Preventing Frame Fingerprinting in Controller Area Network Through Traffic Mutation

Alessio Buscemi*, Ion Turcanu†, German Castignani* and Thomas Engel*
*Faculty of Science, Technology and Medicine (FSTM), University of Luxembourg
†Luxemburg Institute of Science and Technology (LIST), Luxembourg
{alessio.buscemi,thomas.engel}@uni.lu
ion.turcanu@list.lu german.castignani@ext.uni.lu

***Abstract*— The continuous increase of connectivity in commercial vehicles is leading to a higher number of remote access points to the Controller Area Network (CAN) – the most popular in-vehicle network system. This factor, coupled with the absence of encryption in the communication protocol, poses serious threats to the security of the CAN bus. Recently, it has been demonstrated that CAN data can be reverse engineered via frame fingerprinting, i.e., identification of frames based on statistical traffic analysis. Such a methodology allows fully remote decoding of in-vehicle data and paves the way for remote pre-compiled vehicle-agnostic attacks. In this work, we propose a first solution against CAN frame fingerprinting based on mutating the traffic without applying modifications to the CAN protocol. The results show that the proposed methodology halves the accuracy of CAN frame fingerprinting.**

***Index Terms*—Connected Vehicles Security, CAN Bus Reverse Engineering, Fingerprinting, Morphing, Machine Learning**

I. Introduction

Controller Area Network (CAN) is a peer-to-peer message-based network, which is considered the world standard for in-vehicle communication. Despite its popularity, the protocol is not equipped with data encryption nor secure authentication for the Electronic Control Units (ECUs) attached to it [1]. The lack of security was not initially an issue due to the limitation for potential attackers to access the CAN bus physically. However, the recent rapid digitalization of the car sector and the introduction of remote services has consistently augmented the number of wireless access points in the vehicles. In recent years, a number of remote attacks against the CAN bus have been successfully demonstrated [2]–[4].

In spite of the lack of traffic encryption, the interpretation of the data sent and received on the CAN bus by the ECUs is difficult. Every Original Equipment Manufacturer (OEM) encodes the CAN data according to its proprietary specifications and hides the format from the general public. In fact, even if an attacker would gain access to a vehicle's ECU (e.g., by hacking into it), they would still need to decode the format of the CAN frames sent on the bus in order to successfully perform targeted attacks. This drove a number of companies and researchers to specialize in performing manual [5] and semi-automated reverse engineering on it [6]–[9]. Both approaches typically involve a human operator trained to perform specific actions in the vehicle at data collection time.

In one of our recent studies [10], we propose an approach for fully automated CAN bus engineering, which, differently from related work, is context-agnostic, i.e., it does not require any ground truth about the data collection scenario. This solution is based on matching the frames of the target vehicle with the frames of a set of vehicles whose CAN specifications are already known (e.g. through manual reverse engineering). This is possible due to the fact that the ID and content of a frame sent by the same ECU mounted on different vehicles is the same. Such a solution, despite being an optimal tool for researchers and companies working with CAN data, raises further concerns about CAN bus security. Assuming wireless access to the CAN bus (e.g., via a compromised ECU), such reverse engineering can be carried out remotely and without prior knowledge regarding the target vehicle model. This possibility can save a significant amount of time and effort to potential attackers and lead to remote pre-compiled vehicle-agnostic attacking scenarios.

To address these concerns, in a subsequent work [11] we consider the scenario in which a new set of frame IDs is attributed to each newly released vehicle model. In other words, the IDs of frames transporting the same information and sent by the same ECU models present in any two different vehicle models are distinct. This simple countermeasure meets the needs of manufacturers, who are reluctant about adding encryption to the CAN protocol, as it would imply enormous changes in the supply chains. We investigate whether this solution is sufficient to effectively prevent frame matching-based CAN bus reverse engineering. The results show that Machine Learning (ML) classifiers trained to *fingerprint* frames according to the patterns that emerge from their time series can still identify the frame in a target vehicle with an accuracy up to almost 70 %.

