Malware Detection Using Ensemble N-gram Opcode Sequences

https://doi.org/10.3991/ijim.v15i24.25401

Paul Ntim Yeboah[1(✉)], Stephen Kweku Amuquandoh[2], Haruna Balle Baz Musah[1]
[1]Ghana-India Kofi Annan Centre of Excellence in ICT, Accra, Ghana
[2]Kwame Nkrumah University of Science and Technology, Kumasi, Ghana
paul.y.ntim@gmail.com

Abstract—Conventional approaches to tackling malware attacks have proven to be futile at detecting never-before-seen (zero-day) malware. Research however has shown that zero-day malicious files are mostly semantic-preserving variants of already existing malware, which are generated via obfuscation methods. In this paper we propose and evaluate a machine learning based malware detection model using ensemble approach. We employ a strategy of ensemble where multiple feature sets generated from different n-gram sizes of opcode sequences are trained using a single classifier. Model predictions on the trained multi feature sets are weighted and combined on average to make a final verdict on whether a binary file is malicious or benign. To obtain optimal weight combination for the ensemble feature sets, we applied a grid search on a set of pre-defined weights in the range 0 to 1. With a balanced dataset of 2000 samples, an ensemble of n-gram opcode sequences of *n* sizes 1 and 2 with respective weight pair 0.3 and 0.7 yielded the best detection accuracy of 98.1% using random forest (RF) classifier. Ensemble n-gram sizes 2 and 3 obtained 99.7% as best precision using weight 0.5 for both models.

Keywords—malware detection, N-gram, opcode, machine learning, ensemble, grid search

1 Introduction

The surge in malware attacks has become a major threat to internet security. Proliferation in malware attacks could be attributed to the high profit incentives derived from these illicit breaches [1, 2]. A cyber threat report by SonicWall [3] shows that out of the millions of detection engines deployed worldwide, a total of 9.9 billion malware attacks were recorded in 2019 with over 440,000 malware variants. In 2020 SonicWall reported a total of 5.6 billion malware attacks, which is obviously a decline from the previous year. This emerging threat calls for a more sophisticated solution. The signature based method has been the conventional approach for malware detection. With this approach, malware footprint including byte sequences, hashes or anomalies are precomputed and used as a repository for future queries for suspicious files.

Signature-based detection methods face two major drawbacks; first, manual examination of executables and the requirement for regular update of detection engines has become unrealistic due to the volumes of malware released each day. Secondly, malware creators employ obfuscation techniques to generate semantic-preserving malware variants which easily circumvent detection [4, 5]. As described by authors in [5, 6], obfuscation by metamorphism inhibit detection on traditional anti-malware with strategies including variable renaming; changing names of variables, dead-code insertion; inserting sequences like no operation (NOP) instructions, code transposition; rearranging the order of instructions, etc.

Recent breakthroughs in machine learning (ML) in areas such as natural language processing and computer vision have inspired researchers to explore data-driven approach in malware detection. Reports have shown that machine learning based malware detection models could be a better alternative to the traditional methods considering promising results reported in literature including [7, 8, 9, 10]. Such machine learning techniques mostly employ static and dynamic analysis to obtain useful discriminative features to build models to detect malware. In the static analysis, binary file components like raw hexadecimal bytes, operation codes (opcodes), text strings and control flow graph are extracted without the binary being executed [7, 8, 10, 11, 12]. On the other hand, the dynamic approach executes the binary in a controlled environment to collect features like API call traces, network-related information, memory and register usage, etc. [13, 14, 15]. While static analysis could be undermined by obfuscation, dynamic analysis is proven to be resilient against heavily packed malware but could be time consuming.

Various machine learning models with static opcode as input features for malware detection have been proposed [7, 8, 16]. Santos et al. [7] were one of the first to propose the use of opcode sequences for malware detection. In their approach they performed feature selection with information gain and obtained top 1000 n-gram opcode sequences and applied various machine learning classifiers to achieve a performance of 95.9% accuracy with SVM on n-gram of size 2. Quite recently, Manavi et al. [16] demonstrated an image processing method for malware detection using opcode graph and ensemble of SVM and k-nearest neighbour (KNN) models.

In this research, we contribute to existing literature on malware detection with our proposed weighted average ensemble model consisting of natural language processing (NLP) techniques and classical machine learning algorithms. Our major contribution is shown in how our ensemble strategy is implemented. Unlike other ensemble techniques like [17, 18] where models of multiple machine learning classifiers are trained on the same feature set, our proposed ensemble strategy instead trains a single classifier on multiple feature sets obtained from different n-gram sizes of opcode sequences. We employ n-gram a NLP [19] technique to reinforce contextual meaning in opcode sequences. The main contributions of this research are:

- To propose machine learning based malware detection model consisting ensemble of n-gram opcode sequences.
- To evaluate the proposed model and find the optimal *n* sizes of ensemble n-gram opcode sequences that produce the best detection accuracy.

