Ontology Based Anomaly Detection for Cellular Vehicular Communications

Quentin Ricard[1,2], Philippe Owezarski[1]
[1]CNRS-LAAS, Université de Toulouse, Toulouse, France
[2]Continental Digital Services France, Toulouse, France
firstname.lastname@laas.fr

***Abstract*—Intelligent Transportation Systems are being deployed all over the world, providing new applications and services that could prevent accidents, help regulate traffic and the automotive industry in designing energy efficient vehicles. However, enabling vehicles to communicate with the rest of the world ultimately leads to new security challenges with connected vehicles becoming new interesting targets for malicious actors. Thus, safeguards need to be deployed to detect malicious and anomalous activities in vehicular communications. This paper presents an approach to anomaly detection based on an ontological representation of cellular vehicular communication.**

***Index Terms*—intelligent transportation systems, anomaly detection, C-V2X**

I. Introduction

In recent years, the automotive industry and the research community have engaged considerable resources on Intelligent Transportation Systems (ITS) technologies. In fact, vehicles nowadays embed interconnected networks that are responsible of complex tasks in a vehicle such as cruise-control, lane-tracking and assistance braking. These systems make decisions based on different kind of sensors, i.e. radars, cameras, brakes, thermometer. Enabling vehicles to share pieces of information from these sensors with the rest of the digital world facilities would allow the emergence of new services. These services could help improving the efficiency of transport in terms of safety, user experience and fleet monitoring. There are two types of communication channels that were envisioned for this task, namely, vehicle ad-hoc networks (VANET or V2V) and cellular vehicular networks (CVN or C-V2X).

However, introducing new communication channels inside vehicles also creates new opportunities for malicious actors to disturb these new networks. In fact, there has been an ever increasing list of attacks that were successfully conducted against vehicles [1]–[5], some of them even allowed remote exploitation by attackers. Specifically, the Jeep Cherokee hack [6] by Charlie Miller and Chris Valasek had a massive impact in the media and forced automotive companies to take into account cyber-security when designing their vehicles.

Moreover, malicious activities are not the only threat to connected vehicles. For example, in the case of automated driving or collision avoidance systems based on communications. Vehicles and drivers will have to rely on knowledge gathered from the network in order to take complex decisions. Therefore, the integrity of data sent and received by vehicles must be verified and safeguards need to be deployed to prevent anomalies from disrupting these critical services.

Detecting anomalies or intrusions in network communications has been an extensive topic of research in the past decades. The first methods were mostly based on signature [7] and deployed in traditional information and communication technology (ICT) as well as industrial control systems (ICS) networks. While providing good results on well-known attacks, these types of intrusion detection schemes are not able to detect new or variants of known attacks and heavily rely on expert knowledge to build signatures. Therefore, statistical and machine-learning based [8] techniques were envisioned to cope with this limitation. However, these methods suffer from a higher rate of false positives compared to signature based approach, e.g. the classification of benign events as anomalies. Furthermore, these models rely on algorithms often specialized to specific types of anomalies that require expensive computation capabilities [9]. Most importantly, the lack of explainable results and accurate training dataset discourage their use in the industry [9].

However, when considering cellular vehicular networks, we argue that such kind of systems could be successfully embedded in tomorrow's car. In fact, the nature of the communications occurring between vehicles and the rest of the world differs from those of traditional ICT networks. We believe that CVN inherit from both mobile and ICS networks as two types of services namely vehicle-related and user-related, are operating on the same communication channel. Vehicle-related messages are carrying a high semantic meaning as they are dependent on the observation of physical events from sensors of the vehicle. On the other hand, user-related messages will mostly consist of infotainment applications communications such as music streaming, e-mails or map apps, which are closely related to current smartphone traffic. Thus, the content and frequency of vehicle-related communications are more predictable than user-related ones. Therefore, we believe that building an anomaly detector based on a model describing the communication of the vehicle would be beneficial in terms of detection capabilities, adaptation to evolution and explainable results.

In this paper we present an anomaly detection method based on an ontology representations of vehicular communications. The preliminary results on the detection capabilities and the comparisons with other methods show that the use of onto-

logical representation of information has significant impact on anomaly detection.

The rest of this paper is organized as follows: In section II relevant work on ontology for information security and anomaly detection in vehicular networks is presented. Then, section III introduces our ontological representation of the cellular vehicular communication for anomaly detection purposes. Then, section IV introduces the anomaly detection process including its architecture, the algorithms that perform the detection as well the inference rules that are used to build the representation of the anomalies. Then, section V presents the communication dataset along with the attack scenarios that are used for the evaluation of our method whose results are discussed in section VI. Finally, conclusions and future work are presented in section VII

II. Related Work

Ontologies are explicit formal specifications of terms and relations in a particular domain [10]. They have been greatly used in the World Wide Web in order to ease the search for information by automated processes (web crawlers) thanks to the use of expressive languages (RDFS, DAML, OWL...). Such languages enable domain-specific information sharing by experts. Ontologies were used in previous work in the field of information security. In [11], the authors present a survey of the use of semantics for the detection of targeted attacks. In [12], the authors present an ontology for the detection of application level intrusion in the hyper text transfer protocol (HTTP).

