Science Forum (Journal of Pure and Applied Sciences)

Journal Homepage: https://atbuscienceforum.com.ng/index.php/jpas

Techniques for Malware Detection and Classification Using Machine Learning Classifiers

Zaharaddeen S. Iro[1], Nasiru Abubakar[2], Badamasi A. Ahmed[2] and Zahraddeen Sufyan[2]

[1] Department of Information Technology, Federal University Dutse, Jigawa State.

[2]Department of Computer Science, Federal University Dutse, Jigawa State.

Abstract

The growing sophistication and evolving nature of malware present significant challenges to cybersecurity systems across the globe. Traditional signature-based detection techniques are increasingly inadequate against novel and evasive threats, necessitating more intelligent and adaptive approaches. This study proposes a machine learning-based framework for effective malware detection and classification, leveraging behavioral features extracted from dynamic analysis in sandbox environments. The analysis focuses on API call patterns, which serve as a rich source of information for characterizing the runtime behavior of both benign and malicious software. These patterns were transformed into feature vectors to train and test three machine learning algorithms: Support Vector Machines (SVM), Naive Bayes, and Random Forest. The models were evaluated on a comprehensive dataset comprising samples from ten distinct malware families alongside benign executables. Experiments were conducted under both multi-class and binary classification settings to assess the robustness and generalizability of each algorithm. Among the evaluated models, Random Forest consistently delivered the highest performance, achieving an accuracy of 95.69% in multi-class classification and 96.8% in binary classification. It also maintained low false positive and false negative rates, demonstrating its reliability for real-world deployment. SVM also showed strong performance, particularly in the binary setting, where it achieved zero false negatives—a critical factor in minimizing undetected threats. Conversely, the Naive Bayes classifier underperformed due to its reliance on the assumption of feature independence, which proved incompatible with the interdependent nature of API call sequences.The results highlight the superior capability of ensemble-based methods, such as Random Forest, for behavioral malware detection. This study underscores the importance of selecting appropriate feature representations and machine learning algorithms to develop scalable, accurate, and adaptive cybersecurity solutions capable of countering the dynamic landscape of modern malware threats.

Keywords

Malware Detection; Machine Learning; Random Forest; Behavioral Analysis; Feature Selection; Sandbox Environment.

Corresponding Author: Zaharaddeen S. Iro, Department of Information Technology, Federal University Dutse, Jigawa State.,
Email: zs.iro@fud.edu.ng

INTRODUCTION

Due to the swift advancement of the Internet, malware has emerged as a significant cyber hazard in contemporary times. Software that executes harmful activities, including data theft and espionage, is classified as malware. Kaspersky Labs (2017) characterized malware as "a category of computer program intended to compromise a legitimate user's computer and cause damage in various manners" (Chumachenko, 2017; Singh & Singh, 2021). Kaspersky Labs (2016) said that 6,563,145 distinct hosts were compromised, and 4,000,000 unique malware specimens were identified in 2015. Research from Juniper (2016) forecasted that the global cost of data breaches will escalate to $2.1 trillion by 2019. In addition to that, there is a severe drop in the skill level required for malware production; this is due to the availability of attacking tools on the internet. An array of anti-detection tactics exists, along with the capacity to purchase malware on the black market, facilitating the recruitment of additional software attackers (Aliyev, 2010; Knöchel & Wefel, 2022).

Malware protection for computers is a critical cybersecurity responsibility for both individuals and enterprises, as a single attack can lead to compromised data and significant financial losses. Massive losses and frequent attacks require accurate and rapid detection systems. Current static and dynamic approaches do not enable efficient detection, especially when dealing with zero-day assaults. For this reason, machine learning-based algorithms can be employed for malware detection as well as for feature representation and categorization(Shaukat, Luo, & Varadharajan, 2023). This research aims to create a machine learning-based method for malware classification utilizing Cuckoo Sandbox. The sandbox will be employed to extract the behavior of the malware samples, which will serve as input for the machine learning algorithms. The investigation will also discover the optimum feature representation method and how the features should be retrieved, finally differentiating the malware families with the lowest mistake rate. The purpose of this study is to apply machine-based algorithms to detect and categorize malware attacks on computers, as well as determine the optimal feature representation methods.

