Frequency-Based Anomaly Detection for the Automotive CAN bus

Adrian Taylor
Defence R&D Canada
adrian.taylor@drdc-rddc.gc.ca

Nathalie Japkowicz
School of Electrical Engineering and Computer Science
University of Ottawa
nat@site.uottawa.ca

Sylvain Leblanc
Electrical and Computer Engineering Department
Royal Military College of Canada
Email: sylvain.leblanc@rmc.ca

***Abstract*—The modern automobile is controlled by networked computers. The security of these networks was historically of little concern, but researchers have in recent years demonstrated their many vulnerabilities to attack. As part of a defence against these attacks, we evaluate an anomaly detector for the automotive controller area network (CAN) bus. The majority of attacks are based on inserting extra packets onto the network. But most normal packets arrive at a strict frequency. This motivates an anomaly detector that compares current and historical packet timing. We present an algorithm that measures inter-packet timing over a sliding window. The average times are compared to historical averages to yield an anomaly signal. We evaluate this approach over a range of insertion frequencies and demonstrate the limits of its effectiveness. We also show how a similar measure of the data contents of packets is not effective for identifying anomalies. Finally we show how a one-class support vector machine can use the same information to detect anomalies with high confidence.**

***Index Terms*—Anomaly detection, controller area network, automotive, monitoring**

I. Introduction

Automobiles have evolved from purely mechanical devices to connected computing platforms. Modern cars are controlled by multiple embedded processors that communicate over a controller area network (CAN) bus. The computers in cars were long assumed to be safely isolated from the outside world, and the security of those systems was not considered important. But it has become clear in recent years that this is no longer the case. Devices on the CAN bus are now connected to the outside world over both wired and wireless channels. These channels have been exploited to hack cars and cause a variety of effects [1]–[3]. We propose anomaly detection as a means to identify such attacks. Over time manufacturers will respond to this threat with improved security, but detection will always be an important component of defence.

Anomaly detection is well-suited to this problem because automotive CAN traffic is predictable. Most CAN packets are published to the bus at a fixed frequency. Furthermore many attacks require high rate packet injections, motivating recommendations to detect such insertions as a first line of defence [1]. Packet frequency monitoring has been proposed by others for attack detection, but not evaluated [4]. Clearly a high rate of packet injects are detectable, but what are the limits of this approach? An anomaly detection system in a car must maintain a very low false alarm rate. For example, consider a system that evaluates traffic every half second. A false alarm rate of 10^{-4} will produce an alert every hour. A driver will quickly learn to ignore the detector, rendering it useless. We need an understanding of the practical limitations of this approach given these stringent requirements.

Our work evaluates the effectiveness of frequency-based anomaly detection for packet injection attacks. We apply a flow-based method adapted from industrial control system traffic [5] to the CAN bus. Flows measure both frequency and average data content changes for each type of packet, and compare them to historical values to produce an anomaly signal. Our evaluation compares this method's effectiveness over a range of packet insertion types, durations, and frequencies, to determine its limits in detecting both overt and subtle attacks. We also evaluate how a one-class support vector machine (OCSVM) anomaly detector performs using the same information.

The remainder of this paper is structured as follows. In Section II we describe normal and attack traffic. In Section III we explain the flow method in more detail. In Section IV we describe the experimental evaluation and results. In Section V we present our conclusions and suggestions for future work.

II. CAN bus and Attacks

A. Normal CAN bus traffic

The CAN bus standard is commonly used for internal communication in automobiles. A car typically has two CAN buses. One is high speed (e.g. 500 kbps) and is dedicated to engine functions. The other may be lower speed (e.g. 125 kbps) and is dedicated to entertainment and convenience features. The two buses are usually connected through a gateway. The gateway allows restricted communication between the two domains, for example so the dashboard on the low speed bus can display engine alerts sent from the high speed bus. The buses are populated by electronic control units (ECUs). These ECUs broadcast packets that make up the traffic on the bus.

