Advanced Analytics for Connected Car Cybersecurity

Matan Levi
Ben-Gurion University
Be'er Sheva, Israel
matanle@post.bgu.ac.il

Yair Allouche
IBM Security Division
Be'er Sheva, Israel
yair@il.ibm.com

Aryeh Kontorovich
Ben-Gurion University
Be'er Sheva, Israel
karyeh@bgu.ac.il

***Abstract*—The vehicular connectivity revolution is fueling the automotive industry's most significant transformation seen in decades. However, as modern vehicles become more connected, they also become much more vulnerable to cyber-attacks. In this paper, a fully working machine learning approach is proposed to protect connected vehicles (fleets and individuals) against such attacks. We present a system that monitors different vehicle interfaces (Network, CAN, and OS), extracts relevant information based on configurable rules, and sends it to a trained generative model to detect deviations from normal behavior. Using a configurable data collector, we provide a higher level of data abstraction as the model is trained based on events instead of raw data, which has a noise-filtering effect and eliminates the need to retrain the model whenever a protocol changes. We present a new approach for detecting anomalies, tailored to the temporal nature of our domain. Adapting a hybrid approach to the fully temporal setting, we first train a Hidden Markov Model to learn normal vehicle behavior, and then a regression model to calibrate the likelihood threshold for anomaly. Using this architecture, our method detects sophisticated and realistic anomalies, which are missed by other existing methods monitoring the CAN bus only. We also demonstrate the superiority of adaptive thresholds over static ones. Furthermore, our approach scales efficiently from monitoring individual cars to serving large fleets. We demonstrate the competitive advantage of our model via encouraging empirical results.**

***Index Terms*—Anomaly detection, Connected cars, Hidden Markov models, Intrusion detection, Linear regression, Vehicle Cybersecurity.**

I. Introduction

Over the past few years, we have been witnessing a continual transformation of the automotive industry, whereby new technologies are being integrated into vehicles, changing the traditional concept as we know it and improving safety, performance, and efficiency.

However, as vehicles become more connected, they also become more vulnerable to remote cyber-attacks, as researchers have recently pointed out. Koscher et al. [1] demonstrated how, in cases where it is possible to compromise a car's internal network, it becomes possible to control a wide range of essential functions: disabling the brakes, selectively braking individual wheels, stopping the engine, etc. Following the publication of [2] and [3], which detailed a Jeep Cherokee being remotely hacked and stopped on a highway, Chrysler issued a recall for 1.4 million vehicles. Checkoway et al. [4] presented a remote exploitation technique using different attack vectors (e.g., Bluetooth and cellular), which enabled a remote takeover of the vehicle, as well as access to the acoustics inside the cabin and the vehicle's location. Another team managed to hack the wireless interface of the tire pressure monitoring system and use it for eavesdropping and tracking the vehicle [5]. Further research has shown that the passive keyless entry and engine start-up system can also be hacked [6].

Protecting modern vehicles is a challenging task for three main reasons: complexity, connectivity (large attack surface), and legacy (unsafe and outdated technologies).

In [7], it was estimated that a premium-class automobile contains over 100 million lines of code that is executed on 70–100 Electronic Control Units (ECUs), and this number is projected to grow to 200–300 million lines of code in the near future. This means that such vehicles are highly complex machines with many potential vulnerabilities caused by either wrong integration or by human error.

The connectivity revolution makes it possible for modern vehicles to become connected through a wide range of network interfaces, e.g., Wi-Fi, cellular, dedicated short range communication (DSRC), etc. This connectivity allows manufacturers to send over-the-air (OTA) updates, receive diagnostic information, and offer various media services. But, as the number of network interfaces increases, so does the attack surface.

Modern vehicles are controlled and monitored by tens of ECUs that communicate over one or more internal network buses based on the *unsecured* control area network (CAN) protocol which has a limited 1 Mbps bandwidth and a data field of 8 bytes. Since the CAN packets have no source/destination identifier, the ECUs communicate by broadcasting packets on the CAN bus, and each ECU decides whether the packet was intended for it.

The two main lines of defense suggested against such attacks are message authentication/encryption and intrusion detection. The in-vehicle network's nature makes it difficult to adopt message authentication and/or encryption. Given the CAN protocol limitations, any cryptographic message authentication would have too weak of a key to be useful. In [8], researchers suggested a delayed authentication scheme that uses multiple CRC fields to compound a 64-bit CBC-MAC (cipher block chaining message authentication code).