The authors in [13] modelled intrusions in terms of attacks directed towards a particular system component caused by a defined input. Such input has consequences on the system that are divided into two sub-classes, i.e. input validation error and exploit. The attack results are divided in a class of consequences, i.e denial of service, remote to local, user to root or probing. In their case, they use the reasoner rules to detect intrusions while in our case we rely on the ontology to dissect the network traffic to search for specific classes of anomalies in specific sub-set of the whole communication. Thus, we only use the reasoner to build a representation of the detected anomalies. In [14], the authors proposed a method based on an ontology and user-defined rules to represent network security situations in heterogeneous Internet of things (IOT) networks. Their goal is to provide security situation awareness based on multi-source information aggregation. In our case we rely solely on network traffic to detect anomalies in cellular vehicular communication.

In [15], the authors present a multi-agent based anomaly detection system where they use two agents for the detection process. The first agent receives packets translated into an ontological representation and uses the inference engine to detect known attacks signatures that are already inside the ontology. The second agent performs anomaly detection based on a clustering algorithm to discover new attacks.

Another similar multi-agent approach was taken in [16], in their case the authors also created a reaction agent that manages generated alerts to create a prevention model that reconfigures network devices based on a set of reaction rules.

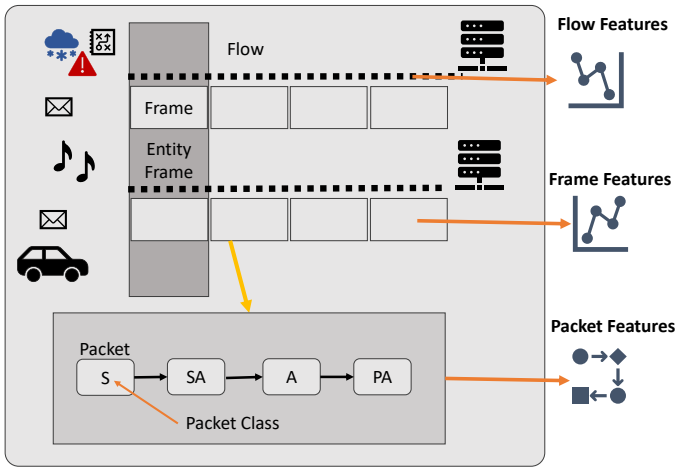

Fig. 1: Graphical Representation of the ontology components.

Lastly, in [17] the authors present an ontology for intelligent network forensics analysis. They designed their ontology following the METHONTOLOGY approach [18]. Their ontology models a knowledge representation of the main concepts involved in network forensic analysis such as the assets of a network, their vulnerabilities, the impact of attacks on these assets as well as attacker motivation and evidences showing malicious behavior.

III. Ontological Model of the Communication and Anomaly Detection

In this section, we introduce our anomaly detection method based on an ontological representation of the communications occurring between vehicles and the rest of the world.

A. Modelling the communications

As stated in our introduction, we show that it is possible to formally describe the communications of a vehicle with the rest of the world. Such a representation allows a better classification of the traffic in order to detect multiple types of anomalies. In fact, since the traffic carries high semantic meaning, it is possible to separate user-related communications from vehicle-related exchanges.

Moreover, vehicle-related traffic can also be distinguished between the different communicating components embedded inside a vehicle. For instance, infotainment system updates can be differentiated from vehicular telemetry messages.

The classes modelled inside our ontology are based on a representation of the communications at different scales as depicted by Figure 1. In our ontology, exchanges between a vehicle and an **Entity** are modelled into **Flows**. These flows are divided into **Frames** of fixed duration. Finally, every **Packet** passing through the vehicle's interface is assigned to a particular **Frame** of a **Flow**. The anomaly detection process uses features that are created when the ontology is populated.

a) Flow Attributes: Flows depict exchanges between the vehicle and another entity, they are defined as follows:

- start date and last date: Timestamps corresponding to the first and last packets seen on the network that are related to the flow.
- remote host: IP address of the remote entity that is communicating with the vehicle.
- host and remote port: ports used for the communication by the host and the remote entity.
- initiated by host: boolean value that shows whether or not the communication was started by the vehicle.
- transport type: type of transport protocol used (currently either UDP or TCP).
- FrameCollection: the list of the frames that were created for the flow
- FlowFeatures: Collection of features regarding the flow (e.g. average number of packets per seconds)

b) Frame Attributes: Frames consist of sequences of packets that are exchanged inside a flow during a specific period. They are defined as follows:

- start date and end date: Timestamps corresponding to the first packet received for the frame, the end date is fixed and corresponds to the $start + 3$ seconds.
- FrameFeatures: Collection of features regarding the frame, these features are used to build the features of the corresponding flow (see d. below).
- Flow: Relational link to the flow that the frame is a part of.
- PacketCollection: Collection of packets that are part of the frame.

c) Packet Attributes: The packets that populate the frames are defined as follows:

- Packet size: represents the size of the packet
- Packet class: represents either the TCP flags or DNS queries and response
- Summary: This attribute is used during the anomaly representation phase and represents a human-readable summary of the packet.
- Timestamp: Date of reception of the packet
- Direction: information on whether packet is inbound or outbound to the vehicle
- Frame: relational link to the frame that the packet is a part of.

