Quest Journals
Journal of Software Engineering and Simulation
Volume 8 ~ Issue 5 (2022) pp: 16-25
ISSN(Online) :2321-3795 ISSN (Print):2321-3809
www.questjournals.org

Research Paper

Malware Detection and Classification System Using Random Forest

[1] Chukunda Dike Chukunda
Department of Computer Science, Rivers State University, Nigeria

[2] Dr. Daniel Matthias
Department of Computer Science, Rivers State University, Nigeria.

[3] Dr. E.O Bennett
Department of Computer Science, Rivers State University, Nigeria.

Corresponding Author: ***Chukunda Dike Chukunda***
Department of Computer Science, Rivers State University, Nigeria

ABSTRACT- *Malware programs attack computer systems, smart mobile devices, and some applications. Malware is a program that needs to be watched out for because it can be a threat to computer users and internet networks. The malware was created to steal personal information about a computer user or control a user's device over a network. Computers are easily infiltrated by various malware programs that can interfere with and even damage user files. Many users are not aware of the entry of malware programs into a computer, one of which is through a network that contains the malware program. The dataset used in this study was taken from the Kaggle Data Set and VirusShare, with a total of 17845 Data in the form of a comma-separated values (CSV) file, captured based on traffic on the network that contains both malware and non-malware. The process of training and testing on the Data Set is carried out using the Tensorflow Tools by making a Binary Classification or creating two classes, namely the malicious class, and the benign class. The method used is Machine Learning by comparing the Random Forest Algorithm, Support Vector Machine, and Bayesian Network, The system was implemented using python programing language for its backend and HTML/CSS for the frontend of the system. The results obtained from the three algorithms show that Random Forest has the highest level of accuracy with a percentage of 99.95%, a precision of 0.998, and a recall of 0.999, with an average detection speed of 3 to 8 seconds, enabling quick and earlier mitigating action to be done before injury.*
Keywords- *Malware Detection, Malware Classification, Machine Learning, Random Forest, Matrix Evaluation*

***R**eceived 01 May, 2022;* ***R**evised 10 May, 2022;* ***A**ccepted 12 May, 2022 © The author(s) 2022.*
P**ublished with open access at* ***www.questjournals.org

I. INTRODUCTION

Computers, mobile phones, and the internet have been commonplace in our everyday lives in the modern era. The majority of companies operate online, using computers and the internet. People and organizations have found this to be a favorable environment. Because of the widespread usage of the Internet, the credibility of our systems and information has grown in importance. This necessitates data continuity, precision, and reliability throughout its life. Data cannot be accessed while in transit, and precautions must be considered to prevent unauthorized persons from altering it [1].

As you would expect, malware has a long and illustrious past. Yisrael Rada may have coined the word malware in 1990, yet these kinds of dangers had been known as computer viruses for decades. Although many of the original infected programs were built as pranks or experiments, hackers now use malware to steal personal, financial, and commercial data. Worse, government agencies are involved to gain access to classified information [2]. We don't go a day without hearing about new varieties of malware that leave a trail of devastation in their wake. Malicious software may appear to be a novel concept. Viruses, worms, and Trojans have infected the majority of computer users in recent years, typically as a result of their machines being targeted. The media equally played a part, reporting on the latest cyber-attacks and virus writer arrests on a more regular basis [3].

On the other hand, malicious technology is not a new phenomenon. Even though the early computers were not infected with viruses, they were nonetheless vulnerable. When information technology was in its infancy, there were not enough people who knew how to manipulate computer systems.

However, as computer devices became more globally used, problems began to emerge. Viruses first appeared in the 1970s on dedicated networks like the ARPANET. Apple's introduction of personal computers in the early 1980s coincided with an increase in computer viruses. Individuals were able to comprehend how computers functioned as more people acquired hands-on access to them, and some people accidentally exploited their newfound knowledge for harmful ends [4]. Viruses have evolved in tandem with technological advancements. Machines have advanced to the point that they are almost unrecognizable in only a few decades. The once-basic computers that booted from a floppy disk have evolved into sophisticated systems capable of transferring massive quantities of data almost rapidly, routing email to hundreds or thousands of addresses, and entertaining people with movies, music, and interactive Web pages. Virus authors have adapted to these developments [5].

II. RELATED WORKS

[6] presented a deep learning-based malware variant identification technique. They successfully resolved the data imbalance issue by using the bat technique to generate grayscale images from malicious code. A convolutional neural network was used to detect and analyze malware pictures, with the model achieving a high level of accuracy, according to the findings. [7] with the same configuration, introduced a more flexible and hybrid deep learning platform approach for successful visual detection of malware. Researchers have to utilize an aggressive learning machine with two hidden layers to detect dangers in malware in safety-critical networks. [8] and [9] offered deep flow, a new deep learning-based technique for detecting malware straight from the information streams of an application.