2. LITERATURE REVIEW

The proliferation of computers, smartphones, and other Internet-enabled contraptions leaves the world defenseless to cyberattacks. Plenty of malware discovery strategies have emerged due to the blast in malware activity. Analysts utilize various large-scale data tools and machine learning methods when attempting to distinguish pernicious code. Traditional machine learning-based malware detection approaches have considerable processing time but may effectively identify newly emerging malware(Shaukat et al., 2023). Feature engineering may become obsolete due to the prevalence of modern machine learning algorithms, such as deep learning. In this study, we examined a variety of malware detection and classification techniques.

Reports by (Radwan, 2019) performed a detection method based on a modified Random Forest algorithm in combination with Information Gain for better feature representation. It should be noticed that the data set consists purely of portable executable files, for which feature extraction is generally easier. The result achieved is an accuracy of 97% and a 0.03 false positive rate.

(Timon, 2019) applied the main investigators in exploiting CNNs for side-channel attacks (SCA), although they were not the learning methods that deployed deep learning approaches such as MLP, CNN, and long short-term memory (LSTM). These techniques include random forest and support vector machine (SVM). The findings of their study show that deep learning is superior to more conventional approaches to machine learning and, as a result, produces good outcomes. The authors show this using two different data sets, one of which is an implementation that does not have any kind of

protection, and the other utilizes a countermeasure for masking.

In addition, the results show that the CNN database, sometimes referred to as the side-channel analysis data set, is in (Das et al., 2019; Staib & Moradi, 2023). This database has been used in the investigations of various researchers and was first presented by the authors. After introducing the data set, they investigated the effect of hyperparameters to find that the CNN and MLP architectures as the most effective.

According to the findings of their study, (Masure, Dumas, & Prouff, 2020) reveal the increase in the volume of the CNN kernel, resulting in better behavior if the network is confronted with misaligned traces. However, they do not explain why increasing the kernel makes the attack more effective, which is strange.

(Picek et al., 2018) compared CNNs' performance against machine learning methods such as Random Forest, XGBoost, and Naïve Bayes. Their main objective is to investigate the circumstances under which CNNs perform better than the other techniques described. According to the findings of their study, CNNs can only improve performance in the aggregate. CNNs are most effective when the traces are not pre-processed, when noise levels are lowered, and when information dimensions are higher. On the contrary, machine learning (ML) schemes could achieve performance that is almost on par with that of CNNs. The discovery that ML methods need noticeably fewer processing resources than CNNs is a significant result. As a result, the researchers have severe reservations about the usefulness of CNNs.

In contrast to a template attack, which generally includes the adversary realigning the traces and selecting the points of interest on their own, this one does not. Because of this, the findings show that CNNs are beneficial even when the traces are misaligned. On the other hand, overfitting is a potential issue due to the size and complexity of the CNN architecture that lies under the surface. They provide two data augmentation algorithms for misaligned traces as a means of generating more training data. Experiments were carried out to illustrate the efficacy of data augmentation options for misaligned traces(Kerkhof, Wu, Perin, & Picek, 2023).

In the work of (Maji, Banerjee, Fuller, & Chandrakasan, 2022), they compared a wide variety of topologies and sets of pieces of information. The results of such experiments showed that no single design succeeds with all data sets. Hence, this remains very necessary in selecting a structure appropriate for issues present at that time. In addition, the authors provide evidence that including distortion within the primary substrates of the networks helps performance by reducing the amount of overfitting that occurs. When working with smaller data sets, this is suggested by using increased noise levels, whereas working with more extensive data sets requires a lower noise level to get the best results.

The researchers of (Shimada, Kuroda, Fukuda, Yoshida, & Fujino, 2022) suggested a completely new CNN framework that uses more domain information obtained through a side-channel attack. Data provided for creating neural networks can be plaintext or ciphertext, and this distinction is determined by the leaky model. The classification block of a CNN architecture is the component that is given the domain information to use as a new feature vector. In the work, the authors compare several architectural concepts offered by various works of literature with and without the architecture that they have provided. They show how a design that uses domain knowledge can improve performance for protected information but not for unprotected information. However, if profile traces are generated using a fixed key, this method is not proper.

(Zaid, Bossuet, Dassance, Habrard, & Venelli, 2021) strongly focus on the need for fine-tuning architecture and hyperparameters; models do not operate correctly without an appropriate configuration. They point out that we cannot realize the full potential of architecture if we do not understand the influence of a hyperparameter and explain why this is the case. The authors provide three visualization methods to solve this problem: weight, gradient, and heat map. These methods are utilized to improve the readability and interpretability of each hyperparameter. These approaches make it simpler to set the hyperparameters by allowing an opponent to determine the influence of each one individually, which in turn makes it easier to tune the hyperparameters. Using these three visualization approaches, they also propose implementation options for protected and unprotected environments.

