COMPARISON OF MALWARE CLASSIFICATION METHODS USING CONVOLUTIONAL NEURAL NETWORK BASED ON API CALL STREAM

Matthew Schofield[1], Gulsum Alicioglu[2], Bo Sun[1], Russell Binaco[1], Paul Turner[1], Cameron Thatcher[1], Alex Lam[1] and Anthony Breitzman[1]

[1]Department of Computer Science, Rowan University, Glassboro, New Jersey, USA
[2]Department of Electrical and Computer Engineering, Rowan University, Glassboro, New Jersey, USA

ABSTRACT

Malicious software is constantly being developed and improved, so detection and classification of malwareis an ever-evolving problem. Since traditional malware detection techniques fail to detect new/unknown malware, machine learning algorithms have been used to overcome this disadvantage. We present a Convolutional Neural Network (CNN) for malware type classification based on the API (Application Program Interface) calls. This research uses a database of 7107 instances of API call streams and 8 different malware types:Adware, Backdoor, Downloader, Dropper, Spyware, Trojan, Virus,Worm. We used a 1-Dimensional CNN by mapping API calls as categorical and term frequency-inverse document frequency (TF-IDF) vectors and compared the results to other classification techniques.The proposed 1-D CNN outperformed other classification techniques with 91% overall accuracy for both categorical and TF-IDF vectors.

KEYWORDS

Convolutional Neural Network, Malware Classification, N-gram Analysis, Term Frequency-Inverse Document Frequency Vectors, Windows API Calls.

1. INTRODUCTION

There are many different forms of malicious applications present in the highly connected software environment of the current technological world. Malicious applications such as viruses are constantly being developed and distributed in an attempt to extract information from computers or networks, and software such as firewalls and antivirus programs are constantly evolving to attempt to protect benign users and software from this threat [1]. Malicious software is constantly being developed and improved, so the classification of malicious applicationis an ever-evolving problem.

Signature-based, behavior-based, and specification-based techniques are commonly used to detect malware. Signature-based detection techniques [1] keep a database that contains the signatures of the known malicious programs and uses these signatures to identify the presence of attacks by matching the signature kept in the database. Suchtechniques offer fast malware detection and require less computational resources. However, it cannot detect new or unknown malware. Behavior-based detection techniques analyze various features such as the source and destination of malware, types of attachments, and statistical features [2]. Behavior-basedmethodshave ability to detect known and unknown malwares, but they require high computational resources such as memory and CPU time [2]. To overcome these disadvantages, a specification-based technique [3], which is basically a behavior-based approach, is developed. Researchers have employed data mining and machine learning techniques and obtained good performance in specification based malware detection and classification with high accuracy scores [4-6]. Figure 1 summarizes the malware detection techniques with their advantage and disadvantages. These methods provide reliable and accurate results, especially for classifying metamorphic malware.

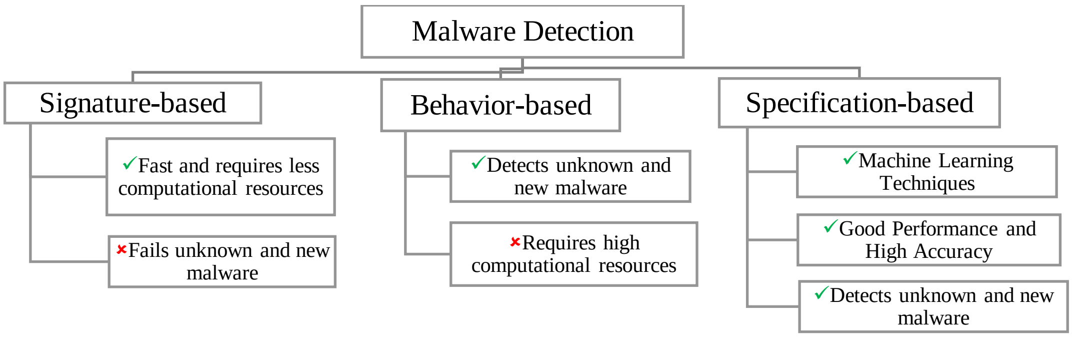

Malware Detection
Signature-based
✓Fast and requires less computational resources
×Fails unknown and new malware
Behavior-based
✓Detects unknown and new malware
×Requires high computational resources
Specification-based
✓Machine Learning Techniques
✓Good Performance and High Accuracy
✓Detects unknown and new malware

Figure 1. The classification of malware detection techniques

Metamorphic malware indicates itself with different sequences in various environments, but it must demonstrate the same behavioral features in all environments. Hence, most of the methods usedbehavioral features for malware classification and detectionrather than structural features [7-8].APIcall sequences can provide considerable information about the behavioral features of malware. Therefore, most of the researchers conducted their study by using API calls to detect and classify malware [9-12]. One benefit of having the ability to classify the type of malware from the system call behavior is to identify a potential malicious source faster and better understand the effects of the malware. Understanding the class of malware to which a malicious program belongs gives administrators of an infected deviceinsights on resolution strategies regarding the attack.

There are three major approaches for malware detection based on the analysis types: static (code analysis), dynamic (behavioral analysis), and hybrid analysis using both static and dynamic attributes of malware. Static malware detection inspects the malware files in binary form and extracts features of the malware. Static analysis decides whether the file is malicious or not by observing the execution path and analyzing the extracted features. To detect malware through dynamic analysis, binary files are run on a virtual environment. Then, the run time behavior of these files is observed. During dynamic analysis, features of the files such as memory usage and system calls are extracted. Since the API call sequences disclose behaviors of malware, they are commonly used in dynamic analysis.

In this paper, we proposed two different approaches to classify the type of malware of a malicious program based on its API call stream.We used a public Windows API call dataset [8] with 8-classes of malware in experimentation. In our first approach, we encoded and mapped API calls into a categorical vector. We used a 1-D Convolutional Neural Network (CNN) andcompared our results with existing methods. We also presented a text-based analysis as a second approach for malware classification by using n-gram analysis to convert API calls to term frequency-inverse document frequency (TF-IDF) vectors. Comparisons are conducted by using CNN,Random Forest (RF), Bernoulli Naïve Bayes (NB), Multinomial NB, Gaussian NB, Decision Tree (DT), AdaBoost, Bagging Classifier, k-Nearest Neighbors (kNN), and Multilayer Perceptron (MLP). We also presented overall comparisons for both approaches.This paper extended our previous publication [13] by presenting both binary and multiclass classification results and adding detailed information related to our approach.

Since the API call sequences have complex nature to be understood by end users, we also proposed a visual approach to clarify the patterns of API calls based on overall malware types. The purpose of the visualization is to form a union between computationally complex algorithms and human intuition [14]. The studyalso investigates the utility of visual analytic approaches in data analysis, exploration, and computational model development.

This paper is organized as follows. Section 2 provides a literature review. Section 3 describes the malware dataset, comprehensively. Section 4 provides a detailed methodology and visual approaches for data exploration. Section 5 presents the results of the study. Finally, Section 6 concludes the paper along with the discussion.

2. BACKGROUND

Since the volume of malware being spread has had rapid growth, many studies have been carried out to analyze and detect malware automatically [3,15-18]. The most common and traditional way to detect malware is for a system to maintain a hash signature-based blacklist of known malware. These signature-based methods fail to detect new, unknown, or obfuscated malware. Data mining and machine learning approaches have overcome the disadvantages of signature-based methods by detecting new and unknown malware with high accuracy and detection rate [5, 18]. Hence, most of the studies used various machine learning algorithms to detect malware accurately and quickly [4,15-17].