To improve the effectiveness of the classifier, researchers used deep learning as a feature extraction method to detect malicious code. [10] Suggested a hybrid deep learning visualization technique, demonstrating that a deep learning-based model can effectively differentiate the behavioral characteristics of several malware families.

[11] Proposed a method for dealing with the complexities of the malware dataset. To increase the scalability of unknown malware detection, this technique employs a multi-level deep learning model that arranges the tree structure of several deep models.

Researchers employed a heterogeneous deep model made up of several layers of associative memory and a weighted auto-encoder with multilayer constrained Boltzmann machines to detect malware, according to [12]. The deep learning model is required to undertake a greedy layer-wise training operation followed by fine-tuning of a supervised parameter to efficiently detect unsupervised features.

[13] Compared single-flow convolutional neural network (CNN) models to dual-flow deep learning methods such as gated recurrent unit fully convolutional network (GRU-FCN). Operating systems (OS) popularity has soared in recent years. However, because it is a common target for malicious software, its popularity comes at the expense of stability.

[14] Developed a malware detection system based on interpretable strings derived from API execution calls, as well as semantic strings that reveal an attacker's intent and goal. A parser was used to extract interpretable text from each PE file, and an SVM ensemble with bagging was utilized to build the detector. The system's performance was evaluated using a dataset created by Kingsoft's anti-virus lab.

[15] suggested a text categorization-based malware classification approach. They began by extracting all n-grams from the training data, with n ranging from 3 to 6. Second, the Fisher Score feature selection technique was used to choose the top 5500 features based on their Document Frequency (DF) score. They then fed the features into a Decision Tree (DT), an Artificial Neural Network (ANN), a Support Vector Machine (SVM), a Random Forest (RF), and a Bayes method (NB).

[16] proposed a method for collecting bytes n-gram characteristics from known dangerous samples, with n ranging from 1 to 8, to aid in the identification of unknown executable. They employed a technique called class-wise document frequency to minimize the feature space because the number of unique n-grams is so huge. Finally, multiple N-gram models were generated using classifiers such as Decision Trees, Nave Bayes, Instance-based Learners, Adaboost, and Random Forests.

[17] developed a method for calculating the information gain of each bytes n-gram in training samples and selecting the K n-grams with the highest information gain as features. Each feature vector property's averages from the malware and benign samples were then calculated individually. Finally, a new piece of software was

assigned to one of the two categories based on the distance between the unknown sample's feature vector and the average vectors of the two categories.

[18] proposed a malware detection approach based on the frequency of opcode sequences and the relevance of such sequences. Each program was represented as a feature vector, with each feature corresponding to a unique 1-g or 2-g. To decrease the number of 2-g characteristics, they used Information Gain to choose the top 1000 2-g traits. To test their strategy, they utilized 1700 malicious and 1000 benign programs. In the end, the Support Vector Machine with Pearson VII proved to be the most accurate.

[19] proposed MutantX-S, a clustering approach based on operation code N-gram characteristics retrieved from malware assembly language source code after disassembly. By combining a hashing method and a close-to-linear clustering approach, MutantX-S increases the scalability of processing very large numbers of malware with high-dimensional attributes. Instead of working with enormous amounts of data, the algorithm simply used prototypes to do agglomerative hierarchical clustering.

According to [20], byte n-grams appear to learn mostly from string information in an executable, namely elements from the PE header. Millions of n-grams (for a greater n), feature selection algorithms tend to select the ones that usually occur to be considered features. This encourages the use of low entropy features such as strings and padding locations.

[21] extended the previous work to detect metamorphic malware. Wavelet analysis was applied to decide the areas where significant changes are made in the entropy values. Afterward, a comparison was made regarding the similarity of the two files using the Levenshtein distance. Hence, given an unidentified piece of software, it would be assigned to the class representing the most related sample in the training set.

[22] centered on creating an effective model for detecting phishing URLs using machine learning techniques The Link Guard algorithm was used to extract real and visual links from the Domain Name System (DNS), and the actual and visual links were compared to see if they were the same. The system was written in the Python programming language. Experiments were carried out using two publicly accessible website databases to test the effectiveness of detecting phishing websites. 3200 website samples were used, with 2127 genuine websites and 1036 phishing websites. The accuracy of the LinkGuard and SVM was 98.8 percent.