In particular, for data sets that include a concealed countermeasure, it is recommended that the CNN kernel measure be modified to equal 50% of the highest delay of the randomized delay. It remains one of the guidelines provided by their method. Some of the most related works are summarized in the table below.

3. METHODOLOGY

This study identified the most effective feature representation, selection techniques, and classification algorithms for distinguishing malware families with high accuracy. The methodology involves five major stages: data collection, sandbox configuration, feature extraction, feature selection, and machine learning model application.

3.1 Data Collection

2,140 files were collected, comprising 1,156 malicious and 984 benign samples. The malware samples were gathered from reputable sources like VirusTotal using unique file hashes from threat intelligence and malware reverse engineering reports. To ensure a diverse dataset, nine malware families were selected: *Dridex, Locky, TeslaCrypt, Vawtrak, Zeus, DarkComet, CyberGate, Xtreme*, and *CTB-Locker*. Benign samples included .exe, .pdf, and .docx files commonly used vectors in malware delivery.

3.2 Sandbox Configuration

Behavioral analysis was performed using Cuckoo Sandbox, running in virtual machines created with VMcloak on VirtualBox. Each VM was equipped with:

- OS: Windows 7 Professional (64-bit)
- RAM: 2 GB
- Software: Adobe Reader 9.0, Flash Player 11.7, Java JRE 7, .NET Framework 4.0, Visual Studio Redistributables (2005–2013)

This setup ensured diverse execution environments to mimic real-world malware behavior.

3.3 Feature Extraction

Features were extracted from sandbox reports and represented as combination matrices of successful/failed API calls and their return codes. Reports triggering fewer than five API calls were discarded. The extraction process logs and timestamps were stored alongside the generated .csv files for traceability.

