**ROAD: The Real ORNL Automotive Dynamometer Controller Area Network Intrusion Detection Dataset With a comprehensive CAN IDS dataset survey & guide**

**Abstract**

Autonomous Cars today use the Controller Area Network (CAN) protocol; however it misses several key security features, such as authentication scheme. To address these issues, a rapidly expanding area of research has formed that aims to identify tampering, anomalies, or assaults on such networks; this field has produced a wide range of unique methodologies and algorithms to solve these issues. The dearth of elevated datasets with genuine labelled attacks is a key hindrance to the advancement of this CAN anomalous intrusion prevention detection method (IDS) research area, making it impossible to analyse, compare, and verify these suggested approaches. The first complete survey of public information CAN intrusion datasets is presented in this paper. We provide a complete overview of each dataset and outline the disadvantages, advantages, and proposed use cases following a thorough review of the data and documentation. Our findings are intended to assist researchers in locating relevant datasets for CAN IDS test. We propose the Real ORNL Automotive Dynamometer (ROAD) CAN Intrusion Dataset, which adds to the current collection of public datasets the first dataset containing real, advanced attacks.

## Introduction

Drive-by-wire automobiles are becoming more common, relying on the continuous communication of small computers known as electronic control units (ECUs). Controller area networks (CANs), which are now ubiquitous in current automobiles, make data sharing between ECUs easier by providing a shared network with a standard protocol. There has been a considerable increase in the amount of research and proposed CAN intrusion detection systems devoted to CAN vulnerability (IDSs). Four recent in-vehicle IDS surveys demonstrate the field's expansion and evolution. These polls' classifications highlight the existing roadblocks to CAN IDS progress. Note that all these polls (even when combined) will not provide a complete picture of the stated distribution, but they do provide a representative sample. Whereas the majority of publications in the CAN security sector, particularly in IDS research, has increased dramatically in recent years, IDS research is hampered by two primary issues: (1) encrypted CAN messages  plus (2) a lack of high-quality, available to the public, actual CAN data with sophisticated assaults. Importantly, the scientific community is starting to solve the CAN signal reverse engineering challenge, in particular to ease CAN IDS research, but also to enable a wide range of downstream automotive technologies (for an overview of these works). While the obscured signal problem is not the focus of this study, it provides important context for the second point, toward which we contribute.

### Problem Statement

With true high-fidelity labelled assaults, obtaining CAN data is tough. There are three reasons why such data isn't available. Except perhaps fabrication (simple message injection) attacks, CAN data is expensive to produce with real attacks. Second, creating genuine CAN attack data poses significant threats to passengers, passers-by, and the vehicle itself. Dynamometers, which allow driving in a secure and regulated laboratory setting, are suitable, but they are expensive. Third, revealing sensitive data is a deterrent. OEMs see their CAN encodings as proprietary information. Furthermore, if additional attacks are discovered, responsible vulnerability disclosure may be required, halting data leakage at the very least. Furthermore, providing data in response to specific attacks may be perceived negatively by OEMs, perhaps leading to legal action if not properly managed.

### Project Scope

ORNL DATASET The ORNL dataset (https://0xsam.com/road/, DOI: 10.13139/ORNLNCCS/1728694) consists of 33 attack captures totaling about 30 minutes, and 12 ambient captures containing about 3 hours of ambient data. All of the data is from a single vehicle, the make/model of which we do not disclose. The published data has been obfuscated in a way that maintains the anonymity of the vehicle, while preserving all important aspects of the data for IDS. During all of the attacks, the vehicle was on a dynamometer, and was actively being driven. Ambient data was collected both on the dynamometer and on roads, while performing a variety of normal and sometimes unusual but benign driving activities (e.g., unbuckled seatbelt or opened door while driving)

**ATTACKS**

|  |  |
| --- | --- |
| **Fuzzing Attacks** | |
| **Attacks** | **Data Size** |
| Fuzzing\_attacks\_1 | 49,342 |
| Fuzzing\_attacks\_2 | 32,351 |
| Fuzzing\_attacks\_3 | 13,238 |
| **Accelerator Attacks** | |
| Accelerator\_attack\_drive\_1 | 204760 |
| Accelerator\_attack\_drive\_2 | 171936 |
| Accelerator\_attack\_reverse\_1 | 202447 |
| Accelerator\_attack\_reverse\_2 | 249413 |
| **Correlated Attacks** | |
| Correlated\_signal\_attack\_1 | 81262 |
| Correlated\_signal\_attack\_1\_masquerade | 79176 |
| Correlated\_signal\_attack\_2 | 69657 |
| Correlated\_signal\_attack\_2\_masquerade | 67517 |
| Correlated\_signal\_attack\_3 | 41829 |
| Correlated\_signal\_attack\_3\_masquerade | 405665 |