d) Flow and Frames Features: The flow and frames features consist mostly of key statistical attributes of the traffic set to a flow. They range from the number packets in a frame, their average size, the ratio of received packet versus sent packets to the mean inter-packet gap. We maintain 44 features drawn from on a recent study [19]. Moreover, based on our knowledge of the underlying processes that trigger the communications, we assign a specific class to the flow, e.g. vehicle or user-related flow.

e) Packet sequences: Network packets carry a lot of information often represented in the Open Systems Interconnection model [20]. Each layer in the model corresponds to a specific function in the communication. In our work we only consider three layers, i.e. network, transport and application. We group session, presentation and application in a single layer for convenience. From each of these layers we extract interesting key information in order to determine the semantic that the packet carries, and assign it to a specific *packet class* inside the ontology. At the time of writing this paper, our classes are based on TCP flags and regarding UDP communications. We only consider dns queries and responses. Every other packets are assigned to a default class.

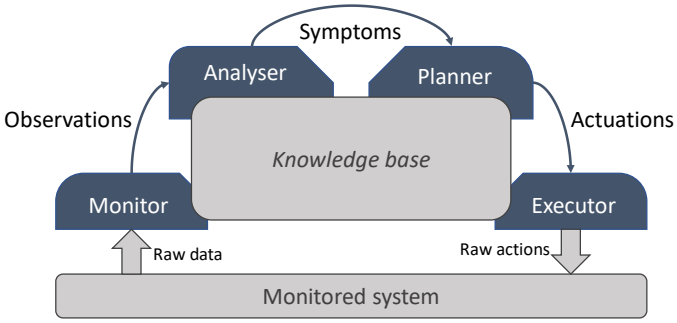

Fig. 2: Mape-k loop for our anomaly detection system.

The use of an ontology representation to model the communications of a vehicle has three purposes in the anomaly detection process. First, it allows us to extract interesting features from the network traffic of the vehicles in a multiple-scale manner (e.g frame-level, packet-sequence level, and entity-level). Secondly, using inference rules on the relations between packets, frames, flows and entities eases the anomaly detection process while also allowing us to build contextual information to represent the anomaly.

IV. Anomaly Detection

A. Architecture of the Anomaly Detection System

We designed our anomaly detection system architecture based on IBM's Monitor, Analyse, Plan and Execute over a shared knowledge (MAPE-K) feedback loop. It was introduced in [21] and updated in [22] as a way to structure autonomic computing systems to ease their management. Mape-k is considered a reference for the modelling of autonomic and self-adaptive systems.

In our case, since the anomaly detection process takes place inside the vehicle, it can be seen as an autonomic computing system. The general architecture of our system is depicted in Figure 2. In this paper, we focus on the monitor, analyser and knowledge base components. Planning and execution components are considered in the future work section VII.

The architecture is defined as follows. We consider a monitored system which, in our case, corresponds to the communication of the vehicle. For each packet that flows through the network interface of the vehicle an event is created and handled by the monitor module. The events are composed of the packet and a timestamp. The monitor extracts

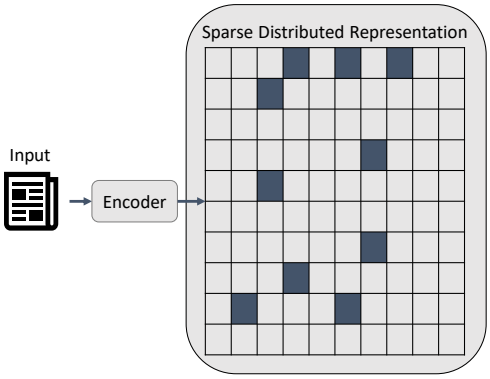

Fig. 3: Encoders transform inputs into sparse distributed representations (SDR), dark colour represents active bits in the SDR.

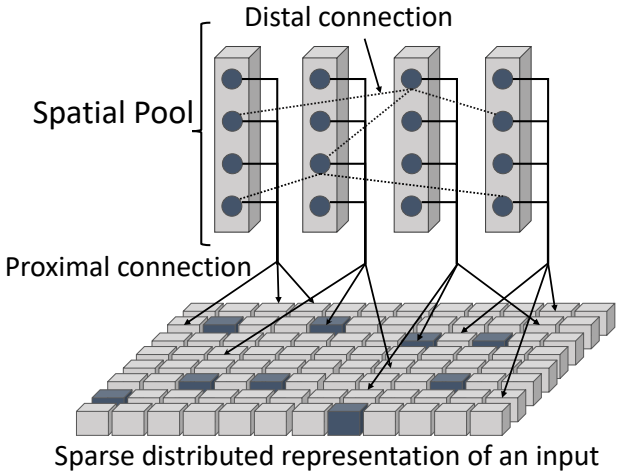

Fig. 4: Spatial pooler representation. Arrows represent the proximal connection of columns of the spatial pool to a particular input of the SDR. Dotted lines represent distal connections between cells inside the spatial pool

from these packets the attributes of the communication and populates the ontology that is stored inside the knowledge-base. Furthermore, the monitor module is also responsible of creating the features of each component (e.g. Flow, Frame, Packet).

Finally, each time a feature is computed, it triggers a symptom that is sent to the analysis module. The analysis module is in charge of running the anomaly detection process based on the symptoms and attributes from the communication that are stored in the knowledge base.