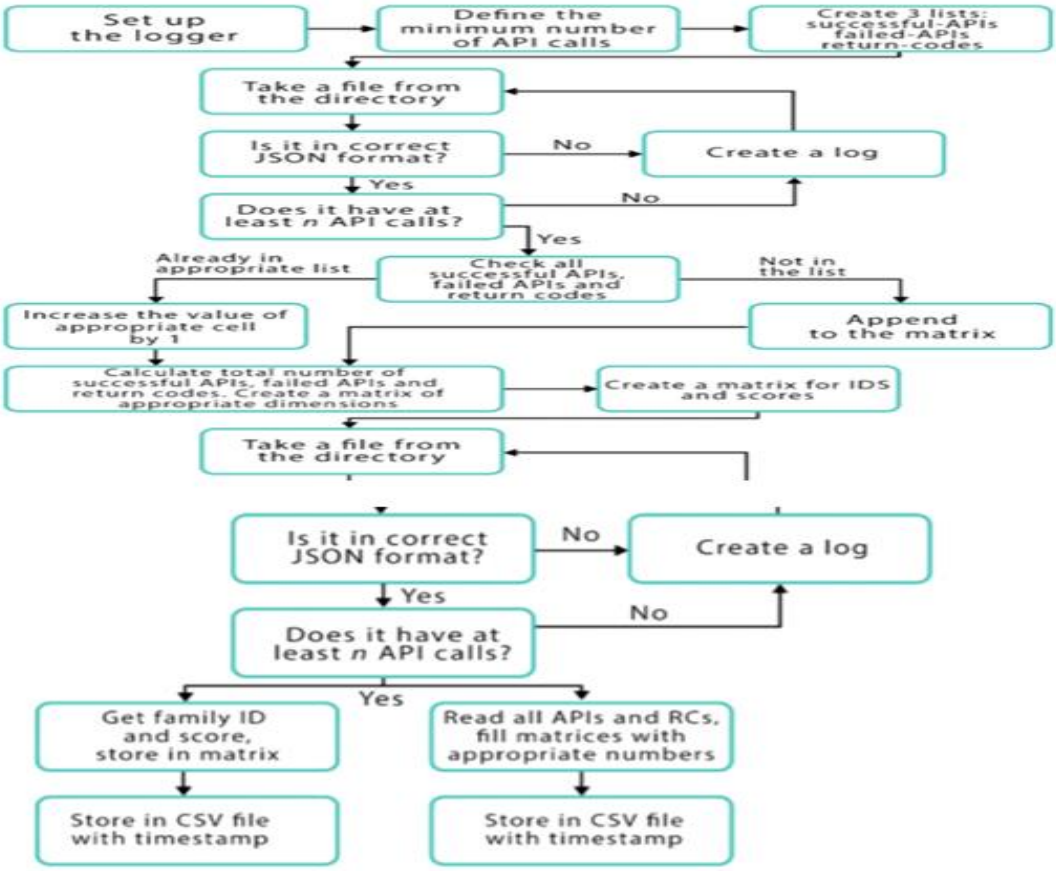

Figure 1. Feature Extraction Process

3.4 Feature Selection

The initial set of 70,518 features was too large for efficient processing. To eliminate irrelevant features, the Boruta algorithm (using the randomForest package in R) was employed. Due to memory constraints, the dataset was divided into manageable subsets, processed individually, and then merged. The final reduced feature set contained 306 features, significantly improving runtime and accuracy.

3.5 Application of Machine Learning Algorithms

With the refined features, the following machine learning algorithms were applied:

- Support Vector Machines (SVM) — kernlab
- Naive Bayes — e1071
- Random Forest — randomForest

4. RESULTS AND DISCUSSION

This section discusses the results of the assessment of the implemented machine learning methods. The accuracy of detection is measured as the percentage of correctly identified instances:

$$Accuracy = \frac{count(Correctly\ identified\ samples)}{count(Total\ samples)}$$

4.1 SUPPORT VECTOR MACHINES (SVM)

The Support Vector Machines achieved the overall accuracy 87.6% for multi-class classification and 94.6% for binary classification. The detailed information about the accuracy of each class produces are found in Table 1.

Table 1 SVM Multiclass Accuracy

Class	Family	Correctly Classified	Incorrectly Classified	Accuracy	Average Cuckoo Score
1	Benign	56	5	91.8%	1.04
2	Dridex	32	5	86.5%	5.26
3	Locky	21	6	77.8%	6.41
4	TeslaCrypt	37	7	84%	6.27
5	Vawtrak	10	9	55.6%	2.66
6	Zeus	31	1	77.5%	6.46
7	DarkComet	48	2	98%	5.15
8	CyberGate	37	3	97.4%	6.57
9	Xtreme	31	2	91.2%	5.15
10	CTB-Locker	22	0	100%	4.76

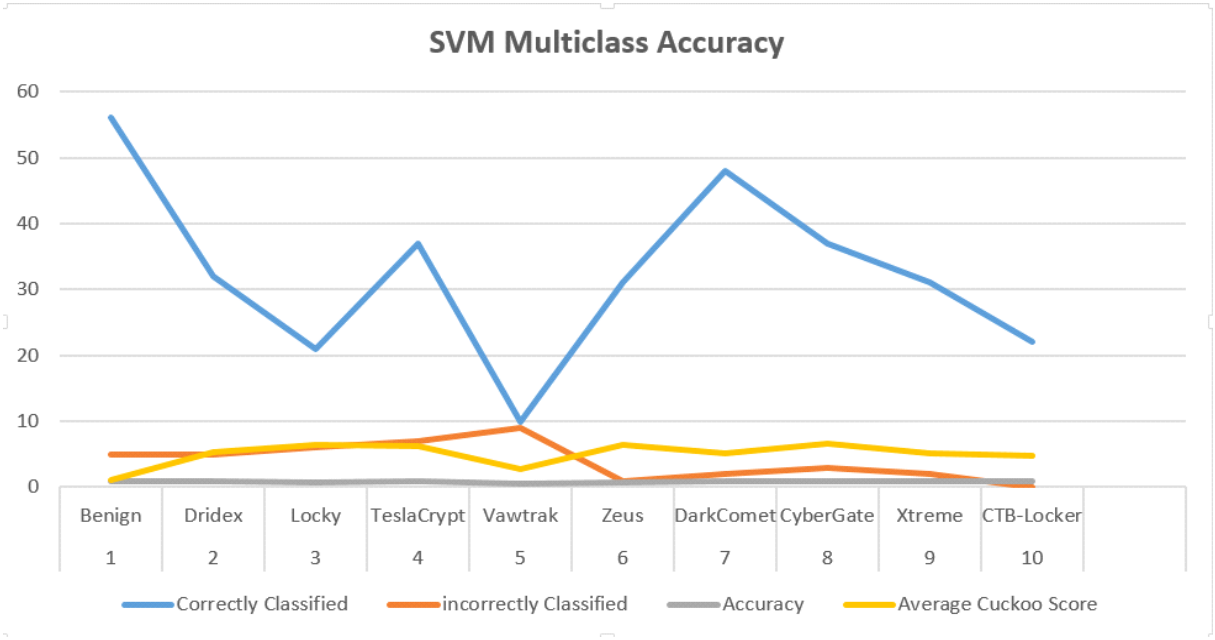

Figure 2: SVM Multiclass Accuracy

Table 2: SVM Binary Classification Cross Table

Selected_apis.testlabels.twoway	1	2	Row Total
1	41 174.102 0.672 1.000 0.111	20 21.631 0.328 0.54	61 0.164
2	0 34.259 0.000 0.000 0.000	310 4.256 1.000 0.939 0.889	310 0.836
Column Total	41 0.111	330 0.889	371