In recent studies, API call sequences are mostly used as a feature to detect malware. Xiaofeng et al. [16] used a combined deep learning and machine learning model for malware behavior analysis with binary classification (i.e., benign and malicious). Random Forest (RF) was used to extract API call features. The combined RF and Long Short-Term Memory (LSTM) model classified malware with 96.7% accuracy. Malware analyses are carried out by using static or dynamic analysis. Researchers have studied to improve the usage of static and dynamic analysis. Han et al. [15] used RF, Decision Tree (DT), k-nearest neighbor, and XGboost methods to detect and classify multiclass malware by combining static and dynamic API calls sequences.The study explains the differences and relation between API sequences and explains malicious behavior types via MalDAE framework.The study achieved a 97.8% detection rate and 94.4% classification accuracy with RF. The study also examined the effect of sequence length on measuring time. Salehi et al. [18] proposed a novel dynamic malware feature selection method based on generating new features to overcome disadvantages of using only static or dynamic analysis. RF, DT, sequential minimal optimization, and Bayesian Logistic Regression (BLR) methods were used, and 98.1% accuracy rate is obtained with BLR by generating three feature sets. As the volume of cyber-attacks with advanced malware increases, it makes malware hard for detection and classification. Bahtiyar et al.[5] proposed a multidimensional machine learning approach to detect advanced malware. They applied various regression models by using five distinguished features: stealthiness, Stuxnet closeness, behavioral instability, conventional malware arsenal and metamorphic engine. The proposed method provided 0.0001 mean squared error rate. Belaoued andMazouzi [19] presented a fast-portable executable system to detect malware by using an efficient feature selection method. They used decision tree, boosted decision tree, AdaBoost, Random forest, and rotation forest algorithms for binary classification. They evaluated the proposed system with different subsets of data and achieved more than 98% accuracy in a short detection time (0.09 seconds). Gupta et al. [20] conducted a comprehensive study to capture malware behaviors based on API calls sequences. The experiments were conducted with five malware classes for 2000 samples. They encoded 534 important API calls to

26 categories (A... Z). N-gram analysis was carried out to extract class specific patterns. They also calculated fuzzy hashed scores with an ssdeep algorithm to use as classification criteria. Yazi et al. [7] generated a dataset contains 8 different malware type with their Windows API call sequences and utilized LSTMclassification models for 8 different types of malware. The highest accuracy rate was obtained with 97.5% in the adware class. Zhang et al. [21] proposed a feature-hybrid malware variants detection method by integrating various features. They extracted bi-gram model and encoded API calls as a frequency vector. The study achieved 90% classification accuracy by utilizing Convolutional Neural Network (CNN). Table 1 presents a summary of the reviewed studies.

Table 1. A summary of the related studies for malware classification

Study	Analysis Method	Problem	Method	Performance
Xiaofeng et al. [16]	Dynamic	Binary	RF, LSTM*	96.7% Accuracy
Han et al. [15]	Hybrid	Binary	RF*,DT, kNN, XGboost	94.4% Accuracy
Salehi et al. [18]	Dynamic	Binary	RF,DT, Sequential Minimal Optimization, Bayesian Logistic Regression*	98.1% Accuracy
Bahtiyar et al. [5]		Feature selection	Linear Regression, Polynomial Regression, RF Regression*	0.0001 MSE rate
Belaoued and Mazouzi[19]	Static	Binary	DT, Boosted DT, AdaBoost, RF, Rotation Forest	98% Accuracy
Gupta et al. [20]	Dynamic	Multiclass	Fuzzy Hashed Scores	96% Accuracy
Yazi et al. [8]		Binary	LSTM	97.5% Accuracy
Zhang et al. [21]	Hybrid	Binary	CNN, NN	90% Accuracy

*Indicates the best performing method

In the light of the literature review, we conducted a comprehensive study by using Windows system API calls dataset [8] to classify malware. API call sequences are used as a feature for malware classification. In the literature, most of the studies are carried out for binary classification to detect malware by utilizing one-vs-rest strategy [7,16,19-20]. However, different types of malware require different precautions and treatments. For this reason, it is crucial to classify and determine the type of a given malware sample. We conducted experiments on both binary and multiclass to detect malware by using various decision trees (DT, RF, AdaBoost DT and Bagging DT), Naïve Bayes (Bernoulli, Multinomial and Gaussian), k-NN, and MLP and a 1D CNN algorithm. We implemented two different data representation approaches: text-based analysis through TF-IDF vectors by using n-grams and binary categorical vector by mapping API calls.

3. Windows Malware API Call Dataset

The Windows Malware API Call Dataset is a public malware dataset containing 7,107 samples of malicious files among eight classes of malware. This dataset is created by running over 20,000 malware with the Cuckoo Sandbox application, which allows any software to be run as if it were in a real environment [8]. Then, the API call sequences produced by the malicious applications on Windows operating system were acquired. The distribution of malware classes is shown in Table 2.

The dataset contains a class label and an arbitrary length list of API call strings for each data entry. For this dataset, API calls represent a system call on the Windows operating system occurring during the runtime of the malicious file. The dataset does not include benign samples.

Table 2. The distribution of malware classes

Malware Type	**Sample Size**
Spyware	832
Downloader	1001
Trojan	1001
Worms	1001
Adware	379
Dropper	891
Virus	1001
Backdoor	1001
Total	**7107**

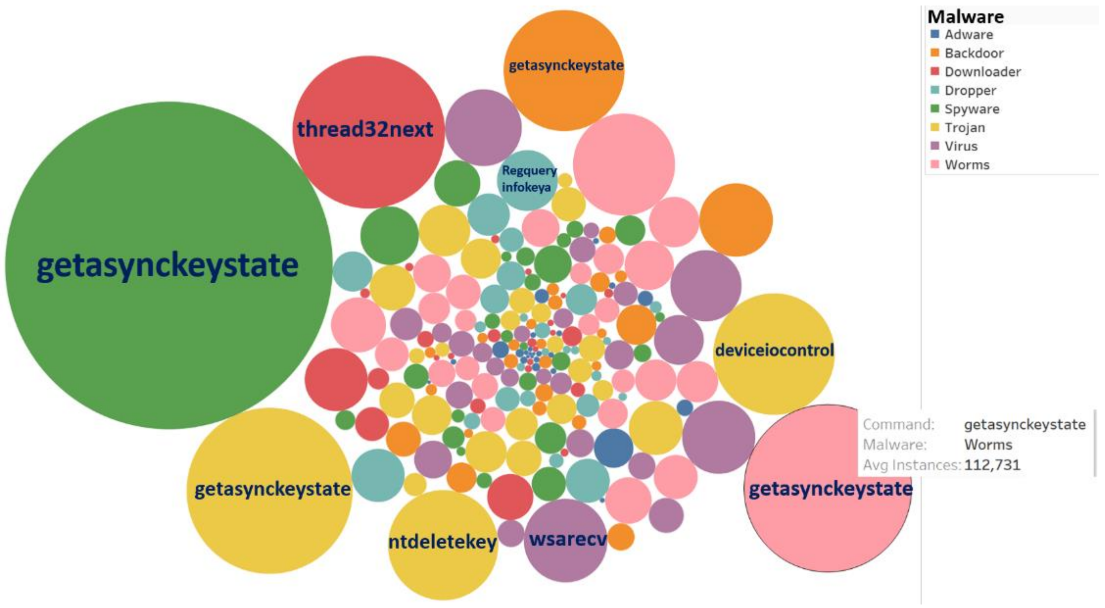


Figure 2. Bubble plot of frequent API calls by class

Exploratory data visualizations are presented to provide some human-readable insight into the breakdown of data features and metrics for the Windows API call dataset by using Tableau [22]. Figure2shows a bubble plot of the most frequent API calls per class. The labels in the bubbles represent the API calls. The size of each bubble indicates the average instances that contain the related API call.GetAsyncKeyState is the most frequent API call for Spyware,Worms, Trojan and Backdoor malware classes, and thread32next is the most frequent call for the Downloader class.Similarly, bar charts were constructed for each class with their record percentage, as seen in Figure3.