Therefore, the anomaly detection process is divided in two parts. The first part is responsible for running the anomaly detection process based on the features, while the second part propagates detected anomalies inside the knowledge base using an inference rule. These two aspects of our systems are detailed in the following subsections.

B. Unsupervised Machine Learning Algorithm: Hierarchical Temporal Memory (HTM)

HTM is an online sequence memory algorithm based on theories of the functioning of human brains and particularly the neocortex. It was originaly introduced by Jeff Hawkins and Sandra Blakeslee in [23]. It was successfully used to detect anomalies in streaming data in [24].

HTM systems are based on three building blocks: encoders, spatial pooling and temporal memory. Encoders are key components of the HTM algorithm, their goal is to encode input for the algorithm into a sparse distributed representation (SDR) as depicted in Figure 3. These representations are composed of bit-arrays defined as follows [25]:

- Semantically similar data should have overlapping bits.
- SDRs are deterministically created, i.e. the same input must result in the same output.
- Output of encoders should have the same dimensionality (length of the bit-array).
- Outputs should have the same number of active bits (sparsity).

These SDRs are used by the spatial pooler for the learning and prediction of the algorithm. The spatial pooler represented in Figure 4 is composed of columns of cells. Each cell has two types of connections, namely proximal and distal. Proximal connections are responsible for connecting a column of the spatial pooler to a particular input. On the other hand, the distal connections connect cells of the columns to other cells of the spatial pooler.

The algorithm uses these connections to learn the transition of patterns inside SDRs. In fact, each time an input is received, columns that are connected to this input are triggered if their proximal connections are connected to an active bit of the SDR. Then each cell of an active column, uses its distal connections to turn into a predictive state all the cells that it is connected to. If the next input triggers these predicted cells, the distal connections are reinforced. Thus, the algorithm is constantly updating itself in order to build a predictive model of the data that it is observing. For each correct prediction it reinforces these links inside the data.

Currently, we run HTM on a combination of feature attributes defined for the frame class. Thus, anomalies are not generated for a single packet but for entire frames of a same flow.

Future work will involve the packet analysis based on the *packet class* that we extract from the traffic. Combining these two approaches will allow us to improve the detection of different types of anomalies. For instance, a volume anomaly, e.g. a denial-of-service, would be detected by comparing different features of the *frames* of a *flow* without having to consider every class of *packet*. In the meantime a sequence anomaly, e.g. a syn-scan, could be detected by analysing unusual sequences of *packets classes*.

C. Anomaly representation and inference rules

a) Anomaly representation: An important element of our model is the representation of anomalies in a way that allows for sharing and understanding generated alerts.

We represent detected anomalies inside our ontology with the following attributes:

- Root: The tuple *Entity*, *Flow*, *Frame* that triggered the anomaly.
- ContaminatedFrames: The list of *Frames* and therefore *Packets*, that are closely related to the anomaly and were also classified as anomalous be cause they were received just after the generated anomaly. We only classify a number of frames (N) received after the generated anomaly in order to avoid generating too large anomalies.
- SuspiciousFlows: The list of *Flows* that the anomalous remote entity might have with the vehicle.
- ContextFrames: The list of *Frames* that were active when the anomaly was detected.

In order to understand the anomalies, we retrieve the context surrounding a detected anomaly using inference rules. Inference rules have two roles in our anomaly detection system. We use them to link a detected anomaly to the corresponding *Packets*, *Frames* and *Flow* inside the knowledge base, allowing us to detect anomalies in a efficient way. Secondly, we use them to build representations of the detected anomalies which could be used to share the anomaly with other detection agents or store them for evidence purposes inside the vehicle.

b) Root inference rules: The inference rules are based on the composition relationship materialized in our ontology by the *partOf* axiom. Said axiom binds *packets* to *frames*, *flows* and *entities*:

- partOf(x, y) $\wedge$ isAnomalous(x) $\rightarrow$ isAnomalous(y)

Therefore, if a *packet* is deemed anomalous, the *frame* as well as the *flow* are also categorized as anomalous. Finally, the entity that took part in the communication with the vehicle is also categorized as anomalous.

We use similar rules to build *ContaminatedFrames*, *SuspiciousFlows* and *ContextFrames*. In fact, if a packet triggered an alert, it would be beneficial for an operator to get:

- All other flows that the anomalous entity uses to communicate with the vehicle.
- Every packets that were also captured during this anomaly.
- Other packets of the anomalous flow.

These rules are defined as follows:

1) ContaminatedFrames:
   - isAnomalous(Frame(x)) $\wedge$ hasDistance(x, d) $\wedge$ lessThan(d, N)
     $\wedge$ isNext(x, Frame(y)) $\rightarrow$ isAnomalous(y)
     $\wedge$ hasDistance(y, add(d, 1))
2) SuspiciousFlows:
   - hasRoot(Anomaly(a), Entity(x)) $\wedge$ hasFlowCollection(x, c) $\wedge$ partOf(c, Flow(f)) $\wedge$ $\neg$ isAnomalous(f) $\rightarrow$ hasSuspiciousFlow(a, f)
3) ContextFrames:
   - hasRoot(Anomaly(a), Frame(x)) $\wedge$ Date(x, T_x) $\wedge$ Start(Frame(y), s) $\wedge$ End(Frame(y), e) $\wedge$ greaterThan(T_x, s) $\wedge$ lessThan(T_x, e) $\rightarrow$ isPartOfContext(a, f)

We limit the number of packets returned by the ContaminatedFrames rule as it could have a detrimental effect on the understanding of the anomaly if it returns too much information, especially because categorizing the *flow* as anomalous will retroactively contaminate every *packet* inside it. Thus, we returns packets depending on the distance (d) between a *frame* where a packet was deemed as anomalous and other *frames*. Therefore, attention is drawn to the closest contextual knowledge of the anomaly.