Table 2 outlines the cross-table for binary classification. The detailed information about binary classification can be found in Tables 2 and 3. As we can see, the number of correctly identified benign instances (true negatives) was equal to 41, correctly identified malicious instances (true positives) – 310, incorrectly identified benign instances (false positives) – 20, incorrectly identified malicious instances (false negatives) – 0.

Table 3: SVM Binary Classification Accuracy

Class	**Correctly identified instances**	**Incorrectly identified instances**	**Accuracy**
Benign	41	20	67.2%
Malicious	310	0	100%

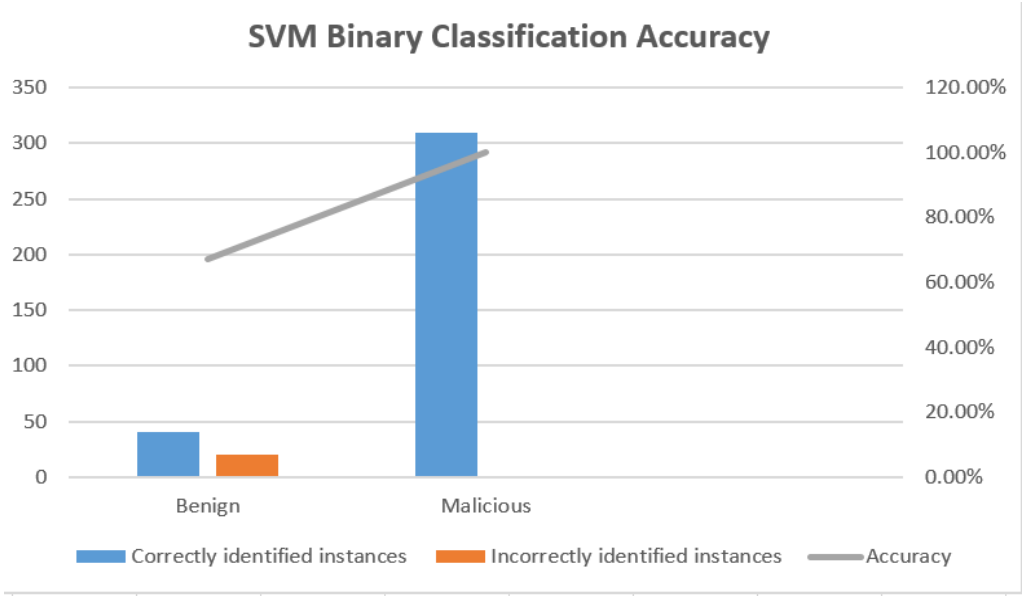

Figure 3: SVM Binary Classification Accuracy

Table 4: SVM Binary Classification Accuracy

True Positives	True Negatives	False Positives	False Negatives
310	41	20	0

4.5 NAIVE BAYES

The Naïve Bayes algorithm was tested, and the resulting accuracy was 72.23% for multiclass classification and 55% for binary classification. The detailed results that outline the accuracy of each of the malware families can be found in Table 5

Table 5: Naive Bayes Multi-Class Classification Accuracy

Class	Family	Correctly Classified	incorrectly Classified	Accuracy	Average Cuckoo Score
1	Benign	34	27	55.8%	1.04
2	Dridex	1	36	2.7%	5.26
3	Locky	25	2	92.6%	6.41
4	TeslaCrypt	33	11	75%	6.27
5	Vawtrak	8	10	44.4%	2.66
6	Zeus	28	12	70%	6.46
7	DarkComet	49	0	100%	5.15
8	CyberGate	37	1	97.4%	6.57
9	Xtreme	31	3	91.2%	5.15
10	CTB-Locker	22	0	100%	4.76