III. METHODOLOGY

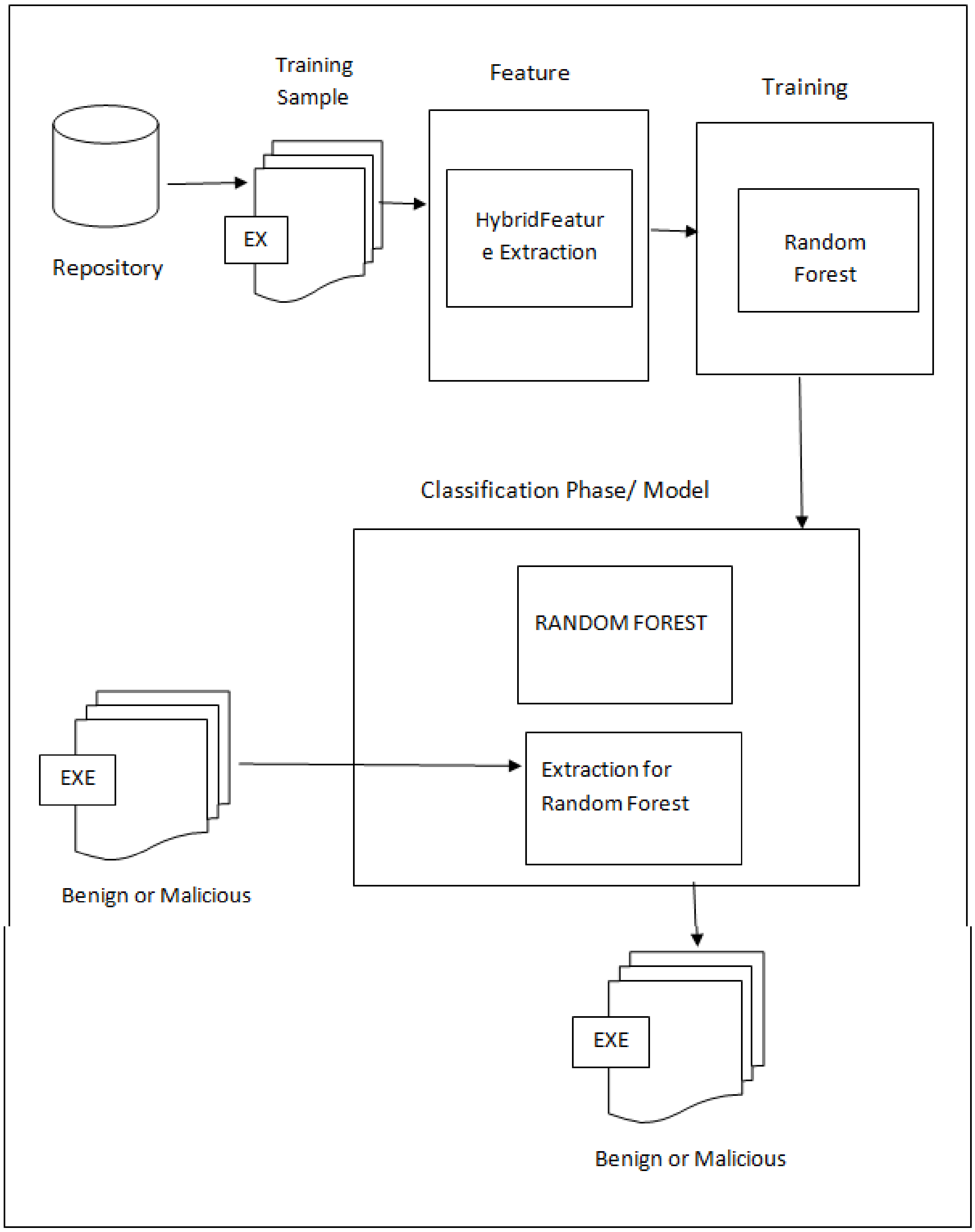

Figure 1: Architecture of the proposed System

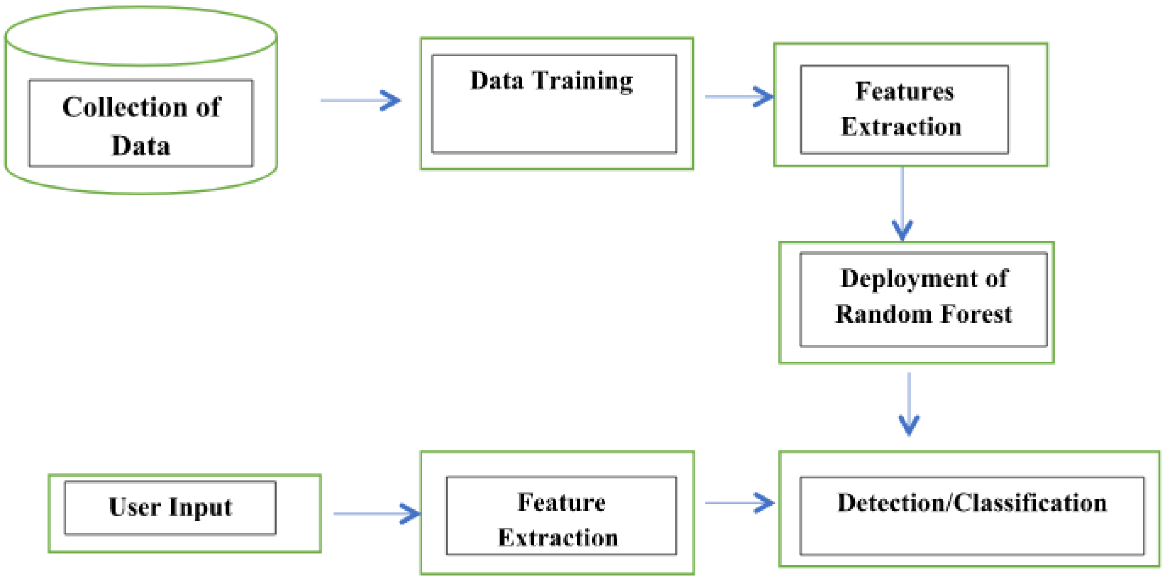

Figure 2: Proposed System Flow Diagram

Collection of Dataset

The gathering of datasets is the initial step. Malware and benign files were gathered to provide the dataset for training the model with a machine learning technique. The databases for dangerous and benign files are combined to create a huge dataset. The benign files dataset has approximately 15,000 files, while the harmful applications dataset contains approximately 30,000 files".

Training of Dataset

The created dataset is utilized to train the model using the random forest technique with the supplied parameters. The data must be arranged, hence data prepossessing is a crucial element of the process. To extract features, the relationship between these datasets must also be discovered. Finding data with missing characteristics and values is also a big component of the process. Because many of the harmful files or software gathered from VirusShare were unable to be adequately decompiled, they were discarded and new files were collected from Contagio.

Feature Extraction

The main aim is to discover malicious permission patterns using a mix of permissions as ML input training datasets of benign and malicious files. The permission-based features have been extracted, and these traits have been used to distinguish between malicious and benign permission requests. The extraction of features is crucial for prediction. The more features you have, the faster your computations will be, as well as the less memory you'll use. This is used to go through the decompiled source code and retrieve the target system used APIs, which are what make up the first feature sets. The model is built using the numbers of permission-related APIs extracted as features from both the malicious and benign file datasets, and it can be used to forecast any file from the test dataset.

Random Forest Algorithm:

A random forest algorithm is used to classify the features after they have been extracted. If we break down the word, it consists of forest, which is a collection of decision trees, and random, which refers to the fact that we are sampling at random. When this approach is applied to a data set, a portion of the data is used as a training set, and the data is clustered into groups and subgroups. A decision tree is a structure that looks like a tree and is created by connecting data points to groups and sub-groups. The program then creates a forest out of several trees. However, each tree is unique since the variables are chosen at random for each split in the tree. Apart from the training set, the remaining data is utilized to forecast which tree in the forest produces the best categorization of data points, and the tree with the highest predictive power is displayed as output. The type of each program is then determined using a set of labels, with 1 denoting malware and 0 denoting benign files. By minimizing the uncertainty of the class labels, the decision tree splits the training set into two subsets with distinct labels at each node.