V. Anomaly Dataset Construction

In order to validate the design of our anomaly detection system, we created a communication dataset based on an emulation environment. This section introduces the communication and anomaly generation procedures that we used to build the dataset.

A. Emulation environment

Anomaly detection applied to network communications is well known to encounter issues when considering training datasets. Quite often, communication datasets used by researchers cannot be published due to privacy concerns or by law restrictions. Therefore, results on new methods for anomaly detection lack repeatability. Furthermore, recent datasets such as the ones introduced in [26] are mostly dedicated to classical ICT networks. In fact, to our knowledge there is no real communication dataset available regarding connected vehicles. Thus, we created our own dataset using an emulation environment dedicated to a cellular vehicular network named **Autobot** [27]. We followed the guidelines introduced by Shivari et al. in [28] and summarized as follows:

- **Realism**, in order to show the effects of attacks and anomalies on the network, they are being emulated in ***AutoBot*** and not post inserted inside a dataset.
- **Evaluation**, since we are in control of the environment, generated datasets are deterministically labelled.
- **Anomalous Activities**, at the time of writing this article we implemented several anomalies such as DNS-tunnelling attack, Network Scans and telemetry anomalies (see subsection V-B).

The emulation runs in a completely isolated environment using docker containers. Each container acts as a vehicle connected to a docker network as depicted in Figure 5. In order to respect cellular network behaviour, `netem` [29] is used to shape the latency and bandwidth of every virtual interface for every container. The latencies and bandwidth were determined based on actual measurements done inside a vehicle cruising on the ring road of Toulouse. **Autobot** allows the emulation of up to 300 vehicles on a server running Ubuntu 16.04 with 32 processors (2.6 GHz) and 64 gigabytes of RAM. In our scenario, we emulate a single vehicle and capture the communications on its virtual interface.

Vehicles run several applications that generate traffic which could appear in connected vehicles in the future.

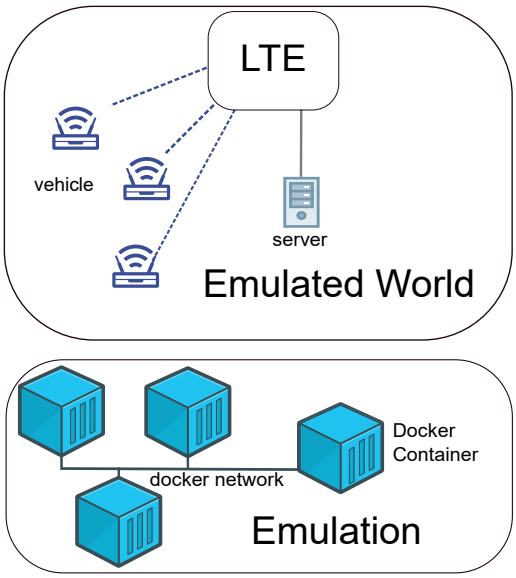

Fig. 5: Representation of the Autobot emulation environment.

a) Vehicle Telemetry: In order to enable the new ITS services presented in the introduction, vehicles will have to share information they gather from their sensors with remote servers. To this end, an exchange format was created by a consortium of automotive industrial. This message format, Sensoris, is meant to enable different parties to share knowledge between each other regardless of the vehicle's manufacturer. We embed inside every vehicle container a Message Queueing Telemetry Transport (MQTT) client that sends Sensoris messages to another container that acts as a server. The data that is sent in these messages was extracted from real-life vehicles that are used on the E-horizon project at Continental Digital Services France for research and development purposes. The data contains information ranging from accelerometer to temperature and GPS coordinates.

We extended Autobot in order to increase the realism of the communication dataset by adding infotainment applications that are likely to exist in future vehicles.

b) Infotainment applications: In fact, in order to improve the experience of users when driving, vehicles will likely provide services that are currently only available on smart-phones. We created a containerized music streaming application based on the Spotify API as well as a map navigation application based on Waze.

B. Emulated Anomalies

We generated anomalies based on scenarios drawn from a survey introduced in [1]. Furthermore, we also performed attacks inspired by real-life situations that were reported in the last few years. However, we want to draw the attention of the reader to the fact that technical details on how to perform the exploits are left out of these reports. In fact, while vehicles tend to be more connected, it is not always possible to perform firmware updates over the air. Thus, we created our own exploits based on the data available. Finally, most attacks

Vehicle IP	Telemetry Server IP	Infotainment Servers IPs	Attacker IPs	Protocols
10.0.0.5	10.0.0.6	104.199.64.136/23 130.211.34.59 130.211.9.172 140.93.5.46 157.240.195.35 172.217.0.0/16 179.60.192.36 216.58.198.67 35.201.119.145 66.102.1.154 74.125.140.155	10.0.0.1 10.0.0.3	mqtt (85.4%) https (13.6%) dns (0.6%) http (0.1%)