|  |
| --- |
| **Max Engine Coolant Attacks** |

|  |  |
| --- | --- |
| Max\_engine\_coolant\_temp\_attack | 61923 |
| Max\_engine\_coolant\_temp\_attack\_masquerade | 61881 |
| **Max Speedometer Attacks** | |
| Max\_speedometer\_attack\_1 | 213397 |
| Max\_speedometer\_attack\_1\_masquerade | 210953 |
| max\_speedometer\_attack\_2 | 145894 |
| Max\_speedometer\_attack\_2\_masquerade | 142754 |
| Max\_speedometer\_attack\_3 | 213551 |
| Max\_speedometer\_attack\_3\_masquerade | 207444 |
| **Reverse Light Off Attacks** | |
| Reverse\_light\_off\_attack\_1 | 67902 |
| Reverse\_light\_off\_attack\_1\_masquerade | 67230 |
| Reverse\_light\_off\_attack\_2 | 99628 |
| Reverse\_light\_off\_attack\_2\_masquerade | 97257 |
| Reverse\_light\_off\_attack\_3 | 140855 |
| Reverse\_light\_off\_attack\_3\_masquerade | 138421 |
| **Reverse Light On Attacks** | |
| Reverse\_light\_on\_attack\_1 | 133237 |
| Reverse\_light\_off\_attack\_1\_masquerade | 131246 |
| Reverse\_light\_on\_attack\_2 | 175945 |
| Reverse\_light\_off\_attack\_2\_masquerade | 172256 |
| Reverse\_light\_on\_attack\_3 | 156052 |
| Reverse\_light\_off\_attack\_3\_masquerade | 153701 |

Our dataset contains attacks requiring detectors of varying types and sophistication. The desired alteration of vehicle functionality was physically verified for all attacks included. To illustrate characteristics of the data

• The Correlated Signal attacks break the relationship between all the usually correlated wheel speed signals in the message, and cause signal discontinuities as well.

• The Max Speedometer attacks cause a signal discontinuity for the target signal without affecting other portions of the message.

• The Reverse Light Off attack breaks relationships of CAN signal. Details and takeaways are documented in the caption. Ambient dynamometer driving includes activities (e.g., reverse, drive, accelerate, brake) and can be used for training detectors and/or testing for false positives. The fuzzing attack should be very apparent to most detectors, but should be accurately detectable with timing/frequency approaches. Our dataset provides the stealthiest possible fabrication attacks. This is the hardest attack we expect a timing/frequency based detector to accurately detect. These fabrication attacks enhance the quality in terms of stealth and realism over what is currently available. Next, each fabrication attack is accompanied by a masquerade version where the ambient messages in correspondence with the injected messages are synthetically removed after the capture. the hacking expertise and time required has stymied any defensive researchers from acquiring such data. Our dataset provides the best possible alternative: real data with real attacks that are physically verified seem to have the ability to produce a bona fide masquerade attack. This provides the best possible alternative and should be T.O. attacks, not detectable by timing-base means. We believe that while these attacks are T.O., they are likely detectable by intra-signal (or per-signal) models— those detectors modeling each signal’s time series. Finally, the Accelerator Attacks are unique examples of CAN data from a functioning vehicle with only the vehicle’s ECUs transmitting messages, but in a compromised manner. There is no disruption of the timing of the CAN messages, thus this attack should be T.O. This attack permits adequate testing of advanced IDSs that must rely on some understanding of the payloads and their correlations with each other. Having data from a wide variety of real vehicles in actual road driving conditions with simple to very sophisticated attacks is of course ideal; thus, many limitations to this dataset are indeed present. Notably, we only present data from a single vehicle. Secondly, the dataset includes only dynamometer data, which is well known to cause subtle differences in actual road driving. Thirdly, we provide attack intervals rather than labeling each message.

As above we have mentioned some of the types of attacks which are in our dataset furthermore we have ambient dataset too which is further divided into multiple datasets so started implementation, as we see The data set is needed to be preprocessed and to do this we make an excel file to perform it. First, we properly put all data. Copy the first variable which is **Unix Timestamp** and paste it on an excel file to perform analysis on it without any processed technique. The second variable is a channel and this kind of variable need to proposed so in this variable, we did remove the ‘can’ and just put its value. The third variable is **Data Field**this also needs to change because it has a hexadecimal value so needs to convert in decimal only. The final variable is ID and we transform it into our label.