Another approach is to build a new protocol, CAN flexible data-rate (FD), which allows flexible data rates and longer payloads [9]. However, due to the fact that messages must be broadcast at a high frequency, the encryption/authentication mechanisms may lead to delays, which could impair vehicle safety. Moreover, authentication techniques for the in-vehicle network would not necessarily prevent attackers from remotely attacking the car and gaining access using its own network interfaces.

Hence, one is led to consider intrusion detection systems (IDSs). However, rule-based IDSs are fragile and cannot cover the full range of abnormal behavior. This shortcoming is mitigated by machine learning anomaly detection systems. One complication is that the system must monitor not only the CAN bus traffic, but rather the different vehicle interfaces (e.g., OS, Network, and CAN).

Our contribution consists of a novel approach that applies cross-system monitoring, adapts data abstraction, and applies HMM to detect anomalies using a new temporal detection technique. Due to the connectivity revolution, attackers can attack a vehicle without leaving any trace on the CAN bus. Therefore, monitoring the CAN bus traffic alone is not enough. We suggest monitoring the vehicle's main interfaces (e.g., OS, Network, and CAN). By such monitoring, we can detect much more sophisticated and complex attacks that combine several attack vectors from the different interfaces and do not necessarily include the CAN bus (e.g., an attacker eavesdropping using a malicious application).

Additionally, we propose to train the model based on events generated from the raw data by a rule-based engine which monitors the different interfaces. By preprocessing the raw data into events, we provide a higher level of data abstraction, which eliminates the need to retrain the model whenever a protocol is updated and helps filter noise. Generating events will also be effective implementation-wise, as we discuss in the different implementation techniques in Section VI.

By modeling time-series data to state changes, we can detect anomalous changes in states which can indicate abnormal behavior. Therefore, we suggest training an HMM as a normal behavior model using events collected from the different interfaces. The actual anomaly detection engine is inspired by [10], and adapted to our fully temporal setting.

Our algorithms are trained on an abstraction of the raw data in the form of "events". This abstraction eliminates the need to retrain the model each time there are protocol updates. Additionally, it acts as a dimensionality reduction technique, helping to reduce bandwidth demand and filter out noise.

In Section VI, we discuss the different techniques to implement our solution. We present the hybrid approach we used, where a rule-based client (data collector) is integrated into the vehicle and sends events to the backend. This technique allows us to monitor car fleets, detect cross-fleet anomalies, and identify correlations between different vehicles.

II. Related Work

A. Frequency-Based Techniques

Several researchers suggested using the CAN packet frequency to detect abnormal activity on the CAN bus. Due to the limited computational power in vehicles, Song et al. [11] proposed a lightweight intrusion detection algorithm based on the fact that each message ID has a regular frequency, and when attackers inject messages into the CAN bus, the message frequency changes abruptly. In [12], Cho and Shin likewise relied on the fact that most in-vehicle network messages are periodic and broadcast over CAN, and suggested exploiting the time intervals of these periodic messages as ECU fingerprints. These methods are mainly effective for periodic messages, so an attacker who injects messages aperiodically may go undetected. Moreover, when the ECU is itself the source of the malicious packets' IDs, attacks may go undetected.

B. Statistical Techniques

Another approach to anomaly detection is to use statistical tools to construct a "normal" baseline and then to identify deviations from the norm. One idea is an entropy-based model [13], [14]. In this regard, researchers have proposed an entropy-based anomaly detection technique. In [13], the entropy of the in-vehicle network was taken as the normal behavior baseline. The basic intuition is that due to the clear and restrictive specification of the in-vehicle traffic, the entropy is relatively low, and therefore, attacks (e.g., changing packet payload, packet injection) would cause the entropy to increase. Marchetti et al. [14] carried out an extensive experimental evaluation using hours of real CAN data. Several limitations of entropy-based approaches were noted. In particular, in order to detect low-volume attacks, one must build an anomaly detector for each message ID. Han et al. [15] divided the data into four categories (engine, fuel, gear, and wheel) and used a one-way ANOVA test to identify abnormal activity.

C. Machine Learning Techniques

Much work has been done in the field of anomaly detection using machine learning techniques [16]. Given the scope of this paper, we restrict our attention to anomaly detection in vehicles and related applications.