2 Related work

For decades, anti-malware creators have relied on signature based methods which quite recently has been proven ineffective against zero-day malware. Breakthroughs in machine learning however have inspired researchers to explore data driven approaches for the task of malware detection and classification [7, 8, 9, 10]. Applying ML for malware detection requires optimal discriminative and representative features to be obtained from binary files. Static and dynamic analysis has become the *de facto* methods for generating representations for executables. The former extracts without executing the binary file components including opcodes, text strings and control flow graph [8, 10, 11, 20], whereas the latter executes the binary to extract representation features like API call traces, network-related information, memory and register usage, etc. [13, 14, 20]. While both modes of feature generation are recommended, static analysis has the limitation of been susceptible to obfuscation. Dynamic analysis on the other hand can be time consuming but does better against obfuscation since the binary is executed for behavioural analysis.

A malware detection and classification research by Fuyong et al. [11] employed static features comprising of raw bytes of binary files and generated n-grams of the byte code attributes which was used as the basis to compare similarity between a test sample and malware or benign files. Vinayakumar et al. [13] proposed malware detection frameworks using static and dynamic extracted features. In their work, different static features including raw byte histogram, byte entropy histogram and strings were used to train classical machine learning and deep neural network models. They obtained 97.0% and 98.9% as best accuracies using random forest (RF) and deep neural network (DNN) respectively. In addition they proposed a dynamic malware detection approach where a Cuckoo sandbox was employed to collect machine activity data of executed binary samples and further used the extracted features to train shallow and deep machine learning models. A behavior-based malware detection model presented by Galal et al. [14] employed API sequences invoked from executed binaries and trained classification algorithms including RF, decision tree (DT) and support vector machine (SVM), which resulted accuracies of 96.89%, 96.14% and 94.8% respectively.

There have been several proposals for malware detection with ML methods using static instruction sequences representations [7, 8, 16, 21]. The work by Santos et al. [7] have shown that machine learning-based malware detection models could be used as a complement to signature-based engines. In their work, instruction codes were proposed as representation for executable binaries for the detection of malware. Authors of [8] also applied different classical ML models for malware categorization using n-gram opcode sequences and recorded an f-measure of 98% as best results using SVM. Similarly, Ni et al. [10] and Lu [21] have shown promising results of 99.26% and 97.87% accuracies respectively on malware classification with deep learning models trained on opcode sequences.

3 Proposed methodology

The methodology to the proposed ensemble model is divided into three main parts: feature extraction, feature selection and ensemble classification model. At the feature

extraction phase, we disassemble the binary file into assembly code (Bin2Asm) and extract discriminative components of the disassembled portions which serve as an indicator of maliciousness. We extract multiple sets of features using n-gram with different n sizes. Afterwards, we reduce the dimensions of the extracted features by selecting optimal n-gram features with most substantive information on the verdict on whether a binary is malware or goodware. Finally, these selected features are used to model an ensemble classification, where model predictions on the trained multi feature sets are weighted and combined on average to detect malware. This procedure is shown in Figure 1.

3.1 Feature extraction

Static and dynamic analyses are the major techniques that can be leveraged to extract features to represent binary files. In this paper we represent binaries by static opcodes. We obtain opcodes from executables by a disassembler. Each file sample is converted into respective assembly versions using the Radare2 disassembler[1] and all the opcodes presented in the disassembled file are extracted as a single sequence. Further, we generate n-grams from the opcode sequences by sliding a window of size *n* across the opcode sequences as shown in Figure 2.

