DOI 10.15622/ia.20.4.4

Y. CHEVALIER, F. FENZL, M. KOLOMEETS, R. RIEKE, A. CHECHULIN, C. KRAUSS

CYBERATTACK DETECTION IN VEHICLES USING CHARACTERISTIC FUNCTIONS, ARTIFICIAL NEURAL NETWORKS AND VISUAL ANALYSIS

Chevalier Y., Fenzl F., Kolomeets M., Rieke R., Chechulin A., Krauss C. **Cyberattack detection in vehicles using characteristic functions, artificial neural networks and visual analysis.**

Abstract. The connectivity of autonomous vehicles induces new attack surfaces and thus the demand for sophisticated cybersecurity management. Thus, it is important to ensure that in-vehicle network monitoring includes the ability to accurately detect intrusive behavior and analyze cyberattacks from vehicle data and vehicle logs in a privacy-friendly manner. For this purpose, we describe and evaluate a method that utilizes characteristic functions and compare it with an approach based on artificial neural networks. Visual analysis of the respective event streams complements the evaluation. Although the characteristic functions method is an order of magnitude faster, the accuracy of the results obtained is at least comparable to those obtained with the artificial neural network. Thus, this method is an interesting option for implementation in in-vehicle embedded systems. An important aspect for the usage of the analysis methods within a cybersecurity framework is the explainability of the detection results.

Keywords: controller area network security, intrusion detection, anomaly detection, machine learning, automotive security, security monitoring.

1. Introduction. Information technology (IT) security and data protection are essential for the Internet of Vehicles [1]. Due to a strong connectivity of vehicles and the dependency on external information sources and services, the attack surface increases for intelligent autonomous vehicles. This is exacerbated by the increasing complexity of modern vehicles with more than 100 electronic control units (ECUs) and more than 100 million lines of code. Thus, vulnerabilities in software are highly likely that could be exploited by a potential attacker. However, it is imperative that an attacker cannot influence safety-critical systems. But it has already been demonstrated in [2] how an attacker can remotely take over ECUs to influence steering and braking. It is therefore very important to improve the security of in-vehicle networks, and as long as there are no effective means to prevent certain attacks, methods should be in place to automatically detect them and respond accordingly. The United Nations Economic Commission for Europe (UNECE) has issued Regulation No. 155, "Cybersecurity and Cybersecurity Management System- [3], which makes cybersecurity mandatory for the approval of new vehicle types. One important requirement is that vehicle manufacturers must implement mechanisms to detect cyberattacks in a privacy-friendly manner. This includes measures to detect denial-of-service attacks, such as when the

Controller Area Network (CAN) bus is flooded or ECUs are crashed by a high message load, and subsequent recovery. Furthermore measures for the detection of malicious messages need to be implemented. In principle, the detection of anomalies in network traffic within a vehicle caused by attackers could be done remotely by sending all internal traffic to a Security Operation Center (SOC). However, this would be problematic from a privacy perspective, it would be inefficient, it would incur high costs, and it might not meet real-time response requirements.

This paper is based on our own preliminary work presented in [4]. We propose a new method for in-vehicle anomaly detection that satisfies the following four requirements: (1) the recognition accuracy should be equivalent to or better than existing IDS systems, (2) the method should be lightweight and resource efficient so that it can be executed on typical ECUs, (3) no hardware changes should be necessary and no additional third-party software libraries should be required (as they may not be available for specific ECUs), and (4) the anomaly detection results should be explainable in order to make informed decisions about countermeasures.

To meet these requirements, we propose a logic analysis method that we compare to an artificial neural network-based method that could likely be used in embedded systems in vehicles. We aim for better accuracy, faster and more resource-efficient message characterization, portability to embedded systems without dependencies on libraries such as Tensorflow, and rule-based reasoning so that message evaluation results related to anomalies can be traced back to the responsible rules. We evaluate the proposed method on data sets of the CAN bus, which is the standard solution for communication between ECUs in vehicles.

The remainder of this paper is organized as follows: Section 2 gives an overview on the background and related work. Section 3 introduces data sets from two different vehicles that have been used to evaluate the proposed method. Section 4 presents the principles of the characteristic functions method while Section 5 describes its implementation and the results of various detection setups. Section 6 describes some results from tests with neural networks in order to provide a benchmark for our work. Finally, Section 7 concludes this paper.

2. Background and Related Work. The security of a system can be improved by reducing its attack surface. In [8, 9], for example, possible break-in points are listed together with suggestions for countermeasures such as cryptography, detection of anomalies and ensuring software integrity by separating critical systems. However, most of the intrusion prevention measures currently under discussion require hardware changes, which is inconsistent

with backward compatibility. Therefore, researchers and industry experts have suggested that in the CAN context intrusion detection should be used in addition to the established security mechanisms [10, 11]. CAN intrusion detection methods can be divided into four categories: ECU imitation detection, specification violation detection, message insertion detection, and sequence context anomaly detection. The work to recognize ECU imitations like [12, 13] uses in most cases some kind of physical fingerprint through voltage or time analysis with specific hardware. This work tries to alleviate the general problems of missing authenticity measures in the CAN bus design and thus complements the work presented in this paper. Specification violation detection requires the normal behavior specification to be available, and therefore the benefit of not generating warnings based on false positives. Specification-based intrusion detection methods can use specific checks, e.g. for formality, protocol and data area [14], a certain frequency sensor [15], a number of network-based detection sensors [16] or specifications of the state machines [17]. Message insertion detection can be based on various technologies, such as the analysis of time intervals of messages [18] or long short-term memory (LSTM) [19]. The methods for detecting sequence context anomalies include process mining [20], hidden Markov models [21, 22], a one class Support Vector Machine (OCSVM) [23], artificial neural networks [24] and detection of anomalous patterns in a transition matrix [25]. In most cases, the authors of the above papers have described experiments with a specific method. However, since the authors use different data sets for their experiments, the results of their work cannot be directly compared. Comparisons of various machine learning (ML) algorithms are included in [6, 26, 27]. OCSVM, Self Organizing Maps, and LSTM are used in [26] while LSTM, Gated Recurrent Units (GRU) and Markov models are used in [27]. OCSVM, SVM, sequential neural networks and LSTM are used in [6]. A detailed overview on intrusion detection systems for in-vehicle networks can be found in [28].

The method of characteristic functions used here has already proven in [4] to be much faster and more resource-efficient than artificial neural network methods. With respect to our previous work presented in [4] which was using training sets from [5] and [6], we now use improved state-of-art log files from [7] which have been designed to include sophisticated attack types which do not disrupt normal timing, and thus would not be detected with a frequency-based intrusion detection system (IDS).

3. In-Vehicle Attacks and Data Sets. To evaluate our work, we used the Oak Ridge National Laboratory's (ORNL) Road data set, which was originally presented in [7]. The data set is presented in form of a set of text

files from which 4 meaningful fields can be extracted, that are then mapped in the structure that is presented in Table 1:

1. *Time*. Capture time – helps to identify the time sequence of messages.
2. *ID*. CAN ID, where the lowest ID has a higher channel priority.
3. *Payload*. $p_1, \ldots, p_8$ – 8 bytes with data.
4. *Type*. Attack label where -1 is attack and 1 is the legal message.

The original data set does not contain ground truth flags for each message, but metadata files that describe the attack in regards to timing, content and target ID. For our method we reformatted the messages and marked each valid message in the log with a `1` and each malicious intrusion message with a `-1`. In the exemplary excerpt of a data log in Table 1 you can see that the first occurrence of a message with arbitration ID `208` is a valid message sent by the responsible ECU, whereas the second occurrence is a malicious message introduced by an intruder.

