Journal of Theoretical and Applied Information Technology

15th December 2023. Vol.101. No 23
© 2023 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN IMPROVED MALWARE VARIANT DETECTION MODEL BASED ON HOMOGENEOUS STATIC HYBRID FEATURES AND A DATA AUGMENTATION TECHNIQUE

AZAABI CLETUS[1], ALEX AKWASI OPOKU[2], BENJAMIN ASUBAM WEYORI[2]

[1]School of Sciences, University of Energy and Natural Resources, Sunyani, Ghana & St. John Boscos College of Education, Navrongo, Ghana. (**Corresponding Author)**
[2]School of Sciences, University of Energy and Natural Resources, Sunyani, Ghana, Email: Email .
[2]School of Sciences, University of Energy and Natural Resources, Department of Computer and Electrical Engineering, Sunyani, Ghana.

Email: cleinhim@yahoo.com, alex.opoku@uenr.edu.gh, benjamin.weyori@uenr.edu.gh.

ABSTRACT

The use of Machine learning has become the de-facto standard for malware defense due to the limitations of signature-based, heuristic-based and other cloud-based techniques. However, poor malware features, class imbalance problems and malware obfuscation remain challenges facing malware researchers. To ensure efficient and resilient detection in the face of these challenges requires novel models that adopt innovative techniques to improve malware detection. The paper proposed an improved novel malware variant detection model based on Homogeneous Multi-Static Hybrid features (HMSHF), obfuscated malware dataset and Synthetic Minority Oversampling Technique (SMOTE). A malware dataset comprising 11678 malware files from virusTotal.com and 3963 benign files obtained from windows environment was used for the study. We extracted 'fine-grained' strings, APIs, and opcode features from static disassembly of the malware dataset. We trained and tested a Random Forest (RF), Support Vector Machine (SVM), GradientBoost (GB), and eXtremGradientBoost (XGB) ensemble algorithms before and after obfuscating the malware dataset. We hybridized the features into HMSHF for training and testing the ensembles before and after the malware was obfuscated. We evaluated the performance of the models using individual features and the hybrid features before and after obfuscations. To overcome the class imbalance problem, we applied the SMOTE technique on the training set with the HMSHF. The proposed hybrid features showed effectiveness and efficiency in classifying malware with 99.87% accuracy without data augmentation and 98.8% accuracy with SMOTE data augmentation. Consequently, the paper concluded that, the proposed technique improved malware detection and demonstrated resilience against obfuscation compared with the state of the art. Thus, the approach can be adopted for the detection of known, unknown and zero-day malware. Notwithstanding the improved performance, this work is not without limitations; the use of feature selection instead of feature extraction, and use of ensembles instead of other Deep learning techniques and SMOTE instead of other data augmentation methods. Thus, future works will adopt the approach and use Principal Component Analysis (PCA) dimensionality reduction techniques; employ deep learning techniques and apply other data augmentation techniques to observe the performance.

Keywords: *SMOTE, Malware, Ensemble Learning, Ransomware, Malware Features, Signature-Based Detection*

1. INTRODUCTION

Globally, there is an exponential growth in malware samples and Potentially Unwanted Software (PUS). These exploits limitations in existing signature-based, heuristics and other non-intelligent detection methods resulting in exposures [1], [2], [3], [4]. The increased malware volumes, variety and complexity coupled with the use of innovative obfuscation techniques renders existing defense systems inefficient and ineffective requiring the use of novel and innovative defense techniques that are more resilient and robust in the face of the prevailing malware ecosystem [5], [6]. Malware are script, codes and or software with malicious intension and aimed at compromising systems mainly yo gain unauthorized access, disclosure and or modification. Malware attacks target military installations, critical infrastructure, corporate entities, and individuals, leading to exposures [3]. Malware exposure compromises confidentiality, integrity, and availability principles, leading to reputational loss, financial loss, legal issues, and compliance issues, among others. According to [2], one of the leading malware-testing labs based in Germany, an average of 450,000 malware and PUS are registered daily. The authors or purveyors of this malware mainly aim at stealing information, disrupting systems, and ensuring command and control of systems, all geared towards compromising information systems for their parochial motivations [7]. Malware as malicious software that performs malicious actions, they comes in the form of codes or scripts [8]. Some examples of malware include, but are not limited to, rootkits, Trojans, worms, ransomware, spyware, and others. These malware are not only increasing in volume but also in complexity and variety and exploits the existing signature-based or signature-matching defense systems[9]. The increased volumes, variety and complexity of current malware renders traditional signature-based defense systems and others such as heuristics and cloud-based techniques inefficient and ineffective. In addition, malware authors are adopting innovative obfuscation techniques to conceal their identities from being detected by these signature-based or matching techniques [9]. These limitations coupled with the need for adaptability and generalizability of the tools and techniques has made automated Machine Learning (ML) techniques the current research focus for automated malware detection [10]; [11].

However, even though ML techniques are extensively explored in malware defense with positive results, there are still some limitations or challenges that militates against the achievement of the full potentials of the techniques. These limitations includes poor features for the training of ML models and subsequent testing of the algorithms, the problem of class imbalance in malware datasets resulting in the ‘accuracy paradox’, and challenges with the detection of obfuscated malware and their variants [12]; [13].

To improve upon the existing works and to fill the identified gaps requires novel approaches that provides efficient features for efficient malware detection and classification. In addition, there is the need to improve the class imbalance problem to overcome the ‘accuracy paradox’ in models leading to improved detection accuracy, since the use of the accuracy metric with imbalance datasets results in models/algorithms being bias towards majority samples at the expense of the minority. Finally, improving resilience of ML models against obfuscated malware by exposing the models to variety of anti-static and advanced obfuscation techniques. Therefore, to improve malware classification, requires the use of innovative techniques that overcomes these gaps based on the application of various schemes that provides efficient and effective malware features, techniques that improves the accuracy paradox such as data augmentation methods, and ample exposure of the learning models to the relevant anti-static and advanced obfuscation techniques.

In recent times, the use of different machine learning approaches to enhance malware detection using static-signature-based features has been explored. In [15], the authors proposed a method for extracting features from a malware dataset and using Random Forest (RF), Deep Neural Network (DNN), and XGBost for classification, and reported an accuracy of 96.3%. Similarly, the authors in [16] proposed an opcode-based frequency (opcode frequency) as a malware vector for classification and used both supervised and unsupervised models involving RF and a DNN model with a reported accuracy of 99.78%. Similarly, the use of a combination of static and dynamic features for the classification of Android malware binaries using GradientBoost, Decision Tree, Nave Bayes, and RF models was proposed and implemented in [17] and reported an accuracy level of 96% with the RF algorithm. While the authors in [4] proposed a small-scale and easy feature extraction that includes the sizes and permissions of PE features to classify malware families with machine-learning algorithms and reported 99.40% with the RF model, Other approaches used heterogeneous hybrids involving static and dynamic features extracted from both methods to train and test machine-learning algorithms [18]; [10]; [19]. In the same vein, [20], proposed the use of permissions with a lightweight technique for malware detection. They experimentally demonstrated its efficiency using real android malware samples. They considered only one aspect of the vulnerabilities of the features and ignored the others such as API Calls. The use of intents was explored in [21] where both explicit and implicit intents as semantically rich features for the encoding of malicious intensions were used for the study. Their proposed system performed encoding and extracted explicit and implicit intents, intent filters and other permissions. Moreover, in [22], the authors explored an approach called MalDozer relying on neural network taking inputs from raw API calls in the order as they show up in the .dex file mainly for android. They reported that during training, their approach could automatically classify malicious patterns using only the sequences of the raw method present in the assembly code. Besides, the authors in [23] experimented with the use of a framework with several multidimensional features from applications useful for malware classification. A multimodal deep learning network based on opcode features APIs, permissions and string features were used. Their experiment with malware dataset from virus share received an accuracy of 98%. Finally, in [24], a hybrid deep learning model using autoencoder and a convolutional neural network (CNN) was proposed to improve accuracy of the detection of malware using multiple features and achieved 98.8% accuracy.