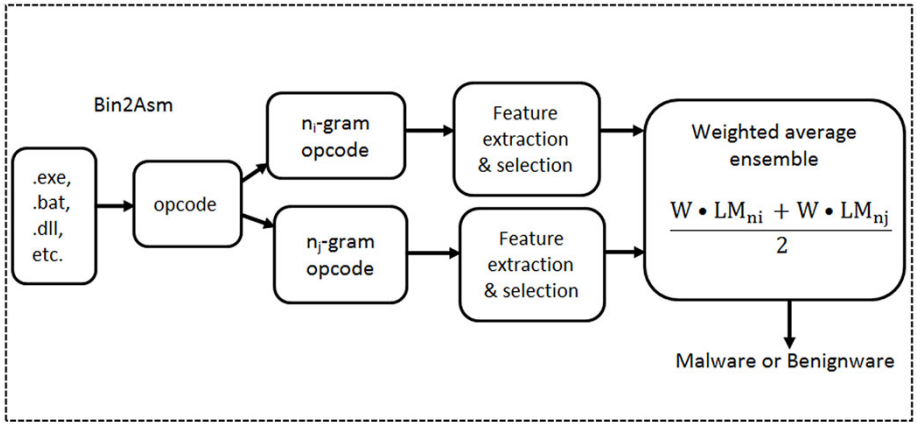

Fig. 1. Proposed ensemble model for malware detection. *ni* and *nj* are used to generate n-gram opcode sequences with different *n* sizes. LM is a single learning classifier (e.g., SVM). Weights (W) are values within the range of 0 to 1 found using a grid search approach

Applying different n-gram sizes on static and dynamic feature representations of binaries have yielded remarkable improvement in performance on ML-based malware detection models [9, 22, 23]. Kang et al. [8] for example achieved an f-measure of 98% on their android malware detection model using n-gram opcode of n size 4. Moskovitch et al. [24] experimental results with n-gram of size 2 outperformed all other n sizes on malware detection with opcode representation. This has been our inspiration for the

[1] https://rada.re/n/.

proposed malware detection model using ensemble n-gram opcode sequences. In this research, we consider n-grams of sizes ranging from 1 to 4 and generate multiple feature sets of opcode sequences for the different *n* sizes.

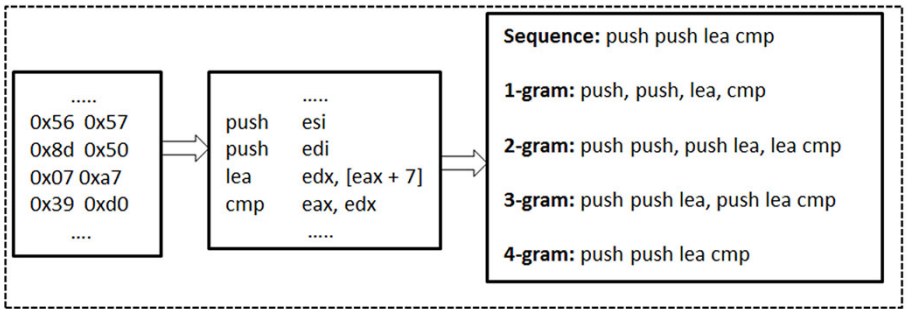

Fig. 2. Example of n-gram opcode sequences generation

In Table 1, we show statistics on the number of unique n-gram opcodes obtained from our dataset, which has a total of 2000 benign and malware samples. Our findings attest to other research [7, 8, 25] that the number of unique n-grams increases proportionally to the size of *n*.

Table 1. Statistics on unique n-gram opcodes for varying *n* values

N	Benign	Malware
1	327	503
2	7390	25496
3	35840	317853
4	93791	740787
5	183002	896491
6	297106	933565
7	427357	950387
8	563455	962821

Since machine learning classifiers only understand features in numerical representations, we vectorize each sample's n-gram opcode sequences using the term frequency-inverse document frequency (TF-IDF) [26, 27]. TF-IDF works by creating a dictionary of unique n-gram opcode sequences and then measures the frequency of occurrence of each unique n-gram opcode within a given sample using the term frequency (TF) and with inverse document frequency (IDF), measures the importance of the unique n-gram opcode on the basis of frequency of occurrence across the entire corpus. Let $D = \{d_1, d_2, d_3, \ldots, d_n\}$ be a set of documents for *n* number of disassembled file samples, and let $d = \{t_1, t_2, t_3, \ldots, t_m\}$ be the output of a disassembled sample, where *m* is the number of n-grams in *d*. The term frequency $TF(t, d)$ as shown in equation (1), computes the frequency of occurrence of *t* (i.e., n-gram opcode) in *d*:

$$TF(t,d) = \frac{count(t)}{|d|} \tag{1}$$

Using equation (2), the measure of importance of n-gram opcode *t* across the entire documents *D* (in our case a corpus of binary samples) is derived using the inverse document frequency *IDF*(*t*, *d*):

$$IDF(t,d) = \log \frac{|D|}{1+|\{d \in D : t \in d\}|} \tag{2}$$

Finally, the true importance of n-gram opcode *t* to a disassembled output *d* in corpus of samples *D* is obtained using TF-IDF which is the product of *TF*(*t*, *d*) and *IDF*(*t*, *D*).