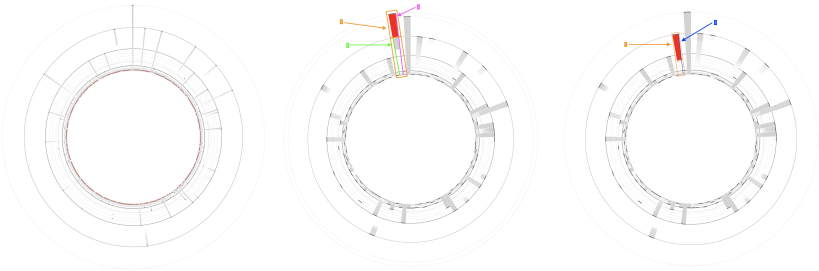
Table 1. Exemplary messages from the ORNL Road data set intrusion scenario `max_speedometer_attack_1`

i	Time	ID	p^1	p^2	p^3	p^4	p^5	p^6	p^7	p^8	Type
1	42.0256	1533	189	221	253	128	126	255	237	218	1
2	42.0271	208	10	115	4	100	136	5	110	0	1
3	42.0282	51	0	6	128	0	12	66	183	208	1
4	42.0282	263	0	0	0	0	0	0	0	0	1
5	42.0282	4095	0	0	0	0	0	0	0	0	1
6	42.0282	14	32	82	150	2	8	9	118	148	1
7	42.0292	208	10	115	4	100	136	255	110	0	-1
8	42.0292	293	144	0	65	31	64	255	163	96	1
9	42.0292	186	6	152	5	4	16	0	2	100	1

Attacks that are presented in the ORNL data set can be divided into 3 categories:

1. *Fuzzing attack* – an attacker injects messages with maximum payloads for many random CAN IDs.
2. *Message injection target attack* – an attacker inject message with a specific CAN ID immediately after the legal message appeared. Thus injected messages are superimposed on legal messages.
3. *Message injection target attack with masquerade* – this attack is similar to the previous one, but legitimate messages were removed. So injected messages replace legal ones.

For a better understanding of what traffic looks like with injected and legal messages, we present these 3 attacks using radial bar chart visualization that was originally presented in [29]. Examples of visualization of these 3 attack types are presented in Figure 1, where malicious messages are marked with red bar color and red bubble. In this visualization, we present attack



(a) Fuzzing attack (b) Message injection attack (c) Masquerade attack

Fig. 1. Different attack scenarios in ORNL Road data set, where red bars and bubbles indicate injected messages and grey bars indicate legal messages

influence by radial time intervals, where each CAN ID is represented as a bar whose height equals the number of messages. Bars consist of arcs that represent payload — the more messages with the same payload the higher is the arc. So solid (or almost solid) arcs depict messages with the same payload, while thin or even transparent (their thickness is less than a pixel) bars depict messages with big payload variety.

Fuzzing attack is the simplest for detection using visual analytics. Usually Fuzzy attack is characterized by many CAN IDs with almost empty bars (see red bubbles in Fig.1a).

For message injection attack there is a pattern in the radial chart – CAN ID with injection have 2 types of message frequency distribution. The first distribution that is without injected messages consists of thin arcs with various payloads. The distribution of injected messages is a solid bar that indicates a lack of variability of payload. We can see how injected messages are superimposed on legal ones by the next patterns depicted in Figure 1b:

1. Sharp difference between frequency (orange indicator #1) – the first part of the bar is very frequent (legal messages) and the second one is not (injected messages).

2. A part of such bar (legal messages – green indicator #2) has almost the same number of messages that other bars.

3. Whole bar (legal messages plus injected – purple indicator #3) does not have the same number of messages that other bars.

For masquerade attack the injection is not clearly detectable by visualization (see Fig.1c). The bars with injected messages look pretty normal except (orange indicator #1) sharp difference between frequency where legal messages have various payload and injected do not. But the (blue indicator #2)

height of the bar looks normal, as the attacker replaced the legal messages with malicious ones.

The ORNL data set also contains "Accelerator Attacks"data sets. This type of attack exploits a vulnerability that puts the ECUs into a compromised state. Therefore, there are no injected messages, so we do not analyze "Accelerator Attacks"data sets in this paper.

4. Principles of the Characteristic Functions Method. Before proceeding to the presentation of characteristic functions, we start by formalising a generic notion of intrusion detection in Section 4.1, and prove that this setting is not usable for exhaustive search in practice. We then present in Section 4.2 criteria that we have considered, and the characteristic functions in Section 4.3

4.1. Formal setting. To formalize this notion, we need to introduce a few notations. We let $\mathcal{P} = [0, 255]^8$ be the set of CAN messages payloads, $\mathcal{I} \subseteq [0, 4095]$ be the set of CAN messages ID, and $\mathcal{B} = \{\bot, \top\}$ denote the respective false and true values. A *log* of length n is a finite sequence $(e_i)_{1 \leqslant i \leqslant n}$ of elements in $\mathcal{I} \times \mathcal{P}$. Let $\mathcal{L}$ be the set of logs. Given a log $L \in \mathcal{L}$ and $\iota \in \mathcal{I}$, we let $\pi_\iota(L)$ be the subsequence of L of elements whose ID is ι. Given a log $L = (e_i)_{1 \leqslant i \leqslant n}$ of length n and $1 \leqslant k \leqslant n$, we denote $L \setminus k = (e'_i)_{1 \leqslant i \leqslant n-1}$ where $e'_i = e_i$ if $i < k$, and $e'_i = e_{i+1}$ otherwise. I.e., $L \setminus k$ is the log L in which the kth event has been removed. Under the same premisses, we denote $L_{<k}$ the log $(e_i)_{1 \leqslant i < k}$.

Definition 1. (Evaluation functions) An evaluation function with memory k*, or* k-evaluation function*, is a function* $\varphi : (\mathcal{I} \times \mathcal{P})^k \to \mathbb{B}$.

We say that a log $L = (e_i)_{1 \leqslant i \leqslant n}$ of length n is *accepted* by a k-evaluation function φ if, for all $k \leqslant i \leqslant n$, we have $\varphi(e_{i-(k-1)}, \ldots, e_i) = \top$. Conversely, for each $k \leqslant i \leqslant n$ such that $\varphi(e_{i-(k-1)}, \ldots, e_i) = \bot$, we say that the event i is an *anomaly*.

Let us now define how an evaluation is applied on a log that may contain anomalies.

Definition 2. (Application of a k*-evaluation function) Given a log* L *of length* n *and a* k*-evaluation function* φ*, the* application of φ on L at step $k \leqslant i \leqslant n$ *is denoted* $\mu(\varphi, L, i)$*. It is defined if* φ *accepts* $L_{<i}$ *and when this is the case, we have:*

$$\mu(\varphi, L, i) = \begin{cases} L, & \text{if } i = n + 1; \\ \mu(\varphi, L, i + 1), & \text{if } \varphi(e_{i-(k-1)}, \ldots, e_i) = \top; \\ \mu(\varphi, L \setminus i, i), & \text{if } \varphi(e_{i-(k-1)}, \ldots, e_i) = \bot. \end{cases}$$

The application of φ *on* L *is denoted* $\mu(\varphi, L)$ *and is equal to* $\mu(\varphi, L, 0)$.

We call the result of the application of an evaluation function φ on a log L the φ-accepted subsequence of the elements of a log L. Elements that have been eliminated are said to have been rejected by φ.

The Intrusion detection problem. We let $\mathbb{L} = \{L_i\}_{i\in\mathbb{N}}$ be a set of logs. The *intrusion detection computation problem* consists in computing a parameter k and a k-evaluation function $\varphi_{\mathbb{L}}$ such that, for all $L \in \mathcal{L}$, we have $\mu(\varphi_{\mathbb{L}}, L) \in \mathbb{L}$. Unsurprisingly, given the generality of the notions introduced, we have:

Theorem 1. Every k-evaluation function reckognises a regular language.

Proof (Sketch) Given a k-evaluation function φ, we construct a finite automaton $\mathcal{A}_\varphi$ as follows:

– All states are final, and are the elements in the $\bigcup_{0\leqslant l\leqslant k-1}(\mathcal{I}\times\mathcal{P})^l$;

– Letters are all the elements in $\mathcal{I}\times\mathcal{P}$;

– There is a transition $(e_1,\ldots,e_{k-1}) \to^e (e_2,\ldots,e_{k-1},e)$ if, and only if, $\varphi(e_1,\ldots,e_{k-1},e) = \top$;

– For $l < k-1$, there is a transition $(e_1,\ldots,e_l) \to^e (e_1,\ldots,e_l,e)$;

– The initial state is the state $()$.