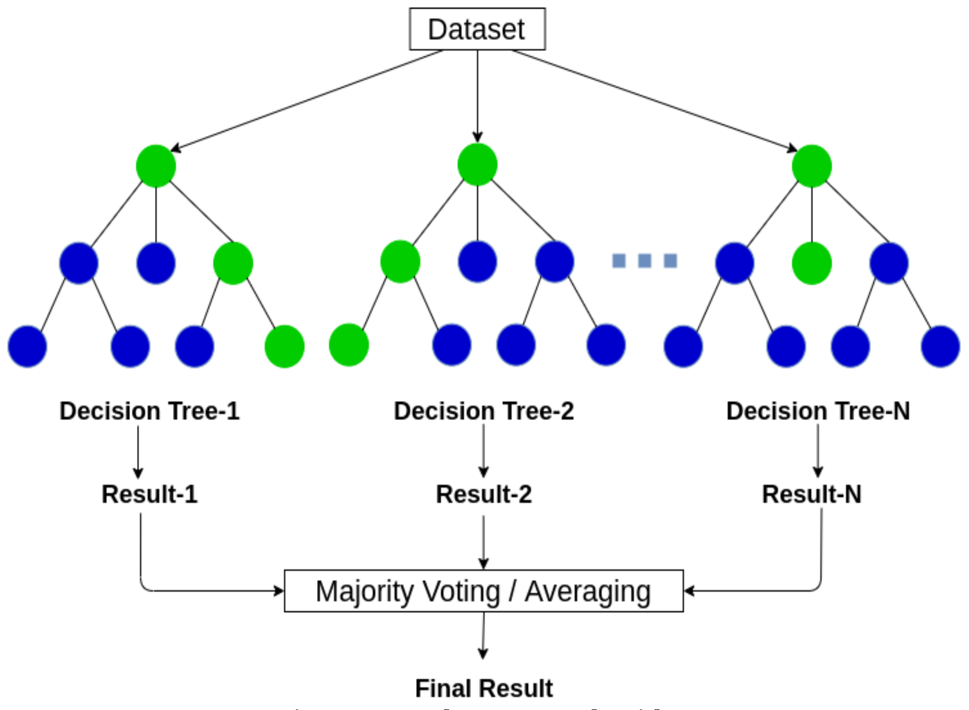

Figure 3: Random Forest Algorithm

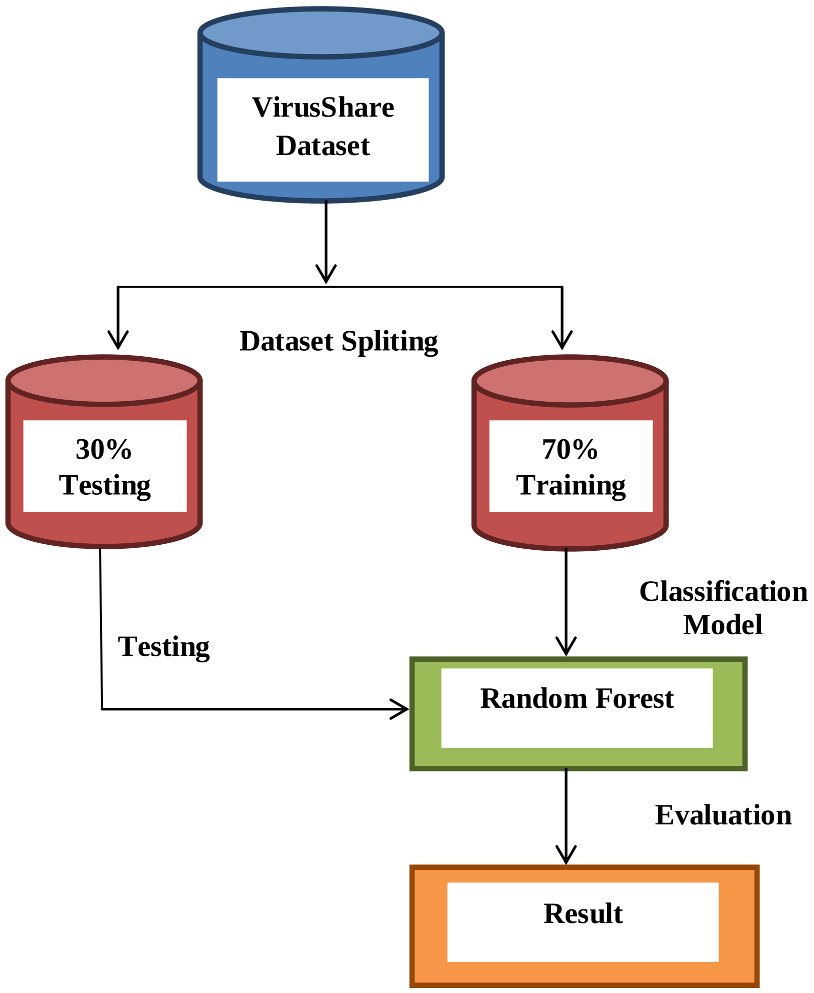

Figure 4: Formulation of Datasets

IV. RESULT AND DISCUSSION

To create a baseline for comparing evaluation metrics between k-fold cross-validation metrics and those achieved by testing on an unknown dataset, a 10-fold cross-validation approach was used with the entire dataset of 345,000 observations. This strategy trains the classifier iteratively on 70% of the training data before testing it on the remaining 30%. After 10 iterations, the results are calculated by computing the mean accuracy across all models. The standard classification measures of Precision (an indicative measure of erroneous positives), Recall (an indicative measure of false negatives), and F-measure are provided in Table 2. (a mean of Precision and Recall). Each has a maximum score of 1.0.