3.2 Feature selection

From statistics on Table 1, we generated a huge number of unique n-gram opcodes from our dataset especially from n-gram sizes 2 upwards. This huge vocabulary size is expected to reflect in the final vector representations obtained from the TF-IDF model. Training a machine learning model on the entire vocabulary may not be the ideal approach since not all the n-gram opcodes may carry substantial information for detection of malware. We therefore apply feature selection to reduce the dimensions of the TF-IDF vector representations and remove less important n-gram opcode features.

To select optimal features, we leverage on information gain [28] to filter out less useful features while we keep the top 1000 most informative features. However, for n = 1, since the total unique n-gram opcodes is less than 1000, we skipped feature selection for single opcode feature. We measure information gain for the n-gram opcode features by entropy [29], which is the measure in the level of uncertainty in a random variable. With entropy, we are able to ascertain the measure in reduction of uncertainty of a class variable (in our case malware or goodware) given a feature variable (i.e., n-gram opcodes).

Information gain (IG) as shown in equation (3) can be measured as the rate of reduction in entropy of a class variable *N* as a result of information provided by feature variable *M* about *N*.

$$IG(N \mid M) = H(N) - H(N \mid M) \tag{3}$$

Here, *H*(*N*) is the entropy of the class variable *N* and $H(N \mid M)$ is the entropy of class variable *N* after observing variable *M*. For two given feature variables *M* and *Q*, feature *M* is regarded as more useful indicator than feature *Q* for a class *G* if $IG(N \mid M) > IG(N \mid Q)$.

4 Ensemble classification model

Research have shown that results of single classifier ML-based models could be reinforced using a hybrid scheme [17, 18, 30]. This hybrid approach also known as ensemble learning enhances results of a single model by fusion numerous classifiers. Common ensemble strategies consist of multiple classifiers trained either on a single or multiple feature sets. The benefit with multi feature sets is that, the ensemble model has the advantage of building a more in-depth discriminative characteristic about the samples. In this study, we employ the ensemble strategy where multiple feature sets generated from different n-gram sizes of opcode sequences are trained using a single classifier.

As shown in equation (4), individual predictions of the single classifier *LM* trained on multi features *ni*-gram opcode and *nj*-gram opcode sequences are weighted and merged on average to produce a final predicted class y .

$$y = argmax(avg(W \cdot y(LM_{ni}), W \cdot y(LM_{nj}))) \tag{4}$$

Here, each model LM_{ni} and LM_{nj} predicts a vector of probabilities, with one probability for each class (i.e., malware or benignware). The predicted probabilities for each model are weighted with weights W values obtained from grid search on a pre-defined range between 0 to 1. Subsequently the argmax function [31] is applied to the weighted average of the predictions to obtain the index of the highest vector value which is the final predicted class (i.e., 0=benignware and 1=malware).

We trained and tested the proposed ensemble model using three different machine learning classification algorithms including: support vector machine (SVM) [32], random forest (RF) [33] and K-nearest neighbour (KNN) [34]. We leveraged the scikit-learn [35] implementation of these classifiers.

5 Dataset generation

Our dataset consists of a balance of 2000 malware and benign windows portable executable (PE) files obtained from a research project by Tuan et al. [36]. The original dataset contains 1000 goodware and 8970 malware, where the malware dataset comprises 5 different malware categories. All the malware samples were collected from a combination of virusshare [37] and malicia-project [38] and the benign binaries downloaded from [39]. Dataset was duly verified with VirusTotal [40]. To generate a balanced dataset, the malware dataset was downsampled by a random selection of 200 samples from each malware category to compose the 1000 malware, however, no sampling was applied to the benign samples.

6 Evaluation metrics

The efficacy of our proposed ensemble malware detection model was measured on the basis of *total accuracy* (shown in equation (5)), which is the number of correct predictions divided by all predictions made and *precision* (shown in equation (6)), which is the proportion of positive predictions correctly identified.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

Where, true positive (TP) is the number of malware samples correctly identified, true negative (TN) is the total benign samples correctly classified, false positive (FP) is the number of benign binaries wrongly identified as malware and false negative (FN) is the number of malware misclassified as legitimate.

7 Experimental results and discussion

For our ensemble model, we trained a single classifier using two separate feature sets obtained from different n-gram sizes (n_i-gram and n_j-gram) of opcode sequences, where n_i and n_j could be any combination pair in the range 1 to 4. In order to derive optimal weight combination W_{ni} and W_{nj} for the ensemble n-gram opcode feature sets, we applied a grid search on a set of pre-defined weights in the range 0 to 1.

To evaluate performance of the proposed ensemble model, we adopted the *K-fold cross validation* which allows partitioning our dataset into *K* subsets and preforming *K* different rounds of learning and testing. We performed cross-validation with $K = 5$ for all ensemble models. Accuracy (acc) and precision (precc) scores for each round was averaged to obtain the global performance for each tested ensemble model.