It is clear that $\mathcal{A}_\varphi$ accepts a log L if, and only if, $\mathcal{A}_\varphi$ accepts L.

As a corollary of Theorem 1, since sets of logs $\mathbb{L}$ are not assumed to be rational, the intrusion detection problem usually does not have a solution. Beyond this formal impossibility, one can also note that the set of possible k-evaluation functions, even for $k = 1$, is too large to be computed explicitely.

For practical purposes, one thus has to rely on heuristics to find evaluation functions that are of practical use to detect intrusion.

4.2. Criteria for relevant evaluation functions. Since it is unlikely that the possible logs of a non-trivial system form a rational langugage, we aim at learning a *flight envelope* for the system under analysis by overapproximating the set of possible logs of the system with a rational language. Towards this end we try to compute, given the values occurring in the different fields of the legitimate messages, what the possible acceptable values are for these fields. Just as to locate a point in space there is an infinite number of possible basis in which the coordinates of the point can be expressed, there is in principle an infinite number of ways of looking at values of the fields and their interactions one with another.

For this implementation, we have focused on two sources of regularity in the messages normally exchanged on the CAN bus:

– as a car is an example cyber-physical system, some field values represent "*physical*"values, and while their range may encompass the whole

set of possible values, they are likely to change slowly from one message of a given ID to the next;

– the ECU communicating over the CAN bus run computer programs, and those programs are likely to test for the presence of a specific value in the message, or its membership in a small set of possible values. Legitimate messages sent on the bus are constructed so as to pass these tests.

These considerations, explored in more details below, led us to consider testing whether the value of a field stays in a small set, and whether the value of its differential stays in a similarly small state. The set of all possible tests is the *test space*. The tests that are consistently passed by all the messages of a given ID in a log are considered to be characteristic of that message ID, and the tests themselves are the *characteristic functions*.

Methodology. Each log file is read twice. In the first reading, anomalies are removed from the log file, and the analyzer computes for each message ID and each field a subset of the characteristics functions so that:

– that subset is small enough;

– each message occurring in the log pass at least one of the test.

When no small subset is available, the analysis of the field is considered to be inconclusive. During the second read, for each message, the monitor scans each field for which at least one of the value or differential analysis was conclusive. Each field is accepted if one of the retained tests on its value succeeds, and is rejected otherwise. The message is accepted if no field has been rejected.

Methodology on the choice of the test space. The first step consists in choosing a set of simple tests that are likely to be relevant. The space of all possible message tests will then be all the possible conjunctions and disjunctions of these simple tests. We model packets by an ID and a sequence of bytes, *i.e.* 256-valued integers. This ID determines a class to which each packet belongs. We assume that all packets in a given class are similar enough so that some tests exist that are valid on all messages on the class and are not vacuous.

In principle the test space encompasses all boolean functions on messages or sequences of messages. However a succinct analysis already delineates a few types of tests that may be useful for the analysis of logs:

– some tests are related to the syntactic content of the packet, such as the presence of a padding constant or the presence of a specific value, denoting *e.g.* a more precise type for the packet;

– some tests are computed on the whole packet, such as an error-correcting code;

– some tests are domain specific and relate to the possible evolution of physical data between consecutive packets or the set of possible values of some data;

– some tests depend on the internal state of the devices, a packet being acceptable at some point of their execution but not at another point.

For the sake of simplicity we consider in this paper only tests performed independently on the different fields of messages, as well as on their ID. That is, we consider only the first and third cases of the preceding list. We are currently working on implementing the second (whole message tests) and fourth (with an online process mining algorithm).

Automatic fields. A field is *automatic* if the device receiving and accepting this packet tests whether the value of the field is equal to a constant in its program. It is expected that, if different packets can be sent from one device to another, at least one automatic field exists so that the receiver can derive the type of the received packet. The statistical characteristic of such fields are that they should have only *a few* legitimate values, and that these values should have no other detectable relations. However, the difference between these values can be arbitrary as it is simply a case of a few bits switching value.

There is obviously some arbitrariness in deciding what *a few* means. Since the tests performed are not based on any hints from the protocol, we have arbitrarily decided to define a small set of different values to be the square root of the total number of possible different values, that is less than 16 values among the 256 possible ones. Tests relevant to automatic fields are *value tests* in which we record all the different values occurring in a field during training. If the number of different values is more than 16, the analysis is considered to be inconclusive, and no value test is performed on that field for that message ID during monitoring. Otherwise we verify during monitoring that the value in that field for a message is among the ones seen during training. To sum up, value tests are a conjunction, on all fields f, of a disjunction $f = v_1 \vee \ldots \vee f = v_k$ with $k \leqslant 16$, or of the $true$ constant $\top$ if more than 16 different values have been encountered.

Physical values. These are values that are assumed to evolve slowly. For these values we assume a bound on the difference between the value present in the current packet *wrt* the value occurring in the last preceding similar packet. For these fields the analyzer keeps track of the value in the last accepted message and compares that value with the one in the current message. As in the case of value tests, these difference tests are performed during monitoring only if a small (less than 16, again based on a square root consideration) number of changes have been observed during the training phase. Re-using the same notation as above, but now denoting f the value

of a field in the last accepted packet, and f' its value in the packet under analysis, difference tests are a conjunction, on all fields f, of a disjunction $(f' - f) = v_1 \vee \ldots \vee (f' - f) = v_k$ with $k \leqslant 16$, or of the $true$ constant $\top$ if more than 16 different values have been encountered for the difference between the values for that field between a message and its predecessor.

Random values. There are fields for which no relation was found in the data set among the ones that were searching for. In the data sets considered, a post-analysis of the rules has shown that in several cases these fields are often related with the physical value fields, and that the data conveyed were actually 2-bytes values. The analyzer does not perform any test on these fields, as per the construction described both the value and the difference tests are reduced to the $\top$ constant for these.

4.3. Characteristic functions. We sum up the presentation above with the following criteria on a sufficiently good evaluation function φ:

1. We can forget by relations between the content of different message IDs defining a log with multiple IDs as the coproduct of logs each restrict to one single ID ι, *i.e.*, each log L is assumed equal to $\bigoplus_{\iota in \mathcal{I}} \pi_\iota(L)$;

2. The decomposition of each payload into a set of meaningful fields means that each φ_ι can further be decomposed into evaluation functions specific each field f, *i.e.*, $\varphi_\iota = \bigwedge_{f \in \mathcal{F}_\iota} \varphi_{\iota,f}$;

3. To take into account physical values, it suffices to assume that each $\varphi_{\iota,f}$ is a 2-evaluation functions, and that it suffices to consider the difference between the present value of field and its former value;

A final criterion, not introduced above but that we believe is necessary for the stability of the learning phase, is to refrain from having forbiden values, *e.g.* saying that the value in the field 0 will never be 129. As a heuristic these criteria can certainly be relaxed, but we have already obtained good results even though they may seem very restrictive.

In order to define characteristic functions, it suffices now to introduce, for each ID ι, a set of fields $\mathcal{F}_\iota$. Each field $f \in \mathcal{F}_\iota$ is a function $f : \{\iota\} \times [0, 255]^8 \to \mathbb{Z}$. Also for each field $f \in \mathcal{F}_\iota$ we introduce two sets of values $V_{\iota,f}$ and $D_{\iota,f}$ which, according to the above discussion, can either be finite and of cardinal between 1 and 16, or $\mathbb{Z}$.