TABLE I: IP Repartition and relevant protocols present inside the dataset

presented in this paper require that vehicles are provided with a public IP address, or alternatively that mobile network operators allow device-to-device communication. However, we consider this pre-requisite to be very plausible as the upcoming 5G[1] allows device-to-device communications.

a) Dataset Generation: At the time of writing this paper, our dataset includes four different scenarios that were performed over a four hour period inside the emulation environment. The traffic captured during this period is summarized in the Table I.

b) Normal Behaviour: In the first hour, we emulate normal communication behaviour during which the vehicle starts the telemetry service as well as the infotainment applications.

c) Network Scans: During the second hour, an attacker tries to find listening ports of a vehicle by sending specially crafted packets. It is often considered as the first step that attackers perform in order to gain access to a system. We used `nmap` to perform the scan.

d) Remote Exploitation: Based on a discovered vulnerability an attacker is able to remotely break into the vehicle infotainment system and perform malicious activity such as shutting down the system or extracting private information about the users of the vehicle. This scenario was drawn from an actual exploit that was done by Computest and reported in [30]. In our case, we decided to perform a data extraction attack via DNS-tunnelling using a tool named `dnscat2`.

e) Anomalies in Vehicle Telemetry: We generate anomalies in the vehicle telemetry messages generated by the vehicles. For this scenario, we emulated a malfunction inside the vehicle's telemetry service that prevents it to send information over the telemetry service. In this case, the connection with the remote server stays active but no messages are sent by the vehicle.

VI. Evaluation

In this section, we present a preliminary evaluation of our anomaly detection system. We highlight the importance of using the inference rules to improve the detection rate of the system. Then, we compare the detection capabilities of our

[1] http://5gaa.org/

Feature	Bit array length (n)	Sparsity	resolution
startframe	2048	41	0.001
pkts/s	2048	41	0.001
meanfpktl	1500	31	0.01
avgpktsize	400	21	0.001
category	400	21	NA

TABLE II: Weights for the encoding of the features inside the spatial pooler.

method against popular machine learning algorithms, e.g. one-class support vector machine (OCSVM) and Density-based spatial clustering of applications with noise (DBSCAN).

A. Feature Selection

In [26], the authors present an analysis of the most important features by class of attack. In order to test the detection capabilities of our method we used the most common important features given the attacks and anomalies that we generated inside our dataset. The features we used are drawn from the frames that populate the ontology and represent the following attributes of the communication occurring for the duration of the frame:

- mean outgoing packet length (meanfpktl)
- the number of packets per second (pkts/s)
- the average packet size (avgpktsize)

The HTM algorithm is based on time series therefore we also use the timestamps that represent the creation date of a frame. Furthermore, in order to give the algorithm knowledge on the type of traffic it is analysing, we defined an encoder for the category of traffic. This encoder represents the class of the flow. Lastly, for the anomaly detection process the algorithm has to predict the values of the number of packets per second. The anomaly score is calculated by comparing the prediction results against the actual value of the next received frame. We consider the detection of an anomaly when this score is superior to some threshold (in our case 0.9).

The weight of each feature inside the encoding of the whole input for the spatial pooler was defined empirically and the values are presented inside the Table II

B. Detection Results

In order to evaluate our model, we perform the anomaly detection based on these features and three different set-ups, i.e. without the ontology, with the ontology and no inference rules, and finally with ontology and the inference rules. These set-ups differ in the way they extract the features from the traffic and how the anomaly detection is performed.

In the first set-up, we extract the features from all the communications occurring during a certain time window, which corresponds to Entity Frame (see Figure 1). The second and third set-ups both use the ontology to extract the features from the traffic. The third set-up uses the inference rules in order to improve the detection results.

The results are presented in the Table III. The label column represents the frames that contain either normal or anomalous

Label	No Ontology	Ontology	
		No Inference	Inference
False Positive Rate	2.3% (97/3291)	3.4% (298/8736)	14% (1228/8736)
Scan	0% (0/2)	0.6% (7/1024)	0.6% (7/1024)
Data extraction	1.4% (3/211)	9.4% (25/264)	39.0% (103/264)
Telemetry	3.7% (1/27)	27.2% (3/11)	90.9% (10/11)

TABLE III: Frames classified as anomalies according to different feature extraction methods.

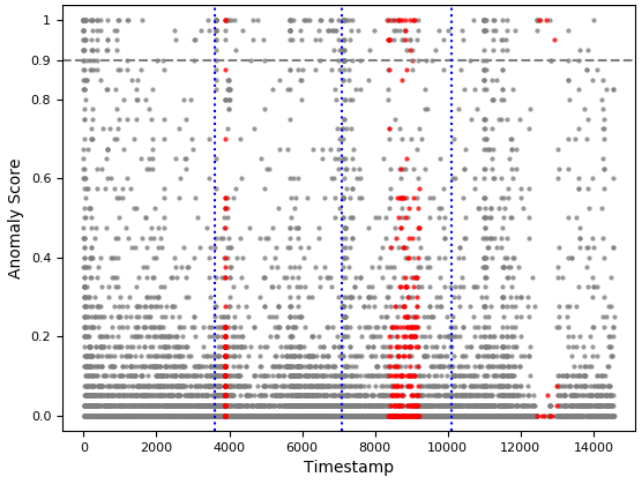

Fig. 6: Detection results for HTM using the ontology without inference.

traffic, the false positive rate row represents the ratio of normal frames that were classified as anomalous. The other columns represent the ratio (n/m) of n frames that were classified as anomalies out of m. We can see that the use of the ontology has a great impact on the true positive rate while having relatively low impact on the false positive rate of the detection. Moreover, the use of the inference rules significantly increases the detection ratio of the telemetry anomaly but also has great impact on the false positive rate. Lastly, it is worth noting that it has no impact on the detection rate of the scan attack.