In this work, we study whether it is possible to make CAN frames less identifiable through traffic mutations, i.e., altering the traffic in such a way that the resultant distribution of values over certain properties of the frames becomes more similar among different frames. The goal of such operations on the CAN traffic is to reduce the recognizability of frames, thus making the ID randomization an effective defense against frame matching-based fingerprinting. The contribution of this work can be summarized as follows:

- We present a methodology to prevent the CAN frames

IEEE International Conference on Communications (ICC 2022)

fingerprinting based on statistical traffic patterns. In this scope, to the best of our knowledge, we are the first to adapt well known concepts, such as *padding* and *morphing*, to the CAN traffic.
- We conduct an extensive evaluation of the presented methodology and discuss the implications that it has on the data transmission performance of the CAN bus.

II. Background and Related Work

CAN bus is a master-less network in which the communication among nodes (ECUs), is based on messages, also called *frames*. The frames are composed of several fields, the most relevant for this work being the ID, Data Length Code (DLC), and payload. The ID uniquely identifies a frame and its priority. The DLC reports the length of the frame payload. In the CAN standard version, the payload is between 0–8 Byte long and typically contains one or multiple signals, which encapsulate the actual vehicle functions. An ECU can send multiple frames with different IDs, but frames with the same ID cannot be sent by different ECUs. All frames associated with the same ID contain the same signals in the same position. We refer to the sequence of frames associated to the same ID as *frame series*. The frames in a frame series are sent periodically according to the frequency set by the source ECU.

The goal of CAN bus reverse engineering is to identify the boundaries of signals within a frame payload and decode their semantic meaning and format. It typically involves human operators activating sensors to trigger events in the vehicle during data collection, injection of diagnostic messages through the On-Board Diagnostics (OBD-II) port and finding correlations between the CAN traffic and the ground truth offered by external sensors [5]–[7], [9]. Recently, it was demonstrated that fully automated reverse engineering can be achieved by finding a correspondence between the frames of a target vehicle and the ground truth of previously reverse engineered vehicles. The match can be found based on the ID of frames (if it has been preserved in the target vehicle [10]), or through ML fingerprinting (if the ID has been changed/anonymized [11]).

In this work, we study a defense against fully automated reverse engineering based on traffic mutation techniques. The goal of traffic mutation algorithms is to conceal the patterns that emerge from traffic features, which ML models can learn and exploit to recognize sensitive information. The two main techniques adopted in the scope of fingerprinting prevention are padding and morphing. Padding consists in augmenting the length of packets in a communication stream to a predefined target size. It was introduced in [12] in response to websites fingerprinting attacks over encrypted traffic based on the size of transmitted packets. The goal of morphing, instead, is to make a set of source processes that need to be protected from fingerprinting resemble another target process, hindering the predictive capability of ML classifiers. In its original implementation [13], morphing matrices are generated offline with convex optimization methods to define how the packets of the source processes should be padded or split.

Traffic mutation approaches have been used as a defense in a multitude of fingerprinting attack scenarios, such as web encrypted traffic [12]–[14], mobile devices and apps [15], VoIP data [16], and Internet of Things (IoT) devices [17]. Among those, it is worth mentioning the adoption by The Onion Router (Tor) of traffic padding between the client and network entry guard as a countermeasure against website fingerprinting [14].

The specifications of the CAN protocol, as well as the constraints that it is subject to, make the data transiting within vehicles differ consistently from the traffic on which mutations techniques have been designed in literature. Namely, the CAN traffic is not encrypted and frames series bring sequential information. In addition, the frame length and sending frequency in a frame series are pre-defined and do not change over time. As a consequence, existing traffic mutation approaches cannot be directly applied to the CAN traffic. In this paper, we present novel padding and morphing techniques that fit the unique characteristics of the CAN bus.

III. Methodology

The frame fingerprinting approach presented in [11] revolves around four distinctive aspects of the CAN traffic: the payload length, the dynamic behavior of the payload bits, the sending frequency, and the frame priority. The approach in [11] makes the following assumption regarding the frame priority: the attribution of new sets of frame IDs in newly released vehicle models should preserve their priority. This leaves no space for manipulation against frame fingerprinting. Therefore, in this work, we operate on the three remaining aspects: (i) padding of the frame payload length, (ii) morphing of the frame sending frequency, and (iii) morphing of the dynamic behavior of the payload bits.