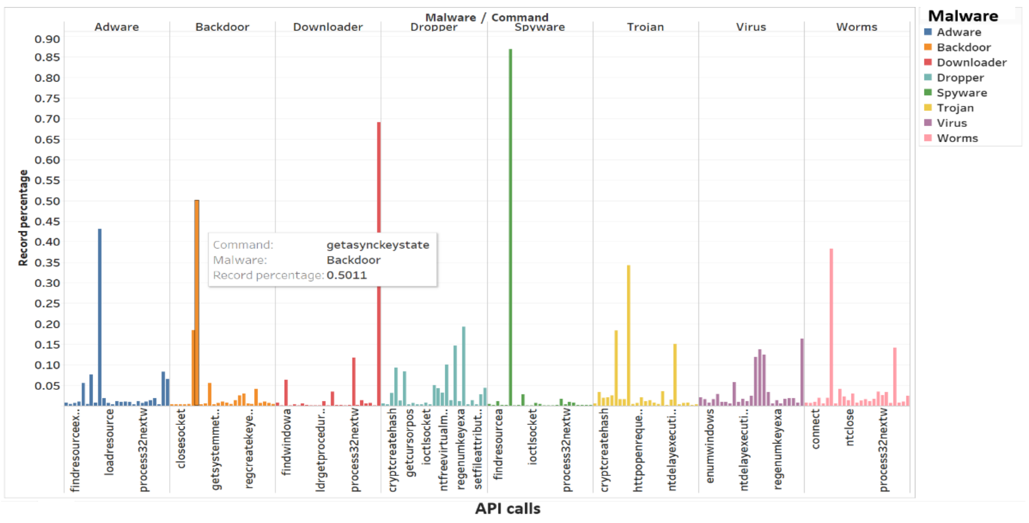


Figure 3. API calls frequency of each malware

However, since these most frequent API calls are often used in many of the mentioned classes, statistical analysis and traditional malware detection techniques may not be sufficient for identifying what class a call stream belongs to. This justifies the need for a more complex system than an expert rule set.

4. METHODS

In this section, we presenttwo malware classification methods using CNN based on categorical and text-based approaches in detail.

4.1. Convolutional Neural Network

Machine learning algorithms arefrequently utilized to classify and detect malware due to their prediction power and good performance. Since the API call streams are high dimensional complex data, deep neural networks are preferred by many studies [23-24]. Deep neural networks can extract features from raw data and classify high dimensional data well. With the increase in technology and computational resources, these algorithms have achieved good performance. Convolutional Neural Networksare one of the most popular deep neural networks [25]. They achieve impressive performance especially for image and voice recognition. CNNs take their name from a mathematical operation called convolution that helps to extract features from the input. The architecture of a CNN is shown in Figure 4. It consists of three main layers: convolutional layer, pooling layer, and fully connected layer.

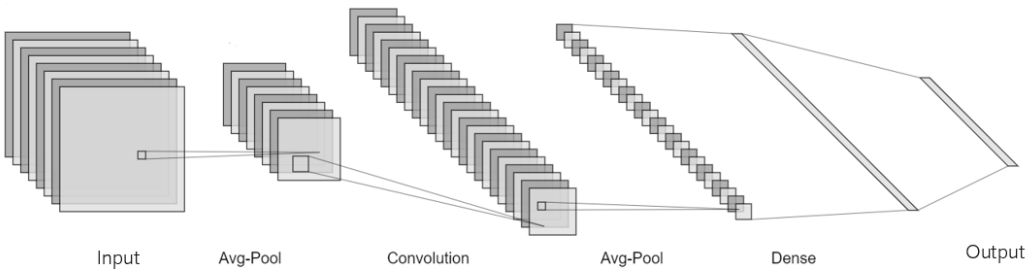


Figure 4. The architecture of a CNN [26]

The convolutional layer uses convolutional filters to create a feature map from an input. Convolution is a mathematical operation, that computes a dot product between the input weights and region that are connected to the input. ReLU (Rectified linear unit function) activation [27] function is commonly used for the outputs of CNNs. Its formula is shown in Eq. (1). This activation function rectifies the input values that have less than zero, by forcing them to be zero [28].

$$\text{ReLU}(x) = x^{+} = \max(0,x) \tag{1}$$

where x is the pre-activation output value of the nodes.

During the convolution operation, there may be information loss on the edges of the input [25]. To prevent this information loss zero-padding is used. It preserves the dimensionality of the outputs by equalizing the generated feature maps to the same size as the input.

The pooling layer is responsible for reducing the dimensionality of the attributes. Pooling is used to gradually decrease the dimensions of the attribute representation. Hence, it provides low computational cost by shrinking memory cost and the number of parameters [28-29].Average pooling and max-pooling are the most popular pooling techniques. While average pooling calculates the average of each feature values of the feature map [29], max pooling instead returns the maximum value of the feature values.

A fully-connected layer behaves like a traditional feed-forward neural network to do classification. This layer uses a Softmax activation function, as seen in Eq (2). Softmax provides an output whose value ranges between 0 and 1 and returns probabilities of each class [27].

$$SoftMax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{2}$$

where x is an input vector of pre-activation values. The letter e represents the base of the natural logarithm system.

The convolutional neural network model architecture used in this study is shown in Figure5. The first layer of the network is a 1-D convolutional layer, this uses convolutional filters to create a feature map from an input API call stream seen as a vector with 279 channels. This layer uses 64 filters with ReLU (Rectified linear unit function) activation function [27], which has an almost linear function and therefore retains the properties of linear models. These properties provide ease to optimize with gradient descent methods.The following two layers flatten the generated feature map using average-pooling [28-29] on segments of length two within the generated feature map.The next layer is a standard feed-forward layer in a deep learning architecture which transforms the generated feature vector using ReLU activation [27]. The final layer then uses softmax activation [27] to generate a vector. Softmax represents the probabilities of the input malware APIcall stream belonging to each class. Each class is statically assigned to a specific index of this probability vector within the training data.

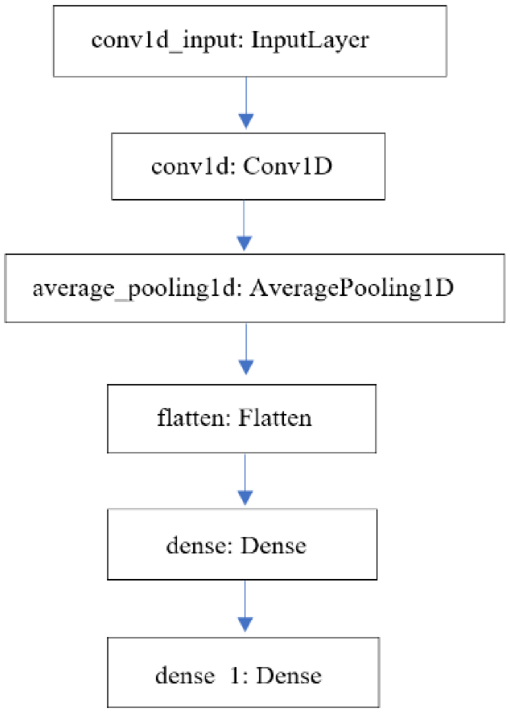


Figure 5. The Model Architecture

Two different analytic features based on categorical vector and TF-IDF vector were used for the API call dataset, and then fed to the proposed CNN to classify the malware.

4.2. Categorical Vector

In the categorical-based approach, the API call stream produced by the same type of malicious program is used in the one-dimensional convolutional neural network (CNN) to classify malware. For the data to be formatted for the CNN, preprocessing was necessary. Each sample was comprised of a list of system API calls. To reduce the dimensionality of the data, all call streams that were longer than 2000 API calls were reduced to thefirst 2000 API calls. This reduction affected 1466 of 6851 records. The dataset contains 278 unique API calls in total.These unique API calls may be repeated throughout the API call stream.

Each unique API callwas encoded into a categorical vector. The lengthof the vector is the same as the number of unique API calls plus one, totaling 279.For each unique API call, 1 was placed in the vector's index corresponding to that call and the rest of the vector was set to 0's. Additionally, a padding vector was created as a categorical vector with a 1 in the previously unused last index. This padding vector is then added to encoded API call streams that have a length less than 2000 until the API call stream's length reaches 2000. The padding vector allows equalizing the API call streams of different lengths to provide batch optimization. Each unique API call mapped to an integer and converted to a categorical vector to use in the CNN. Figure 6 provides a specific example.As seen in Figure6, API call "ldrloaddll" was mapped to 133 and API call "ldrgetprocedureaddress" was mapped to 132 according to their positions in alphabetical order (see Figure 11). Then, using their mapped value, categorical vectors are obtained by replacing with 1 at position 132 and the rest of the index with 0 for the "ldrloaddll" call. For the "ldrgetprocedureaddress" call, we placed 1 at position 133 and 0 at the rest of the indices. These categorical vectors represented the unique API calls, and they were substituted with their corresponding position in each API call sequence.