The dataset was read into the working directory using the pandas library in python and analyzed if null values are using a heatmap function from the seaborn library. Feature extraction was used in reducing the columns of the dataset by selecting just two import features, which are the name and label columns. The name column contains various applications while the label column represents the class for each of the applications. A count plot that displays the number of benign files and malicious files can be seen in figure 5. Tokenizer was used in separating each of the text in the name column into tokens for easy implementation while the LabelEncoder function was used in converting the label column from non-numeric values to the numeric value. The reduced dataset was split into training data and testing data. 85% of the data was used for training, while 15% of the data was used for testing. X_train, Y_train variable was used in holding the training data while X_test, Y_test are used in holding the testing data. X_val and Y_val were also used in validating the data. The training and validation data were passed into the random forest for training a model for detecting malicious software. The model was trained using a batch size of 32 and an epoch value of 20. The performance of the trained model can be seen in Figures 5, 6, 7, 8, and 9.

Table 1: Malware Dataset

Number	Family	Family Name	No. of Variant
1	Dialer	Adialer .C	122
2	Backdoor	Agent.FYI	116
3	Worm	Allaple.A	2949
4	Worm	Allaple.L	1591
5	Trojan	Alueron.genj	198
6	Worm	AutoIT.Autorun.K	106
7	Trojan	C2Lop.P	146
8	Trojan	C2Lop.genG	200
9	Dialer	Dialplatform.B	177
10	Trojan Downloader	Dontovo.A	162
11	Rogue	Fakerean	381
12	Dialer	Instantaccess	431
13	PWS	Lolyda.AA 1	213
14	PWS	Lolyda.AA 2	184
15	PWS	Lolyda.AA 1	123
16	PWS	Lolyda.AT	159
17	Trojan	Malex.gen!J	136
18	Trojan Downloader	bfuscator.AD	142
19	Backdoor	Rbot!gen	158
20	Trojan	Skintrim.N	80
21	Trojan Downloader	Swizzor.gen!E	128

Table 2: Summary of 10-Fold Cross-Validation Classification Results.

ALGORITHM	ACCURACY (%)	PRECISION	RECALL	F-MEASURE
Random Forest	99.95	0.998	0.999	0.9995
Bayesian Network	92.90	0.928	0.929	0.9290
SVM	70.39	0.702	0.703	0.7039

Table 3: Breakdown of Training Data used during Initial Training

Process Classification	Number	Percentage (%)
Malware	95,191	9%
Benign-ware	953,384	91%

Table 4: Breakdown of Malware by Classification used in Training Data

Malware Classification	Number of Samples
Trojan	10
Spayware	18
Unknown	12
Ransomware	17
Backdoor	7
Worm	7

Virus	29

Table 5: Breakdown of Classification Report for Random Forest Training Runs

Precision	Recall	F1	False Positive	False Negatine	Run
1	1	1	1	1	94
1	0.79	88	21.41	0	1
1	0.97	99	2.82	0	1
1	0.98	99	1.64	0	1
1	0.99	99	1.29	0	1
1	0.99	1	0.92	0	2

Table 6: Breakdown of Malware Classifications Used During Live Testing Of Including Detection Rates

Malware Classification	Number of Samples	Detection Ration (%)
Trojan	47	99.8%
Ransomware	30	100%
Spyware	30	99.8%
Backdoor	22	100%
Virus	74	100%
Adware	15	99.9%

```
[Text(0, 0, 'Benign'), Text(1, 0, 'Malware')]
```

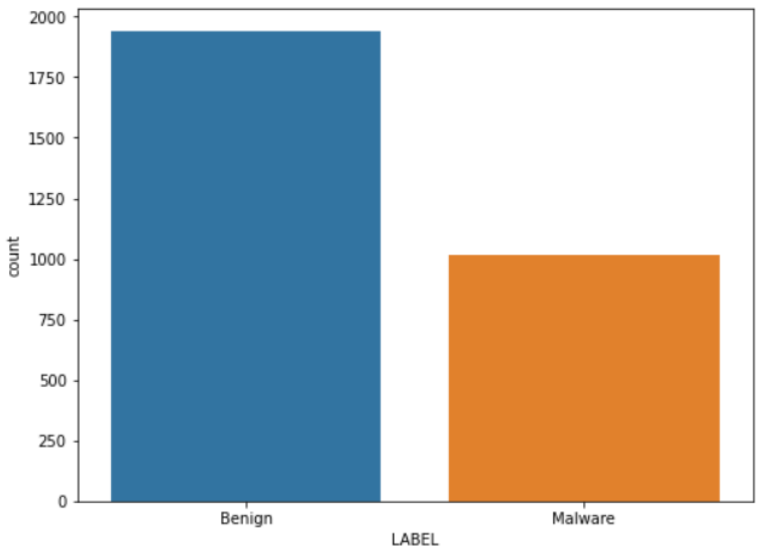

Figure 5: Count Plot of Benign and Malware

These show the total number of malicious and begin files present on the dataset.