A. Payload Length Padding

The frame length, expressed by the DLC field, is a feature that can be used to fingerprint frames whose payload is shorter than 8 Byte – the maximum frame length. Hereafter, we refer to this set of frames as *short* and to those whose length is 8 Byte as *long*. In this work, we make the following assumptions:

- The length of the payload can only be increased, as a reduction would cause a loss of information.
- The extra padding bits should be appended at the end of the frame to preserve the original location of signals within the payload. This allows the receiving ECU to correctly interpret the actual signals contained in the payload and discard the rest of the bits.

Based on the assumptions, we study two solutions:

(a) All payloads are padded to the maximum length of 8 Byte. This is the most straightforward solution, but which also adds the most overhead.

(b) Short payloads, whose length is inferior to a certain threshold τ are padded to τ, while those whose length is superior than τ are padded to 8 Byte length, where $\tau < 8\,\mathrm{Byte}$. The parameter τ should be chosen to be an optimal trade-off between reduction of fingerprinting accuracy and added overhead.

Since each vehicle has a different distribution of long and short frames – as well as different distribution of payload length among the short frames – we argue that an optimal universal threshold τ cannot be identified. On the contrary, it is reasonable to define τ for each vehicle based on the distribution of the frames lengths. For this purpose, the algorithm calculates the quartiles Q_i of payload lengths of all frame series and assigns τ based on the value of a chosen quartile Q_i.

To designate the content of the padded bits, multiple approaches can be followed:

1) Set all bits to a constant value, i.e. always 0 or 1;
2) Define the status of each bit randomly, for each frame;
3) Adopt a particular heuristic to set the bit values.

We argue that approach (1) allows to easily identify how many bits have been padded for each frame, thus nullifying the benefits of the padding. As a consequence, in this work we evaluate approaches (2) and (3). In particular, regarding (3), we follow the algorithm presented in Section III-C.

B. Sending Period Morphing

Frames having the same ID and that carry periodic signals are typically being sent according to a pre-defined period. However, due to hardware imprecision of clocks embedded within ECUs, the frames deviate from the target sending frequency. On the one hand, this deviation (or *offset*) from the defined sending frequency can be used as an intrusion detection mechanism [18]. On the other hand, the frame sending period and the corresponding offset can be exploited as a feature for frame fingerprinting, as demonstrated in [11].

In this work, we make the following assumptions:

1) The sending period of the frames can only be reduced (i.e. the sending frequency is increased). In fact, incrementing the sending period of frames carrying critical information about the vehicle status would cause ECUs reacting less promptly, thus threatening the vehicle's safety.
2) The car manufacturer can synchronize all ECUs according to the new set of sending frequencies of all frames.

Based on these assumptions, we design Algorithm 1 to morph the sending frequency of all frames. The algorithm takes in input a reference CAN log R and an integer P. R is a sample of the original traffic of the vehicle with no mutation applied. P defines the number of target sending periods to which the frames series will be morphed. If we assume P_o is the number of discrete sending periods extracted from the original trace R, then we must have $P < P_o$.

Algorithm 1 Send Frequency Morphing

Input: Ref. Trace R, # of Target Sending Periods P
Output: Send Period-Morphed Frames

1: *periods* ← get_periods(R)
2: *offsets* ← extract_offsets(R, *periods*)
3: *target_periods* ← quantiles(*periods*, P)
4: *target_offsets* ← assign(R.*frame_ids*, *target_periods*)
5: *target_K* ← compute_K(*new_periods*, *target_offsets*)
6: **for** *frame* **in** *CAN_traffic* **do**
7: *morph_frame* ← morph(*frame*, *target_offsets*, *target_K*)
8: **end for**

R allows OEMs to preliminarily collect sending periods and calculate the mean standard deviation of the sending periods of all frames (lines 1-2). Then, the set of sending periods of R is ordered and divided in P quantiles. The ECUs are set to send the frames according to the target periods, which are computed based on these quantiles (line 3). In particular, each frame ID must be associated with a target period such that (i) the target period is inferior to its original period, and (ii) the difference between the target period and the original period is minimal in order to reduce the traffic overhead.