The whole data set is in one column which belongs all the value is in it. So need to be separated one by one, used spilt function to able separate it from each other, continue it by giving the values to the algorithm. We separate it by using the pandas' data frame str spilt. The logic is that separated it by using space between the values in them. The second approach is that by # because it has # so separate from each other and continue with the () technique. In final the data saved on our drive. Then the next operator is set role, it is used for choosing the label form data set in our data set by selected the data filed label. To continue we have to split data before applying machine learning algorithm. so splitted the data for training and testing 70% for training and 30% for testing. in final we applied algorithms **Random forest, Decision Tree** ,**SVM and KNN** in jupyter notebook.

**Random Forest:**

Random Forest is a well-known supervised machine learning algorithm. In machine learning, it could be used for both classification and regression. It is focused on ensemble learning, which would be the process of integrating numerous classifiers to tackle complex difficulties and reduce the precision. Rather than depending on a single decision tree, the random forest analyzes every tree's forecast and forecasts the final outcome depending on the absolute majority of forecasts. The more trees in the forest, the more accurate it is and the issue of over fitting is avoided.

**Decision Tree:**

Decision Tree is a supervised learning method which can be used to solve both classification and regression problems; however it is most commonly employed to solve classification issues. The Decision Node and the Leaf Node are indeed the two types of nodes of a Decision tree. Leaf nodes are indeed the result of such decisions and therefore do not include any more branching, while Decision nodes are being used to take a decision and have several branches. We utilize the CART method, which means for Classification and Regression Tree algorithm, to form a tree. A decision tree merely wants to know and divides the tree into sub trees based on the response (Yes/No).

**SVM**:

SVM (Support Vector Machine) is among the most widely utilized Supervised Learning techniques for Classification and Regression issues. The SVM algorithm's purpose is to find the optimum line or decision boundary for categorizing n-dimensional space into classes so that additional data points can be readily placed in the appropriate category in the foreseeable. A hyper plane is the name for the optimal choice boundaries.

There are two main types of SVM:

Linear SVM: Linear SVM is a classifier that is used for linearly separable data, which implies that if a dataset can be categorized into two categories using only a single horizontal line, it is called linearly separable data, and the predictor is named Linear SVM.

Non-linear SVM: Non-linear SVM is mostly used for non-linearly separated data, which implies that if a dataset can't be categorized using a horizontal line, it's non-linear data, and the classifier utilized is called Non-linear SVM.

**KNN:**

Among the most fundamental Ml Algorithms is the K-Nearest Neighbor algorithm, which is based on the Supervised Learning approach. The K-NN approach assumes that perhaps the fresh case/data and current cases are related and puts the new case in the most suitable category with the segments. It also is called as a lazy learning algorithm because it doesn't immediately understand from the training set; instead, it stores the dataset and uses it to categorize it once the time arrives.

These are the accuracies that we are achieving by doing cross validation from random forest, as we can see that from both of these two algorithms our accuracy increases by we increase the hyper parameter values , if we analysis the graphs of these two algorithms we find that in random forest our accuracies varies between 94% to 80% its mean that in random forest gave accuracy up to 80 to 94% while on the other hand in decision tree accuracies varies between 95% to 88% here we find that both algorithms are efficient they are working properly but we find that decision tree is better than random forest as it is not varies to much as like random forest and give answer accurately.

**Accuracies With Respect to Algorithms without Hyper Parameter Tuning:**

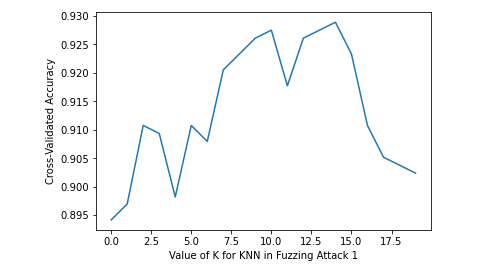
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algorithm | KNN | Decision Tree | Random Forest |  | SVM |
| Fuzzing Attack 1 | 80.5% | 61.3% | 61.2% |  | 20.05% |
| Fuzzing Attack 2 | 79.4% | 60.03% | 60.03% |  | 20.5% |
| Fuzzing Attack 3 | 68.5% | 49.3% | 49.4% |  | 20.09% |
| Max Engine Coolant Attack | 76.7% | 57.7% | 57.6% |  | 20.09% |
| Max engine Coolant Temp Attack | 78.5% | 58.9% | 58.8% |  | 20% |