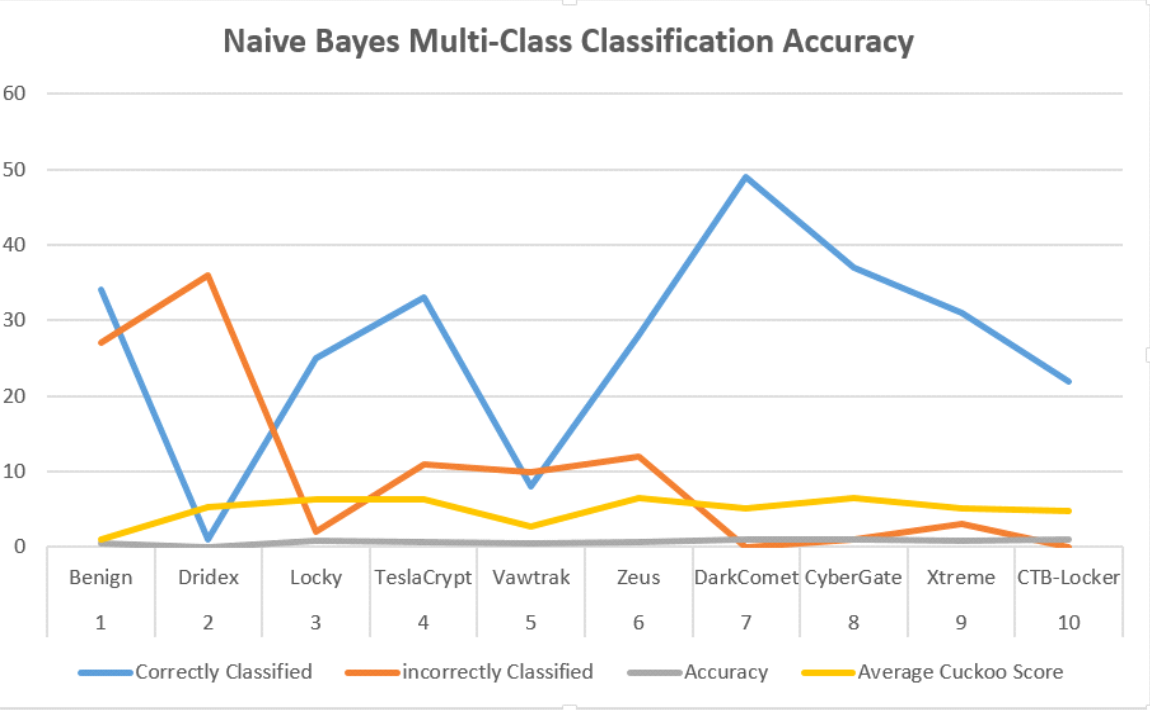

Figure 5: Naive Bayes Multi-Class Classification Accuracy

For binary classification, the algorithm performed poorly. The number of correctly identified benign instances (true negatives) was 61, correctly identified malware instances (true positives) 143, incorrectly identified benign instances (false positives) 0, and identified malware instances (false negatives) 167. The detailed results can be found in Figure 5, Tables 6.

Table 6: Naive Bayes Binary Classification Cross-Table

Selected_apis.testlabels.twoway	prediction 1	2	Row Total
1	61 14.747 1.000 0.268 0.164	0 23.512 0.000 0.000 0.000	61 0.164
2	167 2.902 0.539 0.732 0.450	143 4.672 0.461 1.000 0.385	310 0.836
Column Total	228 0.615	143 0.385	371

Overall, the Naive Bayes algorithm performed poorly. From Table 5, the accuracy of multiclass classification was 72.23%, and of binary classification, only 55%. This result is insusceptible for real world detection. In addition to that, a number of false negatives, in other words, malware files that were incorrectly marked as benign filed, reached 167 of the total number of files. In a real environment, such result would cause huge malware epidemics in a short amount of time.

Table 7: Naive Bayes binary classification accuracy

Class	**Correctly identified instances**	**Incorrectly identified instances**	**Accuracy**
Benign	61	0	100%
Malicious	143	167	46.1%

Table 8: Naive Bayes binary classification accuracy

True Positives	**True Negatives**	**False Positives**	**False Negatives**
143	61	0	167

Most likely, such a bad accuracy results from a high dependence between features. As we know, the main drawback of the Naive Bayes algorithm is that each feature is treated independently, although in most cases, this cannot be true. In our case, most likely certain APIs are dependent on each other, i.e. $APIn$ cannot be triggered without $APIm$. That is the most probable reason of the bad result of the Naive Bayes algorithm.