But since we know that each ECU introduces an additional distinct offset that can help fingerprinting its frames, an additional defense mechanism is needed. Specifically, for each target period the highest standard deviation among all frame series associated to that period is chosen as a target mean offset (line 4). Let σ_c be the standard deviation of a frame series F, π its target period, and σ_t its target standard deviation. To morph F to have overall sending frequency standard deviation equal to σ_t while keeping its sending period $\sim \pi$, the ECU must aim at sending each frame every $\pi \pm r$, where r is randomly generated following a uniform distribution between $[-K, K]$. K is calculated (line 5) as follows:

$$K = \sqrt{3(\sigma_c^2 - \sigma_t^2)} \tag{1}$$

C. Bit Flip Rate Morphing

Given a frame series F, the Bit Flip Count (BFC) of a payload bit b indicates how many times b flips (changes its status from 0 to 1 or vice versa) throughout F. The Bit Flip Rate (BFR) is then calculated as $BFC/(f-1)$, where f is the length of F [6]. The work in [11] demonstrates that the number of bits that flip at least once during the trace and the mean BFR constitute valuable features to fingerprint the frames. In the current work, we perform morphing on the payload bits to hide the patterns associated to the BFR.

Every bit within a signal is relevant to the interpretation to information contained in the signal. On the contrary, *unused bits* – as defined in [6], [7] – are bits that do not belong to any signal and never flip. Since we do not want to alter the actual information carried by the payload, the unused bits are the only ones on which we can operate. In this work, we make unused bits flip in such a way that it makes the overall dynamicity of the frames series to resemble among each other. To be noted that, if payload padding is applied too, the added bits are considered unused and, therefore, can be employed for the morphing.

The pseudocode related to the morphing of BFR is presented in Algorithm 2. The algorithm gets in input a reference trace R, the list of signals S, and an integer B. Similarly to the sending frequency morphing (see Section III-B), R is a sample of the original traffic of the vehicle. S is the list of signals that can be found in R, as reported in the specifications owned

by the OEM. B defines the number of target mean BFRs to which the frames series will be morphed.

The frames in a series can only be mutated in such a way that the overall mean BFR is higher than the original one. In fact, trying to decrease the BFR would necessarily imply a manipulation of the bits of the signals, thus causing a loss of information. As a consequence, R should be dynamic enough to ensure that the original mean BFR of the frame series in the CAN traffic can be morphed to the B targets.

The mean BFR of all frames series are initially calculated on R (line 1). The set of all mean BFRs of R is ordered and divided in B quantiles. The target mean BFRs are then extracted according to the quantiles (line 2). Subsequently, for each frame series, the unused bits are extracted based on the ground truth (line 3). Finally, the BFR of each frame series in the CAN traffic can be then morphed to its target (lines 5-11).

To morph a frame series F to have a mean BFR equal to a target mean BFR T, a number of unused bits n must flip for each frame in F. Prior to sending a frame F_i, the ECU defines n by taking into consideration the BFR achieved until F_{i-1} and the information encapsulated in F_i (line 6). Once n is calculated, a subset S of the unused bits in the frame is selected, such that $|S| \leq n$ (line 7). All the bits indicated by S are then flipped and the so altered frame is sent (line 8).

IV. Performance Evaluation

In this section, we evaluate the efficacy that each of the techniques presented in Section III have against frame fingerprinting singularly and combined. To validate our approach, we employ a set composed of 10 s CAN logs from 427 different vehicle models produced by 28 distinct automotive makers. The traces have been collected with a `PCAN-USB FD` in a static context, i.e. parked vehicles, with no action performed by a human operator. Due to the data collection conditions, some signals are never triggered, thus not making any of their bits to flip. We refer to the frames series whose payload bits never flip as *inactive*. Since the presence of inactive frames in our dataset impacts the fingerprinting performance evaluation – namely, the classifier cannot exploit the features based on the payload dynamicity – we present the results achieved on active and inactive frames separately.