Notwithstanding the plethora of studies using ML techniques as catalogued above, a critical analysis of the works shows that, not much effort is placed on the provision of efficient features from homogeneous sources such as static-static hybridization, the handling of the accuracy paradox and the exposure of the models to sufficient anti-static and advanced obfuscation methods as explained in the proceeding paragraphs

Firstly, the analysis of the performance measures used in the evaluation of the models show that, most authors employ the accuracy metric as a performance measure with class imbalance datasets. This results in the usual ‘accuracy paradox’ where the models tend to skew or be bias towards the majority sample at the neglect of the minority class to be predicted. Using the accuracy metric with an imbalance class dataset without the treatment of the imbalance results in poor classification accuracy. This occurs where the model is bias towards the majority class at the expense of the minority.

In addition, whiles a plethora of works employs heterogeneous hybridization of features, there is less focus on the use of homogeneous hybridization of features for efficient and improved malware detection. Thus, the paper demonstrates that the use of homogeneous hybridization of static-based features results in improved effectiveness and efficiency in malware detection accuracy.

Besides, the use of obfuscation as a means to outwit automated detection systems where the malware changes its form and shape (mutation) resulting in polymorphic, metamorphic, oligomorphic and other variants of known malware remain a challenge in the current ML research [14]; [20]; [26]. Improving the resilience and robustness of the models against obfuscation requires adequate exposure of the models to various obfuscation techniques during the training phase of the modelling.

Consequently, this paper filled these gaps by proposing an improved malware variant detection model based on homogeneous multi-static hybrid features and a data augmentation technique aimed at improving efficiency and resilience of models against malware attacks and variants of known and unknown malware. To achieve this goal, the following objectives guided the study:

1. To provide efficient malware features with high feature importance for efficient malware classification
2. Apply data augmentation technique to improve malware classification and avoid the ‘accuracy paradox’ associated with class imbalances in malware dataset.
3. Improve malware variant detection by using obfuscation techniques in malware detection and classification.

Following these objectives, the paper provided efficient homogeneous hybrid features with high feature importance for efficient malware variant detection and classification. We applied the SMOTE data augmentation technique to overcome the problem of class imbalance, thereby overcoming the ‘accuracy paradox’, and improved malware detection. Finally, we demonstrated the resilience and robustness of the approach to malware variant detection by exposing the models to various forms of obfuscated malware samples.

We achieved this by proposing a HMSHF with ensemble classifiers based on the hybridization of string, Opcode, and API Call features extracted using static disassembly techniques for the training and evaluation of RF, SVM, GB, and XGB classifiers. The purpose was to provide efficient malware features for the efficient classification of malware and variants, improve performance accuracy, reduce false-positive rates, achieve resilience and robustness against malware obfuscation and other evasive techniques, and ultimately overcome the ‘accuracy paradox’. We obtained an imbalanced malware dataset comprising 11,678 malware and 3,963 benign ware from virusTotal.com and malwr.com. The dataset was preprocesses and Features were obtained based on the static disassembly of the malware dataset. Unlike the usual case of using all the features from the disassembly, we extracted only 'fine-grained' features; features with the highest predictive capability for the training of the models. We extracted static API Calls, Opcodes, and String features to train and test a RF SVM, GB, and XGB ensemble techniques. This is because our aim was to improve accuracy and stability where ensembles have shown to be good at that [27]. Using the individual features as our baseline mark, and to test the classification accuracy of our features in classifying malware, we evaluated the performance of the models on the individual features before and after the malware was obfuscated. Having obtained the baseline results with the features, we then hybridized or integrated the three extracted features into a integrated feature set and used it to train and test the models before and after obfuscating the malware dataset. The performance of the hybrid was compared with the individual features. The results of the proposed approach performed better than the individual features. With our dataset being imbalanced, we applied SMOTE to the training dataset to overcome the problem of imbalanced data and avoid overfitting and/or the "accuracy paradox." We evaluated the performance of the ensembles on accuracy and false-positive rate (FPR) on the hybrid features. Lastly, we compared the performance of our hybrid approach with that of the cited literature. From the results, our proposed approach (HMSHF) demonstrated the efficiency of the features for efficient malware classification, demonstrated improvement over the 'accuracy paradox', and finally, improved classification by the implementation of the SMOTE data augmentation technique, and showed resilience and robustness against malware obfuscation by exposing the models to various obfuscation techniques for learning and generalization.

On the basis of these achievements, our approach shows moderate novelty, the findings and results are exciting, and have some value, and our work therefore makes a modest contribution to the body of knowledge in malware detection in particular and the use of artificial intelligence (AI) or ML in cybersecurity in general. The moderate to high relevance of the approach to malware practitioners, its innovativeness, the rigorous technical procedures employed contributes to malware defense in particular and cybersecurity in general. Specifically, this study contributed to knowledge as follows:

- We proposed a novel (HMSHF) comprising strings, opcodes, and API call features through static disassembly of malware binaries resulting in improvement in malware and malware variant detection. By this, we have contributed to malware feature engineering, which is essential in building efficient classifiers for automated detection of malware, and offers unique feature set for known, unknown and variants of known malware.
- The application of SMOTE in balancing the imbalance dataset used in our study overcome the 'accuracy paradox' inherent in the imbalance dataset resulting in providing true performance accuracy compared to the cited state literature. Malware environments are highly imbalance as well as the datasets used for the experiments. This imbalances tends to make algorithms to be bias towards the majority class. This results in the 'accuracy paradox'. To overcome this phenomenon requires oversampling and under sampling techniques. Hence, the use of the SMOTE technique as data augmentation improved the accuracy of the models, which resulted in fare classification of both minority and majority sample in the dataset.
- Demonstrated the resilience of the proposed approach in overcoming malware obfuscations that malware authors employ to evade detection in signature-based detection systems. The use of static and advanced obfuscation techniques are used by malware authors to compromise systems. The use of different obfuscation techniques and the ability of the models to show high and same classification performance demonstrate the resilience and robustness of the model against malware and malware variant attacks.

Organization of the Paper.

We structured the rest of the paper as follows: we present the Materials and Methods in Section 2; in Section 3, the Results and Discussion is presented; whiles in Section 4, we present the Conclusion and future works of the study.

2. MATERIALS AND METHODS

This section of the paper discusses the methods and materials of the study following the methodological framework as shown in Figure 1 below. The methodology starts by providing: (1) context and motivation for the proposed approach, (2) description of the malware dataset and the preprocessing activities, (3) static disassembly of malware samples to extraction features using IDA Pro (4) feature selection technique used to select features (5) feature integration or hybridization (6) feature representation (7) selection and training of the models (8) evaluation of the performance. The description of each stage of the process follows.

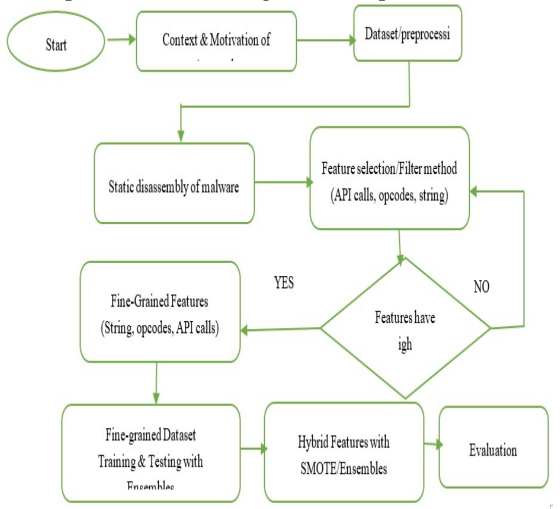

Start
Context & Motivation of
Dataset/preprocessi
Static disassembly of malware
Feature selection/Filter method (API calls, opcodes, string)
YES
NO
Features have igh
Fine-Grained Features (String, opcodes, API calls)
Fine-grained Dataset Training & Testing with Ensembles
Hybrid Features with SMOTE/Ensembles
Evaluation

Figure 1. Methodological Framework Of The Study.