As shown on Table 2, the overall best results in terms of accuracy was obtained by RF trained with gini criteria, which yielded an accuracy of 98.1% using ensemble n-gram opcode sizes 1 and 2 with weight pair 0.3 and 0.7 respectively. The overall precision best score was 99.7%, which was obtained with RF-gini using n-gram sizes 2 and 3. The other classifiers also obtained accuracies greater than 95%. SVM trained on rbf kernel yielded 97% as the best accuracy for models trained with SVM using ensemble n-gram sizes 1 and 3 with weight pair 0.6 and 0.4 respectively, and the best precision score of 96.7% using n-grams 1 and 2 with respective weights 0.6 and 0.4. Similarly, ensemble models trained with KNN with k neighbors=5 recorded best accuracy of 98% and precision of 98.4% using n-gram sizes 1 and 2 with respective weights 0.4 and 0.6. Generally, our model recorded higher precision scores as compared to accuracy. The higher precision score is an indication that our model performed better at classifying malware samples correctly, thus, the model has a minimal false positive alarm rate. On the other hand, the relatively lower accuracy scores can be attributed to the false negatives (i.e., the number of malicious samples misclassified as legitimate).

Finally, in Figure 3, we show that the overall best performing model in terms of accuracy obtained a mean area under curve (AUC) score of 1.

Table 2. Results for different ensemble n-gram opcode sequences with different combinations of *n* sizes

Classifier	W_{ni}	W_{nj}	Metric	n_i=1, n_j=2	n_i=1, n_j=3	n_i=1, n_j=4	n_i=2, n_j=3	n_i=2, n_j=4	n_i=3, n_j=4
SVM (RBF)	**0.1**	**0.9**	Acc	0.965	0.966	0.955	0.967	0.955	0.954
			Precc	0.961	0.960	0.948	0.963	0.949	0.948
	0.2	**0.8**	Acc	0.965	0.967	0.956	0.965	0.957	0.957
			Precc	0.959	0.960	0.950	0.959	0.952	0.950
	0.3	**0.7**	Acc	0.965	0.967	0.958	0.964	0.959	0.958
			Precc	0.961	0.961	0.951	0.954	0.955	0.952
	0.4	**0.6**	Acc	0.966	0.968	0.958	0.962	0.959	0.959
			Precc	0.961	0.963	0.949	0.947	0.955	0.953
	0.5	**0.5**	Acc	0.966	0.969	0.960	0.960	0.965	0.962
			Precc	0.965	0.961	0.953	0.941	0.962	0.957
	0.6	**0.4**	Acc	0.967	**0.970**	0.963	0.615	0.964	0.965
			Precc	**0.967**	0.966	0.958	0.610	0.960	0.959
	0.7	**0.3**	Acc	0.965	0.965	0.960	0.528	0.964	0.965
			Precc	0.963	0.963	0.956	0.528	0.960	0.960
	0.8	**0.2**	Acc	0.960	0.960	0.957	0.528	0.964	0.965
			Precc	0.957	0.955	0.950	0.528	0.961	0.961
	0.9	**0.1**	Acc	0.959	0.959	0.958	0.528	0.964	0.966
			Precc	0.953	0.953	0.952	0.528	0.960	0.963
KNN (K=5)	**0.1**	**0.9**	Acc	0.978	0.947	0.920	0.947	0.920	0.920
			Precc	0.983	0.929	0.989	0.929	0.894	0.894
	0.2	**0.8**	Acc	0.978	0.952	0.935	0.938	0.935	0.920
			Precc	0.983	0.938	0.909	0.914	0.911	0.894
	0.3	**0.7**	Acc	0.979	0.958	0.936	0.932	0.938	0.915
			Precc	0.984	0.945	0.911	0.900	0.913	0.884
	0.4	**0.6**	Acc	**0.980**	0.962	0.950	0.926	0.949	0.915
			Precc	**0.984**	0.952	0.934	0.889	0.932	0.884
	0.5	**0.5**	Acc	0.980	0.969	0.958	0.927	0.954	0.917
			Precc	0.983	0.965	0.952	0.893	0.947	0.892
	0.6	**0.4**	Acc	0.980	0.967	0.958	0.766	0.954	0.908
			Precc	0.982	0.962	0.947	0.745	0.942	0.875
	0.7	**0.3**	Acc	0.979	0.964	0.960	0.613	0.954	0.758
			Precc	0.982	0.961	0.954	0.613	0.946	0.750
	0.8	**0.2**	Acc	0.980	0.959	0.958	0.611	0.956	0.754
			Precc	0.982	0.961	0.956	0.621	0.954	0.765
	0.9	**0.1**	Acc	0.979	0.956	0.956	0.519	0.955	0.497
			Precc	0.982	0.956	0.956	0.424	0.955	0.313