C. Results interpretation

The fact that the scan attack is not detected very well by the system while using the ontology to build the features of the frame is mostly due to the fact that these features are based on a particular flow. In fact, when an attacker performs a scan he sends specially crafted packets to different ports of the victim. In this case, our ontology will build a flow each port that the attacker tries. Furthermore, since the inference rule for contamination is based on the flow, the detection ratio for the scan cannot be improved by this method.

The high rate of false positive can be explained by a close analysis of the dataset. The Figure 6 represents the anomaly detection results of the HTM algorithm using the ontology

Label	HTM		OCSVM		DBSCAN	
	S1	S2	S1	S2	S1	S2
False Positive Rate	6.6%	3.4%	0.16%	37.1%	68.3%	0%
Scan	0.1%	0.6%	97.7%	97.7%	0%	0%
Data extraction	6.1%	9.4%	0%	96.2%	100%	0%
Telemetry	9%	27.2%	0%	90.1%	100%	0%

TABLE IV: Frames classified as anomalies according to different feature extraction methods.

without inference. The horizontal dotted line represent the detection threshold used to classify an anomaly. The vertical blue lines represent the timestamps where the emulation environment has to restart for the next scenario (see section V). Lastly, the red dots represent the anomalous frames inside the dataset.

As previously stated, the HTM algorithm is unsupervised. Therefore, it does not need training and is able to start the predictions as it discovers the dataset. Thus, the fact that there is a high number of anomalies detected in the beginning of the dataset is normal. Furthermore, the fact that all communications are stopped abruptly at the end of each scenario generates numerous anomalies when new communications are encountered in the following scenario.

This behavior is also shown by the detection of the telemetry anomaly that is generated during the last scenario. The anomalies generated during this scenario show that the first few frames are classified as anomalous before being treated as benign, then right after the anomaly stops the algorithm detects a few anomalies. This shows that the HTM algorithm is able to adapt itself to new communication patterns, the drawback of such behaviour is that it also triggers false positive when the pattern changes in normal situations.

D. Comparison with OCSVM and DBSCAN

We ran a comparison benchmark of the HTM algorithm with two other algorithms, One-class support vector machine and Density-based spatial clustering of applications with noise. The hyper-parameters for OCSVM and DBSCAN were computed by minimizing a cost function over a search space representing the hyper-parameters using hyperopt [31]. The cost function is the number of anomalies detected by the algorithms that has to be minimized.

The results of this benchmark are shown in Table IV. We tested the different algorithms based on every feature available inside the ontology (S1) and based on the features presented in the previous results (see Table III), i.e. packets per seconds, mean outgoing packet length and mean packet size.

Thus, we can see that the HTM algorithm is the only one to detect at least one frame of every anomaly while maintaining a relatively low false positive rate using every feature available. However, we note that OCSVM is very efficient at recognizing the scan attack with very low false positive rate while using every feature.

Nevertheless, when reducing the number of features the detection results of the OCSVM and DBSCAN algorithms become unreliable with a very high rate of false positive. On the contrary, the detection of the telemetry anomaly by the HTM algorithm are improved when a smaller set of features are used. Using fewer features also improves the false positive rate of HTM.

E. Limitations

These preliminary results describe a promising first step in the implementation of a complete anomaly detection system. However, several limitations remain. First, the size of the communication dataset and the range of attacks that were performed leads to difficult training for the DBSCAN and OCSVM because there is not enough occurrences of the anomaly classes. However, the results obtained by the HTM algorithms lead us to think that it will perform well under larger datasets. Finally the false positive rate also needs to be reduced for such a method to be embedded successfully inside vehicles.

VII. Conclusion and Future Work

A. Conclusion

In this paper, we introduced an anomaly detection method based on an ontological representation of cellular vehicular communication. We showed how using a semantic approach to packet analysis could reduce the number of features that are needed for the anomaly detection process. Moreover, we presented how our model is integrated to an anomaly detection system based on IBM's Mape-K loop [21], [22] that uses the Hierarchical Temporal Memory algorithm for the detection process in combination with inference rules.

In order to evaluate our approach we presented our dataset creation method based on a emulation environment dedicated to cellular vehicular networks. Finally, we presented a performance comparison against other algorithms used for network anomaly detection using our dataset.

These encouraging preliminary results constitute a first step in the implementation of a complete anomaly detection system that will be able to run on its own inside vehicles. Several subsequent work is still needed in order to fulfil this goal; it is discussed in the next subsection.