2.1. Context and Motivation for the Proposed Approach

This sub-section discussed the context and motivation for the proposed approach. Specifically, we gave a brief description of static analysis and its limitations. The obfuscation methods or techniques used by malware attackers to evade detection. Finally, we present the motivation and justification for the use of our proposed approach.

2.1.1. Static Malware Analysis and Limitations

The first stage in malware analysis is usually static techniques using tools in the form of cryptographic hashes, fuzzy hashes, import hashes, and others [26]. It is the analysis of a malware binary without the actual execution of the code. Static malware detection is based on conventional malware analysis techniques based on the extraction of static features using reverse engineering methods. To perform static analysis, therefore, requires unpacking or decoding the malware sample, through which features can be obtained. Examples of static features: pefiles, strings, opcodes, n-grams, and function length frequencies. Static analysis techniques are faster, cheaper, and always the first point of call for malware analysts. However, their limitations or drawbacks include but are not limited to obfuscation (packing, crypting), dynamic code loading, which results in poor detection accuracy, and susceptibility when a malware binary is obfuscated. This is because these techniques (signature- based or matching) assumes that, once a malware is identified, its byte sequence remains the same. This is not the same with the current malware as they can change their shape and form depending on the environment. Hence, this requires the use of machine learning and other adaptable methods that learn from experience instead of static codes [26]; [5]. Notwithstanding the limitations in the face of obfuscation and other evasive techniques, the use of static disassembly is known to reveal some malware features that can be used for training adaptive algorithms [26].

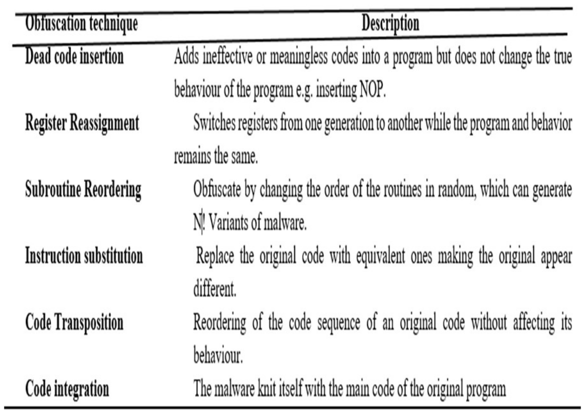
Thus, we envisaged that using different static analysis techniques to extract some features and training ensembles has the potential to improve performance accuracy and resilience against obfuscation and other evasive techniques employed by malware developers to exploit static techniques. To be able to develop novel tools and techniques, requires an understanding of the the types of obfuscation techniques employed by malware attackers and how we employed obfuscation in our dataset to test the resilience of the models against malware obfuscation.

2.1.2. Obfuscating the Malware

Malware obfuscation is the deliberate hiding of a malware binary's identity with the goal of evading detection by the detection system. This is essential because malware defenders have developed methods throughout time to prevent malware from causing damage to their computing systems. Its created to avoid detection by anti-virus software or malware detectors disguise Malware, opening the door to exploitation. [12] claims that several methods are used to mask the genuine nature of malware. He indicated that, malware is obfuscated using techniques such as, dead code insertion, code transposition, subroutine reordering, instruction

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

replacement, register reassignment and others. The strategies and their descriptions are listed in Table 1.



Obfuscation technique	Description
Dead code insertion	Adds ineffective or meaningless codes into a program but does not change the true behaviour of the program e.g. inserting NOP.
Register Reassignment	Switches registers from one generation to another while the program and behavior remains the same.
Subroutine Reordering	Obfuscate by changing the order of the routines in random, which can generate N! Variants of malware.
Instruction substitution	Replace the original code with equivalent ones making the original appear different.
Code Transposition	Reordering of the code sequence of an original code without affecting its behaviour.
Code integration	The malware knit itself with the main code of the original program

For the purpose of this study, we used dead code insertion and code transposition to disguise the malware. We inserted null codes into the binaries and in some of the samples we transposed the codes. The aim was to conceal the identity of the malware samples to observe the performance of the models after obfuscation. The use of the obfuscation techniques changes the byte sequence and hash values of the malware binary making it difficult for the signatures to be matched for detection. However, the potency of the malware is still maintained in the masked state. This phenomenon has posed detection challenges requiring the use of novel adaptable techniques with generalizability, and memorability for improved malware detection. Hence, the motivation for our proposed approach to use the static disassembly to obtained relevant features to train adaptable methods for improved detection.

2.1.3. Motivation for Our Proposed Homogeneous Hybrid Approach

Current arguments on the use of heterogeneous hybrids between static and dynamic environments with base machine learners have been explored with some considerable success [19]. However, the issue of poor features, class imbalances, and the use of obfuscation techniques remain some of the challenges [28]. We argued that static disassembly of malware samples can reveal relevant features that can be used to improve malware classification. By statically disassembling the malware binaries, extracting only 'fine-grained' instead of the usual collection of all features revealed in the disassembly process can produce relevant features for effective and efficient malware classification. Additionally, the use of SMOTE data augmentation technique to rebalance the dataset leads to improvement in malware classification accuracy and overcome the class imbalance problem where the models tend to be bias towards majority sample at the expense of the minority sample. Thirdly, we proposed that to improve resilience against obfuscation requires the exposure of the models to the various obfuscation techniques during the training phase of the Ensemble techniques. The choice of the ensembles is based on the fact that, the traditional of conventional ML techniques are largely unstable in non-stationary environments, susceptible to bias, variability and noise. Therefore, motivated by this, we proposed a HMSHF with ensemble and data augmentation technique (SMOTE) for efficient and resilient malware classification. The purpose was to provide efficient features for malware classification, using data augmentation technique (SMOTE) to overcome the problem of class imbalance resulting 'accuracy paradox', and the demonstration of resilience and or robustness of the proposed technique against obfuscated malware detection.

2.2. Malware Dataset and Preprocessing

This section discusses the modelling process from including datasets and processing activities, feature-engineering process, the description of the ensemble models, the performance evaluation methods.

2.2.1. Dataset and Preprocessing

For this experiment, we collected malware samples from two main sources; VirusTotal.com and for a four year period (2017-2019) and (2019-2021). This was necessary because malware landscape is evolving and revolving and new, novel and variants of known malware come into being [29]. By including these malware samples, we are sure to have almost all new and variants of known malware. To obtain benign samples for this study, we extracted these files from the windows operating systems. Using virusTotal.com, we checked whether a sample is benign or malicious when all the virus scanners flags it as malware or non-malware. Consequently, we combined these malware and benign ware to form our experimental dataset comprising 11,678 malware and 3,963 benign ware and made up of different malware families. Tables 3 and 4 shows the malware categories and the total dataset size used for the study respectively.

Table 2. Malware categories and Samples

Malware category	Samples
Addware	2108
Backdoors	998
Downloaders	797
Virus	883
Trojans Spy	1123
Trojan Droppers	234
Worms	1732
Spyware	974
Others	2829
Sub-Total	**11,678**
Benign ware	3963
Total Dataset	**15, 641**

Table 3. Composition of the Experimental Dataset.