(Continued)

Table 2. Results for different ensemble n-gram opcode sequences with different combinations of *n* sizes *(Continued)*

Classifier	W_{ni}	W_{nj}	Metric	n_i=1, n_j=2	n_i=1, n_j=3	n_i=1, n_j=4	n_i=2, n_j=3	n_i=2, n_j=4	n_i=3, n_j=4
RF(Gini)	**0.1**	**0.9**	Acc Precc	0.980 0.984	0.977 0.983	0.973 0.973	0.974 0.980	0.973 0.975	0.973 0.976
	0.2	**0.8**	Acc Precc	0.980 0.984	0.978 0.983	0.974 0.975	0.977 0.987	0.974 0.976	0.973 0.976
	0.3	**0.7**	Acc Precc	**0.981** **0.984**	0.979 0.983	0.976 0.979	0.975 0.992	0.978 0.981	0.977 0.982
	0.4	**0.6**	Acc Precc	0.980 0.983	0.979 0.984	0.977 0.983	0.969 0.994	0.978 0.984	0.978 0.986
	0.5	**0.5**	Acc Precc	0.981 0.983	0.979 0.984	0.977 0.983	0.958 **0.997**	0.978 0.983	0.977 0.985
	0.6	**0.4**	Acc Precc	0.980 0.982	0.979 0.984	0.979 0.983	0.739 0.589	0.979 0.984	0.978 0.984
	0.7	**0.3**	Acc Precc	0.980 0.982	0.979 0.982	0.979 0.983	0.626 0.400	0.979 0.984	0.979 0.984
	0.8	**0.2**	Acc Precc	0.980 0.982	0.978 0.982	0.980 0.983	0.471 0.00	0.979 0.984	0.978 0.984
	0.9	**0.1**	Acc Precc	0.979 0.982	0.978 0.982	0.978 0.982	0.471 0.00	0.979 0.984	0.978 0.984

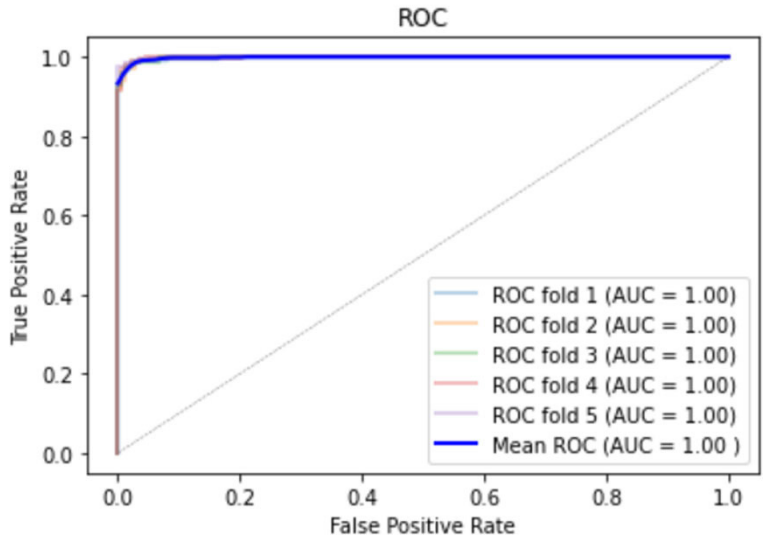

Fig. 3. Receiver operating characteristic (ROC) curve for the overall best performing ensemble model in terms of accuracy, which is RF with W_{ni} = 0.3 and W_{nj} = 0.7

To measure the true performance of the ensemble models, we compare results to classifiers trained on individual n-gram feature sets. We found that, the ensemble models outperformed models trained on individual n-gram opcode sequences. The best results

in terms of accuracy for the models trained on single feature set as shown in Table 3 was obtained with RF, which yielded an accuracy of 97.6% and a corresponding precision of 98.1% using n-gram of size 2. Though individual models achieved promising results, the ensemble models as evaluated and shown (see bold RF, SVM and KNN results on Table 2) earlier produced slightly higher detection accuracies compared to models trained on individual n-gram opcode sequences.