Algorithm 2 BFR morphing

Input: Ref. Trace R, Signals S, # of Target Mean BFR B
Output: BFR-Morphed Frames

1: *BFRs* ← get_mean_bfr(R)
2: *target_BFRs* ← quantiles(*BFRs*, B)
3: *unused_bits* ← extract_unused_bits(R, S)
4: **for** *frame* **in** *CAN_traffic* **do**
5: *curr_frame_series* ← *CAN_traffic*[*frame.id*]
6: *curr_BFR* ← bfr(*curr_frame_series*)
7: *n* ← n_bits_to_flip(*frame*, *curr_BFR*, *target_BFRs*)
8: *S* ← bits_to_flip(*n*, *unused_bits*)
9: *morph_frame* ← morph(*frame*, S)
10: **end for**

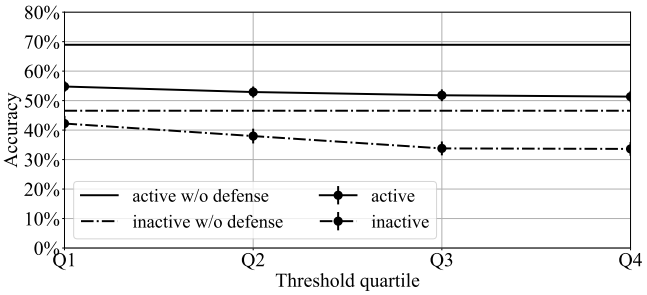

Figure 1. Comparison between the fingerprinting accuracy on the original traffic and *payload-padded* traffic for different quartile values chosen to set τ.

The traces contain a total of 33 034 frame IDs. However, ground truth information is not available for all of them. In fact, the ground truth has been extracted through manual reverse engineering and it is partial, i.e. it does not contain all the signals that can be actually found in the vehicles. For the validation, we follow a *leave-one-out-cross-validation* approach, i.e. the classifier is iteratively trained on all vehicles in the test set except one, which is used as a target for the fingerprinting. In addition, for all the presented results we report the confidence interval at 95 %.

To assess the fingerprinting performance, we choose the *accuracy* metric, which corresponds to the percentage of correct predictions over the total number of test samples. It is to be noted that our fingerprinting problem is under Open Set Recognition (OSR) assumptions, since not all the frames that are found in a target vehicle can be associated to a known class by the classifier [19]. This reflects the possibility for an attacker to target a vehicle mounting ECUs not yet encountered (i.e., sending unknown frames). In this paper, the accuracy refers solely to the frames that have been matched with a known label/class.

To fingerprint the frames, we feed a Random Forest (RF) classifier with a depth of 200 with the same set of features described in [11]. This classifier was proven to achieve the highest accuracy in the scope of frame fingerprinting among a set of selected classifiers, such as Extreme Value Machine (EVM), PI-Support Vector Machine (SVM), and Fully Connected Neural Network (FCNN) [11]. Given the presence of unknown samples, the classifier is set with a rejection threshold $= 0.2$, i.e., all predictions with a confidence score inferior to 20 % are discarded.

A. Evaluation of payload length padding

We evaluate the impact that padding the payload of the frames has on the fingerprinting accuracy of the RF classifier. For this analysis, we set the algorithm to choose the content of the padding bits randomly. Figure 1 compares the mean accuracy obtained on the original traces (without defense) with the results achieved by applying payload padding on all the tested vehicles (with our proposed defense mechanism in place). The figure shows how the performance varies based on the choice of the quartile used to the determine the threshold τ (see Section III-A). It is to be noted that $Q4$ represents the case in which all frames are padded to 8 Byte (solution (a),

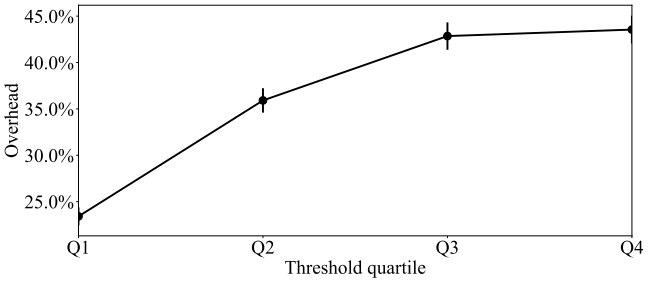

Figure 2. Communication overhead introduced by *payload-padded* traffic for different quartile values chosen to set τ.