4.6 RANDOM FOREST

The last algorithm that was implemented was the Random Forest algorithm. The algorithm resulted in a good accuracy of predictions, 95.69% for multi-class classification and 96.8% for binary classification. Detailed information about the performance of each class can be found in Table 9 and Fig. 6.

Table 9: Random Forest Multiclass Classification Accuracy

Class	Family	Correctly Classified	incorrectly Classified	Accuracy	Average Cuckoo Score
1	Benign	58	3	95%	1.04
2	Dridex	35	2	94.6%	5.26
3	Locky	25	2	92.6%	6.41
4	TeslaCrypt	44	0	100%	6.27
5	Vawtrak	15	3	83.3%	2.66
6	Zeus	35	5	87.5%	6.46
7	DarkComet	49	0	100%	5.15
8	CyberGate	38	0	100%	6.57
9	Xtreme	34	0	100%	5.15
10	CTB-Locker	22	0	100%	4.76

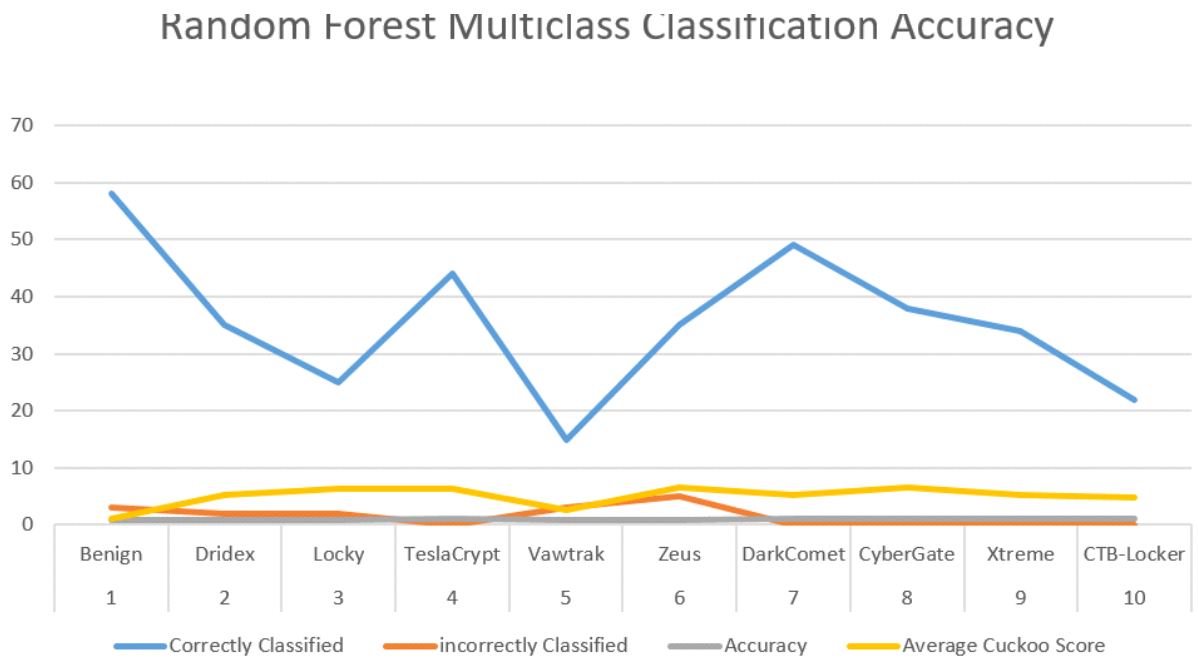

Figure 6: Random Forest Multiclass Classification Accuracy

In the binary classification problem, the result achieved was 96.8%. More specifically, the number of correctly identified benign instances (true negatives) reached 52, correctly identified malware instances (true positives) 307, incorrectly identified benign instances (false positives) 9, and incorrectly identified malware instances (false negatives) 3. Detailed information can be found in Figure 6, Table 9, and 10.

Table 10: Random Forest Binary Classification Cross-Table

Selected_apis.testlabels.twoway	Prediction 1	2	Row Total
1	52 204.055 0..852 0.945 0.140	0 35.516 0.148 0.028 0.024	61 0.164
2	3 40.153 0.010 0.055 0.008	307 6.989 0.990 0.972 0.827	310 0.836
Column Total	55 0.148	316 0.852	371

Table 11: Random Forest Binary Classification Accuracy

Class	**Correctly identified instances**	**Incorrectly identified instances**	**Accuracy**
Benign	52	9	85.2%
Malicious	143	3	99%

Table 12: Random Forest Binary Classification Accuracy

True Positives	**True Negatives**	**False Positives**	**False Negatives**
307	52	9	3

The Random Forest algorithm resulted in the highest accuracy among the other algorithms. It achieved 85.2% and 99% accuracy for multiclass and binary classifications, respectively. However, some false negatives are still present – their number is equal to three.