Definition 3. (Characteristic functions) A characteristic function *for an ID ι with a set of fields $\mathcal{F}_\iota$ is a 2-evaluation function $\varphi_\iota(e, e')$ of the form:*

$$\varphi_\iota(e, e') = \bigwedge_{f \in \mathcal{F}_\iota} \bigvee_{v \in V_f} (f(e') = v) \ \wedge \bigwedge_{f \in \mathcal{F}_\iota} \bigvee_{v \in D_f} (f(e') - f(e) = v).$$

Finally, the log analysis function is the function φ such that $\mu(\varphi, \bigoplus_{\iota in \mathcal{I}} \pi_\iota(L)) = \bigoplus_{\iota in \mathcal{I}} \mu(\varphi_\iota, \pi_\iota(L))$.

Implementation. Characteristic functions, and thus the log analysis functions they define, have been implemented in C. As a first step, the log is translated if necessary into a binary file which is then mapped to an array of structures using `mmap` call, with each structure representing a packet. Records in this array are then analyzed independently by two modules, one tracking for each ID and for each field the number of different values, until the threshold 16 is reached, the other tracking the differences between consecutive values, again for each ID and for each field of that ID. Each analysis module constructs a balanced binary tree mapping an ID and a field to the result of the analysis on this ID for this field. The `monitor` module then uses this structure to parse and iterate over another log file to classify each packet as to whether it should be accepted or not. The complexity of treating each event in this architecture is $\Theta(\log |\mathcal{I}|)$, as we assume the number of fields is bounded, and thus the number of elements in the balanced binary trees is $\Theta(|\mathcal{I}|)$. Thus the treatment time for a log of N events with K different IDs is $\Theta(N \cdot \log K)$, both for learning and monitoring.

Memroy footprint. During training the entire log file is virtually available in memory, and we rely on the operating system to optimize speed and memory consumption. During the rule evaluation the memory needed by the monitor is linear to both the number of different IDs and in the number of fields within the payload. We note however that since each φ_ι is a 2-evaluation function, both the learning and the monitoring can be performed online, with space requirements of $\Theta(\log |\mathcal{I}|)$.

5. Implementation and Evaluation of the Characteristic Functions Method. Our characteristic functions method attempts firstly to classify messages into classes, and secondly to characterize messages in a given class by the set of rules they are required to pass. The `monitor` module only implements tests that are satisfied by all messages in a given class. The classification tool then outputs the specific rules that are to be used in message classification for each individual class. This information is provided in a human-readable format and may potentially be useful in future research as well.

First, it permits to compute the probability that a random message satisfies all the tests in the class, and thus allows us to evaluate the robustness of the monitor against the injection of random messages. Assuming that in a given class there are n fields classified as automatic and m fields classified as physical, and that tests on fields all accept the maximum of 16 values, a random message in that class has a probability $(\frac{16}{256})^{n+m} = 2^{-4\cdot(n+m)}$ to be

Table 2. Intrusion detection results

Scenario	**CF**			**NN**			**NN'**		
Measure	F_1	PPV	TPR	F_1	PPV	TPR	F_1	PPV	TPR
CSA_1	1	1	1	1	1	1	0	0	0
CSA_2	1	1	1	1	1	1	0	0	0
CSA_3	1	1	1	1	1	1	0	0	0
CSA_{1m}	1	1	1	1	1	1	0.999	1	0.999
CSA_{2m}	1	1	1	1	1	1	0.999	1	0.999
CSA_{3m}	1	1	1	1	1	1	0.999	1	0.999
$Fuzz_1$	1	1	1	0.999	0.998	1	0.993	0.994	0.992
$Fuzz_2$	1	1	1	0.996	0.992	1	0.981	0.980	0.983
$Fuzz_3$	1	1	1	0.987	0.975	1	0.974	0.957	0.991
$MECTA$	1	1	1	0	0	0	0	0	0
$MECTA_m$	1	1	1	0	0	0	0	0	0
MSA_1	1	1	1	1	1	1	0	0	0
MSA_2	1	1	1	1	1	1	0	0	0
MSA_3	1	1	1	1	1	1	0	0	0
MSA_{1m}	1	1	1	1	1	1	0.138	0.989	0.074
MSA_{2m}	1	1	1	1	1	1	0.466	0.998	0.304
MSA_{3m}	1	1	1	1	1	1	0.115	0.997	0.061
$RLoff_1$	1	1	1	1	1	1	0	0	0
$RLoff_2$	1	1	1	1	1	1	0.001	1	0.001
$RLoff_3$	1	1	1	1	1	1	0	0	0
$RLoff_{1m}$	1	1	1	1	1	1	0	0	0
$RLoff_{2m}$	1	1	1	1	1	1	0.438	1	0.280
$RLoff_{3m}$	0	1	0	1	1	1	0.556	1	0.385
$RLon_1$	1	1	1	0.990	1	0.980	0	0	0
$RLon_2$	1	1	1	1	1	1	0	0	0
$RLon_3$	1	1	1	1	1	1	0	0	0
$RLon_{1m}$	0.001	1	0	0.987	1	0.975	0.997	0.995	0.999
$RLon_{2m}$	0.340	1	0.204	1	1	1	0.994	0.997	0.991
$RLon_{3m}$	0.578	0.487	0.712	1	1	1	0.998	0.995	0.999

accepted. This small but non-negligible probability explains the occurrences of false negatives in Table 2, where evaluation results for this approach are labelled with `cf` for *characteristic functions*.

Scenario: log-file with simulated attacks; Precision (Positive Predictive Value) $PPV = \frac{TP}{TP+FP}$; Recall (True Positive Rate) $TPR = \frac{TP}{TP+FN}$; F1 Score : $F1 = 2 * \frac{PPV*TPR}{PPV+TPR}$) Different classification scenarios are *characteristic functions* (`cf` or neural networks trained with either the original ORNL intrusion sets (`nn`) or randomly introduced intrusion messages (`nn'`).

Second, given that the rules generated implement simple tests, it is also in theory possible for a human to better understand the system by looking at the rules produces, and eventually produce new (and less generic) tests beyond those described in this paper. A side result of this is that it is also quite easy to build a fake traffic that will be accepted by a monitor once we know its rules.

Third, it permits to focus further classification work on classes for which only a few fields are tested. For example, some poorly classified messages

seem to be frames in a more complex *Multi-Frame Message* (MFM). To handle this case we plan in future works to implement MFM protocol recognition. Also, and though this is outside of the scope of this paper, a manual analysis of the rules produced and of the messages in these classes strongly suggests new test functions, such as counter and checksum detection, to handle these currently poorly handled cases.

In addition to the discussion above, the results of experiments in Table 2 show next to no *false positive* classifications. This is further visualized in Figure 2, where only one column for the characteristic functions method, here marked as `logan`, can be seen. The evaluation results show that though arbitrarily selected, the heuristic threshold of 16 is not too high as it does not classify a field that contains random values into an automatic field, *i.e.* no over-fitting has been observed. This however should not be interpreted as an impossibility for our method to suffer from over-fitting. Especially a training data set which is too short would tend to produce illegitimate value tests, *e.g.* for the fields recording the timestamp of the packet. The high number of *false positives* on one intrusion scenario however, shows a behaviour yet to completely evaluated, where intrusions that alter existing messages instead of only introducing new messages potentially cause the internal state of the *characteristic functions* classifier to reject every message after the first malicious intrusion message has been classified. In a real-life scenario this would potentially not be harmful, due to the fact, that an intrusion has to be detected in order for this to occur.

For a better understanding of the classifier's errors, we visualized the number of FP and FN in a form of bar charts that are presented in Figures 2 and 3, so one can see how errors are distributed over different attack scenarios and classifiers. The `logan` classifier seen said Figures corresponds to the *characteristic functions* approach, whereas `nn_orig` and `nn_fuzzed` correspond to the different training scenarios for the neural network approach, discussed in Section 6.

We also map classification results in form of radial bar charts where blue represent TP and TN, red – FP, and orange – FN. The example is presented in Figure 4. All classification results in form of radial bar charts are available via the link https://guardeec.github.io/ornl_dataset_vis/visualization.html. You can select a data set and classifier type to view the corresponding result. The CAN ID is displayed by clicking on the bar.