Dataset type	Numberofthesamples	Category
Malware	11678	Varied (from 2017-2021)
Benign ware	3963	Varied (from2017-2021)
Total Dataset	15,641	Mixed

From the data exploration stage, it was realized that, the malware dataset was imbalanced. This phenomenon if not managed well may result in what is usually known as the "accuracy Paradox" where the prediction would be skewed or bias towards the majority sample. Consequently, the composition of the dataset shows that it is imbalance made up of 74.66% and 25.34% making it mildly skewed as shown in figure 2. To overcome data imbalance to obtain true accuracy of models, there are a number of methods to apply as a means of resampling. The most common are undersampling and oversampling with most users adopting the oversampling techniques relative to the undersampling. This is because; undersampling may remove instances of data with relevant information [25]. For this reason, we opted to use oversampling technique known as Synthetic Minority Oversampling Technique (SMOTE).

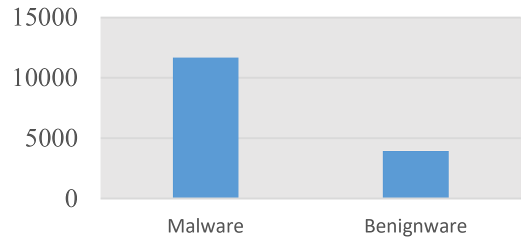


Figure 2. Imbalance Dataset

2.2.2. Applying SMOTE

To overcome this imbalance in the data, a number of under sampling and over sampling techniques can be applied based on the merits and demerits [30]. For the purpose of this study, we used the SMOTE using SMOTE algorithm in python. This creates synthetic data points from the minority sample to balance the dataset. This approach using the algorithm results in balancing the data as shown in figure 3. The two data samples are now balanced which leads to prediction of both malware and benign ware equally and presents the true prediction or classification of the models. By this approach, the bias and or skewedness of the model is prevented resulting in overcoming the 'accuracy paradox'.

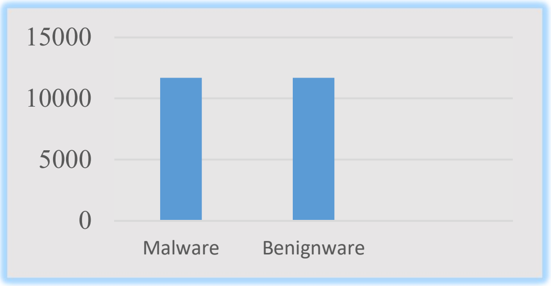


Figure 3. Balanced Data After Applying SMOTE.

Thus, to test the proposed approach, we applied this data augmentation techniques at the hybrid level for improved accuracy using sklearn. The code snippet of the SMOTE algorithm from the sklearn is as shown below in figure 4:

```
From imbalance.over_sampling import SMOTE
Counter = counter (y_train)
Print ('before', counter)
# oversampling the traini dataset using SMOTE
Smt = SMOTE ()
#X_train, y_train =smt.fit _resample (X_train, y_train)
X_train_sm, y_train_sm = smt.fit_resample(X_train, y_train)
```

Figure 4. Code Snippet of the Smote Algorithm

2.3. Feature Engineering.

Feature Engineering (FE) the use of domain knowledge to identify or select and transform the most relevant variables or attributes from a given raw data for training MLs for predictive modelling. It refers to the process of representing a real-world problem in a manner that makes it possible for machine learning techniques [32]. Thus, in searching for relevant features, FE process transforms the extracted features/data into relevant representative features that underlies the problem leading to improvement in the performance of the ML.

2.3.1. Static disassembly of Malware and Feature Selection

Features play a critical role in machine learning. [31] suggest that machine-learning detectors start and die based on the features. Malware features can be obtained using static analysis or dynamic analysis [26]. To obtain features for the training of our ensemble models, we extracted three features: strings, opcodes, and API calls. For the purpose of this work, we used static disassembly of malicious binaries to extract the needed string, opcode, and

API call features as input for the training of our ensemble models. Generally, most works tend to use all the features revealed during disassembly to train models. This is problematic because, some of the features are not representative of the malware but packed and crypted features that does not contribute to the predictive value of the model. This often result in poor prediction or classification and results in redundancy and the usual *curse of dimensionality*.
Thus, we extracted only the 'fine-grained' features with high predictive importance to form our feature set. This approach was relevant to obtain the needed features to train our models, which are the inputs for the models to learn from and use that experience to generalize [31]; [5]. Hence, increasing the quality of our prediction, avoiding redundancy of features and improved processing time. In all, 16 features from each of the Strings, Opcodes, and API Call features were retained after the feature selection process as shown in Table 4.

Table 4. Extracted String, Opcode and API Call Features

Features	Number of sample
Strings	16
Opcodes	16
API Calls	16
Total Features	48

2.3.2. Feature Selection

In ML application and in real-life, it is seldom that the attributes in a given dataset would all have usefulness in the prediction of a phenomenon. Thus, when there are many attributes it results in redundancies that reduces the generalizability of the model, impact negatively on the accuracy of the model of classifier and also results in the model complexity leading to high computational resource requirements[31]; [26]. Feature selection is a technique of finding the best set of features that enables the building of optimized models of a studied phenomenon. For example, given data N, with input dimension d, and selected dimension k, selecting features k that gives the most information of the problem and discarding the others, i.e. (d-k) dimensions [31].
These techniques can be Filters, wrappers and embedded and each of them has their potentials and limitations. Filter-based selection methods checks for correlation with the dependent variable. They are faster and less overfitting. They include information gain, chi-square, and variance methods. In wrapper techniques, models are build and the best model is chosen. Examples of these are genetic algorithms (finding subset of the features with relevant information and using them). Recursive techniques (removes all the weakest features with low feature importance) and sequential methods (add highest features together). They are computationally expensive and prune to overfitting. However, they have best performance and select only optimal features. Embedded feature technique also makes a model and select the model features with the best feature. They are faster as filters, have more accuracy that filters, less prune to overfitting.
String features are plain text in nature usually encoded in executable files used by malware attackers. They are usually found in windows systems such as Get*LayOut, SendMailFail, SendMail, GetCurrentProcess.* Thus, to extract string features we used the String Utility from Microsoft to search for the executable files or binaries from both ASCII and Unicode. In addition, we used IDA Pro for the automatic disassembly of the codes to obtain the string features. The disassembly revealed several features. However, a critical analysis of the features showed that most of the revealed features were packed and obfuscated and would not contribute to the predictive capability of the model, hence, only feature with high importance was used.
Opcodes (operational code) is an instructional level machine language used to identify operations to be executed. We disassembled the collected files to extract the instruction level sequence in assembler language using ndisasm tool. Examples of opcode sequences include Mov, Pop, and Push [17].
Finally, we disassembled Portable Executable (PE) files from dynamic link libraries of windows systems in the Win32 PE binaries. We extracted API Call features using windows systintemals found in Microsoft systems. The table below show the sample API Call features extracted for the study. Sample of all the Features are as shown in table 5.

Table 5. Sample Features From Static Analysis Methods.

Type of Analysis	Extracted Features
String Feature	*CreatProcessA,Sleep, ExitProcess, OpenMutexA,*
Opcode Feature	*Add,SetNo, Lodsq,Move,Test, Stosq, Syscalls*
API Calls	*GetHostByAddr, GetHostByName,Select, send, Connect*

Therefore to reduce the dimensionality of the features extracted , we used a feature selection Fisher's Score which is one of the filter methods. it is one of the most widely used supervised feature selection methods. this algorithm works by calculation the score of the features and ranking them in descending order of magnitude. Based on this rank, we extracted the relevant features for the training and integration. The code snippet of the Fisher Score algorithm is as shown figure 5.

```
from skfeature . function. similarity_based
import fisher_score
import matplotlib.pyplot as plt
% matplotlib inline

# calculating scores
Ranks =fisher_score. Fisher_score (X, Y)

#plotting the ranks
Feat_impotances =pd. Series(dataframe.
Columns(0: len(dataframe. Columns)-1])
Feat_impotances .plot(kind='barh', color
= 'teal')
Plot. Show ()
```

Figure 5. Code snippet of the Fisher's Score Algorithm

2.3.3. Hybrid Integrated Malware Feature set and Representation

To obtained uniform features for the training of our models, the extracted strings, opcodes sequence and API Call features extracted were integrated into a HIMF. This is shown diagrammatically in figure 6 below. The features were concatenated and vectorized into binary 0 and 1 for benign and malware respectively.