Confusion Matrix

The confusion matrix shows the number of prediction results of the classification problem. It shows the summary of the number of correct and incorrect predictions with a count value broken down by down. The confusion matrix is a technique for summarizing the performance of a classification algorithm. This is because classification accuracy alone can be misleading if an unequal number of observations in each class. The confusion matrix of the random forest can be seen in figure 6

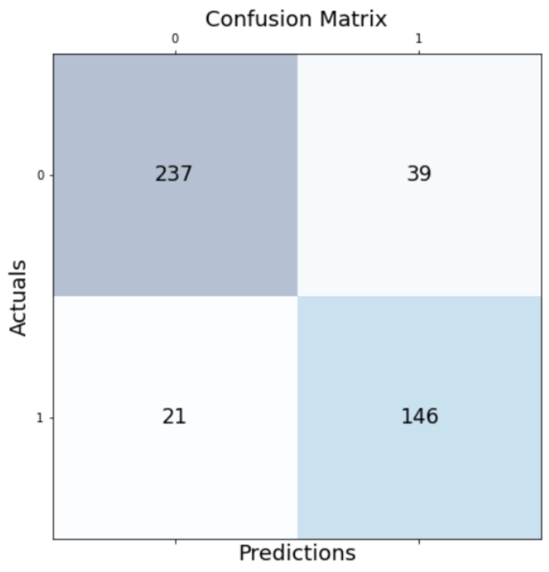

Figure 6: Confusion Matrix

The confusion matrix shows that the model classified the benign class correctly 237 times and falsely 39, and it classifies the malicious class correctly 146 times and falsely 21.

Performance Analysis

The performance of the trained random forest model was carried out by plotting a classification report on the trained dataset. The Classification report is used to measure the quality of predictions from the random forest Model to check how many predictions are True and how many are False. More specifically, True Positives, False Positives, True Negatives, and False Negatives are derived while making a prediction. The classification report for the random forest model for malware detection and classification can be seen in figure 7 below

The classification report shows the accuracy level of the test data to be 87%, precision for the benign class to be 92%, and that of the malicious files to be 79%. The support shows the total number of the prediction made by the random forest to be 276 for benign class and 167 for malware class.

Evaluation and Validation Results

Cross-Validation is a very powerful tool. It helps in giving a better use of data, and it gives much more information about the performance of the random forest. In complex machine learning models, it's sometimes easy not to pay enough attention and use the same data in different steps of the pipeline.

V. CONCLUSION

Malware, including in mobile and smart devices, has become more sophisticated and greater in frequency during recent years. Despite the existence of several defensive tools and procedures, malware detection, analysis, and classification remain difficult tasks because malware producers continue to hide information in attacks or adapt cyber-attacks to avoid newer security measures. It is critical to evaluate malware behavior and categorize samples to design an effective program to prevent malware attacks. A deep learning technique for malware classification employing random forest (RF) architecture has been developed for this purpose.

VI. RECOMMENDATION

The study of malware will never go away due to the everyday new creation of malware. Future research should continue to deepen it to improve understanding of the phenomenon, for better proactive protection. In general, malware detection on devices is expected to be a popular topic in the future. Because of the growing use of cellphones, an increasing number of individuals will be at risk from forthcoming malware. And, as these devices gain additional capabilities, their use will expand, resulting in an increasing number of victims. Creating online stores, as Apple and Android have done, was a decent first step in protecting customers from harmful software. Before being made available in these stores, testing is performed to ensure that submitted programs do not include any undesired behavior. Unfortunately, malware authors have devised methods to circumvent these safeguards, emphasizing malware's pervasiveness. On-device protection is the only way to retain security at an acceptable level in these situations. On-device detection was not possible on a broad scale of the device five years ago owing to inadequate hardware. Devices now run at 1 GHz, and new dualprocessor architectures have been announced, allowing for on-device detection. As a result, future research will focus on on-device detection to protect smartphone users from viruses.

REFERENCE

[1]. Alazab, M.; Venkataraman, S. & Watters, P. (2010). "Towards Understanding Malware Behaviour by the Extraction of API calls", Accepted to the IEEE 2nd Cybercrime and Trustworthy Computing Workshop (CTC2010), 2010.

[2]. Stolfo, S.; Wang, K. & Li, W. (2006). "Towards stealthy malware detection", Malware Detection, Advances in Information Security, Springer, 27, 231-249.

[3]. Venkatraman, S. (2009). "Autonomic Context-Dependent Architecture for Malware Detection", Proceedings of International Conference on e-Technology (e-Tech2009), International Business Academics Consortium, ISBN 978-986-83038-3-6, 8-10 January, Singapore, 2927-2947.