B. Future Work

First, the training dataset needs to be improved in order to test a larger range of attacks and anomalies that our method is able to detect. To this end, we plan testing the algorithm under a greater range of attacks, such as other types of scans and denial-of-service attacks as well as ransomware attacks.

Secondly, we will embed new communication applications such as updates over-the-air to improve the realism of the dataset. We believe that these types of applications will become more and more popular. In fact, to manage vehicle fleets and provide users with new applications, updates will need to be sent over the air to vehicles. Such applications are

already used. For example during Hurricane Florence[2] Telsa car owners were allowed to increase the range of their vehicles if they were located in the path of the Hurricane.

Moreover, to improve the detection results an optimization of the features and parameters of the HTM algorithm will be conducted. To embed other key aspects of the communication that are often used as attack vectors such as HTTP keywords or wrongly crafted packets, the ontology will also be extended.

Finally, to build a complete anomaly detection system, future work on our design will involve the alert sharing process that could be put in place in order to help other vehicles detect more efficiently anomalies that were already experienced by other vehicles.

ACKNOWLEDGMENTS

The authors wish to thank Silvia Gil Casals for her contribution to the comparison benchmark and her advises throughout this work and Continental Digital Services France for funding this work.

REFERENCES

[1] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, "Comprehensive experimental analyses of automotive attack surfaces."

[2] I. D. Foster, A. Prudhomme, K. Koscher, and S. Savage, "Fast and vulnerable: A story of telematic failures."

[3] S. Jafarnejad, L. Codeca, W. Bronzi, R. Frank, and T. Engel, "A car hacking experiment: When connectivity meets vulnerability," in *Globecom Workshops (GC Wkshps), 2015 IEEE*. IEEE, 2015, pp. 1–6.

[4] S. Mazloom, M. Rezaeirad, A. Hunter, and D. McCoy, "A security analysis of an in-vehicle infotainment and app platform."

[5] S. Bayer, T. Enderle, D.-K. Oka, and M. Wolf, "Security crash test-practical security evaluations of automotive onboard it components," *Automotive-Safety & Security 2014*, 2015.

[6] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," 2015.

[7] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[8] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[9] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010, pp. 305–316.

[10] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[11] R. Luh, S. Marschalek, M. Kaiser, H. Janicke, and S. Schrittwieser, "Semantics-aware detection of targeted attacks: a survey," *Journal of Computer Virology and Hacking Techniques*, vol. 13, no. 1, pp. 47–85, 2017.

[12] A. Razzaq, H. F. Ahmed, A. Hur, and N. Haider, "Ontology based application level intrusion detection system by using bayesian filter," in *Computer, Control and Communication, 2009. IC4 2009. 2nd International Conference on*. IEEE, 2009, pp. 1–6.

[13] J. Undercoffer, A. Joshi, and J. Pinkston, "Modeling computer attacks: An ontology for intrusion detection," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2003, pp. 113–135.

[14] G. Xu, Y. Cao, Y. Ren, X. Li, and Z. Feng, "Network security situation awareness based on semantic ontology and user-defined rules for internet of things," *IEEE Access*, vol. 5, pp. 21 046–21 056, 2017.

[15] I. Brahmi, H. Brahmi, and S. B. Yahia, "A multi-agents intrusion detection system using ontology and clustering techniques," in *IFIP International Conference on Computer Science and its Applications*. Springer, 2015, pp. 381–393.

[16] G. A. Isaza, A. G. Castillo, and N. D. Duque, "An intrusion detection and prevention model based on intelligent multi-agent systems, signatures and reaction rules ontologies," in *7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009)*. Springer, 2009, pp. 237–245.

[17] S. Saad and I. Traore, "Method ontology for intelligent network forensics analysis," in *2010 Eighth International Conference on Privacy, Security and Trust*. IEEE, 2010, pp. 7–14.

[18] M. Fernández-López, A. Gómez-Pérez, and N. Juristo, "Methontology: from ontological art towards ontological engineering," 1997.

[19] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features." in *ICISSP*, 2017, pp. 253–262.

[20] H. Zimmermann, "Osi reference model-the iso model of architecture for open systems interconnection," *IEEE Transactions on communications*, vol. 28, no. 4, pp. 425–432, 1980.

[21] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, no. 1, pp. 41–50, 2003.

[22] A. Computing *et al.*, "An architectural blueprint for autonomic computing," 2006.

[23] J. Hawkins and S. Blakeslee, *On intelligence: How a new understanding of the brain will lead to the creation of truly intelligent machines*. Macmillan, 2007.

[24] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[25] S. Purdy, "Encoding data for htm systems," *arXiv preprint arXiv:1602.05925*, 2016.

[26] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *ICISSP*, 2018, pp. 108–116.

[27] Q. Ricard and P. Owezarski, "Autobot: An emulation environment for cellular vehicular communications," in *Proceedings of the 2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications*. IEEE Computer Society, 2019.

[28] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *computers & security*, vol. 31, no. 3, pp. 357–374, 2012.

[29] S. Hemminger *et al.*, "Network emulation with netem," in *Linux conf au*, 2005, pp. 18–23.

[30] Computest, "The connected car-ways to get unauthorized access and potential implications," *Online: https://www.computest.nl/wp-content/uploads/2018/04/connected-car-rapport.pdf*, 2018.

[31] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[2]https://bit.ly/2Wy9PeO