As can be seen in Table 2, the results are very encouraging against the different attacks considered. It is to be noted that using knowledge of the results and models from the analysis modules, it would potentially be easy to

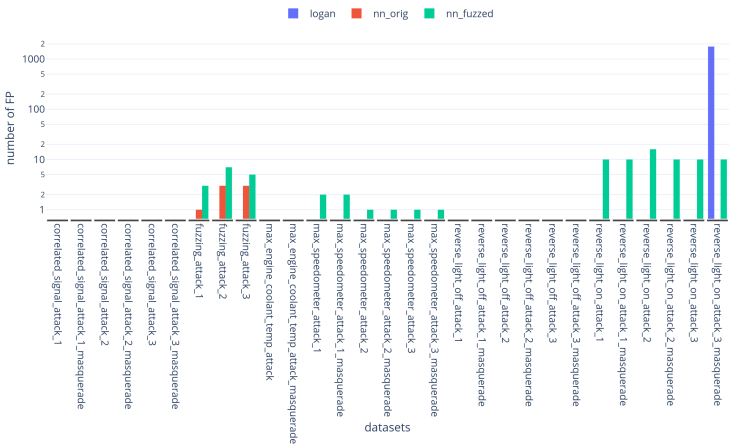


Fig. 2. Classification FP errors for different data sets

construct attacks (*i.e.*, introduction of additional malicious messages on the bus) that follow a pattern that will be accepted by the analyzer.

In addition to the intrusion detection performance we have also evaluated the number of classified messages per second from all test scenarios, which averaged at approx. 1700 messages per second. This test was performed on a Raspberry Pi 3 Model B to test the performance of the classifier on a device similar to what could be used as an edge node in a vehicle.

6. Baseline Benchmark: Artificial Neural Network. As a benchmark for the evaluation of our approach we implemented an artificial neural network approach using the Tensorflow Keras API [30, 31]. Neural networks are the standard for deep learning and can model very complex nonlinear relationships. A fully connected neural network utilizes a number of layers with each layer supporting an arbitrary number of neurons. Data is propagated from the input to the output layer using weighted connections between the neurons of these layers. Specifically a multilayer perceptron (MLP) based on the `Sequential` model from the `keras` Tensorflow package with two hidden layers of 25 neurons each was used. This results in a model with ~ 1200 trainable parameters. We specifically designed the network to perform well on weaker in-vehicle edge devices. As the activation function for the hidden layers we selected *rectified linear unit* (ReLU), which is computationally cheap. In total we trained on the data set for a learning phase of 20 epochs, with a validation split of 0.2, so 20%

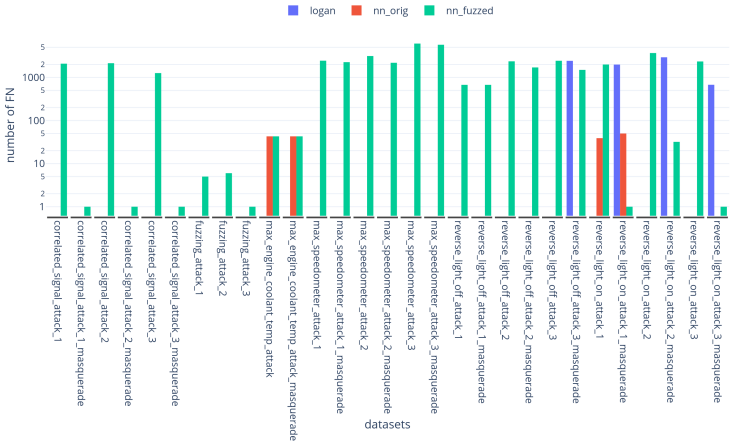


Fig. 3. Classification FN errors for different data sets

of the input data was used for validation. For the *loss* function of the model we decided to use *binary cross-entropy*, which is based on a classification of values between 0 and 1 and is best suited for binary classification, as is required for our training data. In addition to that the *Adam* [32] optimizer was used.

The data logs are then preprocessed into a structure where for each message m^i with arbitration ID i and payload p_{m^i} a input vector (i, p_{m^i}, p_{m-1^i}) with the payload of the previous message with the same ID, is created. With first message of each ID, where there is no previous payload available, a vector of zeros is used as payload. The timing of the message is disregarded in this approach. This structure was selected to make the neural network approach as comparable to the characteristic functions approach possible, by providing the same information for the classifying process as in the case with characteristic functions.

For each of the intrusion scenarios described in Section 3 a separate model was trained, where scenarios that consist of more than one log file are merged into one model. For the training only the non modified log files from the ORNL road data set were used, due to the fact that the masquerade intrusions alter the structure of the log and can potentially impede training.

To improve our evaluation using neural networks we have designed two different evaluation scenarios. One scenario utilizes the original data logs for training data, while the other uses artificially generated intrusion messages,

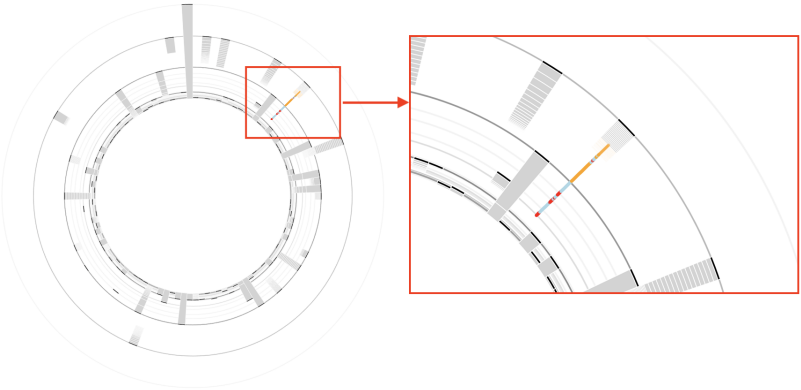


Fig. 4. Classification errors mapped on radial bar chart: blue – TP and TN, red – FP, orange – FN

based on the original intrusions and introduced with a normal distribution over the course of the complete log file. This does not alter the content of the intrusion messages introduced and figuratively represents the case that in real operation the attack is known but the context of the attack is not. Both scenarios have been additionally trained on a version of their respective log files, where all intrusion messages have been deleted in an attempt to improve the classification of normal driving behaviour and minimize the number of false positive classification.

In total six models were trained per evaluation scenario, namely `CSA` (correlated signal attack), `Fuzz` (fuzzing attack), `MECTA` (max engine coolant temp attack), `MSA` (max speedometer attack), `RLoff` (reverse light off attack) and `RLon` (reverse light on attack).

The results are shown in Table 2, where the evaluation scenario using original log files is annotated with *nn* and the scenario using artificially generated messages with *nn'*.

As expected the results for the *nn* scenario are in most scenarios near perfect, except for the `MECTA` intrusion scenario, which contained too few intrusion messages for reliable training. In most other scenarios all introduced intrusion messages were classified correctly as intrusions, as indicated by a value of 1 in the TPR_{nn} column. For the fuzzing attacks a below a 1 value in the PPV_{nn} column indicates the occurrence of *false positive* values in classification, a close observation here shows misclassification here happens mostly at the beginning of the log, where a zero value vector was used in the input vector for the message, as described above.

The results for the training with artificially introduced intrusion data, which can shown in the *nn'* column of Table 2, are more diverse. The classifier models have shown that often for intrusion scenarios, where additional messages were introduced to obfuscate the normal behaviour, the classification performance of the models trained on artificially placed intrusion is zero or close to zero. This shows that the context of the messages, most importantly the previous message is decisive for correct classification. For the masquerade scenarios of each intrusion, many of the introduced and modified messages were classified correctly. A high *positive predictive value* here shows that the number of false positive classifications is close to zero, whereas the *true positive rate* varies significantly with different log files. These scenarios show, that despite the low classification rates on the non-masquerade versions of the logs, the models are able to detect derivations from normal behaviour in the log files.