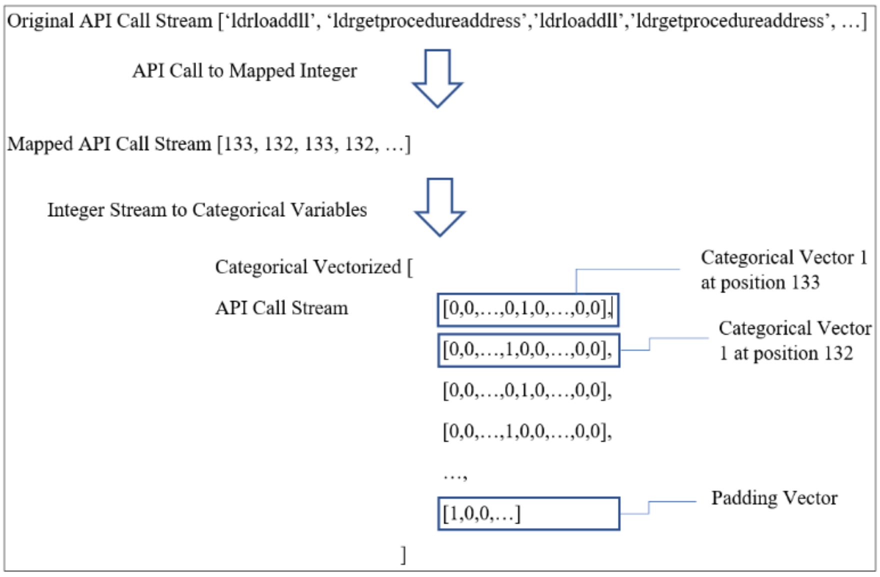

Original API Call Stream ['ldrloaddll', 'ldrgetprocedureaddress','ldrloaddll','ldrgetprocedureaddress', …]
API Call to Mapped Integer
Mapped API Call Stream [133, 132, 133, 132, …]
Integer Stream to Categorical Variables
Categorical Vectorized [
API Call Stream
[0,0,…,0,1,0,…,0,0],
[0,0,…,1,0,0,…,0,0],
[0,0,…,0,1,0,…,0,0],
[0,0,…,1,0,0,…,0,0],
…,
[1,0,0,…]
]
Categorical Vector 1 at position 133
Categorical Vector 1 at position 132
Padding Vector

Figure 6. Encoding methodology for categorical vector

The proposed 1-D CNN model and encoding methodology can be used to classify multiclass malware API call streams. Converting a list of words to a categorical variable can be utilized in any text data. This encoding provides ease in the analysis and classification. After encoding, the index of the largest probability within the output probability vector will correspond to the predicted class.

4.3. TF-IDF Vector

The text-based analysis is the othercomputational approach used for the API call dataset. Feature extraction and selection is an important step of classification through machine learning algorithms. The API call dataset includes only the lists of calls during malicious files running and do not includedirect features. Therefore, we need to extract features from the API call sequences to apply machine learning algorithms for malware classification and detection. Byte n-grams are one of the most popular approaches to extract features from text data. N-grams, described in [30], are sequential series of words or terms in an ordered sequence. N-grams are commonly used in text mining and natural language processing problems. Since the API calls consists of a sequence of calls in text, we conducted a text-based approach for feature extraction as well. In this study, N-grams can be created from a moving window of API calls alongthe call streams in the dataset. Due to the nature of the API call streams (hundreds or thousands of terms in which calls were often repeated sequentially), 10-grams, sequences of ten terms, were used.Figure7provides general structure of n-grams. We provided the first three grams for API call streams as an example with a small piece of API call sequence. As seen in Figure7, when a unigram is utilized (N=1), each API call is grouped with only 1 API call. Consequently, we obtained 4 unigrams, namely"getAsyncKeyState","ldrloaddll","ldrgetprocedureaddress", and "thread32next". Similarly, for bigram (N=2), API calls are grouped with only 2 consecutive calls. In this case, we obtained 3 bigrams: "getAsyncKeyState, ldrloaddll", "ldrloaddll, ldrgetprocedureaddress", and "ldrgetprocedureaddress, thread32next". In trigram (N=3), we obtained 2 trigrams containing 3 consecutive API calls including "getAsyncKeyState, ldrloaddll, ldrgetprocedureaddress" and "ldrloaddll, ldrgetprocedureaddress, thread32next". In a similar manner10-grams are obtained from a moving window of API calls along the call streams. To parse API call streams, we modeled call streams to represent each n-gram as n words, where n is the position of the API call.

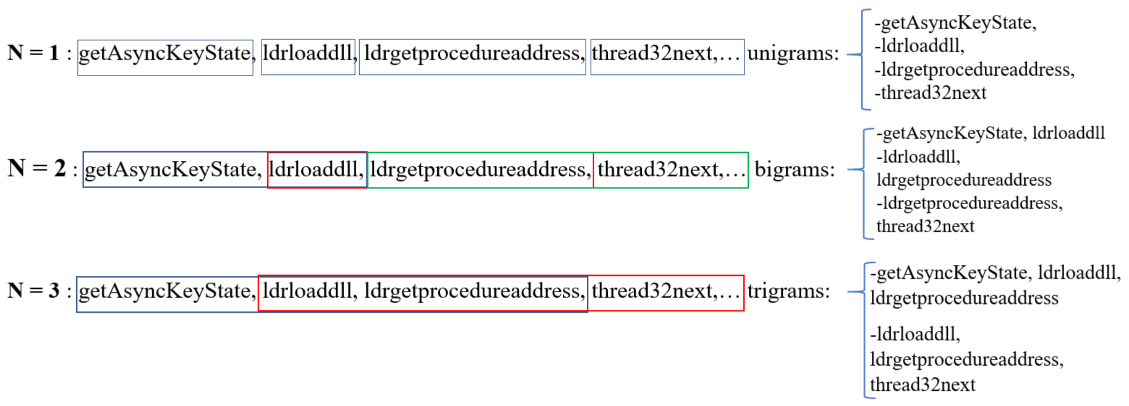

N = 1 : getAsyncKeyState, ldrloaddll, ldrgetprocedureaddress, thread32next,…
unigrams:
-getAsyncKeyState,
-ldrloaddll,
-ldrgetprocedureaddress,
-thread32next
N = 2 : getAsyncKeyState, ldrloaddll, ldrgetprocedureaddress, thread32next,…
bigrams:
-getAsyncKeyState, ldrloaddll
-ldrloaddll, ldrgetprocedureaddress
-ldrgetprocedureaddress, thread32next
N = 3 : getAsyncKeyState, ldrloaddll, ldrgetprocedureaddress, thread32next,…
trigrams:
-getAsyncKeyState, ldrloaddll, ldrgetprocedureaddress
-ldrloaddll, ldrgetprocedureaddress, thread32next

Figure 7. N-grams representation

These identified vectors of terms were translated to Term Frequency, Inverse Document Frequency (TF-IDF) vectors. TF-IDF vectors helps to determine the importance of words by assigning weights to the words [31]. Term frequency is used to calculate the number of times a term is present in a document [32]. Inverse document frequency is a measure of information provided by words. IDF assigns a value to the word according to their frequency in the corpus. If a word is occurs frequently, then IDF assigns less weight and if a word occurs infrequently thena higher weight is assigned. The formulation of TF, IDF and TF-IDF are provided in Eq. (3), Eq. (4) and Eq. (5), respectively [31].

$$TF(x) = \log(x+1) \tag{3}$$

$$IDF(x) = \log(\frac{1+p}{1+d})+1 \tag{4}$$

$$TF\text{-}IDF = TF * IDF \tag{5}$$

where x is a vector of raw frequencies, p is the number of tracks in the X set, and d is a vector that counts the tracks where every 10-grams appears.

