava, J. (2003). A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 25–36. SIAM.

Mahalanobis, P. C. (1936). On the generalised distance in statistics. *Proceedings of the National Institute of Sciences of India, 1936*, pages 49–55.

Mantere, M., Uusitalo, I., Sailio, M., and Noponen, S. (2012). Challenges of machine learning based monitoring for industrial control system networks. In *2012 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 968–972. IEEE.

Marchetti, M., Stabili, D., Guido, A., and Colajanni, M. (2016). Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–6. IEEE.

Pevný, T. (2016). Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102(2):275–304.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471.

Shyu, M.-l., Chen, S.-c., Sarinnapakorn, K., and Chang, L. (2003). A novel anomaly detection scheme based on principal component classifier. In *Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM03)*. Citeseer.

Stolfo, S. J. (1999). KDD cup 1999 dataset. *UCI KDD repository. http://kdd.ics.uci.edu*.

Taylor, A., Japkowicz, N., and Leblanc, S. (2015). Frequency-based anomaly detection for the automotive can bus. In *2015 World Congress on Industrial Control Systems Security (WCICSS)*, pages 45–49. IEEE.

Theissler, A. (2017). Detecting known and unknown faults in automotive systems using ensemble-based anomaly detection. *Knowledge-Based Systems*, 123:163–173.

Vapnik, V. N. (1995). The Nature of Statistical Learning Theory. *Springer*, 8(6):187.

Weber, M., Klug, S., Sax, E., Maas, J., and Zimmer, B. (2018). Embedded Hybrid Anomaly Detection for Automotive CAN Communication. To be published in: Embedded Real Time Software and Systems (ERTS^2) 2018.

Xiao, Y., Wang, H., Zhang, L., and Xu, W. (2014). Two methods of selecting Gaussian kernel parameters for one-class SVM and their application to fault detection. *Knowledge-Based Systems*, 59:75–84.