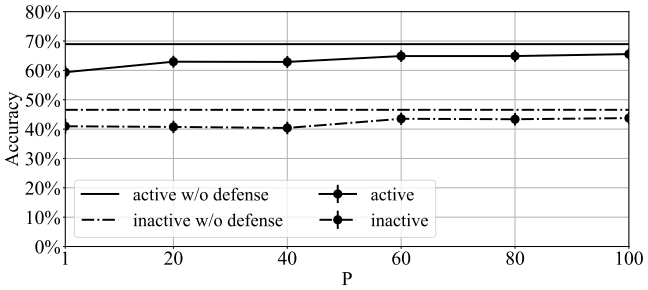

Figure 3. Comparison between the fingerprinting accuracy on the original traffic and sending *period-morphed* traffic for different values of P.

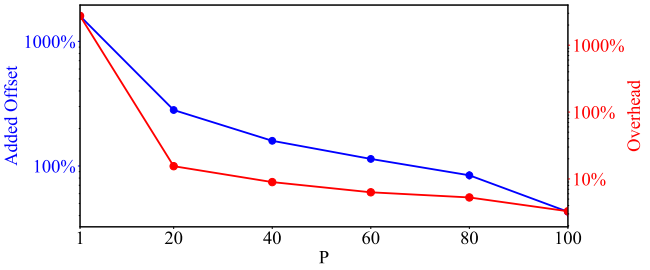

Figure 4. Overhead and added deviation from the target sending period according to the quartile chosen to set τ.

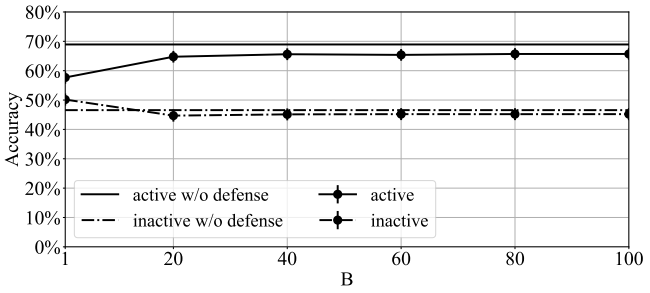

Figure 5. Comparison between the fingerprinting accuracy on the original traffic and *BFR-morphed* traffic for different values of B.

Section III-A). The figure highlights that, for both active and inactive frames, the defense improves when the quartile to determine the threshold is higher. It is worth noting that the choice of τ impacts more the fingerprinting on the inactive frames (loss of accuracy of 4 % and 13 % for, respectively, Q_1 and Q_4) than on the active frames (loss of accuracy between 14 % and 18 %).

We also evaluate the impact that padding the payload of the frames has on the transmission overhead. In particular, Figure 2 shows that the overhead increases from a minimum of 23.4 % (Q1) to a maximum of 43.5 % when all payloads are padded to 8 Byte. This means that slightly improving the defense mechanism comes at a higher cost in terms of communication overhead on the CAN bus.

B. Evaluation of sending period morphing

We evaluate the efficacy of morphing the frame sending period against frame fingerprinting by comparing the results with the performance obtained on the original non-morphed CAN traffic. Figure 3 illustrates how the fingerprinting performance varies according to the number of quantiles P used to select the target periods (see Section III-B). The figure highlights that the fingerprinting accuracy increases with P. The reason is that the higher the number of target periods, the closer the sending period of each frame is to its original value on average. The fingerprinting performance on the active and inactive frames is reduced by a maximum of, respectively, 9.5 % and 6.5 % for P = 1, and it follows a similar trend when varying P. As a matter of fact, this morphing technique does not alter the content of the frames and, therefore, impacts similarly the active and inactive frames.