Packets contain an ID field and data bytes, as well as additional bits for error correction, control, and low-level bus logic. Bus arbitration is handled with a simple system: the lower ID has priority. If two ECUs attempt to transmit at the same time, the one sending a higher ID packet will detect the

978-1-908320-58/2/$31.00 ©2015 IEEE

lower ID packet and wait until the bus is free before sending its message. Here we assume all messages are well-formed and are thus only concerned with the ID an data fields. The time sequence of these ID and data field pairs is the raw data we are analyzing to identify anomalies. We do not have access to the data dictionary used by the manufacturer of our test vehicle, and so assume no knowledge about the meaning of each packet.

We analyzed bus traffic from our test vehicle, a late-model SUV. Each ID was observed to appear on the bus at a fixed frequency. Only minor deviations in each ID's period were observed over about half an hour of observations.

Aside from driving the vehicle, no actions (opening windows, turning lights on/off, etc.) were taken. It is possible that such actions would cause additional non-periodic traffic to appear on the bus, but we restrict the analysis here to only periodic traffic. The methods presented here would need to be re-evaluated for buses with non-periodic traffic.

B. Attack traffic

A variety of attacks have been described in the literature. Miller and Valasek are particularly explicit in the descriptions of their methods [1], which include reflashing ECUs, sending extra control packets, and using diagnostic packets to achieve various ends. Our detector evaluates CAN packets and does not directly measure physical layer characteristics of the bus, or other hardware in the vehicle. We consider only how an attack would manifest on the CAN bus while the car is driving. This excludes for example attacks on the wireless interfaces of the car [3]. However secondary attacks that leveraged such access would appear on the bus and be detectable. As viewed from the bus, all published attacks produce some change in packet traffic: unexpected packets appear, expected packets do not appear, and/or the packet data contents are unusual. We discuss each of these cases with the goals of identifying which effects are in scope for this work, and how we will simulate them.

Unexpected packets come in two varieties: diagnostic packets, or additional (or missing) instances of the normal periodic IDs. Diagnostic packet insertions account for a large number of attacks, but in principle they are easy to detect because they should not occur under normal driving conditions. We assume a practical detector would alert on such packets and could be disabled when the car is legitimately being serviced. Other attacks are of the first variety: additional normal-looking packets inserted onto the CAN bus at a high rate. The high rate is necessary because the malicious packets must compete with normal messages for attention by their intended recipient. Missing packets are more rare; we have seen only one example of such an attack [6]. In this attack, a rogue controller on the bus silences its target with clever use of CAN arbitration rules. Missing packets could thus be evidence of an attack designed to silence an ECU so that it can be replaced by a counterfeit one.

Most attacks also require specific data contents to be effective. It is possible that an attack could consist of packets arriving at the expected time but containing data with malicious intent, e.g. sent by a compromised device. Detecting anomalies in the data contents of packets is an important topic but our focus is on attacks that affect packet frequency. We do not address pure data field changes here.

Thus the scope of this investigation is limited to the insertion of unexpected normal packets, and the erasure of normal packets. High-rate packet insertions correspond to the majority of published attacks that are not trivially detectable.

We inserted packets into and deleted packets from data captures from a CAN bus as described in Section IV-B. Our traffic modifications are not designed to elicit any specific effects; rather, we are interested solely how sensitive our detector is to detecting packet insertions and deletions. We could extrapolate from our results to predict the detectability of any attack that inserts or removes normal packets on the CAN bus.

III. Flow-based anomaly detection

Our anomaly detector works by calculating statistics about ongoing network traffic and comparing them with historical values. The collections of statistics are called flows. In network traffic analysis, flows contain statistics about a complete communication between two endpoints. For example a flow might contain the number and frequency of packets, the quantity of data exchanged, and the duration of the connection. However the flow concept must be modified for the CAN bus, because traffic all traffic is broadcast, the endpoints are unknown, and packets are sent constantly so there is no end of the transaction.