After 10-grams are obtained, TF values are calculated for each API call. Then IDFs are calculated and multiplied with TF values to get TD-IDF values. These TF-IDF vectors were used as inputs for a suite of decision tree classifiers for both binary cases (i.e., Trojan versus Not Trojan, and so on) and multiclass classifications. Severalclassifiers were used including RandomForest (RF), BernoulliNaïve Bayes (BNB), MultinomialNB (MNB), GaussianNB (GNB), DecisionTree (DT)Classifier, AdaBoostClassifier, BaggingClassifier, kNN, and MLP. Additionally, the proposed1-D CNNwas used to classify the malware by using TD-IDF vectors. The same CNN structure as described in the previous section was used but its input layer was changed to accept TF-IDF vectors in a larger length of 14666. The Python scikit-learn library was used for the implementations of n-grams, TF-IDF vectors, and all of the classifiers.

5. RESULTS

This section provides classification results from several ML approaches including the proposed 1-D CNN by using categorical and TF-IDF vectors, respectively. F-1 scores and various performance metrics were used to evaluatethe ML methods.

5.1. Categorical Vector

To utilize the 1-D Convolutional Neural Network, API callswere converted to categorical variables, as described earlier. An 80-20 train-test split of API call stream data was adopted. The proposed 1-D CNN model achieved an accuracy of 91.0%, a macro F1 score of 91.3%, and a weighted F1 score of 91.0%. We compared our proposed algorithm with Random Forest (RF) classifier, Logistic Regression (LR), Support Vector Machine (SVM), and k-nearest neighbor (kNN) with 3, 5, and 7 neighbors. A comparison of our model and the existing classification algorithms is shown in Figure8. The 1-D CNN outperformed the other algorithms.The algorithm that performed most competitively with CNN was the Random Forestwhich achieved an accuracy of 89.8%(-1.2%), a macro F1 score of 90.3%(-1.0%), and a weighted F1 score of 89.8%(-1.2%). Naïve Bayes algorithm has the lowest accuracy and F1 scores. For kNN, an increase in the number of neighbors affected accuracy scores, adversely. The confusion matrix for the CNN is shown in Figure9.Thematrix shows that each malware has a low number of false positives and false negatives.

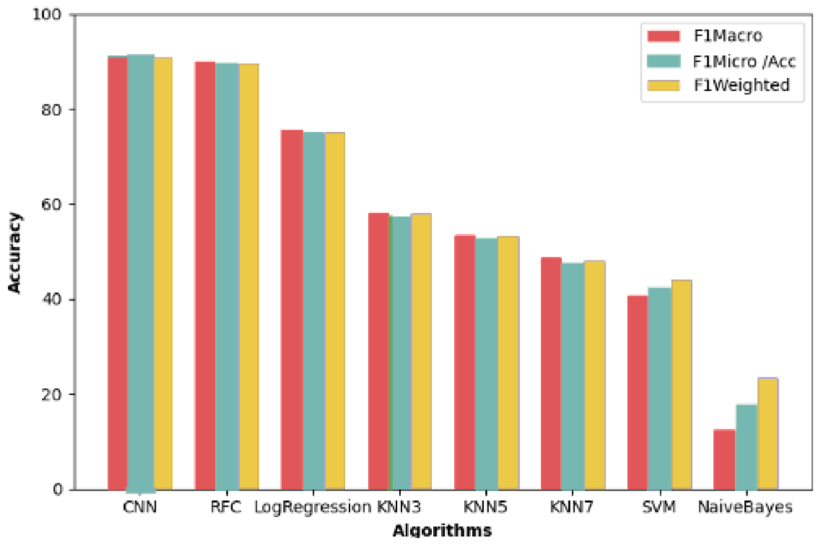


Figure 8. Traditional models compared to our 1-D CNN predicting the types of multiclass malware using TF-IDF vectors

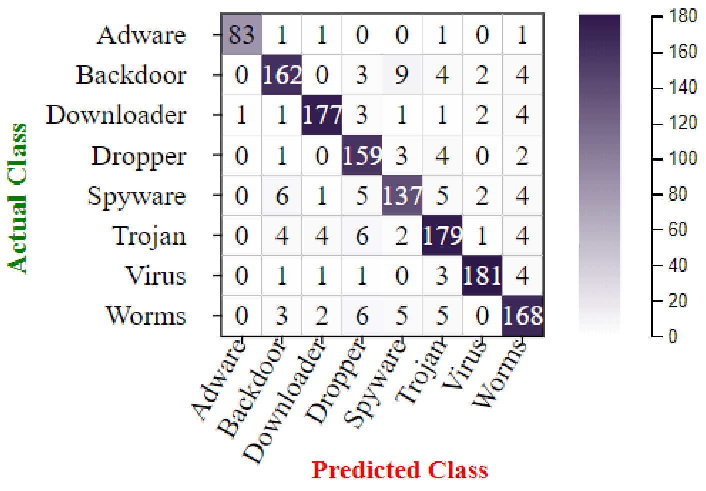


Figure 9. 1-Dimensional convolutional neural network confusion matrix

5.2. TF-IDF Vector

When using TF-IDF feature extraction, CNN outperformed the other classifiers including AdaBoost DT, Random Forest, Bernoulli Naïve Bayes (NB), Multinomial NB, Gaussian NB, Decision Tree Classifier, Bagging Classifier, kNN, and MLP. The proposed 1-D CNN model has achieved an accuracy of 91.0%, a macro F1 score of 91.3%, and a weighted F1 score of 91.0%.This TF-IDF based CNN performed competitively with the previously described CNN, which received API call streams as input directly. This is likely the case as the CNN receiving the API call stream directly can produce a more effective feature map than the TF-IDF feature extraction. The achieved accuracyfor high dimensional categorical and text-based vectors indicate that the proposed 1-D CNN can be used to classify malware in real-world applications.

Of the suite of the decision tree classifiers, AdaBoost performed the best among other algorithms.Figure 10 shows the normalized confusion matrix and area under the receiver operating characteristics (AUC ROC) curve for AdaBoost classifier in multiclass case. MLP has the worst accuracy values, specifically at classification of Adware, Dropper, and Spyware. Table 3summarized these results on class accuracy in detail.

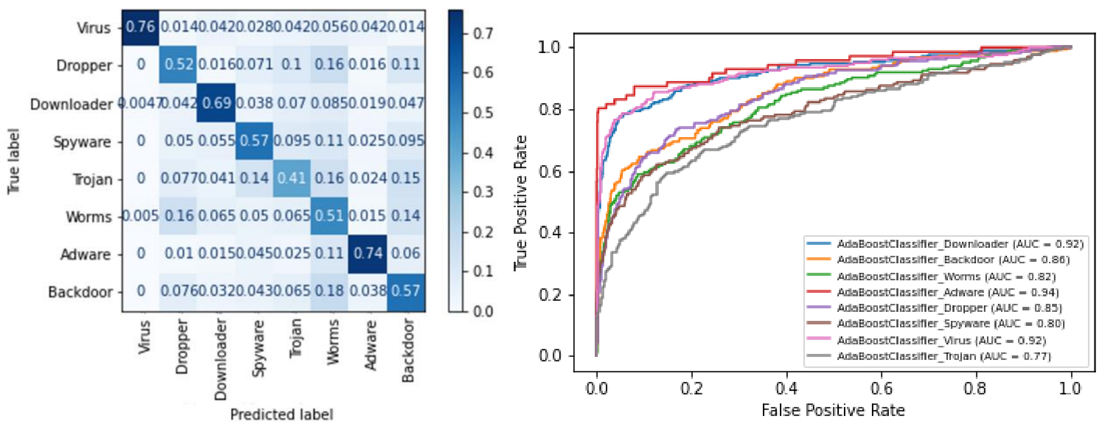


Figure 10. Confusion matrix (left) and AUC ROC curve (right) for AdaBoost Classifier

The diagonal of the confusion matrix in Figure 10 shows the normalized of true positives. According to this, virus, adware, and downloader classes have the highest true positives with 0.76, 0.74, and 0.69 accuracy values, respectively. While the AdaBoost classifier predicted these classes with relatively high accuracy, it predicted trojan, worms, and dropper classes with low accuracy scores. The AUC ROC curve is another performance measurement to assess the performance of a classifier, especially for multiclass problems. True positive rate, in other words, sensitivity, is an outcome where the model correctly predicts the positive class. False positive rate indicates false alarm that falsely rejecting the true instances. The AUC ROC curve is used to evaluate the model according to its ability to distinguish classes. The higher value (close to 1) indicates that the model has better performance to distinguish classes. Figure 10 (right) shows that the AdaBoost classifier has a high ability to distinguish adware, virus, and downloader classes, as seen in the confusion matrix as well.