1) Classification-Based Techniques: In the automotive context, Theissler [17] proposed using a one-class SVM with the radial basis function (RBF) kernel to learn baseline normal behavior and classify deviations as anomalies. The resulting classifier is applicable to sequences of events, but does not detect point anomalies.

2) Clustering-Based Techniques: In [18], Li et al. suggested using Gaussian mixture model (GMM)-based clustering to detect flights with abnormal patterns. The clusters are then characterized using their temporal distribution.

3) Deep Learning Techniques: Several articles proposed anomaly detection using deep learning in the automotive field [19], [20]. In [19], Kang and Kang suggested an intrusion detection system (IDS) for in-vehicle networks based on deep neural networks (DNN). Unsupervised deep belief networks

(DBN) were used to initialize the DNN parameters as a preprocessing stage. The dataset was created using a packet generator, and anomalies were injected by manipulating packets and adding some Gaussian noise. Another deep learning technique, proposed by Taylor et al. [20], used long short-term memory (LSTM) recurrent neural network (RNN) to detect attacks on the CAN bus. This approach works with raw data, unlike the reduced-data abstraction proposed herein.

4) Sequential Techniques: Given the nature of our data, time-series anomaly detection techniques seem like a natural approach. The Hidden Markov Model (HMM) is a popular and powerful tool for modeling and analyzing time-series data. HMMs have been widely used in different applications, such as for gene prediction [21], protein structure prediction [22], speech recognition, and weather forecasting. A fair amount of work has also been done in time-series anomaly detection in various research areas.

An HMM was used for monitoring patients' health conditions, predicting future clinical episodes, and initiating alerts [23], [24]. An HMM was also applied to host-based anomaly detection. Warrender et al. [25] compared the performance of four methods for modeling normal program behavior and detecting intrusions based on system calls. They used several techniques and showed that the HMM achieved the best accuracy on average. In another paper, Hu et al. [26] suggested an efficient HMM training scheme for system-call-based anomaly intrusion detection.

A previous work, related to our approach, is [27], which proposed an anomaly detection method based on the HMM for the CAN bus raw data. However, this method was only tested against sudden decreases/increases in speed and RPM anomalies.

Berlin et al. [28] introduced the idea of a security information and event management system (SIEM) for connected vehicles located in the backend, but did not specify any concrete implementation. We will discuss the different deployment techniques for our solution, their tradeoffs, and the architecture we used in Section VI.

III. Proposed Detection Technique

Having trained an HMM, it remains to specify how to identify anomalous events based on the model likelihood score. Most commonly, a static threshold is used (either for single observation or for sequences), and scores crossing the threshold are flagged as anomalous. However, a static threshold can be inaccurate in many applications such as temporal data with time and history-sensitive characteristics.

We propose a different approach for identifying anomalous events using an additional regression model, based on work done in [10] on temporal anomaly detection in database accessibility. The training set is divided into two parts: P_1 and P_2. The first part, P_1, will be used to train the HMM, while the second part, P_2, will be used to build a regressor that will predict the log-likelihood for time interval t. After the HMM is generated, we will build the training set for the regression from P_2, as described in Algorithm 1.

Algorithm 1 BuildRegressor

```
Input: HMM model $H$, training set $P_2 = \{(O_1^1, \dots, O_{n_1}^1), \dots, (O_1^k, \dots, O_{n_k}^k)\}$
Output: Regression model $\hat{w}$
1: for $j \leftarrow 1$ to $k$ do
2:     for $i \leftarrow 1$ to $n_j$ do
3:         $LL_{ij} \leftarrow getLogLikelihood(H, (O_1^j, \dots, O_i^j))$
4:         $v_{ij}$ is a temporal feature vector created for the $i$th observation in the $j$th drive (e.g., time since drive start, consecutive arrival times, etc.)
5:     end for
6: end for
7: $\hat{w}$ is built using a least squares optimization problem:
       $$\hat{w} \leftarrow \arg\min_w \sum_{j=1}^{k} \sum_{i=1}^{n_j} (\langle w, v_{ij} \rangle - LL_{ij})^2$$
8: return $\hat{w}$
```

Upon receiving a new event, we compare the event log-likelihood computed from the HMM model to the the regression model predicted log-likelihood. When observing an abnormal value with respect to the regression model $\hat{w}$, the system can issue an alert.