We modify the concept by adopting of the approach of Valdes and Cheung, who evaluated flows for anomaly detection in industrial control system traffic [5]. Each CAN flow collects the following features:

- ID: The packet ID
- N_p: the number of packets in the flow
- μ_h: the average Hamming distance between successive packet data fields
- σ_h^2: the variance of the Hamming distance between successive packet data fields
- μ_t: the average time difference between successive packets
- σ_t^2: the variance of the time difference between successive packets
- T_t: the time difference test value
- T_h: the Hamming distance test value
- T: the combined time difference and Hamming test values

The test values are t-tests that compare the time and Hamming difference means to their historical values:

$$T_x = \frac{\mu_x - \mu_{Hx}}{\sqrt{\frac{\sigma_x^2}{n}}} \tag{1}$$

where μ_x and σ_x^2 are replaced with the Hamming or time difference mean and variance, and μ_{Hx} is the historical mean for the relevant variable. In our experiments we compare the effectiveness of the individual and combined test values. The

historical means are calculated over the data captures reserved for training.

We initially chose a one-second window to calculate the flows. This is a reasonable time scale to alert on anomalous traffic of interest to the driver. However some IDs arrive at a rate close to or equal to once per second. The statistics of these slow flows calculated over a second are unusable. For simplicity we exclude these from the trials and include only those with a period equal to or below 50ms. These fast IDs account for over 90% of the traffic on the bus. Also for simplicity we focus solely on the fast CAN bus for these experiments. A practical system would need to account for both time scales and buses, e.g. by dividing flow analysis into message frequency classes, or analyzing each flow individually at its own time scale.

The window advances in half-second increments, generating test statistics twice per second. Each window is assigned a score equal to the highest test value over all the flows. A sequence of window scores are combined to produce a single anomaly score by taking the log sum, as recommended in [7]:

$$A(S_q) = \frac{1}{|\mathbf{T}|} \sum_{i=1}^{|\mathbf{T}|} \log T_i \tag{2}$$

where $\mathbf{T} = T_1, T_2 \ldots T_{|\mathbf{T}|}$ is the vector of scores, and A is the anomaly score for the sequence S_q.

A. One-class support vector machine

To compare with the simple approach above, we trained a one-class support vector machine (OCSVM) to classify flows. The OCSVM learns the distribution of a single class from training data [8]. It can then be used to classify new data as in or out of the training class. We use same flows above as a basis for feature vectors for the OCSVM. The test statistics and IDs are omitted, and the flows are concatenated over IDs to produce a single feature vector for each window. Thus the feature vector for each window contains the count, mean and variance of time differences, and mean and variance of Hamming differences for each ID. We used the scikit-learn Python library [9] OCSVM implementation for our experiments.

IV. Experiment

A. Data

We collected CAN bus data from a 2011 Ford Explorer. Traffic was captured simultaneously from the two CAN buses in the vehicle. During the captures the vehicle was was started, driven at low speed for about five minutes, and then stopped. No user controls (e.g. turning on the lights, opening windows) were activated in any trip. This trip was repeated five times. We used data from the first three captures for calculating historical statistics and analysis. For the OCSVM experiments, the first three captures were used to generate training data. Segments from the fourth and fifth captures were used to generate simulated attack data.

B. Simulating attack traffic

We simulate attack traffic by modifying captured data. New packets are inserted into the capture record, starting at a random time, for a given duration and insertion frequency. If the bus is not free at those times, the new packet is queued for insertion at the next time the bus would have been free. The existing packets timestamps are then modified according to three rules:

1) If an inserted packet would have caused an existing packet to be delayed: the existing packet time stamp is advanced to when the bus is again available.
2) If multiple packets are delayed, they are put in a queue. When the simulated bus is free, packets in the queue are put on the bus according to normal priority rules.
3) If a packet is added to the queue, and an older packet with the same ID is present, the older one is discarded.

The data bytes of the inserted packets are copied from an earlier instance of that packet ID. Erased packets are simply removed from the capture with no other modifications. In these experiments, we did not modify the data contents of existing packets.

C. Experiment procedure

The goal of our experiments is to determine the sensitivity of our detector to a range of packet insertion rates and durations. The test data was divided into three-second-long segments. About half the segments were randomly selected to be unaltered, and so constitute normal traffic. The other half were modified to include additional packets or to remove packets. For each modified segment, a single packet ID was selected for insertion or erasure according to the rules described above. All IDs were selected the same number of times.