[4]. Mamoun Alazab, Andrii Shalaginov, Abdelwadood Mesleh & Albara Awajan (2020), 'Intelligent mobile malware detection using permission requests and API calls', *Future Generation Computer Systems*, vol. 107, pp. 509–521, doi:10.1016/j.future.2020.02.002 [Q1, IF 5.768, ERA2010 RANK - A]

[5]. Moser, A., Kruegel, C. & Kirda, E. (2020). **Limits of static analysis for malware detection** 23rd Annual Computer Security Applications Conference, pp. 421-430 Miami Beach, Florida, USA.

[6]. Ahmed, F., Hameed, H., Shafiq, M. & Farooq, M. (2009) **Using spatio-temporal information in API calls with machine learning algorithms for malware detection** Proceedings of the 2nd ACM workshop on security and artificial intelligence. ACM, 55 - 62.

[7]. Aslan, O. & Samet, R. A. (2020). Comprehensive Review on Malware Detection Approaches. IEEE Access, 8, 6249 – 6271.

[8]. AV Test (2015) [Online] Available from: https://www.av-test.org/en/statistics/malware/

[9]. Bayer, U., Moser, A., Kruegel, C. & Kirda, E. (2006) **Dynamic analysis of malicious code.** Journal of Computer Virology, 2 (1), 67-77.

[10]. Casey, E. (2011). Foundations on Digital Forensics. Ch. 1. Retrieved from: http://booksite.elsevier.com/samplechapters/9780123742681/Chapter_1.pdf

[11]. Christie, C. J. (2006). 'Former UBS computer system manager gets 97 months for unleashing 'logic bombs' on company network. US DOJ Press Release (News).

[12]. Continella, A., Guagnelli, A., Zingaro, G., De Pasquale, G., Barenghi, A. & Zanero, S. (2016). **Shieldfs: A self-healing, ransomware-aware filesystem** Proceedings of the 32nd annual conference on computer security applications, ACM, New York, NY, USA 336 – 347. ACSAC '16.

[13]. Cui, Z., Xue, F., Cai, X., Cao, Y., Wang, G. & Chen, J. (2018). Detection of Malicious Code Variants Based on Deep Learning. IEEE Trans. Industrial Information, 14, 3187 – 3196.

[14]. Dahl, G., Stokes, W., Deng, L. & Yu, D. (2013). **Large-scale malware classification using random projections and neural networks** 2013 IEEE international conference on acoustics, speech and signal processing, IEEE, 3422 – 3426.

[15]. Damodaran, A., Troia, F., Visaggio, C., Austin, H. & Stamp, M. (2017). **A comparison of static, dynamic, and hybrid analysis for malware detection.** Journal of Computer Virology and Hacking Technology, 13 (1), 1 – 12.

[16]. David, O. & Netanyahu, S. (2017). **Deepsign: Deep learning for automatic malware signature generation and classification** 2015 International Joint Conference on Neural Networks (IJCNN), 1 – 8.

[17]. Dini, G., Martinelli, F., Saracino, A. & Sgandurra, A. D. (2021). **MADAM: a Multi-level Anomaly Detector for Android Malware** Springer Berlin Heidelberg, Berlin, Heidelberg 240 – 253.

[18]. Matrosov, A., Rodionov, E., Harley, D. & Malcho, J. (2011). Stuxnet Under the Microscope (Revision 1.31). Technical report. Retrieved from www.go.eset.com/us/resources/ white-papers/Stuxnet_Under_the_Microscope.pdf.

[19]. Mukamurenzi, M. N. (2008). Storm Worm: A P2P Botnet, NTNU (Norwegian University of Science and Technology) white paper.

[20]. Naeem, H., Ullah, F., Naeem, M. R., Khalid, S., Vasan, D., Jabbar, S. & Saeed, S. (2020). The detection of malware in the industrial internet of things is based on a hybrid image visualization and deep learning approach. Ad Hoc Network, 105, 102154.

[21]. Poulsen, K. (2013). Slammer worm crashed Ohio nuke plant network, Security Focus, available online at: http://www.securityfocus.com/news/6767

[22]. Arshad, H., Khan, M. A., Sharif, M. I., Yasmin, M., Tavares, J. M. R. S. & Zhang, Y. D. (2020). Satapathy, S.C. A multilevel paradigm for deep convolutional neural network features selection with an application to human gait recognition. Expert System, e12541.

[23]. Dumka, N., Matthias, D., Bennett, E. O., (2021). An efficient Model for detecting Uniform Resource Locator (URL) phishing using machine learning technique. International Journals of Computer Techniques, 8, 1- 3.