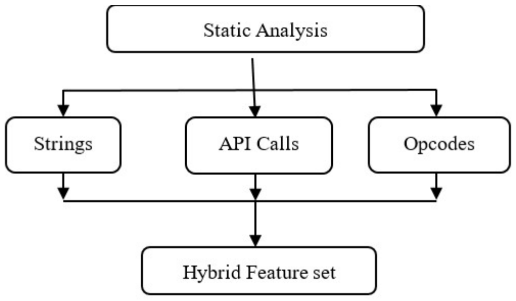


Figure 6. Architecture of our Proposed Hybrid Features

For machine-learning algorithms to be able to learn the patterns, the features needs to be converted into a vector form. For malware features such as strings, Opcodes and API Calls, the creation of representative vector can be done using a number of ways including frequency feature vector, frequency weighted feature representation and or binary feature representation [28]. For the purpose of this work, we used the binary frequency vector approach as described by [28] and [26]. In binary representation, the features are represented as a binary feature F signifies the presence of absence of a malware binary given a resulting feature vector as $V_{Fb}= (bs_1, bs_2 \ldots\ldots\ldots\ldots\ldots bs_n)$, where bs_i is 1 if F contains an instance of s_i or 0 if otherwise, where n is the size of the sample or dataset. With this, our chosen ensembles were trained with the training dataset and tested with the testing dataset. The complete feature set and their importance in percentages is shown in figure 7 below.

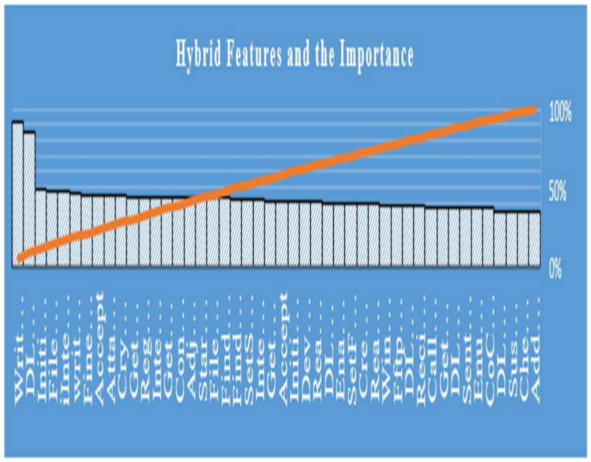


Figure 7. Proposed Hybrid Features

2.4. Selecting and Description of the Ensemble Algorithms

Four-machine learning algorithms were explored; Random Forest (RF) Model, Support Vector Machine (SVM), GradientBoost and eXtremeGradient Boost were trained using 80% for training and 20% for testing. We chose these techniques because they are good at binary classification. The train–test process was necessary to ensure that the model achieve optimum performance. This was to avoid over generalization of the model or the under fitting or overfitting problem usually encountered in machine learning. The next section discusses the algorithms used in the experiment.

2.4.2. Random Forest (RF)

RF model is used in many fields such as banking, health care, stock market prediction, e-commerce and cybersecurity with very good success (Balram, Hsieh & Mcfall, 2019). Security community has used decision tree-based algorithm but it is always not used alone. Many trees are trained and used together to make a prediction in a fashion known as Random Forest. This ensures that each tree sees the data differently to improve detection outcomes. To make the prediction as to whether a binary is malware or benign, the trees are allowed to vote and

Journal of Theoretical and Applied Information Technology
15th December 2023. Vol.101. No 23
© 2023 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the most popular vote wins. More precisely, suppose $D = \{(x_1,y_1),\dots,(x_n,y_n)\}$ is data sampled from $\mathbb{R}^d \times S$-valued random vector (X,Y) with possibly known distribution, for some positive integer $d \geq 1$. The objective of the random forest (RF) algorithm is to predict y from x using an ensemble $h = \{h_1(x),\dots,h_k(x)\}$ of classifiers for decision trees. The decision tree for the classifier $h_k(x)$ is determined by the parameter $\theta_k = \left(\theta_{k_1},\dots,\theta_{k_p}\right)$ that is a realization of a known random variable θ, that describes the subsets of the data set D that constitute the decision trees of the classifiers in h. Thus

$$h_k(x) = h_k(x|\theta_k).$$

Each tree in the random forest cast a vote for the most popular class $y \in S$ for the input data x. The vote of the k^{th} tree is the output of the classifier $h_k(x)$. The class with the most votes wins. Therefore the decision function is then given as

$$H(x) = arg \max_{y \in S} \sum_{i=1}^{k} I\left(h_i(x) = y\right), \quad (2)$$

where $I(h_i(x) = y)$ is the indication function. The margin function for the best class $y \in S$ of a random forest is given

$$m(x,y) = P_\theta(h(x|\theta) = y) - \max_{u \neq y} P_\theta(h(x|\theta) = u), \quad (3)$$

where P_θ is the probability distribution of the decision tree generating random variable θ. Note that $m(x,y)$ is the measure of extent the probability of votes for best class exceeds the probability for the next-best class. Therefore, the generalization error e takes the

$$e = P_{x,y}(m(x,y) < 0) \leq \frac{Var_{x,y}(m(x,y))}{\left(\mathbb{E}_{x,y}(m(x,y))\right)^2} \quad (4)$$

where $P_{x,y}$, $\mathbb{E}_{x,y}$ and $Var_{x,y}$ are respectively the probability distribution, expectation and variance of the random vector (X,Y). The inequality follows from the Chebyshev's inequality.

2.4.3. Support Vector Machine

Consider the following training data $x_1, x_2, \dots, x_n \in \mathbb{R}^D$, the D-dimensional Euclidean space. The aim is to classify the points as

$$S = \{(x_1,y_1),(x_2,y_2),\dots,(x_m,y_m)\} \subset \mathbb{R}^D \times \{-1,+1\},$$

with the help of a separating hyperplane w·x+b=0. A training data x_i will get

$$y_i = \begin{cases} +1, & if\ x_i\ is\ above\ the\ hyperplane \\ -1, & if\ x_i\ is\ below\ the\ hyperplane \end{cases}$$

The training points closest to the separating hyperplane are called the support vectors and the objective of the Support Vector Machine is to orientate the hyperplane, by tuning w and b, to ensure it is as far as possible from the support vectors. Thus, we want to maximize the margin

$$\rho = \min_{x \in \{x_1,x_2,\dots,x_m\}} \frac{|w \cdot x + b|}{\|w\|} \quad (5)$$

By choosing w and b such that the support vectors satisfying $|w \cdot x + b| = 1$ we get that

$$\rho = \frac{1}{\|w\|}$$

Therefore, the resulting maximization problem becomes

$$(1) \quad \max_{w,b} \frac{1}{\|w\|}$$

subject to $y_i(w \cdot x_i + b) \geq 1$, for all $i = 1,\dots,m$

which *is* equivalent to

$$\min_{w,b} \tfrac{1}{2}\|w\|^2 \quad (11)$$

subject to $y_i(w \cdot x_i + b) \geq 1$, for all $i = 1,2,\dots,m$

The Lagrangian to this convex quadratic optimization problem is

$$L(w,b,\alpha) = \tfrac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i \left(y_i(w \cdot x_i + b) - 1\right) \quad (6)$$

with $\alpha_i \geq 0$, for all $i = 1,2,\dots,m$.