5.3. Overall Comparison

Table 3 presents the overall comparison in terms of class accuracy for the two approaches of feature extraction. When comparing CNN performance between TF-IDF vector with categorical vector, the highest class accuracy varied depending on the type of malware. The performances are competitive with differences in accuracy ranging from 0.26%-7.32% for eight classes, respectively. The highest accuracy scores are highlighted in bold, and scores in the 2nd rank are underlined for each type of malware. While the proposed CNN classified downloader, dropper, trojan, virus, and worms with the highest accuracy by using categorical vector, it classified adware, backdoor, and spyware with the highest accuracy by adopting TF-IDF vector. Specifically, CNN performed the best at the classification of malware type, suggesting that this form of malware is the most unique relative to its counterparts. Table 3 shows that MLP has the worst accuracy scores, especially for adware, dropper and spyware classes. Among traditional ML algorithms, AdaBoost has the best overall accuracy score. Table 4 shows the binary classification results for TF-IDF vector by adopting one-vs-rest strategy.

Table 3. Performance comparisons on class accuracy scores

	Categorical Vector	**TF-IDF Vector**									
Malware Type	**CNN**	**CNN**	**AdaBoost**	**Bagging**	**BNB**	**DT**	**GNB**	**kNN**	**MLP**	**MNB**	**RF**
Adware	95.40%	**98.17%**	76.00%	48.00%	51.00 %	45.00 %	70.00 %	70.00 %	0%	32.00%	48.00%
Backdoor	88.00%	**88.26%**	52.00%	46.00%	40.00 %	40.00 %	53.00 %	57.00 %	46.00%	52.00%	62.00%
Downloader	**93.20%**	91.14%	69.00%	54.00%	51.00 %	50.00 %	59.00 %	67.00 %	64.00%	64.00%	52.00%
Dropper	**94.10%**	86.78%	57.00%	60.00%	27.00 %	37.00 %	23.00 %	45.00 %	1.50%	29.00%	35.00%
Spyware	85.60%	**88.54%**	41.00%	38.00%	23.00 %	11.00 %	25.00 %	32.00 %	0%	8.30%	17.00%
Trojan	**89.50%**	84.04%	51.00%	45.00%	11.00 %	16.00 %	18.00 %	32.00 %	20.00%	32.00%	16.00%
Virus	**94.80%**	92.26%	74.00%	66.00%	75.00 %	41.00 %	77.00 %	62.00 %	72.00%	63.00%	80.00%
Worms	**88.90%**	87.69%	57.00%	52.00%	28.00 %	78.00 %	34.00 %	49.00 %	29.00%	32.00%	58.00%

Table 4. Performance comparisons on class accuracy scores for binary classification

TF-IDF Vector								
Malware Type	**AdaBoost**	**Bagging**	**BNB**	**DT**	**GNB**	**MLP**	**MNB**	**RF**
Adware vs Rest	**76.00%**	63.00%	59.00%	68.00%	**76.00%**	0%	20.00%	55.00%
Backdoor vs Rest	52.00%	44.00%	**87.00%**	44.00%	86.00%	0%	17.00%	22.00%
Downloader vs Rest	64.00%	57.00%	53.00%	58.00%	**77.00%**	0%	49.00%	49.00%
Dropper vs Rest	43.00%	26.00%	**89.00%**	43.00%	78.00%	0%	9.00%	3.00%
Spyware vs Rest	34.00%	22.00%	56.00%	27.00%	**83.00%**	0%	0%	0.6%
Trojan vs Rest	34.00%	27.00%	51.00%	27.00%	**77.00%**	0%	0.5%	6.00%
Virus vs Rest	66.00%	60.00%	**85.00%**	59.00%	82.00%	3.00%	49.00%	25.00%
Worms vs Rest	49.00%	37.00%	**78.00%**	41.00%	**78.00%**	0%	12.00%	13.00%

As seen in Table 4, Gaussian Naïve Bayes, Bernoulli Naïve Bayes, and AdaBoost algorithms have better accuracy scores compared to other traditional ML algorithms. The best accuracy scores for each malware are marked as bold. On the other hand, MLP, MNB and RF have the worst accuracy scores. Comparing with the results from the multiclass case, GNB and BNB algorithms work well in binary classification problemfor the API calls stream.

6. VISUAL ANALYTICS METHOD

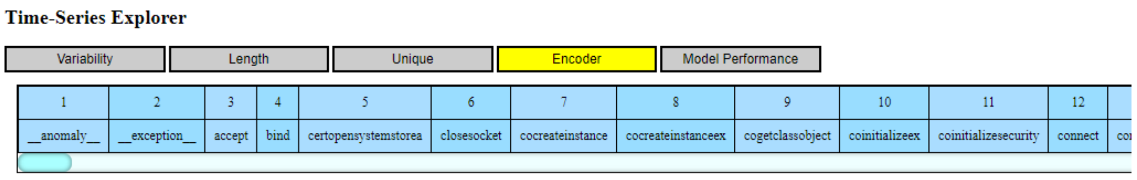
Due to the high dimensional and complex nature of the API call sequence data, visual analytics techniques were used to help understand the data initially. This research used a variety of visual approaches to explore the dataset as well as the computational performance of the analytical model. Four metrics were adopted in the experiment including length, unique instances, varibility and call sequencefor the malware API dataset. Thus, a visual tool that can highlight and compare the metrics for each malware type would help to analyze the dataset initially and assist in the selection of computation model for machine learning. Tableau [22] was used for data exploration and preliminary analysis of feature correlations. D3.js, a JavaScript tool, was used to develop an interactive, web-based visualization platform for exploring the dataset and its computational performance.

6.1. Visual Platform for Malware Feature Exploration

To analyze API call streams, a visual platform using D3 was developed. API calls were mapped as time seriessince they are recorded during malicious files activity. We determined using length, uniqueness, and variability features in investigating malware behaviors and developed the visual representations of these features in the platform. Besides, we created an encoder view to assist in the converting process from API calls to categorical vectors. The encoder view shows API calls in alphabetical order with their categorical value, as seen in Figure 11. The encoder view helps us to see the unique API calls and their assigned integer values. The encoder for the time series data can also be used to inspect any errors.

The distribution of the length, unique instances, and variability metrics is viewed per malware class within theAPI call stream dataset. These metrics are defined as follows:

- Length: Number of entries in a time-series record
- Unique Instances: Number of unique entries in a time-series record
- Variability: Number of instances in a time-series whereentry n differs from entry n + 1, divided bylength of the time-series minus 1.