Every experiment tested a different insert duration and rate. The rate is not absolute, but rather is a multiple of the insertion ID's average packet frequency. Thus for an experiment with a given insertion rate, the absolute insertion frequencies vary over the IDs. We generated simulated insertions with durations ranging from 100ms to 1s. For each duration, packets were inserted at 1x, 5x, and 10x their average rate. The raw test data was reused for each combination of insertion parameters. Reusing the raw data is not ideal, but we had insufficient data to perform completely independent experiments. In practice the data varied very little over each capture, so we believe the effect of reusing data for simulation made little difference in the results.

We evaluate performance using Receiver Operating Characteristic (ROC) graphs and the Area Under Curve (AUC) measure. ROC curves provide a way to compare different methods that is independent of the decision threshold. A ROC plots a methods false positive rate against its true positive rate for a range of decision thresholds. The ideal discriminator is at the top left of the graph, with 100% detection and zero false-positives. The diagonal of the graph represents random guesses and is the worst possible case. The area under curve (AUC) is simply the area under a ROC curve. The AUC condenses the

curve to a single number, which makes it easier to compare a large number of results. However the AUC can obscure important differences at particular false positive rates. As was discussed in the introduction, performance at the lowest false positive rates may be all that matters for a practical application.

D. Results

The first result was that the Hamming distance test statistics was counterproductive in detecting anomalies. The test statistic combining timing and Hamming measures had no discriminative power between normal and inserted traffic. Consequently we only evaluated results using the time test statistic for the flow detector.

ROC curves for for the time statistic detector are shown in Fig. 1 for a range of test cases. Only a cross section of cases are shown; Table I shows the AUC for all results. We see a steady improvement in discrimination as the number of insertions increases. However only in the longest insertion case (1s) are the detection rates practically useful. The erasure case is notably more difficult to detect with this method.

Fig. 2 shows the OCSVM detector results for a selection of test cases, and Table II gives the AUC for all test cases. The OCSVM was able to perfectly detect all test cases where the duration of the alteration was 0.5s or higher. The performance on the remaining cases is also clearly superior to the test statistic.

The superior performance of the OCSVM merits further investigation. Was the OCSVM able to make use of the additional information in the flows, such as the packet count? Or did it simply do a better job of classifying based on the timing information? To answer this question we ran the experiment with every possible combination of features. The top five feature combinations in a representative experiment (0.2s insert duration, 10x insertion rate) were:

1) μ_t
2) μ_t, N_p
3) μ_t, σ_t^2
4) μ_t, N_p, σ_h^2
5) μ_t, σ_h^2.

We note that the mean time is included in the top five, and is by itself the best feature in this experiment. These results suggest that the mean time between packets provides all the discriminatory power being used by the OCSVM. At the other extreme, the Hamming measures by themselves perform little better than a random guess. Surprisingly, the packet count feature was equally weak. We conclude that the OCSVM is making better use of the timing information, and not taking advantage of additional data such as the interpacket timing variance or packet count.

Finally, to challenge the OCSVM further we ran the trials again with shorter flow windows. Using just the mean and variance of the interpacket timing as features, the experiment was repeated with flow windows of 0.5 and 0.2s durations. Surprisingly, performance actually improved as the window size went down. For a 0.2s window, insertions of just three packets were reliably detected. We re-ran the experiments using the t-test detector with shorter windows as well, but found that this reduced performance significantly.

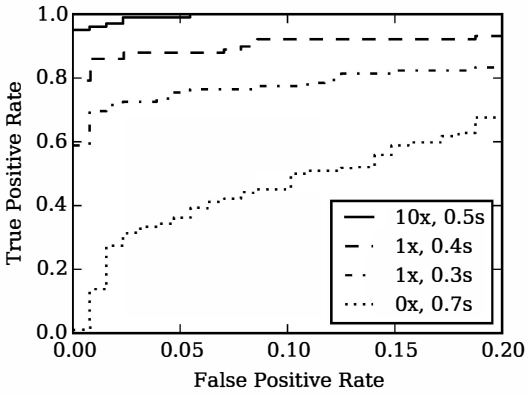

Figure. 1. ROC curves for the detector using the amalgamated max time test statistic. The curves show a cross-section of results from Table I.

TABLE I
FREQUENCY DETECTOR WITH MAX TIME TEST AUC RESULTS. 0X RATE DENOTES ERASURE.