IV. Experiments

A. Datasets

In order to collect a large amount of data and simulate car fleets, we built a small simulation for connected vehicles. The simulator is based on the well-known traffic simulator SUMO (Simulation of Urban MObility) [29] which has been widely used in many different research studies and projects. On top of the simulations generated by SUMO, we integrated a data collector for each vehicle. The data sent to the backend consists of events that are collected and transmitted from each vehicle. As discussed before, the use of events instead of raw data adds a layer of abstraction that helps us filter noise and avoid protocol change issues.

Event Definition: An event is a well-defined occurrence in the vehicle, generated by one of the on-board security systems (in charge of collecting different sensors' data and processing them into events). Each event consists of an event type (each type is associated with a unique ID) and attributes. Event types include login, logout, door open, door close, file access, running app, install app, app update, USB inserted, network usage, etc. Each event contains different attributes. A *file access* event contains attributes such as file type (root, protected, and public), action (read, write, and execute), etc. An *install app* event contains source/destination packets, target IP, etc. The full list of events and attributes is described in Appendix A, which can be found online.

Story Definition: The drive can be described as a sequence of events sent from the vehicle to the backend. We define a *story* as a sequence of events that together describe a scenario. A *drive* is therefore composed of stories.

In order to enrich the simulator and adapt it to connected vehicles, we added the capability to insert stories that describe communication of connected vehicles—new vehicles will login as the drive starts and logout as it ends. Therefore, we added login and logout stories. Connected vehicles will also install and run various applications in the background such as weather, GPS, and music. Thus, we added stories such as *installing application*, *playing music (e.g., from stream, USB, and phone)*, *GPS access*, *open flows (e.g., weather)*, *open ports*, etc. Another fundamental feature of connected vehicles is the ability to download road maps while driving; therefore, a *download map* story was added. Moreover, since connected vehicles will need frequent firmware updates, over-the-air updates (OTA) and USB firmware updates stories were added. Appendix B, which can be found online, contains the full and detailed list of stories.

We simulated a group of vehicles which carried out more than 4,000 drives around the city, where each drive consists of events that occurred in the vehicle and were sent to the backend.

B. Adding Noise

As we aim to test our implementation in as realistic an environment as possible, noise was inserted into the data. We randomly (yet in a logically consistent way) injected network usage, open flows (weather and GPS applications communicating with servers), connected devices (Bluetooth communication, USB successful/unsuccessful insertions), open/closed ports, different file accesses, drive cancellation (the driver entered the vehicle, but exited before he started driving, perhaps because he forgot something outside), playing music during the drive from USB/Bluetooth/Stream, etc.

C. Data Transformation

We tested two different transformation techniques for converting events to suitable HMM training data.

1) EventID Transformation: As a baseline, and in order to examine whether rest of the features improved/reduced the algorithm performance, the model is trained based only on the event IDs.

2) Discrete Transformation: This method takes the different event attributes and transforms the event into a discrete feature vector using configurable "buckets" (users will be able to define the bucket sizes and limits). The main reason for defining an upper limit instead of finding the max value in the dataset is that it is possible for an attribute to reach an extremely high maximum value, while most values are much lower; this could cause the majority of the values to be associated with a small group of buckets, and make the feature redundant.

For these experiments, we used the event ID, velocity, file access options, open flows, vendor type, etc. Velocity was split into five buckets (0–5, 5–10, 10–20, 20–50, 50+). Open flows was split into three buckets (low, medium, high). File accesses were split according to the access type (read, write, and exe) and the file type (public, protected, and root). Vendor type was split into known (white list) and unknown, etc.

D. Attack Scenarios

As described above, each drive could be described as a sequence of events sent from the vehicle to the backend. An unknown sequence of events (such as missing events in a known sequence, unrealistic order of events, or even known sequence of events with attribute values that are not in accordance with normal behavior) could be a sign of suspicious activity. We can now describe several different types of attack scenarios that could be tested in our the model.