Figure 4 shows how morphing the sending period of frames impacts the performance of the CAN transmission in terms of added overhead and the mean clock offset added. The figure highlights that the number of reference sending periods is inversely proportional to the overhead and to the added deviation. In fact, the more reference periods, the inferior is the difference with the original sending period on average, and less frames are morphed to high offset targets.

C. Evaluation of BFR morphing

We evaluate the efficacy of BFR morphing against fingerprinting, by comparing the accuracy obtained by the classifier on the morphed CAN traffic with the results achieved on the original traffic at the varying of B. Given the incompleteness of the ground truth, for this evaluation we consider as inactive all frames that contain only bits that do not belong to known signals and that never flip throughout the traces.

Figure 5 highlights that, in the case of active frames, the fingerprinting performance increases with B and, thus, efficacy of the morphing decreases. In particular, for $B = 1$, the accuracy is reduced by circa 11 %, while for $B = 100$, it is decreased by ≈ 3 %. In fact, the higher the number of mean BFR values used as targets for the morphing, the less the difference with the original dynamicity of the frames series. On the contrary, the performance obtained by the classifier on inactive frames remains stable. As a matter of fact, the features based on BFR are irrelevant for the fingerprinting of inactive frames. To be noted that, unlike the other two approaches, BFR morphing does not introduce any additional overhead on the communication channel.

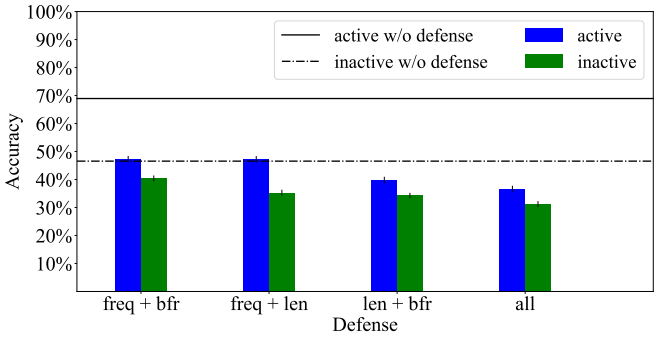

Figure 6. Fingerprinting accuracy when different combinations of the proposed defenses are applied.

D. Evaluation of combined approaches

After analyzing the impact that padding and morphing have on the frame fingerprinting accuracy based on the different characteristics of the CAN traffic, we evaluate the impact that combinations of these approaches have overall. Following the results shown in Sections IV-A, IV-B and IV-C, we select $\tau = Q1$, $P = 40$, and $B = 1$. We consider this as a fair trade-off between the decrease in the fingerprinting accuracy and the negative impact that such operations have on the communication overhead.

Figure 6 compares the fingerprinting accuracy obtained for different combinations of the three proposed approaches with the accuracy achieved on the original traffic. To be noted that when frame padding and BFR morphing are combined, the content of the padded bits is chosen according to the heuristic defined by the morphing. The results presented in the figure confirm that combining multiple approach leads to lower fingerprinting performance. The best result is obtained when all the operations on the CAN traffic are applied. In particular, the mean fingerprinting accuracy on active and inactive frames shrinks, respectively, from 68.9 % to 36.6 % for active frames and from 46.6 % to 31.2 % for inactive frames.

V. Conclusion

In this paper, we present a methodology to reduce the efficacy of CAN frames fingerprinting performed by ML classifiers without applying modifications to the protocol. Our approach is based on mutating the CAN traffic by (i) padding the length of frame payloads, (ii) morphing the frame sending frequency, and (iii) morphing the dynamic behavior of the payloads. The proposed methodology is validated on real CAN logs from 427 different vehicle models. The performance evaluation shows that our approach decreases the fingerprinting accuracy of the classifiers from up to 70 % to less than 40 %. While our method does not completely nullify the classifiers' fingerprinting capability, the obtained results highlight that traffic mutations are a promising study direction for the prevention of CAN frame matching-based reverse engineering. Future work should be conducted to further decrease the frame fingerprinting accuracy, while also paying attention to the overhead and offset in the frame sending frequency.