	Rate			
Duration (s)	0x	1x	5x	10x
0.3	0.7314	0.9072	0.9332	0.9564
0.4	0.7600	0.9669	0.9768	0.9636
0.5	0.7883	0.9825	0.9979	0.9988
0.7	0.8100	0.9890	0.9991	0.9998
1.0	0.8720	0.9999	1.0000	1.0000

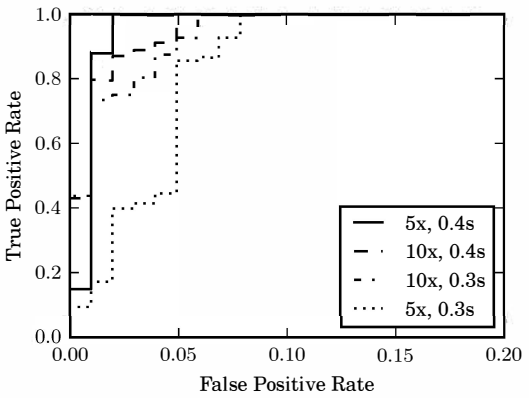

Figure. 2. ROC curves for the OCSVM detector. The curves show a cross-section of results from Table II.

TABLE II
OCSVM DETECTOR AUC RESULTS. 0X RATE DENOTES ERASURE. CASES WITH AUC OF 1 ARE NOT SHOWN.

	Rate			
Duration (s)	0x	1x	5x	10x
0.3	0.9620	0.9715	0.9625	0.9856
0.4	0.9874	0.9900	0.9905	0.9893

V. Conclusion

Our key findings in the use of flows for anomaly detection are as follows. First, for this kind of traffic insertion, the Hamming distance of data packets was an unreliable measure of normality. Only the interpacket timing statistic was reliable for detecting inserted packets. However a simple t-test using this statistic did not detect anomalies at practical false alarm rates. An OCSVM using the same information was able to detect very short packet insertions with acceptable false alarm rates. However, to show we can achieve the extremely low false alarm rates discussed in the introduction, we would need to repeat these experiments with much more data.

What constitutes a realistic threat is not addressed here. Some classes of attacks, such as those in the data fields of packets, would not be detectable with these methods. However we note that most of the attacks revealed in the literature so far would be perfectly detectable.

Finally our data did not include non-periodic packet types. If a normal packet is not periodic then detection of extra insertions could be more challenging. The flow methods described here may be sufficient, but their performance would depend on the nature of the non-periodic packet timing. A complete solution must be capable of dealing with all these cases. Only with more data gathered from more vehicles can these challenges be addressed.

References

[1] C. Miller and C. Valasek, "Adventures in Automotive Networks and Control Units," IOActive Labs Research, Tech. Rep., Aug. 2013. [Online]. Available: http://blog.ioactive.com/2013/08/car-hacking-content.html

[2] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental Security Analysis of a Modern Automobile," in *2010 IEEE Symposium on Security and Privacy (SP)*, 2010, pp. 447–462.

[3] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the 20th USENIX conference on Security*, ser. SEC'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 6–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028067.2028073

[4] M. Muter, A. Groll, and F. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *2010 Sixth International Conference on Information Assurance and Security (IAS)*, 2010, pp. 92–98.

[5] A. Valdes and S. Cheung, "Communication pattern anomaly detection in process control systems," in *IEEE Conference on Technologies for Homeland Security, 2009. HST '09*, 2009, pp. 22–29.

[6] A. G. Illera and J. V. Vidal, "Dude WTF In My Can," presented at Black Hat Asia 2014, Singapore, Mar. 2014. [Online]. Available: https://youtu.be/Y1YmJ0ZYMic

[7] V. Chandola, V. Mithal, and V. Kumar, "Comparative Evaluation of Anomaly Detection Techniques for Sequence Data," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ser. ICDM '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 743–748. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2008.151

[8] B. Schlkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the Support of a High-Dimensional Distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001. [Online]. Available: http://dx.doi.org/10.1162/089976601750264965

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and d. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, p. 28252830, Oct. 2011. [Online]. Available: http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html