- **Out-of-order:** Testing out-of-order events that could indicate abnormal behavior. A simple example is the "car entry" story. Since an ignition event cannot appear before the driver disables the alarm, the following sequence of events is not possible:

  {*car unlock, door open, door close,* ***ignition****, seat belt on,* ***alarm off***}
- **Out-of-context:** A simple example of this is a sudden decrease/increase in speed. A more complex example is an extensive outbound communication with an unknown IP in an unusual context, which could indicate a malicious program communicating with a C&C server, or attacker.
- **USB firmware update attack:** Cars' firmware can be updated using a USB which contains an authentication mechanism (as was done after the Chrysler hacking and recall [2], [3]). Researchers [2] found an exploit in the USB update key exchange mechanism which allows them to take the original USB and insert it, wait for the authentication process to finish, extract the original USB, and replace it with a malicious one. The malicious USB is authenticated and can access the car's systems. The following sequence of events can indicate such behavior:

  {*USB insert, authentication process,* ***USB removed, USB inserted****,* ***running unsigned file from USB****, file access, USB extract*}
- **Communication with unknown vendor:** A fundamental concept of connected vehicles is the ability to download road maps from known vendors while driving. This can be very helpful when the vehicle's sensors which are responsible for monitoring the surroundings fail to do so. Anomalous activity could be communication with an unknown vendor (e.g., malware inside the system trying to disguise communication with the C&C server), followed by an abnormal network communication and/or unknown processes running in the background.
- **OTA malicious updates:** OTA updates will also be available in connected cars. A malicious update caused by attackers could cause the installation of new applications, abnormal network communication, etc.
- **Malicious application installation:** There are a wide variety of ways to trick a user into installing a malicious application. One way is to upload a benign version to the store, and after installation, it updates itself with the malicious content and/or tricks the user into granting it high privileges. This could cause a sudden drift in network usage, file accesses, etc.

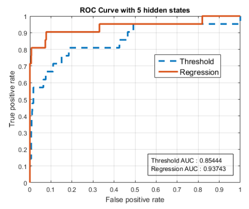


Fig. 1: EventID transformation using 5 hidden states

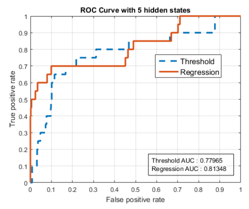


Fig. 2: Discrete transformation using 5 hidden states

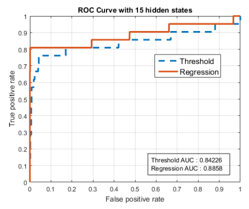


Fig. 3: EventID transformation using 15 hidden states

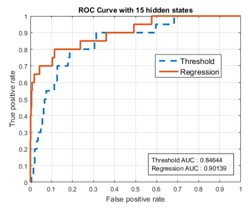


Fig. 4: Discrete transformation using 15 hidden states

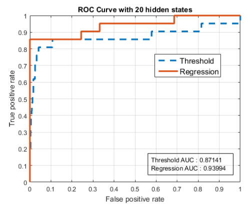


Fig. 5: EventID transformation using 20 hidden states

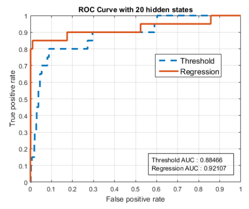


Fig. 6: Discrete transformation using 20 hidden states

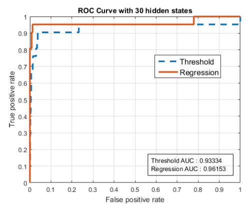


Fig. 7: EventID transformation using 30 hidden states

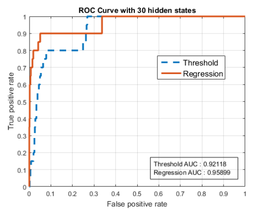


Fig. 8: Discrete transformation using 30 hidden states

TABLE I: AUC results under given transformation technique, detection method, and number of hidden states of the HMM

Transformation	Detection technique	Hidden States Number			
		5	15	20	30
EventID	Threshold	0.85	0.84	0.87	0.93
EventID	**Regression**	**0.94**	**0.89**	**0.94**	**0.96**
Discrete	Threshold	0.78	0.84	0.88	0.92
Discrete	**Regression**	**0.81**	**0.9**	**0.92**	**0.96**

E. HMM Model Generation

We built various HMM models with a different number of hidden states. Then we calculated each model's log-likelihood on a test set with $K = 3$ fold cross validation. After comparing the different models, our algorithm chose the model with the best log-likelihood.

F. Tests

In the following experiments, our focus was on two main goals:

- To test the developed algorithms and system architecture.
- To analyze the dynamic temporal threshold performances.

We tested HMM models with different numbers of hidden states both for the *eventID transformation* and the *discrete transformation*. Models were tested against a test set with 1,000 different benign and anomalous drives. In order to generate anomalous drives, we injected multiple variations of the anomaly types described in IV-D into benign drives.