The results here highlight the complexity of the intrusion scenarios from the ORNL Road data set. The *nn'* model evaluation signifies that even if the message structure of an intrusion scenario is known, the correct classification is non-trivial.

To provide a better performance comparison to the CF approach not only in terms of classification accuracy, but also in regards to time performance we have also run all evaluations on a Raspberry Pi 3 Model B for the ANN classifier. On this relatively low-performance device the ANN was only able to evaluate an average of 150 messages per second, which is less than a tenth of the performance shown by the CF classifier.

7. Conclusion. We have seen in previous work [6] that artificial neural network approaches to anomaly detection deliver good results but that it is hard to implement this kind of detection in-vehicle because of restrictions with respect to on-board resources of typical ECUs used in vehicular systems. Thus, we have started to analyze logs using a bind and branch approach that was very accurate but lacked robustness. From this experience we built a log analyzer in C that focused on payload bytes having either a small set of different values or a small set of possible changes. We have evaluated this characteristic functions approach on state-of-the-art CAN bus intrusion data from real-life intrusion scenarios and obtained results that are significantly more robust and accurate in comparison to a standard implementation of an artificial neural network classifier. The evaluations regarding the time performance of both approaches have also shown a significant margin between both approaches with characteristic functions being able to evaluate ten times more messages with the same time compared to even relatively small artificial neural network.

The approach has shown to classify small derivation from standard behaviour in log files well without the occurrence of *false positive* classifications, which is an important trait for the integration in a real automotive environment. As an extension of our work in [4], we have shown here that even timing opaque attacks from the sophisticated ORNL data set which do not disrupt normal timing nor CAN ID distributions can be found with our method.

We will work in the near future on refining the analysis to *guess* the functions employed by the devices to test whether the packet shall be accepted. We plan to extend our approach to CAN with flexible data-rate (CAN-FD) which is an extension of the original CAN bus protocol with higher bandwidth. Furthermore, we work on a hybrid method where artificial neural networks are used offline to improve the rules of a rule-based in-vehicle IDS.

References

1. Müller-Quade J., Backes M., Buxmann P., Eckert C., Holz T. *et al.* Cybersecurity research: Challenges and course of action. *Tech. rep., Karlsruher Institut für Technologie (KIT).* 2019.
2. Miller C., Valasek C. Remote exploitation of an unaltered passenger vehicle. *Tech. rep., IOActive Labs.* August 2015.
3. UN Regulation No. 155 [Uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system]. Available at: www.eur-lex.europa.eu/eli/reg/2021/387/ojOnline. (accessed 29-Apr-2021).
4. Chevalier Y., Rieke R., Fenzl F., Chechulin A., Kotenko I. Ecu-secure: Characteristic functions for in-vehicle intrusion detection. Proceedings of the International Symposium on Intelligent and Distributed Computing. 2019. pp. 495–504.
5. Hacking and Countermeasure Research Lab (HCRL). [Car-Hacking Dataset for the intrusion detection]. Available at: http://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset. (accessed 28-Jun-2018).
6. Berger I., Rieke R., Kolomeets M., Chechulin A., Kotenko I. Comparative study of machine learning methods for in-vehicle intrusion detection. Proceedings of the ESORICS 2018 International Workshops, CyberICPS 2018 and SECPRE 2018, Barcelona, Spain, September 6-7, 2018, Revised Selected Papers. 2019. vol. 11387. pp. 85–101.
7. Verma M., Iannacone M., Bridges R., Hollifield S., Kay B. Combs F. Road: The real ornl automotive dynamometer controller area network intrusion detection dataset (with a comprehensive can ids dataset survey & guide. ArXiv preprint arXiv:2012.14600. 2020.
8. Studnia I., Nicomette V., Alata E., Deswarte Y., Kaâniche M., Laarouchi Y. Security of embedded automotive networks: state of the art and a research proposal. Proceedings of the SAFECOMP 2013 - Workshop CARS of the 32nd International Conference on Computer Safety, Reliability and Security. 2013.
9. Wolf M., Weimerskirch A., Paar C. Security in Automotive Bus Systems. Proceedings of the Workshop on Embedded Security in Cars. 2014. pp. 1–13.
10. ENISA Cyber security and resilience of smart cars. *Tech. rep., ENISA*. 2016.
11. Metzker E. Reliably detecting and defending against attacks. Available at: https://assets.vector.com/cms/content/know-how/_technical-articles/Security_Intrusion_Detection_AutomobilElektronik_202003 _PressArticle_EN.pdf. (accessed 28-Apr-2021).

12. Choi W., Joo K., Jo H., Park M., Lee D. Voltageids: Low-level communication characteristics for automotive intrusion detection system. *IEEE Transactions on Information Forensics and Security.* 2018. vol. 13. pp. 2114–2129.
13. Cho K., Shin K. Fingerprinting electronic control units for vehicle intrusion detection. Proceedings of the 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016. 2016. pp. 911–927.
14. Larson U., Nilsson D., Jonsson E. An approach to specification-based attack detection for in-vehicle networks. Proceedings of the Intelligent Vehicles Symposium, 2008 IEEE. 2008. pp. 220–225.
15. Hoppe T., Kiltz S., Dittmann J. Security threats to automotive CAN networks – practical examples and selected short-term countermeasures. *Reliability Engineering & System Safety.* 2011. vol. 96. pp. 235–248.
16. Müter M., Asaj N. Entropy-based anomaly detection for in-vehicle networks. Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV). 2011. pp. 1110–1115.
17. Studnia I., Alata E., Nicomette V., Kaâniche M., Laarouchi Y. A language-based intrusion detection approach for automotive embedded networks. Proceedings of the 21st IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2015). 2014. pp. 1–12.
18. Song H., Kim H., Kim H. Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. Proceedings of the 2016 international conference on information networking (ICOIN). 2016. vol. 3. pp. 63–68.
19. Wei Z., Yang Y., Rehana Y., Wu Y., Weng J., Deng R. IoVShield: An Efficient Vehicular Intrusion Detection System for Self-driving. Proceedings of the International Conference on Information Security Practice and Experience. 2017. pp. 638–647.
20. Rieke R., Seidemann M., Talla E., Zelle D., Seeger B. Behavior analysis for safety and security in automotive systems. Proceedings of the Parallel, Distributed and Network-Based Processing (PDP), IEEE Computer Society. 2017. pp. 381–385.
21. Levi M., Allouche Y., Kontorovich A. Advanced analytics for connected cars cyber security. Proceedings of the 87th Vehicular Technology Conference (VTC Spring), IEEE. 2017. vol. abs/1711.01939.
22. Narayanan S., Mittal S., Joshi A. Obd securealert: An anomaly detection system for vehicles. Proceedings of the IEEE Workshop on Smart Service Systems (SmartSys 2016). 2016. pp. 1–7.
23. Theissler A. Anomaly detection in recordings from in-vehicle networks. Proceedings of Big Data Applications and Principles First International Workshop, BIGDAP 2014. 2014. vol. 23. P. 26.
24. Kang M., Kang J. A novel intrusion detection method using deep neural network for in-vehicle network security. Proceedings of the 83rd Vehicular Technology Conference (VTC Spring), IEEE. 2016. pp. 1–5.
25. Marchetti M., Stabili D. Anomaly detection of CAN bus messages through analysis of id sequences. Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV). 2017. pp. 1577–1583.
26. Chockalingam V., Larson I., Lin D., Nofzinger S. Detecting attacks on the CAN protocol with machine learning. *Annu EECS.* 2016. vol. 558. no.7.
27. Taylor A., Leblanc S., Japkowicz N. Probing the limits of anomaly detectors for automobiles with a cyber attack framework. *IEEE Intelligent Systems.* 2018. vol. 33. no. 2. pp. 54–62.
28. Al-Jarrah O., Maple C., Dianati M., Oxtoby D., Mouzakitis A. Intrusion detection systems for intra-vehicle networks: A review. *IEEE Access.* 2019. vol. 7. pp. 21266–21289.