Note that for all $i = 1,2,\dots,m$,

$$\alpha_i[(y_i(w \cdot x_i + b) - 1 = 0. \quad (7)$$

This last condition implies that $\alpha_i \geq 0$, for support vectors x_i and $\alpha_i = 0$ if x_i is not a support vector. Taking the gradients of L with respect to w and b and putting them to zero result in the equations

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i \quad (8)$$

and

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \quad (9)$$

Putting the above equations into $L(w,b,\alpha)$ results in the dual optimization problem,

$$\max_{\alpha} L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i x_j)$$

subject to: $\sum_{i=1}^{m} \alpha_i y_i = 0$, and $\alpha_i \geq 0$, for all $i = 1,2,\dots,m$.

Substituting the unique solution $\bar{\alpha}$ of the above dual problem into the above expression for w gives us required expression for

$$w = \sum_{i=1}^{m} \bar{\alpha}_i y_i x_i \quad (10)$$

For the expression for b observe that if x_i is a support vector then

$$y_i(w \cdot x_i + b) = y_i\left(\left(\sum_{j=1}^{m} \bar{\alpha}_j y_j x_j\right) x_i + b\right) = 1 \quad (11)$$

Thus, if x_i is a support vector then

$$\sum_{j=1}^{m} \bar{\alpha}_j y_j (x_j \,.\, x_i) + b = y_i \quad (12)$$

Hence $b_i = y_i - \sum_{j=1}^{m} \alpha_j y_j \,(x_j . x_i)$ (13)

Now averaging over all the various support vectors we get

$$b = \frac{1}{N_{SV}} \sum_{i \in SV} \left(y_i - \sum_{j=1}^{m} \bar{\alpha}_j y_j (x_j \cdot x_j)\right) \quad (14)$$

where SV is the index set of all support vectors and N_{SV} is the number of support vectors.

2.4.4. AdaBoost

AdaBoost is one of the most popular ensemble techniques (Balram, Hsieh & Mcfall, 2019). It is the best-known family of algorithms using boosting (Adaptive Boosting). In adaptive boosting (AdaBoost), an initial classifier is trained on a training set. The weights of incorrectly classified samples are incremented. A second classifier is again trained on the dataset that contains the incremented or updated weights. This iterative or recursive process terminates when the predetermined estimator number is reached or an optimal predictor is found. The down side of the model is the fact that, it is sequential and cannot be executed in parallel. However, due to its skill and stability in prediction, we used it for our classification. The description of the AdaBoost algorithm is given below.

Let us consider the labeled data $\{(x_{1,}y_{1,}), \dots, (x_N\, y_{N,})\}$, with $x_i \in \mathbb{R}^d$, $d \geq 1$ and $y_i \in S = \{-1, +1\}$. The objective of Adaboost algorithm is to use a collection of weak classifies $\{h_1, h_2, \dots, h_K\}$ and optimally constructed weights $\{\alpha_1, \alpha_2, \dots, \alpha_K\}$ to generate a strong classifier

$$h(x) = \text{sign}\left(\sum_{k=1}^{K} \alpha_k\, h_k(x)\right) \quad (15)$$

The weights $\alpha_1, \alpha_2, \dots, \alpha_K$ are generated incrementally with the help of the following algorithm:

```
Initialize by putting $w_i^{(1)} = 1$ for all $i = 1,2,3,\dots,N$
    for k=1 to K do
        Fit classifier $h_k$ to data to minimize the error
            $\varepsilon_k = \frac{\sum_{i=1}^{N} w_i^{(k)} I(h_k(x_i) \neq y_i)}{\sum_{i=1}^{N} w_i^{(k)}}$
        where $I(h_k(x_i) \neq y_i)$=1 if $h_k(x_i) \neq y_i$ and 0 otherwise
            $\alpha_k = \log\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$
        for all $i = 1,2,3,\dots,N$, do
            $w_i^{(k+1)} = w_i^{(k)} e^{\alpha_k I(h_k(x_i) \neq y_i)}$
        end for
    end for
```

Note that the updated weight $w_i^{(k+1)}$ is the same as $w_i^{(k)}$ if the classifier h_k correctly classified the data pointx_i, otherwise the weight $w_i^{(k+1)}$ is $w_i^{(k)}$ scaled by e^{α_k}.

After the training and testing the four algorithms their performances were measured.

2.4. Training and Testing the Ensembles.

Ensemble algorithms are also called committee-based learners. When the data is non-stationary and in the presence of class imbalances, the use of conventional ML techniques produces suboptimal performance [27]. When this happens, the need for the use of combination of models is recommended. He content that an ensemble is always better than the base classifiers. Therefore, the main purpose of ensembles is to improve the accuracy of the models compared with the individuals. [27] suggested that, ensembles can be combined in many ways; majority voting where the modal classifier is considered(the class with highest frequency).

The need for ensembles in ML is established as means for more accurate prediction than the base or individual classifiers. This dates back to the 1990s [28]. They are known as committees whose aim is to aggregate the classification or prediction of the individual classifiers. They include random forest, gradient boosting and others. Generally, ensembles can take the form of bagging, boosting, stacking and heterogeneous ensembles. Many authors in the current literature apply machine-learning approaches to malware classification, and the approach is on a continuum [8]; [18]; [5]. Our analysis of the various literature and an initial experiment conducted on the use of machine learning and homogeneous features led us to identify four algorithms for this study. After obtaining the dataset, conducting Exploratory Data Analysis (EDA) or preprocessing, and the feature engineering, the need for model selection was obvious. The Random Forest (RF) Model, Support Vector Machine (SVM), AdaBoost, and eXtreme Gradient Boost ensemble algorithms. We chose these techniques because they are good at binary classification and robust against imbalanced data. After the selection of the models, we trained them using the dataset and features based on the simple train-test split. We divided the dataset into two, using 80% of the data for training and 20% for testing.

We trained and tested the models first with the individual features before and after the malware, dataset was obfuscated. This was to test the performance accuracy of the models and form the baseline performance of the features. Secondly, we trained and tested the models on the hybrid features before and after the obfuscation of the malware dataset. The rationale of this was to observe the performance of the proposed approach in handling obfuscated malware and to compare the performance with the performance of the models on the single features. Thirdly, having established the performance of the models, the need to deal with the problem of class imbalance that leads to the 'accuracy paradox' was considered. Since our dataset is imbalanced, as witnessed in many malware detection environments, the use of ensemble techniques with SMOTE and other imbalance data handling techniques would lead to improvements in classification performance [25]. Thus, we applied the SMOTE technique on the training set with the hybrid features and tested to observe the performance of the models. This was to observe the effect or impact of the accuracy paradox without and with data augmentation. Fourthly, the performance of the proposed technique is compared with cited literature.

2.5. Performance Evaluation

Therefore, having evaluated the performance of the models on the individual features before and after the malware dataset was obfuscated, we evaluated same using the hybrid features before and after the obfuscation, and the performance of the models with the hybrid and SMOTE. The performance of the models were obtained from the confusion matrix or the contingency table.

To be able to determine the efficiency of a model, it has to be evaluated using performance metrics. These refers to standards used to assess the characteristic and the behavior of an artifact [31]. The following performance metrics were used; Accuracy, Sensitivity/Hit Rate/Recall/True positive Rate (TPR), False positive Rate(FPR)/Precision, True Negative Rate/specificity(TNR). These measures are obtained from the confusion matrix or contingency table as shown in table 3 below.