The experiments were conducted in both offline and online modes: **full drives** (offline) and **drive prefixes** (online).

Offline mode: When testing full drives, the system waits until the driver has stopped the vehicle and logged out, then it pulls the full sequence of events that occurred during the drive (and were stored in the backend) and tests it against the vehicle's HMM and regression models.

Online mode: In drive prefix mode, the system tests each new event as soon as it is sent from the vehicle, and gives an alert when the accumulated prefix of events is identified as anomalous.

To achieve the second goal and evaluate the dynamic temporal threshold performances, we performed same tests on both static thresholds and on our dynamic temporal threshold.

Since the ROC-AUC and F-measure are generally insensitive to the imbalance nature of the data, they were chosen to compare the results between different models with different hidden states, and between the two transformation techniques. ROC curves are presented in Figures 1 to 8 and summarized in Table I. F-measure results are presented in Figures 9 and 10 and summarized in Tables II and III.

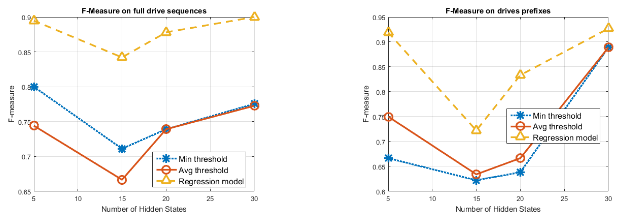
V. Results

Analyzing the results: As can be observed from the graphs, both transformation techniques reached a high AUC of 0.96. The *eventID transformation* reached an F-measure higher than 0.9 both in offline and online modes. The *Discrete transformation* reached an F-Measure of 0.83 in offline mode, and 0.86 in online mode.

Although the F-measure is higher for the *eventID transformation*, the *discrete transformation* advantage lays with its ability to identify anomalies that could not be identified with *eventID transformation* such as: applications accessing files not permitted for them, unusual behavior in different contexts (events that are legitimate in specific speed/flow contexts), etc.

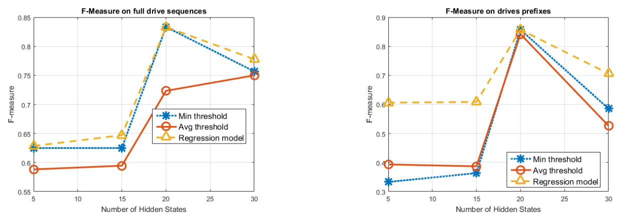
Based on our results, we can achieve True Positive Rate (TPR) higher than 0.9 with False Negative Rate (FNR) smaller than 0.1 and False Positive Rate (FPR) smaller than 0.0009. We can also achieve a TPR higher than 0.85 without any FP.

The best results were achieved indisputably by using the regression model temporal threshold. The regression model was superior to any static threshold in all tests, detection modes, and model measurement methods.



Offline comparison Online comparison

Fig. 9: EventID transformation F-measure results



Offline comparison Online comparison

Fig. 10: Discrete transformation F-measure results

TABLE II: F-measure results with eventID transformation

Offline/Online	Detection technique	Hidden States Number			
		5	15	20	30
Offline	Avg. Threshold	0.74	0.67	0.74	0.77
Offline	Min. Threshold	0.8	0.71	0.74	0.78
Offline	**Regression**	**0.89**	**0.84**	**0.88**	**0.9**
Online	Avg. Threshold	0.75	0.63	0.67	0.88
Online	Min. Threshold	0.67	0.62	0.64	0.88
Online	**Regression**	**0.92**	**0.72**	**0.83**	**0.93**

TABLE III: F-measure results with discrete transformation

Offline/Online	Detection technique	Hidden States Number			
		5	15	20	30
Offline	Avg. Threshold	0.58	0.59	0.72	0.75
Offline	Min. Threshold	0.62	0.62	0.83	0.76
Offline	**Regression**	**0.63**	**0.65**	**0.83**	**0.78**
Online	Avg. Threshold	0.39	0.39	0.84	0.53
Online	Min. Threshold	0.33	0.36	0.86	0.59
Online	**Regression**	**0.61**	**0.61**	**0.86**	**0.71**

Comparing related works results: Our approach uses data abstraction from multiple interfaces (CAN, network, and OS) which is significantly different from all related works that focus on raw data from the CAN bus. Neththerteless, we wished to compare our technique against other related works' anomalies which were relatively simple. The vast majority of these anomalies were sudden decreases/increases in speed. Some also tested out-of-sync messages from different ECUs. Thus, we tested our method against various sudden decreases/increases in speed that caused out-of-context scenarios and against out-of-sync messages that caused out-of-order scenarios (by disturbing the normal message frequency). Our method managed to identify all of the above anomalies without any false positives or false negatives.