29. Kolomeets M., Chechulin A., Kotenko I. Visual analysis of CAN bus traffic injection using radial bar charts. Proceedings of the 1st IEEE International Conference on Industrial Cyber-Physical Systems (ICPS-2018). 2018. pp. 841–846.
30. Abadi M., Barham P., Chen J., Chen Z., Davis A., Dean J., Devin M., Ghemawat S., Irving G., Isard M., et al. Tensorflow: A system for large-scale machine learning. Proceedings of the 12th USENIX symposium on operating systems design and implementation (OSDI 16). 2016. pp. 265–283.
31. Chollet F. Keras. Available at: https://github.com/fchollet/keras. (accessed 28-Apr-2021).
32. Kingma D., Ba J. Adam: A method for stochastic optimization. ArXiv preprint arXiv:1412.6980. 2014.

Chevalier Yannick — Dr. Hab., Assistant Professor, AI department, Université de Toulouse, IRIT. Research interests: formal methods for security, verification of cryptographic protocols. The number of peer reviewed publications — 46. ychevali@irit.fr; www.irit.fr/ Yannick.Chevalier/; 118, route de Narbonne, 31062, Toulouse, Cedex 7, France; office phone: +33-561556091.

Fenzl Florian — Researcher, Department Cyber-Physical Systems Security, Fraunhofer Institute for Secure Information Technology SIT. Research interests: automotive security, artificial intelligence, machine learning for anomaly detection. The number of peer reviewed publications — 2. florian.fenzl@sit.fraunhofer.de; Rheinstrasse 75, 64295 Darmstadt, Germany; office phone: +49-6151869116.

Kolomeets Maxim — PhD Student, Juniour Researcher, Laboratory of Computer Security Problems, St. Petersburg Federal Research Center of the Russian Academy of Sciences. Research interests: security data visualisation, social networks security. The number of peer reviewed publications — 21. kolomeec@comsec.spb.ru; 14th line of V.O. 39, 199178, Saint-Petersburg, Russia; office phone: +7(812)328-7181.

Rieke Roland — Dr. rer. nat., Senior Scientist, Department Cyber-Physical Systems Security, Fraunhofer Institute for Secure Information Technology SIT. Research interests: design principles for secure, scalable systems and model-based predictive security analysis. The number of peer reviewed publications — 46. roland.rieke@sit.fraunhofer.de; Rheinstrasse 75, 64295 Darmstadt, Germany; office phone: +49-6151869116.

Chechulin Andrey — PhD, Leading Researcher, Laboratory of Computer Security Problems, St. Petersburg Federal Research Center of the Russian Academy of Sciences. Research interests: computer network security, intrusion detection, analysis of vulnerability, security visualization, embedded systems security. The number of peer reviewed publications — 80. chechulin@comsec.spb.ru; 14th line of V.O. 39, 199178, Saint-Petersburg, Russia; office phone: +7(812)328-7181.

Krauss, Christoph — Dr., Professor, Head, Department Cyber-Physical Systems Security, Fraunhofer Institute for Secure Information Technology SIT. Darmstadt University of Applied Sciences. Research interests: automotive security and privacy, railway security, intelligent energy networks security, trusted computing, network security, efficient and post quantum cryptography. christoph.krauss@sit.fraunhofer.de; Rheinstrasse 75, 64295 Darmstadt, Germany; office phone: +49-6151869116.

Acknowledgements. This research is supported by the German Federal Ministry of Education and Research (BMBF) and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE and by the BMBF project VITAF (ID 16KIS0835). Additionally, the project leading to this application has received funding from the European Union's Horizon 2020 research, innovation programme under grant agreement No 883135 and by the budget project 0073-2019-0002.

УДК 004.056.5 **DOI** 10.15622/ia.20.4.4

Я. Шевалье, Ф. Фенцль, М.В. Коломеец, Р. Рике, А.А. Чечулин, К. Краус

ОБНАРУЖЕНИЕ КИБЕРАТАК В ТРАНСПОРТНЫХ СРЕДСТВАХ С ИСПОЛЬЗОВАНИЕМ ХАРАКТЕРИЗУЮЩИХ ФУНКЦИЙ, ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ И ВИЗУАЛЬНОГО АНАЛИЗА

Шевалье Я., Фенцль Ф., Коломеец М.В., Рике Р., Чечулин А.А., Краус. К. **Обнаружение кибератак в транспортных средствах с использованием характеризующих функций, искусственных нейронных сетей и визуального анализа.**

Аннотация. Возможность подключения автономных транспортных средств к сетям порождает новые возможности для атак и, следовательно, потребность в развитии методов кибербезопасности. Таким образом, важно обеспечить, чтобы мониторинг сети в транспортном средстве включал в себя возможность точно обнаруживать вторжение и анализировать кибератаки на основе данных о транспортных средствах и журналов событий транспортных средств с учетом их конфиденциальности. В статье предложен и оценен метод, использующий характеризующую функцию и проведено его сравнение с подходом, основанным на искусственных нейронных сетей. Визуальный анализ соответствующих потоков событий дополняет оценку. Несмотря на то, что метод с характеризующей функцией на порядок быстрее, точность полученных результатов, по крайней мере, сравнима с таковой, полученной с помощью искусственной нейронной сети. Таким образом, этот метод представляет собой перспективный вариант для реализации во встраиваемых системах автомобиля. Кроме того, важным аспектом использования методов анализа в рамках кибербезопасности является объяснимость результатов обнаружения.

Ключевые слова: бзеопасность сети контроллера, обнаружение вторжений, обнаружение аномалий, машинное обучение, автомобильная безопасность, мониторинг безопасности.

Шевалье Янник — Dr. Hab., доцент, кафедра искусственного интеллекта, университет Тулузы, IRIT. Область научных интересов: формальные методы безопасности, верификация криптографических протоколов. Число научных публикаций — 46. ychevali@irit.fr; www.irit.fr/Yannick.Chevalier; 118, route de Narbonne, 31062, Тулуза, Cedex 7, Франция; р.т.: +33-561556091.

Фенцль Флориан — научный сотрудник, департамент безопасности киберфизических систем, институт безопасных информационных технологий им. Фраунгофера (SIT). Область научных интересов: автомобильная безопасность, искусственный интеллект, машинное обучение, обнаружения аномалий. Число научных публикаций — 2. florian.fenzl@sit.fraunhofer.de; Rheinstrasse 75, 64295, Дармштадт, Германия; р.т.: +49-6151869116.

Коломеец Максим Вадимович — аспирант, младший научный сотрудник, лаборатория проблем компьютерной безопасности, Федеральное государственное бюджетное учреждение науки "Санкт-Петербургский Федеральный исследовательский центр Российской академии наук". Область научных интересов: визуализация данных безопасности, безопасность социальных сетей. Число научных публикаций — 21. kolomeec@comsec.spb.ru; 14-я линия В.О. 39, 199178, Санкт-Петербург, Россия; р.т.: +7(812)328-7181.

Рике Роланд — Dr. rer. nat., старший научный сотрудник, департамент безопасности киберфизических систем, институт безопасных информационных технологий им. Фраунгофера (SIT). Область научных интересов: принципы проектирования безопасных масштабируемых систем и прогнозный анализ безопасности на основе моделей. Число научных публикаций — 46. roland.rieke@sit.fraunhofer.de; Rheinstrasse 75, 64295, Дармштадт, Германия; р.т.: +49-6151869116.

Чечулин Андрей Алексеевич — кандидат технических наук, доцент, ведущий научный сотрудник, лаборатория проблем компьютерной безопасности, Федеральное государственное бюджетное учреждение науки "Санкт-Петербургский Федеральный исследовательский центр Российской академии наук". Область научных интересов: безопасность компьютерных сетей, обнаружение вторжений, анализ уязвимостей. визуализация данных безопасности, защита встроенных устройств. Число научных публикаций — 80. chechulin@comsec.spb.ru; 14-я линия В.О. 39, 199178, Санкт-Петербург, Россия; р.т.: +7(812)328-7181.