Table 6: Confusion Matrix

Predicted	Actual Malware	Non-Malware
Malware	True Positive(TP)	False Positive(FP)
Non-Malware	False Negative(FN)	True Negative(TN)

Sensitivity is the proportion of correctly classified malware samples in the dataset. False Positive Rate = 1-specificity. Accuracy is the proportion of correctly classified observations in the dataset and is represented as shown in equation 17. Other metrics such as True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate(TNR) are as shown as in equation 18-20.

$$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \ldots (16)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \ldots\ldots\ldots\ldots\ldots..(17)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \ldots\ldots\ldots\ldots\ldots. \quad (18)$$

$$\text{F1} - \text{Score} = \frac{2 * \text{TP}}{2 * \text{TP} + \text{FP} + \text{FN}} \ldots.(19)$$

For the purpose of this work, we used accuracy, false positive rate as performance metrics that is recommended if accuracy is the aim of the work. The next section presents the results of the study.

3. RESULTS AND DISCUSSION OF THE STUDY.

The study proposed an improved malware variant detection model based on the use of improved homogeneous hybrid features, SMOTE data augmentation technique and exposure to various malware obfuscation techniques. The purpose was to provide efficient homogeneous static-based malware features, employ SMOTE data augmentation technique to overcome the accuracy paradox, and exposing ensemble algorithms to sufficient obfuscation techniques during the training of the models to improve resilience against obfuscated malware attacks. This section present the results and discussion of this experiment using charts, figures, tables and other visuals.

3.1. Performance of Ensemble Techniques on Individual Features Malware Obfuscation

In order to be able to test the accuracy of the extracted features, we used each of them in training and testing of the models. We evaluated the performance of the models on the individual, or mono features in classifying malware samples (strings, opcode sequence, and API call features). As depicted in figure 8, the string features showed relatively good classification performance with the dataset, with the eXtreme Gradient Boost ensemble scoring 85.6% accuracy and 3.2% false positives. The rest of the models scored below with relatively high false-positive values compared with the XGB Model.

The relative moderate performance of the models might be because the malware dataset comprised a variety of malware families with different features or attributes, while the models were trained on single features. Thus, using a single feature for the training meant that the models would not gain sufficient experience to be able to generalize with other new malware samples, resulting in poor classification accuracy. This finding or result is consistent with or supported by [27]; [12], who suggested that malware detectors start and die based on features and that if poor and insufficient features are used to train a model, it leads to suboptimal performance. This requires that when training a model, there be sufficient features with enough variety to enable the model to generalize with the unseen data or sample later on.

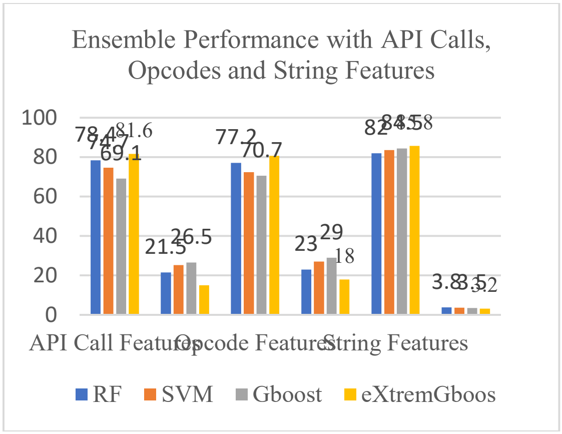


Figure 8. Performance Of Ensembles On Individual Features Before Obfuscation

3.2. Performance of Ensemble Techniques on Individual Features after Malware Obfuscation

Similarly, we tested performance the models' resilience or robustness to obfuscated malware using the individual features. The malware samples were obfuscated and the experimented repeated to determine how resilient the models are to malware obfuscation or changes. As indicated in the figure 9, the performance of the models remains unchanged with the obfuscated dataset.

This result suggests that, unlike static and or signature-based detection systems that are unable to adapt to new and previously unknown malware, our approach using the ensemble produces adaptability, learnability, and robustness with respect to changes that occurred to the malware dataset. Hence, the changes in the malware binaries had no effect on the predictability of the models. This is in tandem with [27] who opined that, the exponential growth in malware volumes, variety and complexity leading to variants of known malware requires the use of ML techniques.

Consequently, the changes introduced into the malware by the obfuscation did not affect the classification potential of the models. This is in tandem with [29]; [26]; [27], who hold the view that the limitations of the existing signature-based detection system are a result of malware authors adopting obfuscation and other encryption techniques, leading to variants of malware such as polymorphic, metamorphic, and oligomorphic that evade detection.

As signature-based static techniques, including many anti-virus scanners, operate on the assumption that, once a malware is identified, its characteristics remain the same for its entire lifespan, the use of obfuscation defeats such a position as the technique leads to new and or variants of the known malware, making detection impossible due to the inflexibility. Thus, the use of the adaptable ensemble classifiers with our proposed technique allows the machines to learn and adapt to the changes introduced in the obfuscated malware, leading to the maintenance of the same classification performance. Therefore, using the features as input and training the ensemble ensured that they were able to generalize to detect even hitherto unseen malware. This is required to detect not only known malware but also unknown malware, including zero-day vulnerabilities.

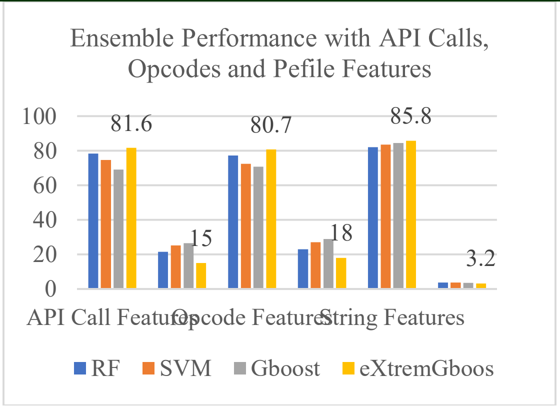


Figure 9. Performance of Ensembles on Individual Features after Obfuscation

3.3. Performance of Ensemble Techniques with Hybrid Features before and after Malware Obfuscation

In order to improve the performances of the models, we proposed a hybridization of string, opcode, and API call features, all extracted from static disassembly, to form an integrated feature set. We used these features to train and test the ensemble techniques in harmony with [26]; [10], who suggested that the weaknesses in signature-based detection methods could be improved by the integration of the techniques in malware analysis. As shown in Figure 10, all the models performed moderately well, with the eXtreme Gradient Boost ensemble outperforming the others in classification accuracy and false positive rates.

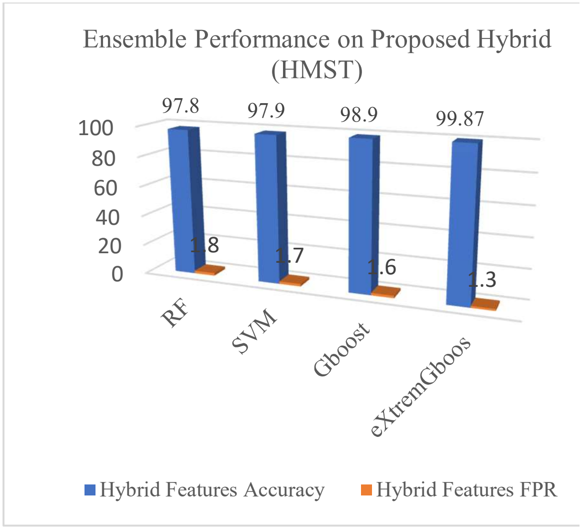


Figure 10. Performance Of The Proposed Approach Before Obfuscation

The relative good performance of the models might be a demonstration of the fact that ensembles always outperform their individual classifiers and are good for non-stationary datasets. The XGBoost algorithm, which is an extension of the usual gradient boosting, has shown to be resilient and robust with large datasets due to its scalability. Thus, with the relatively larger dataset and due to its scalability, the XGB demonstrates superior classification performance in terms of accuracy and false-positive rates compared with the others. This is in line with [22], who suggested that the ability of XGB to handle large data coupled with its scalability makes it a good choice to reduce the residual error of estimators leading to improved prediction and a suitable candidate for parallel computing and cloud computing, where malware attacks are common

3.4. Performance of Ensemble Techniques on Hybrid Features after Malware Obfuscation

Similarly, the performance of the models was evaluated after the malware dataset was obfuscated. This was to test the resilience and robustness of the homogeneous hybrid in classifying obfuscated malware and malware variants such as polymorphic, metamorphic, oligomorphic, and other mutant malware. The results of the models with the obfuscated malware dataset are shown in Figure 11. As depicted from the figure, there was no variations in performance of the models after the obfuscation. Again, the XGB model led in performance accuracy and false-positive rates. As explained earlier, the lack of variation in performance of the models and the relatively high performance accuracy demonstrates the resilience and robustness of the homogeneous hybrid technique or features as an efficient technique for malware classification, which is in line with [28].