VI. Deployment

There are two main approaches for deploying our solution: on-board or on the backend. Those approaches present a trade-off between several aspects including detection latency, detection scale, footprint (resources), and bandwidth consumption.

The on-board architecture will provide real-time anomaly detection due to the fast communication between the data collector and the detection engine. However, this method is resource-intensive on the edge units. Furthermore, it is far more challenging to serve car fleets and detect cross-fleet anomalies using this architecture. On the other hand, in the backend architecture, serving car fleets and detecting cross-fleet anomalies is much easier, but there is a certain latency in the data transmitted from vehicles to the cloud. Moreover, this approach requires enormous bandwidth.

Thus, our solution had to overcome two major challenges:

1) To apply real-time anomaly detection using an HMM, overcoming the low computational power and limited memory resources.
2) To detect anomalies in real-time for a car fleet.

Therefore, we chose to use a hybrid approach, where a rule-based client (the data collector) is integrated into the vehicle and sends events to the backend. However, as vehicles evolve, computational power and memory resources limitations will no longer be an obstacle, and this solution could be integrated fully inside the vehicle.

To address these challenges, we suggest a hybrid platform with an HMM anomaly detection mechanism:

By moving the detection mechanism to the cloud, we can overcome the low computational power and limited memory resources of the vehicle. However, transmitting raw CAN/OS/Network packets to the cloud will require enormous bandwidth and a high transmission rate, perhaps overwhelming the backend. Therefore, we suggest using a light-weight component that could be integrated into the vehicle that can extract important data based on configurable rules, and transmit it back to the backend with all the selected features, which means our data will not be raw packets; rather, it would be at a higher level of abstraction and include only relevant data extracted from applications, network traffic, chosen sensors, CAN bus, etc.

The backend would store HMMs learned for a large number of vehicles. By using a cloud-based platform, we can serve car fleets and possibly apply advanced analytics for identifying correlations between vehicles and identify in-progress attacks on vehicles using anomalies detected from other vehicles. Using this approach, we can either train an HMM for each car and store it in a distributed database, or build a model for groups of cars with common characteristics using clustering-based techniques. Figure 11 describes the system's high-level architecture.

VII. Conclusion

This work has introduced a full working HMM solution for increasing cybersecurity in connected cars with a unique

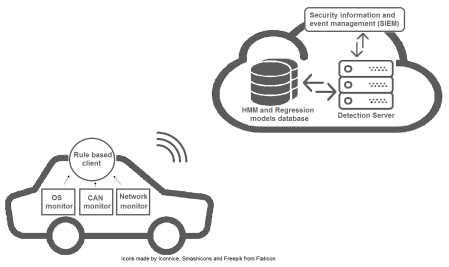

Security information and event management (SIEM)
HMM and Regression models database
Detection Server
Rule based client
OS monitor
CAN monitor
Network monitor
Icons made by Iconnice, Smashicons and Freepik from Flaticon

Fig. 11: High-Level System Architecture

approach handling the vehicle's collected data and a new temporal-based detection technique. For each vehicle in the fleet, important data is collected using a rule-based engine. The collected data is tested (either offline or online) against an HMM model trained on the vehicle's normal behavior. In the search for a smart detection technique, we built a regression model based on temporal features (e.g., time since drive start, consecutive arrival times, etc.), which predicted the expected log-likelihood at time t and compared the result with the actual log-likelihood. Although the system performed very well, even with static thresholds, the temporal detection technique proved its superiority and provided significantly improved results.

Our system monitors not only the CAN bus, but also monitors different vehicle interfaces (OS, Network, and CAN), and was proven to be highly capable of detecting real-life complicated anomalies which involve a wide number of features from different interfaces.

The deployment method we used to implement our solution is capable of monitoring car fleets. Furthermore, it allows us to train HMMs either for individuals or for groups of cars with common characteristics using a clustering preprocessing stage.