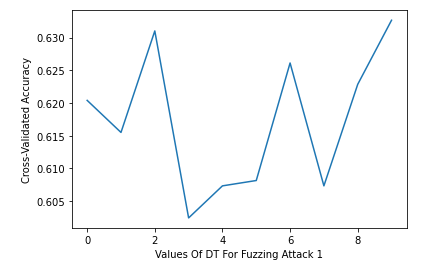
**Accuracies With Respect to Algorithms with Hyper Parameter Tuning:**

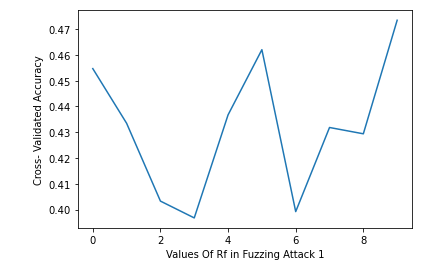
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algorithm | KNN | Decision Tree | Random Forest |  | SVM |
| Fuzzing Attack 1 | 90.5% | 61.3% | 43.2% |  | 20.05% |
| Fuzzing Attack 2 | 79.4% | 60.03% | 60.03% |  | 20.5% |
| Fuzzing Attack 3 | 76.68% | 65.3% | 55.8% |  | 20.09% |
| Max Engine Coolant Attack | 78.6% | 59.3% | 41.8% |  | 20.8% |
| Max engine Coolant Temp Attack | 78.9% | 58.9 | 58% |  | 20.3% |

Analyzing Graphs:

Fuzzing Attack\_1:

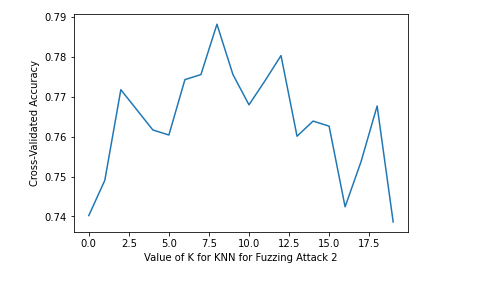


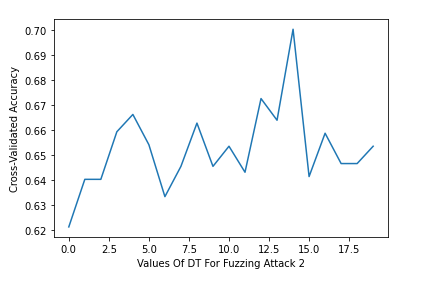


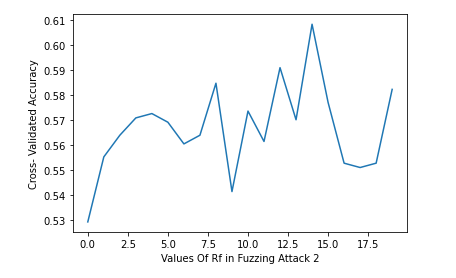


As we can see that in fuzzing Attack 1 dataset SVM stuck at 20% machine is not learning from SVM on this dataset that’s why we can say that this algorithm is not suitable for this dataset, if we see decision tree and random forest classifier both are giving almost similar accuracy but both stuck on 60%, even when we tune hyper parameters accuracy remains same from both of these algorithms we didn’t see any variations by tuning too, But on the other hand if we see KNN it’s good machine is learning from KNN with our dataset it is working well when we implement without hyper parameter it gives 80% accuracy which is good overall and after tuning of the hyper parameter we get accuracy around 93% which is very good. So, we can say that KNN works expectedly well for our dataset.

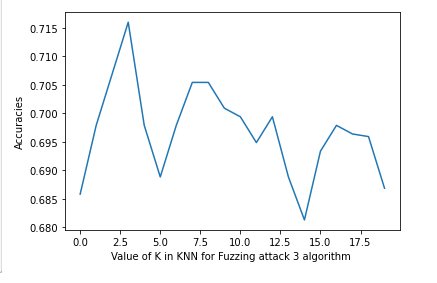
Fuzzing Attack 2:

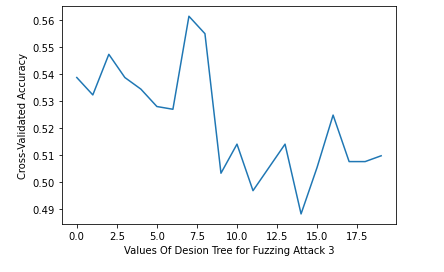




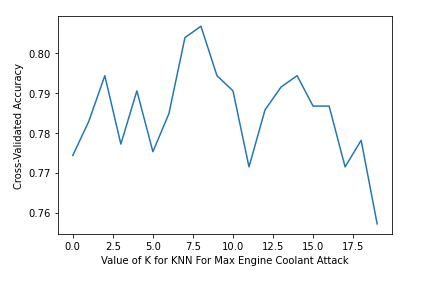


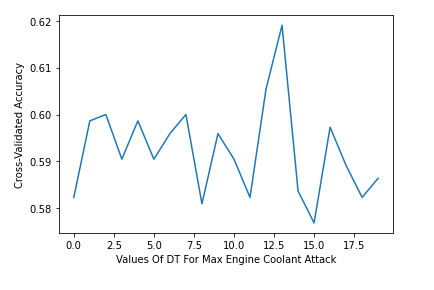
Fuzzing Attack 3:

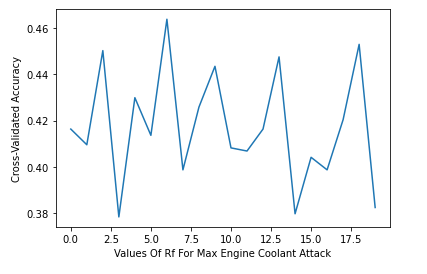




Max Engine Coolant Attack:

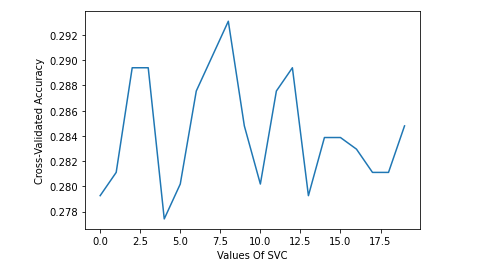




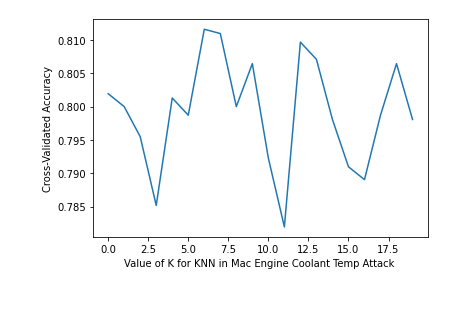


Max Coolant Engine Temp Attack:

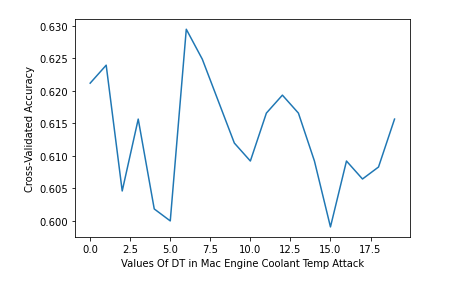
SVM:



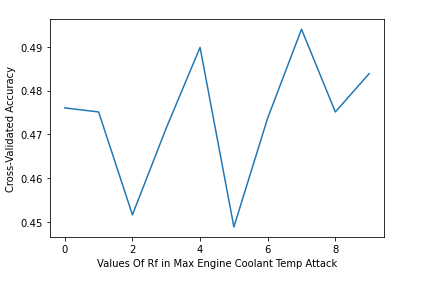
KNN:



DECISIONTREE:



RANDOM FOREST:



From above graphs we noticed that SVM is not suitable for our datasets similarly decision tree and random forest are also not good for our dataset because from these three algorithms accuracy stuck and machine not learning from the dataset we cannot select these three algorithms on the other hand KNN is only algorithm from which we are getting variations in accuracy and we can say that KNN algorithm is the best algorithm among these four algorithm.

### Used Technologies

##### PYTHON:

##### Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Mostly python is used for machine learning and data science.

**ANACONDA:**

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

## 

### Conclusion

### Here we can conclude that only Knn algorithms (K nearest neighbor) work exceptionally well on the dataset while on the other hand remaining three algorithms (SVM ,Decision Tree and Random Forest )are not working well as they only give 50% accuracy even after applying hyper parameters tuning and took too much time for implementation .So, I can say that KNN is good for our dataset. And we will work with KNN for future implementation.

### Future Works

In this work we are presenting comprehensive survey of publicly available CAN intrusion datasets. Based on a thorough analysis of the data and documentation, for each dataset we will provide a detailed description and enumerate the drawbacks, benefits, and suggested use cases. Our analysis is aimed at guiding researchers in finding appropriate datasets for testing a CAN IDS.

* Will work for other datasets which are remaining.
* Come up with proper computational resources.
* Will suggest best Algorithm.
* Will publish our research work.