Table 3. Results for individual n-gram opcode sequences

Classifier	Metrics	1-gram	2-gram	3-gram	4-gram
SVM	Acc Precc	0.935 0.907	0.964 0.959	0.961 0.960	0.938 0.961
KNN (K=5)	Acc Precc	0.958 0.962	0.960 0.960	0.943 0.936	0.943 0.936
RF (Gini)	Acc Precc	0.974 0.982	**0.976** **0.981**	0.971 0.977	0.971 0.977

8 Conclusion

In this paper, we evaluated ensemble of n-gram opcode sequences for malware detection. From our experiments, we found that ensemble of multiple feature sets of n-gram opcode sequences yielded higher detection results compared to classification models trained on individual n-gram opcode sequences. The best ensemble models were able to detect malware with an accuracy of 98.1% and 99.7% in terms of precision.

For future work, we will wish to explore other representation learning methods for malware detection. We would want to particularly concentrate on deep learning models for automation of feature extraction and learning of raw static representations of binaries.

9 References

[1] FitchRatings. https://www.fitchratings.com/research/insurance/ransomware-attacks-growing-global-security-financial- threat-17-05-2021 [online; accessed 07-June-2021].

[2] Convware. https://www.coveware.com/blog/ransomware-attack-vectors-shift-as-new-software-vulnerability-exploits-abound [online; accessed 11-June-2021].

[3] SonicWall. https://www.sonicwall.com/news/2020-sonicwall-cyber-threat-report/ [onlne; accessed 11-June-2021].

[4] M. Chouchane, A. Lakhotia. Using engine signature to detect metamorphic malware. ACM workshop on Recurring malcode, ACM New York, NY, USA, 2006, pp. 73–78. https://doi.org/10.1145/1179542.1179558

[5] Q. Zhang, D. Reeves. MetaAware: Identifying metamorphic malware. Annual computer security applications conference (ACSAC), 2007, pp. 411–420. https://doi.org/10.1109/ACSAC.2007.9

[6] P. Szor. The art of computer virus research and defense, 2005, pp. 244–265.

[7] I. Santos, F. Brezo, X. Ugarte-Pedrero, P. G. Bringas. Opcode Sequences as Representation of Executables for Data-mining-based Unknown Malware Detection. Information Science, 231, 2011, pp. 1–32. https://doi.org/10.1016/j.ins.2011.08.020
[8] B. Kang, S. Y. Yerima, S. Sezer, K. McLaughlin. N-gram Opcode Analysis for Android Malware Detection. International Journal on Cyber Situational Awareness, 1(1), 2016, pp. 1–19. https://doi.org/10.22619/IJCSA.2016.100111
[9] E. B. Karbab, M. Debbabi. MalDy: Portable, data-driven malware detection using natural language processing and machine learning techniques on behavioral analysis reports. DFRWS. Digital investigation, 28, 2019, pp. S77–S86. https://doi.org/10.1016/j.diin.2019.01.017
[10] S. Ni, Q. Qian, R. Zhang. Malware identification using visualization images and deep learning. Computers and security, 77, 2018, pp. 871–885. https://doi.org/10.1016/j.cose.2018.04.005
[11] Z. Fuyong, Z. Tiezhu. Malware detection and classification based on n-grams attribute similarity. IEEE International Conference on Computational Science and Engineering (CSE), 2017, pp. 793–796. https://doi.org/10.1109/CSE-EUC.2017.157
[12] D. Yuxin, Z. Siyi. alware detection based on deep learning algorithm. Neural Comput. Appl, 31 (2), 2019, pp. 461–472. https://doi.org/10.1007/s00521-017-3077-6
[13] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, S. Venkatraman. Robust intelligent malware detection using deep learning, 7, 2019, p. 46720. https://doi.org/10.1109/ACCESS.2019.2906934
[14] H. S. Galal, Y. B. Mahdy, M. A. Atiea. Behavior-based features model for malware detection. Journal of Computer Virology and Hacking Techniques, 12 (2), 2016, pp. 59–67. https://doi.org/10.1007/s11416-015-0244-0
[15] R. Agrawal, J. W. Stokes, M. Marinescu, K. Selvaraj. Robust neural malware detection models for emulation sequence learning, 2018, pp. 1–13. https://doi.org/10.1109/MILCOM.2018.8599785
[16] F. Manavi, A. Hamzeh. A method for malware detection using opcode visualization. Artificial Intelligence and Signal Processing Conference (AISP), 2017. https://doi.org/10.1109/AISP.2017.8324117
[17] E. A. Amer, I. Zelinka. An ensemble-based malware detection model using minimum feature set. Mendel, 25(2), 2019, p. 4. https://doi.org/10.13164/mendel.2019.2.001
[18] A. O. Christiana, B. A. Gyunka. Optimizing android malware via ensemble learning. i-JIM, 14(9), 2020, pp. 61–74. https://doi.org/10.3991/ijim.v14i09.11548
[19] D. Jurafsky, J. H. Martin. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 2000, pp. 189–231.
[20] D. Gilbert, C. Mateu, J. Planes. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. Journal of Network and Applications, 153, 2020, pp. 2–20. https://doi.org/10.1016/j.jnca.2019.102526
[21] R. Lu. Malware detection with LSTM using Opcode language. ArXiv, 2019.
[22] Y. M. Kwon, J. J. An, M. J. Lim, S. Cho, W. M. Gal. Malware Classification Using Simhash Encoding and PCA (MCSP). Symmetry, 12(830), 2020, pp. 1–12. https://doi.org/10.3390/sym12050830
[23] M. Z. Mas'ud, S. Sahib, M. F. Abdollah, S. R. Selamat , R. Yusof. An evaluation of n-gram system call sequence in mobile malware detection. ARPN Journal of Engineering and Applied Sciences, 11(5), 2016, pp. 3122–3126.
[24] R. Moskovitch, C. Feher, N. Tzachar, E. Berger, M. Gitelman. Unknown malcode detection using opcode representation. European Conference on Intelligence and Security Informatics, EuroIsI 2008, pp. 204–215. https://doi.org/10.1007/978-3-540-89900-6_21