Acknowledgement

We acknowledge support from the National Research Fund (FNR) under grant number PRIDE15/10621687. We thank Xee for the provided datasets we used to validate our solution.

References

[1] W. Wu, R. Li, G. Xie, et al., "A survey of intrusion detection for in-vehicle networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 919–933, 2019.

[2] G. Brindescu. "DARPA Hacked a Chevy Impala Through Its OnStar System." (2015), [Online]. Available: https://www.autoevolution.com/news/darpa-hacked-a-chevy-impala-through-its-onstar-system-video-92194.html (visited on 04/02/2021).

[3] S. Woo, H. J. Jo, and D. H. Lee, "A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 993–1006, 2015.

[4] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. S 91, 2015.

[5] C. Quigley, D. Charles, and R. McLaughlin, "CAN Bus Message Electrical Signatures for Automotive Reverse Engineering, Bench Marking and Rogue ECU Detection," in *SAE Technical Paper*, SAE International, Apr. 2019.

[6] M. Marchetti and D. Stabili, "READ: Reverse engineering of automotive data frames," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 1083–1097, 2018.

[7] M. D. Pesé, T. Stacer, C. A. Campos, E. Newberry, D. Chen, and K. G. Shin, "LibreCAN: Automated CAN Message Translator," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, ACM, 2019, pp. 2283–2300.

[8] A. Buscemi, I. Turcanu, G. Castignani, R. Crunelle, and T. Engel, "Poster: A Methodology for Semi-Automated CAN Bus Reverse Engineering," in *13th IEEE Vehicular Networking Conference (VNC 2021)*, IEEE, Nov. 2021, pp. 125–126.

[9] W. Choi, S. Lee, K. Joo, H. J. Jo, and D. H. Lee, "An Enhanced Method for Reverse Engineering CAN Data Payload," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 4, pp. 3371–3381, 2021.

[10] A. Buscemi, I. Turcanu, G. Castignani, R. Crunelle, and T. Engel, "CANMatch: A Fully Automated Tool for CAN Bus Reverse Engineering based on Frame Matching," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 12 358–12 373, Nov. 2021.

[11] A. Buscemi, I. Turcanu, G. Castignani, and T. Engel, "On Frame Fingerprinting and Controller Area Networks Security in Connected Vehicles," in *IEEE Consumer Communications & Networking Conference (CCNC)*, Virtual Conference: IEEE, Jan. 2022.

[12] M. Liberatore and B. N. Levine, "Inferring the Source of Encrypted HTTP Connections," ser. CCS '06, Alexandria, Virginia, USA: Association for Computing Machinery, 2006, pp. 255–263.

[13] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis.," in *NDSS*, Citeseer, vol. 9, 2009.

[14] T. O. Router. "Tor 0.3.1.7." (2017), [Online]. Available: https://blog.torproject.org/tor-0317-now-released (visited on 10/18/2021).

[15] L. Chaddad, A. Chehab, I. H. Elhajj, and A. Kayssi, "App traffic mutation: Toward defending against mobile statistical traffic analysis," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2018, pp. 27–32.

[16] W. B. Moore, H. Tan, M. Sherr, and M. A. Maloof, "Multi-class traffic morphing for encrypted voip communication," in *International Conference on Financial Cryptography and Data Security*, Springer, 2015, pp. 65–85.

[17] I. Hafeez, M. Antikainen, and S. Tarkoma, "Protecting IoT-environments against traffic analysis attacks with traffic morphing," in *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, IEEE, 2019, pp. 196–201.

[18] K.-T. Cho and K. G. Shin, "Fingerprinting Electronic Control Units for Vehicle Intrusion Detection," in *25th USENIX Security Symposium (USENIX Security 16)*, Aug. 2016, pp. 911–927.

[19] C. Geng, S. Huang, and S. Chen, "Recent Advances in Open Set Recognition: A Survey," *CoRR*, vol. abs/1811.08581, 2018.