5. CONCLUSION

In this study, we evaluated three prominent machine learning algorithms—Support Vector Machines (SVM), Naive Bayes, and Random Forest—to detect and classify malware based on API call features extracted from sandbox environments. Each model was assessed on both multi-class and binary classification tasks.

The results demonstrate that Random Forest outperformed the other algorithms across all evaluation metrics. It achieved the highest accuracy in both classification tasks (95.69% multi-class, 96.8% binary) and maintained a low false negative rate, making it a highly effective choice for practical malware detection scenarios. The Support Vector Machine also exhibited strong performance, especially in binary classification, where it recorded zero false negatives, highlighting its potential for environments where preventing malware evasion is critical.

In contrast, Naive Bayes showed significantly lower performance. Its core assumption of feature independence does not align well with the correlated nature of API call data, leading to a high number of false negatives and poor classification of several malware families. As such, it is not recommended for real-world malware detection systems.

In conclusion, for malware classification using API-based behavioral features, ensemble learning methods such as Random Forest offer robust, scalable, and highly accurate solutions. These findings underscore the importance of algorithm selection and feature dependence in building reliable and secure machine learning-based cybersecurity systems. Future work may explore deep learning-based models, adversarial robustness, and automated feature engineering to enhance real-time malware detection and resilience against evolving cyber threats.

REFERENCES

Aliyev, V. (2010). Using honeypots to study skill level of attackers based on the exploited vulnerabilities in the network.

Chumachenko, K. (2017). Machine learning methods for malware detection and classification.

Das, D., Golder, A., Danial, J., Ghosh, S., Raychowdhury, A., & Sen, S. (2019). *X-DeepSCA: Cross-device deep learning side channel attack*. Paper presented at the Proceedings of the 56th Annual Design Automation Conference 2019.

Kerkhof, M., Wu, L., Perin, G., & Picek, S. (2023). No (good) loss no gain: Systematic evaluation of loss functions in deep learning-based side-channel analysis. *Journal of Cryptographic Engineering, 13*(3), 311-324.

Knöchel, M., & Wefel, S. (2022). *Analysing attackers and intrusions on a high-interaction honeypot system*. Paper presented at the 2022 27th Asia Pacific Conference on Communications (APCC).

Maji, S., Banerjee, U., Fuller, S. H., & Chandrakasan, A. P. (2022). *A threshoid-impiementation-based neural-network accelerator securing model parameters and inputs against power side-channel attacks*. Paper presented at the 2022 IEEE International Solid-State Circuits Conference (ISSCC).

Masure, L., Dumas, C., & Prouff, E. (2020). A comprehensive study of deep learning for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 348-375.

Picek, S., Samiotis, I. P., Kim, J., Heuser, A., Bhasin, S., & Legay, A. (2018). *On the performance of convolutional neural networks for side-channel analysis*. Paper presented at the Security, Privacy, and Applied Cryptography Engineering: 8th International Conference, SPACE 2018, Kanpur, India, December 15-19, 2018, Proceedings 8.

Radwan, A. M. (2019). *Machine learning techniques to detect maliciousness of portable executable files*. Paper presented at the 2019 International Conference on Promising Electronic Technologies (ICPET).

Shaukat, K., Luo, S., & Varadharajan, V. (2023). A novel deep learning-based approach for malware detection. *Engineering Applications of Artificial Intelligence, 122*, 106030.

Shimada, S., Kuroda, K., Fukuda, Y., Yoshida, K., & Fujino, T. (2022). *Deep Learning-Based Side-Channel Attacks against Software-Implemented RSA using Binary Exponentiation with Dummy Multiplication*. Paper presented at the ATAIT.

Singh, J., & Singh, J. (2021). A survey on machine learning-based malware detection in executable files. *Journal of Systems Architecture, 112*, 101861.

Staib, M., & Moradi, A. (2023). Deep learning side-channel collision attack. *IACR Transactions on Cryptographic Hardware and Embedded Systems, 2023*(3), 422-444.

Timon, B. (2019). Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 107-131.

Zaid, G., Bossuet, L., Dassance, F., Habrard, A., & Venelli, A. (2021). Ranking loss: Maximizing the success rate in deep learning side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 25-55.