[25] R. K. Shahzad, N. Lavesson, H. Johnson. Accurate Adware Detection using Opcode Sequence Extraction. Sixth International Conference on Availability, Reliability and Security, 2011, pp. 189–195. https://doi.org/10.1109/ARES.2011.35
[26] C. Sammut, G. I. Webb. Encyclopedia of Machine Learning. Springer, Boston, MA, 2011. https://doi.org/10.1007/978-0-387-30164-8
[27] M. McGill, G. Salton, Introduction to modern information retrieval, McGraw-Hill, 1983.
[28] J. R. Quinlan. C4.5: Programs for machine learning. Morgan Kaufmann. 1993.
[29] C. E. Shannon. A mathematical theory of communication. Bell System Technical Journal, 27(4), 1948, pp. 623–656. https://doi.org/10.1002/j.1538-7305.1948.tb00917.x
[30] A. M. Garcia, R. Lara-Cabrera, D. Camacho. Android malware detection through hybrid features fusion and ensemble classifiers: the AndroPyTool framework and the OmniDroid dataset. Information fusion, 52, 2018. https://doi.org/10.1016/j.inffus.2018.12.006
[31] SciPy.org. https://docs.scipy.org/doc/numpy-1.9.1/reference/generated/numpy.ma.MaskedArray.argmax.html. [online; accessed 05-July-2021].
[32] V. N. Vapnik, An Overview of Statistical Learning Theory. IEEE transactions on neural network, 10(5),1999, pp. 988–999. https://doi.org/10.1109/72.788640
[33] L. Breiman, Random forests, Machine learning, 45, 2001, pp. 5–32. https://doi.org/10.1023/A:1010933404324
[34] B. Clarke, E. Fokoue, H. H. Zhang. Principles and theory for data mining and machine learning. 2009. https://doi.org/10.1007/978-0-387-98135-2
[35] Scikit-learn. Machine Learning in Python. https://scikit-learn.org/stable/. [online; accessed 05-July-2021].
[36] A. P. Tuan, A. T. H. Phuong, N. V. Thanh, T. N. Van. Malware Detection PE-Based Analysis Using Deep Learning Algorithm Dataset. Figshare, 2018.
[37] VirusShare. https://virusshare.com/
[38] Malicia Project. Malware in Cybercrime. http://malicia-project.com/dataset.html
[39] CNET. https://download.cnet.com/windows/
[40] Virustotal. https://www.virustotal.com/

10 Authors

Paul Ntim Yeboah graduated with a B.Sc. degree in Communication Network and Security from the University of Information Science and Technology, "St. Paul the Apostle", Ohrid, North Macedonia, in 2014 and received an M.Sc. degree in Telematics (Information Security) from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 2017. He is currently a Lecturer in Cybersecurity at the Ghana-India Kofi Annan Centre of Excellence in ICT (AITI-KACE).

Stephen Kweku Amuquandoh holds a B.Sc. degree in Mathematics and Computer Science from the University of Ghana, and an M.Sc. degree in Information Technology from the Kwame Nkrumah University of Science and Technology, Kumasi, Ghana.

Haruna Balle Baz Musah obtained a B.Sc. degree from the University of Ghana in 1999, and M.Sc. degree from Norfolk State University, USA, in 2012. He is currently a researcher and a faculty member at the AITI-KACE.

Article submitted 2021-07-10. Resubmitted 2021-09-20. Final acceptance 2021-09-20. Final version published as submitted by the authors.