1	2	3	4	5	6	7	8	9	10	11	12	
__anomaly__	__exception__	accept	bind	certopensystemstorea	closesocket	cocreateinstance	cocreateinstanceex	cogetclassobject	coinitializeex	coinitializesecurity	connect	co

Figure 11. A representation of the encoding map for a given time-series

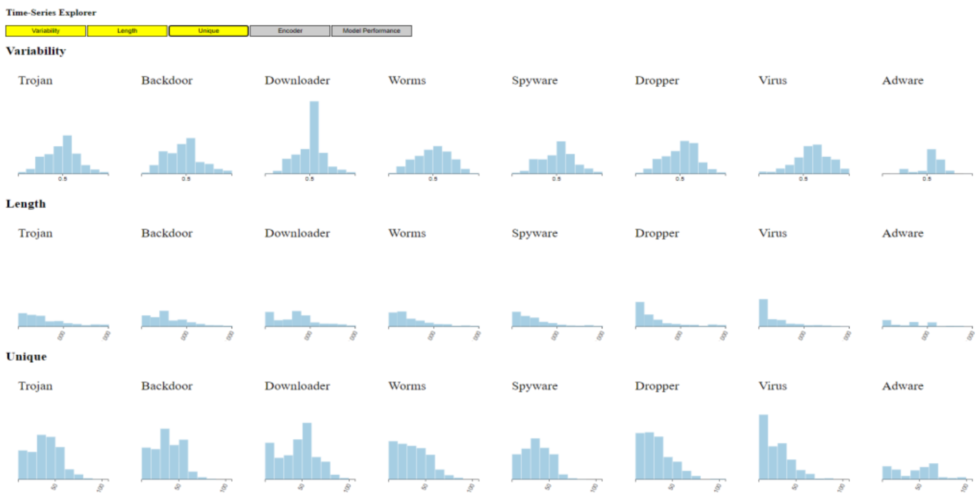


Figure12. The selected metric distributions of malware over the system API calls

Figure12 shows the length, uniqueness, and variability metrics for each malware. These metric buttons can be activated simultaneously. This helps to analyze API calls data comprehensively. Variability metric indicates that downloader (3rd) and adware (8th) have different variability among other types of malware.This justifies that Adware has the best accuracy score with both approaches of feature extractions under CNN because it has a lower than usual variability.We also added Model Performance button to reach the confusion matrix of 1-D CNN. The tool has ability to analyze any time series data with similar structure. In this applicationa user can upload a dataset of time series data and find out how a model performs in classification of each record through our designed metrics. The visualization will help users to view various attributes in determining the model performance.

The platform also includes a 3D view of API call sequences, as seen in Figure 13. This tool allows users to visualize randomly selected data or data from top N samples. The 3D view also provides selection, filter, adding, and removal features to users for further examination. This visual strategy assists in data exploration and pattern search analysis.

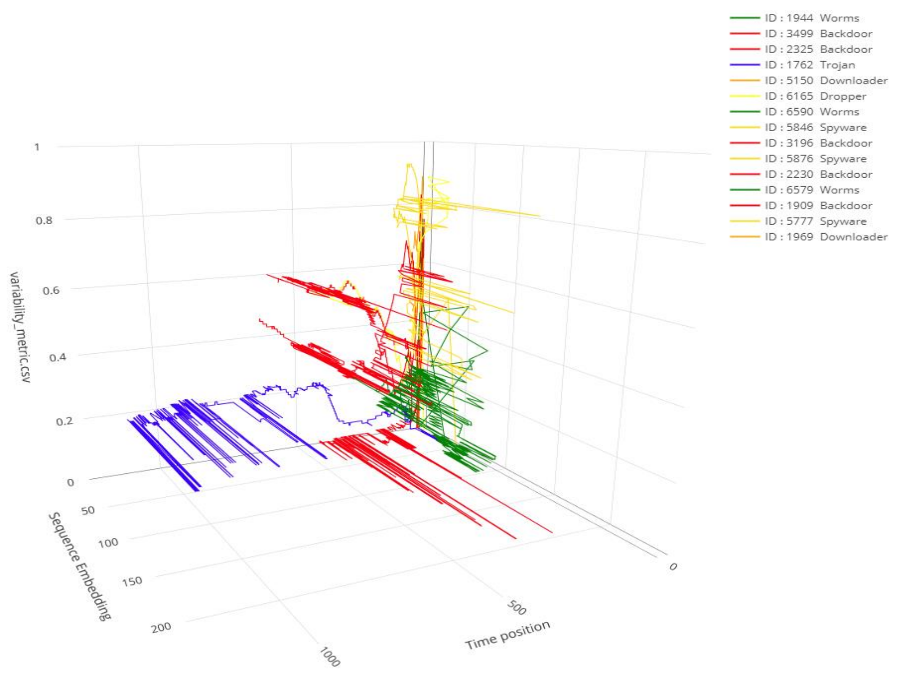


Figure 13. 3-D representation of Sequential API calls

7. CONCLUSION AND DISCUSSION

We utilized two different approachesof feature extraction to classify a malware API dataset at both binary and multiclass levels. In our first approach, we encoded API call streams by converting them to categorical variables. We proposed a 1-D CNN and compared its results with other ML methods includingRandom Forest, Logistic Regression, Support Vector Machine, Naïve Bayes and k-NN. In the second approach, we conducted text-based analysis by using n-grams. We used 10-grams to extract features from API calls and converted them to TD-IDF vectors. A set of classifiers includingAdaBoost DT, Random Forest, Decision Tree, and Bernoulli Naïve Bayes, Multinomial NB, Gaussian NB, Bagging Classifier, kNN, and MLP are used to classify malware based on n-gram analysis. Then the proposed CNN was used to classify the malware using the TD-IDF vectors and the results are compared.

This research has ultimately produced a ConvolutionalNeuralNetwork model that achieved above90% accuracy in classifying the type of malwareusing a malicious program based on its system call stream. For the purpose of identifying and classifying malware, the results of this research demonstrate the advantages of using a CNN to label the type of malware that a malicious system API call stream belongs to. When using the categorical vector, the proposed CNN outperformedRF, LR, SVM, and k-NN. While using TF-IDF vector, CNN also achieved a higher accuracy thanAdaBoostDT, Random Forest, Bernoulli Naïve Bayes (NB), Multinomial NB, Gaussian NB, Decision Tree Classifier, Bagging Classifier, kNN, and MLP. While the second-best performing method is RF for categorical vector analysis, AdaBoost has the second-best accuracy score for the TF-IDF vector. Overall comparisons indicate that both feature extraction approaches produced competitive performance on classification of malware. Depending on the type of malware, the class accuracy varies between 0.26%-7.32%, respectively between using a categorical vector input or TF-IDF vector input to a CNN.The results also suggest that the high impact feature in an API call stream executed by a malware is related to both call variability and call sequence.

We also demonstrate a visual analysis platform for time-series data to assist our machine learning model choices. The D3 visualization tool aided in human understandingof the API call stream data by visualizing both encoder view and generated features, i.e., length, uniqueness, and variability. The distributions of the variability,uniqueness,andlengthofcallstreamscan be inspected using the visualizer. The tool also helps other time series formatted data as well.3-D sequential data visualizer also allows us to see API calls pattern over time by examining through filtering and sampling functions.

Future work includes expanding the length of call streamsand including non-malicious programs for classification. An anti-virus system can adapt the model to aid in attack attribution by quickly gaining an understanding as to the type of malware it is dealing with. Finally, a goal for improving the visual analytic process of this research is to use the correlations identified in the D3 tool to adjust the computational model so that it can be more successful in differentiating between similar classes.

8. ACKNOWLEDGEMENT

This project is partially supported by ASRC Federal Inc.

REFERENCES

[1] Daniel Gibert, Carles Mateu, & Jordi Planes, (2020) “The rise of machine learning for detection and classification of malware: Research developments, trends and challenges”, Journal of Network and Computer Applications. 10.1016/j.jnca.2019.102526.

[2] Zahra Bazrafshan, Hashem Hashemi, Fard Hazrati, Mehdi Seyed, & Ali Hamzeh, (2013) “A survey on heuristic malware detection techniques”, 2013 5th Conference on Information and Knowledge Technology. 113-120. 10.1109/IKT.2013.6620049.

[3] Jyoti Landage, & M. P. Wankhade, (2013) “Malware and Malware Detection Techniques : A Survey”, International journal of engineering research and technology, 2.

[4] DainiusCeponis, & Nikolaj Goranin,(2019) “Evaluation of Deep Learning Methods Efficiency for Malicious and Benign System Calls Classification on the AWSCTD”,Security and Communication Networks,2317976:1-2317976:12.

[5] SerifBahtiyar, Mehmet BarisYaman, & Can Yilmaz Altinigne, (2019)“A multi-dimensional machine learning approach to predict advanced malware”, Comput. Networks, 160,118-129.