This work can serve as a basis for applying advanced analytics for identifying correlations between vehicles, as well as in-progress attacks on vehicles using anomalies detected from other vehicles.

References

[1] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham *et al.*, "Experimental security analysis of a modern automobile," in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010, pp. 447–462.

[2] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, 2015.

[3] A. Greenberg. (2015) Hackers remotely kill a jeep on the highway. [Online]. Available: https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

[4] 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." in *USENIX Security Symposium*. San Francisco, 2011.

[5] R. M. Ishtiaq Roufa, H. Mustafaa, S. O. Travis Taylora, W. Xua, M. Gruteserb, W. Trappeb, and I. Seskarb, "Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study," in *19th USENIX Security Symposium, Washington DC*, 2010, pp. 11–13.

[6] A. Francillon, B. Danev, and S. Capkun, "Relay attacks on passive keyless entry and start systems in modern cars," in *in proceedings of the 18th anual network and distributed system security symposium. the internet society*. Citeseer, 2011.

[7] R. N. Charette, "This car runs on code," *IEEE spectrum*, vol. 46, no. 3, p. 3, 2009.

[8] D. K. Nilsson, U. E. Larson, and E. Jonsson, "Efficient in-vehicle delayed data authentication based on compound message authentication codes," in *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*. IEEE, 2008, pp. 1–5.

[9] F. Hartwich *et al.*, "Can with flexible data-rate," in *Proc. iCC*, 2012, pp. 1–9.

[10] E. Gutflaish, A. Kontorovich, S. Sabato, O. Biller, and O. Sofer, "Temporal anomaly detection: calibrating the surprise," *arXiv preprint arXiv:1705.10085*, 2017.

[11] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network," in *Information Networking (ICOIN), 2016 International Conference on*. IEEE, 2016, pp. 63–68.

[12] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection." in *USENIX Security Symposium*, 2016, pp. 911–927.

[13] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 1110–1115.

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

[15] M. L. Han, J. Lee, A. R. Kang, S. Kang, J. K. Park, and H. K. Kim, "A statistical-based anomaly detection method for connected cars in internet of things environment," in *International Conference on Internet of Vehicles*. Springer, 2015, pp. 89–97.

[16] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[17] A. Theissler, "Anomaly detection in recordings from in-vehicle networks," *BIG DATA AND APPLICATIONS*, p. 23, 2014.

[18] L. Li, R. J. Hansman, R. Palacios, and R. Welsch, "Anomaly detection via a gaussian mixture model for flight operation and safety monitoring," *Transportation Research Part C: Emerging Technologies*, vol. 64, pp. 45–57, 2016.

[19] M.-J. Kang and J.-W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," *PloS one*, vol. 11, no. 6, p. e0155781, 2016.

[20] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 130–139.

[21] M. Stanke and S. Waack, "Gene prediction with a hidden markov model and a new intron submodel," *Bioinformatics*, vol. 19, no. suppl_2, pp. ii215–ii225, 2003.

[22] V. De Fonzo, F. Aluffi-Pentini, and V. Parisi, "Hidden markov models in bioinformatics," *Current Bioinformatics*, vol. 2, no. 1, pp. 49–61, 2007.

[23] P. Jiang, J. Winkley, C. Zhao, R. Munnoch, G. Min, and L. T. Yang, "An intelligent information forwarder for healthcare big data systems with distributed wearable sensors," *IEEE systems journal*, vol. 10, no. 3, pp. 1147–1159, 2016.

[24] A. R. M. Forkan and I. Khalil, "Peace-home: Probabilistic estimation of abnormal clinical events using vital sign correlations for reliable home-based monitoring," *Pervasive and Mobile Computing*, 2017.

[25] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," in *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*. IEEE, 1999, pp. 133–145.

[26] J. Hu, X. Yu, D. Qiu, and H.-H. Chen, "A simple and efficient hidden markov model scheme for host-based anomaly intrusion detection," *IEEE network*, vol. 23, no. 1, pp. 42–47, 2009.

[27] S. N. Narayanan, S. Mittal, and A. Joshi, "Obd_securealert: An anomaly detection system for vehicles," in *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.

[28] O. Berlin, A. Held, M. Matousek, and F. Kargl, "Poster: Anomaly-based misbehaviour detection in connected car backends," in *Vehicular Networking Conference (VNC), 2016 IEEE*. IEEE, 2016, pp. 1–2.

[29] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012.