Краус Кристоф — Ph.D., профессор, руководитель, отдел безопасности киберфизических систем, институт безопасных информационных технологий им. Фраунгофера (SIT). Дармштадтский университет прикладных наук. Область научных интересов: автомобильная безопасность и конфиденциальность, безопасность железных дорог, безопасность интеллектуальных энергетических сетей, надежные вычисления, сетевая безопасность, эффективная и постквантовая криптография. christoph.krauss@sit.fraunhofer.de; Rheinstrasse 75, 64295 Дармштадт, Германия; р.т.: +49-6151869116.

Поддержка исследований. Это исследование поддержано Федеральным министерством образования и исследований Германии (BMBF) и Государственным министерством высшего образования, исследований и искусств земли Гессен в рамках совместной поддержки Национального исследовательского центра прикладной кибербезопасности ATHENE и проекта BMBF VITAF (ID 16KIS0835). Кроме того, проект получил финансирование в рамках исследовательской программы Европейского Союза Horizon 2020, инновационной программы в рамках грантового соглашения № 883135 и бюджетного проекта 0073-2019-0002.

Литература

1. *Müller-Quade J., Backes M., Buxmann P., Eckert C., Holz T. et al.* Cybersecurity research: Challenges and course of action // Tech. rep., Karlsruher Institut für Technologie (KIT). 2019.
2. *Miller C., Valasek C.* Remote exploitation of an unaltered passenger vehicle // Tech. rep., IOActive Labs. August 2015.
3. UN Regulation No. 155 [Uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system]. URL: www.eur-lex.europa.eu/eli/reg/2021/387/ojOnline. (дата обращения: 29.04.2021).
4. *Chevalier Y., Rieke R., Fenzl F., Chechulin A., Kotenko I.* Ecu-secure: Characteristic functions for in-vehicle intrusion detection // Proceedings of the International Symposium on Intelligent and Distributed Computing. 2019. pp. 495–504.
5. Hacking and Countermeasure Research Lab (HCRL). [Car-Hacking Dataset for the intrusion detection]. URL: http://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset. (дата обращения: 28.05.2021).
6. *Berger I., Rieke R., Kolomeets M., Chechulin A., Kotenko I.* Comparative study of machine learning methods for in-vehicle intrusion detection // Proceedings of the ESORICS 2018 International Workshops, CyberICPS 2018 and SECPRE 2018, Barcelona, Spain, September 6-7, 2018, Revised Selected Papers. 2019. vol. 11387. pp. 85–101.
7. *Verma M., Iannacone M., Bridges R., Hollifield S., Kay B. Combs F.* Road: The real ornl automotive dynamometer controller area network intrusion detection dataset (with

a comprehensive can ids dataset survey & guide // ArXiv preprint arXiv:2012.14600. 2020.

8. *Studnia I., Nicomette V., Alata E., Deswarte Y., Kaâniche M., Laarouchi Y.* Security of embedded automotive networks: state of the art and a research proposal // Proceedings of the SAFECOMP 2013 - Workshop CARS of the 32nd International Conference on Computer Safety, Reliability and Security. 2013.
9. *Wolf M., Weimerskirch A., Paar C.* Security in Automotive Bus Systems // Proceedings of the Workshop on Embedded Security in Cars. 2014. pp. 1–13.
10. ENISA Cyber security and resilience of smart cars // Tech. rep., ENISA. 2016.
11. *Metzker E.* Reliably detecting and defending against attacks. URL: https://assets.vector.com/cms/content/know-how/_technical-articles/Security_Intrusion_Detection_AutomobilElektronik_202003 _PressArticle_EN.pdf. (дата обращения: 28.05.2021).
12. *Choi W., Joo K., Jo H., Park M., Lee D.* Voltageids: Low-level communication characteristics for automotive intrusion detection system // IEEE Transactions on Information Forensics and Security. 2018. vol. 13. pp. 2114–2129.
13. *Cho K., Shin K.* Fingerprinting electronic control units for vehicle intrusion detection // Proceedings of the 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016. 2016. pp. 911–927.
14. *Larson U., Nilsson D., Jonsson E.* An approach to specification-based attack detection for in-vehicle networks // Proceedings of the Intelligent Vehicles Symposium, 2008 IEEE. 2008. pp. 220–225.
15. *Hoppe T., Kiltz S., Dittmann J.* Security threats to automotive CAN networks – practical examples and selected short-term countermeasures // Reliability Engineering & System Safety. 2011. vol. 96. pp. 235–248.
16. *Müter M., Asaj N.* Entropy-based anomaly detection for in-vehicle networks // Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV). 2011. pp. 1110–1115.
17. *Studnia I., Alata E., Nicomette V., Kaâniche M., Laarouchi Y.* A language-based intrusion detection approach for automotive embedded networks // Proceedings of the 21st IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2015). 2014. pp. 1–12.
18. *Song H., Kim H., Kim H.* Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network // Proceedings of the 2016 international conference on information networking (ICOIN). 2016. vol. 3. pp. 63–68.
19. *Wei Z., Yang Y., Rehana Y., Wu Y., Weng J., Deng R.* IoVShield: An Efficient Vehicular Intrusion Detection System for Self-driving // Proceedings of the International Conference on Information Security Practice and Experience. 2017. pp. 638–647.
20. *Rieke R., Seidemann M., Talla E., Zelle D., Seeger B.* Behavior analysis for safety and security in automotive systems // Proceedings of the Parallel, Distributed and Network-Based Processing (PDP), IEEE Computer Society. 2017. pp. 381–385.
21. *Levi M., Allouche Y., Kontorovich A.* Advanced analytics for connected cars cyber security // Proceedings of the 87th Vehicular Technology Conference (VTC Spring), IEEE. 2017. vol. abs/1711.01939.
22. *Narayanan S., Mittal S., Joshi A.* Obd securealert: An anomaly detection system for vehicles // Proceedings of the IEEE Workshop on Smart Service Systems (SmartSys 2016). 2016. pp. 1–7.
23. *Theissler A.* Anomaly detection in recordings from in-vehicle networks // Proceedings of Big Data Applications and Principles First International Workshop, BIGDAP 2014. 2014. vol. 23. P. 26.

24. *Kang M., Kang J.* A novel intrusion detection method using deep neural network for in-vehicle network security // Proceedings of the 83rd Vehicular Technology Conference (VTC Spring), IEEE. 2016. pp. 1–5.
25. *Marchetti M., Stabili D.* Anomaly detection of CAN bus messages through analysis of id sequences // Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV). 2017. pp. 1577–1583.
26. *Chockalingam V., Larson I., Lin D., Nofzinger S.* Detecting attacks on the CAN protocol with machine learning // Annu EECS. 2016. vol. 558. no.7.
27. *Taylor A., Leblanc S., Japkowicz N.* Probing the limits of anomaly detectors for automobiles with a cyber attack framework // IEEE Intelligent Systems. 2018. vol. 33. no. 2. pp. 54–62.
28. *Al-Jarrah O., Maple C., Dianati M., Oxtoby D., Mouzakitis A.* Intrusion detection systems for intra-vehicle networks: A review // IEEE Access. 2019. vol. 7. pp. 21266–21289.
29. *Kolomeets M., Chechulin A., Kotenko I.* Visual analysis of CAN bus traffic injection using radial bar charts // Proceedings of the 1st IEEE International Conference on Industrial Cyber-Physical Systems (ICPS-2018). 2018. pp. 841–846.
30. *Abadi M., Barham P., Chen J., Chen Z., Davis A., Dean J., Devin M., Ghemawat S., Irving G., Isard M., et al.* Tensorflow: A system for large-scale machine learning // Proceedings of the 12th USENIX symposium on operating systems design and implementation (OSDI 16). 2016. pp. 265–283.
31. *Chollet F.* Keras. URL: https://github.com/fchollet/keras. (дата обращения: 28.04.2021)
32. *Kingma D., Ba J.* Adam: A method for stochastic optimization // ArXiv preprint arXiv:1412.6980. 2014.