[6] GyuwanKim, Hayoon Yi, JanghoLee, YunheungPaek, & Sungroh Yoon, (2016) “LSTM-Based System-Call Language Modeling and Robust Ensemble Method for Designing Host-Based Intrusion Detection Systems”, ArXiv, abs/1611.01726.

[7] AhmetYazi, Ferhat Ozgur Catak,& EnsarGul,(2019) “Classification of Methamorphic Malware with Deep Learning (LSTM)”,10.1109/SIU.2019.8806571.

[8] Ferhat OzgurCatak,&AhmetYazi,(2019) “A Benchmark API Call Dataset for Windows PE MalwareClassification”, https://arxiv.org/abs/1905.01999.

[9] EslamAmer,&Ivan Zelinka,(2020) “A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence”, Computers & Security. 10.1016/j.cose.2020.101760.

[10] YuntaoZhao, Bo Bo, Yongxin Feng, ChunYu Xu, & Bo Yu,(2019) “A feature extraction method of hybrid gram for malicious behavior based on machine learning”, Secur. Commun. Netw.

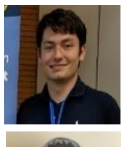
[11] Chang Choi, ChristianEsposito, MungyuLee, & JunhoChoi, (2019) “Metamorphic malicious code behavior detection using probabilistic inference methods”, Cognit. Syst. Res. 56, 142–150.

[12] AsgharTajoddin, & SaeedJalili, (2018) “HM3alD: polymorphic Malware detection using program behavior-aware hidden Markov model”, Appl. Sci. 8 (7), 1044.

[13] Matthew Schofield, Gulsum Alicioglu, Russell Binaco, Paul Turner, Cameron Thatcher, Alex Lam & Bo Sun, (2021) “Convolutional Neural Network For Malware Classification Based On API Call Sequence”, In proceedings of 2021 the 14th International Conference on Network Security & Applications. Computer Science & Information Technology (CS & IT). Zurich, Switzerland.

[14] Jeffrey Heer, Micheal Bostock, & Vadim Ogievetsky,(2010) "A Tour through the Visualization Zoo", ACM Queue, 8, 20.
[15] WeijieHan, Jingfeng Xue, YongWang, LuHuang, ZixiaoKong, & Limin Mao, (2019) "MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics", Comput. Secur., 83, 208-233.
[16] LuXiao-Feng, ZhouXiao, Jiang Fangshuo, Yi Sheng-wei,&ShaJing,(2018) "ASSCA: API based Sequence and Statistics featuresCombinedmalwaredetectionArchitecture",Procedia Computer Science, 129, 248-256.
[17] MatildaRhode, Pete Burnap, & Kevin Jones, (2018) "Early Stage Malware Prediction Using Recurrent Neural Networks",Comput. Secur., 77,578-594.
[18] ZahraSalehi, Ashkan Sami, & Mahboobe Ghiasi, (2017) "MAAR: Robust features to detect malicious activity based on API calls, their arguments and return values", Eng. Appl. Artif. Intell., 59, 93-102.
[19] MohamedBelaoued, & SmaineMazouzi, (2016) "A Chi-Square-Based Decision for Real-Time Malware Detection Using PE-File Features", JIPS, 12,644-660.
[20] Sanchit Gupta, Harshit Sharma, & Sarvjeet Kaur, (2016) "Malware Characterization Using Windows API Call Sequences",SPACE.
[21] Jixin Zhang, Zheng Qin, Hui Yin, Lu Ou, & Kehuan Zhang, (2019) "A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding", Comput. Secur., 84,376-392.
[22] Tableau Software. (2020). Retrieved from www.tableau.com.
[23] Kolosnjaji Bojan, Zarras Apostolis, Webster George, & Eckert Claudia, (2016) "Deep Learning for Classification of Malware System Call Sequences", In: Kang B., Bai Q. (eds) AI 2016: Advances in Artificial Intelligence. Lecture Notes in Computer Science, vol 9992. Springer, Cham. https://doi.org/10.1007/978-3-319-50127-7_11.
[24] Catak Ferhat Ozgur, Yazı Ahmet Faruk, Elezaj Ogerta & Ahmed Javed, (2020) "Deep learning based Sequential model for malware analysis using Windows exe API Calls", PeerJ Computer Science 6:e285 https://doi.org/10.7717/peerj-cs.285.
[25] Albawi Saad, Mohammad Tareq Abed, & Al-Zawi Saad, (2017), "Understanding of a convolutional neural network", 2017 International Conference on Engineering and Technology (ICET), Antalya, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.
[26] "http://alexlenail.me/NN-SVG," 2016. (Accessed 20 December 2020).
[27] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, & Stephen Marshall, (2018) "Activation Functions: Comparison of trends in Practice and Research for Deep Learning", ArXiv, abs/1811.03378.
[28] Yinzheng Gu, Chuanpeng Li, & Jinbin Xie, (2018) "Attention-aware Generalized Mean Pooling for Image Retrieval", ArXiv, abs/1811.00202.
[29] Mark Cheung, John Shi, Lavender Jiang, Oren Wright, &Jose Moura, (2019) "Pooling in Graph Convolutional Neural Networks", 53rd Asilomar Conference on Signals, Systems, and Computers, 462-466.
[30] WilliamCavnar, & John Trenkle, (1994) "N-gram-based text categorization", Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval. Vol. 161175.
[31] Raymond Canzanese, Spiros Mancoridis, &Moshe Kam, (2015) "Run-time classification of malicious processes using system call analysis", 10th International Conference on Malicious and Unwanted Software (MALWARE), Fajardo, 2015, pp. 21-28.
[32] ShahzadQaiser, & Ramsha Ali, (2018) "Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents", International Journal of Computer Applications, 181, 25-29.

Authors



Matthew Schofield is currently enrolled at Rowan University pursuing his B.S/M.S degree in Computer Science anticipating graduation in December 2021. He is currently working on his master's thesis on Deep Reinforcement Learning in Incentivization Systems. His research interests are in Machine Learning and Deep Reinforcement Learning.

GulsumAlicioglu received M.Sc. Degree in Industrial Engineering from Gazi University, Turkey, in 2018. Currently, she is a Ph.D. candidate at the Department of Electrical and

Computer Engineering of Rowan University, USA. Her research interests aredata visualization, machine learning, and explainable artificial intelligence.

Bo Sun is an associate professor of Computer Science and led the project effort of this paper.She received her B.S. in Computer Science from Wuhan University, her M.S.in Computer Science from Lamar University, and her Ph.D. in Modeling and Simulation from Old Dominion University. Her research interests include Visual Analytics and Data Visualization.

Russell Binaco graduated from Rowan University with an M.S. in Computer Science in Spring 2020. He now works as a software engineer for Innovative Defense Technologies, and as an adjunct for Rowan University. At Rowan, he earned undergraduate degrees in Computer Science and Electrical and Computer Engineering. He has also been published in the Journal of the International Neuropsychological Society for research using Machine Learning to classify patients' levels of cognitive decline with regards to Alzheimer's Disease.

Paul Turner received his B.S. in Computer Science from Rowan University in 2018 and is currently enrolled in an M.S. program at the aforementioned University. His interests include machine learning, text mining, and cloud computing.

Cameron Thatcher received his B.S in Computer Science from Rowan University in 2019 and is currently pursuing his M.S. in Computer Science at Rowan University. His research interests include Machine Learning and Data Mining.

Alex Lam is currently attending Rowan University pursuing his B.S/M.S degree in Computer Science and Data Analytics. He has also been published in the 3rd ACM SIGSPATIAL International Workshop on Analytics for Local Events and News (LENS'19) for research in identifying real-world events using bike-sharing data.

Anthony Breitzman holds an M.A. in Mathematics from Temple University, and an M.S. and Ph.D. from Drexel University. He is an associate professor of Computer Science at Rowan University and his research interests are Data Mining, Text Mining, Machine Learning, Algorithm Design, Convolution Algorithms, and Number Theory.