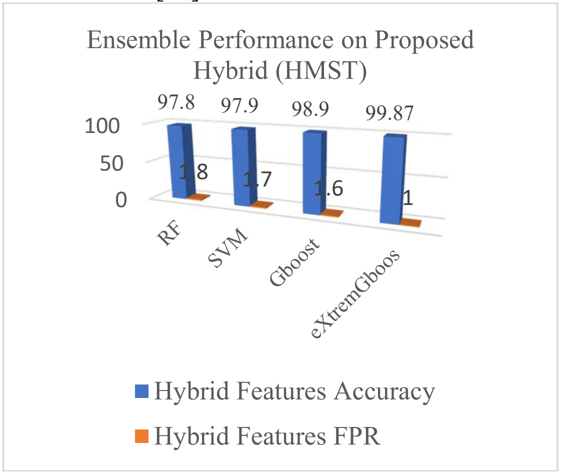


Figure 11. Performance Of The Proposed Approach After Obfuscation

3.5. Comparing the performance of the proposed approach with the individual features

Similarly, comparing the performance of the hybrid with the individual features, there was a vast improvement in the performance of the proposed technique over the individual features, demonstrating the ability of the proposed approach to detect and classify many malware categories. As shown in Figure 12, the hybrid approach largely outperformed the individual features. This is necessary because with the current increase in malware volumes, variety, and complexity, there is a need for automated tools to not only target a single malware family but to be able to detect and classify a variety of these malware forms. Thus, the variety of malware included in our dataset and the high performance in terms of classification accuracy and false positive rates demonstrate the resilience of the approach in handling diverse malware families. This is necessary to avoid the adoption and installation of many detection tools at the same premises for malware analysis and defense.

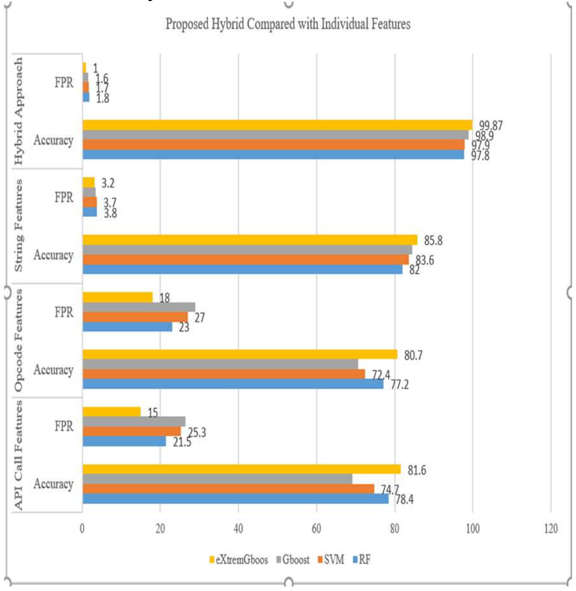


Figure 12. Comparison Of Hybrid With Individual Features

3.6. Overcoming the 'Accuracy Paradox' in Class Imbalance using the Proposed Hybrid Features

To overcome the class imbalance problem in the dataset, data augmentation techniques such as oversampleing or undersampling methods can be applied. Undersampling is where some random samples are extracted from the majority class; whiles in oversampling, synthetic samples are added to the minority class. Each of these have their trade-offs depending on the circumstances. Oversampling methods mostly used due to the advantage explained earlier. Thus, we applied SMOTE technique, which is an oversampling method to rebalance the dataset and reduce the skewedness of the model and provides for the true classification accuracy.

As depicted in table 7, the accuracy of the models after the application of the SMOTE technique reduced. These results suggest that, with the imbalance dataset, there was some form of bias, which lead to the higher accuracy. However, when the data augmentation technique (SMOTE) was applied, the true classification accuracy of the models were obtained that led to the reduction of the accuracy. This position is in tandem with [27], who suggested that to efficiently manage the imbalance data asymmetry, requires the use of sampling methods that rebalances the dataset leading to better performance of the classifiers.

Table 7. Performance of Models without and with SMOTE

Ensemble Learner	Accuracy without SMOTE (%)	Accuracy With SMOTE (%)
RF	97.8	94.6
SVM	97.9	95.7
GB	98.9	96.9
XGB	99.8	98.8

This implies that, with imbalance dataset as experienced in most security domains, the use of accuracy as a performance measure or metric without data augmentation may result in inaccurate results when accuracy is used as performance measure. [27] contents that, under such circumstances the use of Area Under Curve(AUC) and F1-Score or measure provides a better evaluation measure.

3.7. Comparison of the Proposed Approach with similar cited Literature

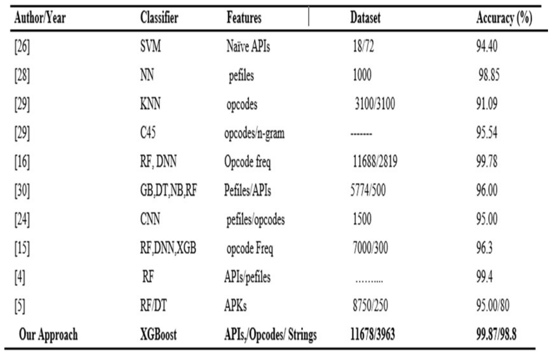
Generally, since our approach is largely different from the rest of the works, comparison is a challenge. Under such conditions, the recommended strategy is the use similar works and situate the new work in context. Consequently, to situate the performance of our approach within the context of literature, we compared the performance of our technique with similar studies from 2017 to date. As depicted in Table 8, our approach performed comparably better with all the cited literature. It is to be noted that [29] achieved 99.78% with the use of only opcodes. This is contrary to our approach, in which we used multiple features (strings, opcodes, and API calls) where we obtained 99.87% without data augmentation and 98.8%. This allows our

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

model to gain more training experience with a variety of features, which results in the detection of a variety of malware compared with their approach, which has limitations in detecting malware without opcode features. In addition, all authors used imbalanced data for their models; however, none of them used data balancing techniques to avoid skewed and biased classification. Consequently, their classification accuracy may be subject to the accuracy paradox.

Overall, our proposed approach used relatively larger dataset size, efficient homogenous features, we used the SMOTE data augmentation technique in handling the imbalance data; avoiding the usual 'accuracy paradox' and tested the resilience of our approach with obfuscated malware samples. In our approach we showed that our classification accuracy is unbiased and unskewed, which is necessary for handling imbalance data. This observation is consistent with [30], who suggest that one way of overcoming the data imbalance problem is to use ensemble with the oversampling techniques and variants of these techniques.

Table 8. Comparison Of Our Proposed Technique With Cited Literature.



Author/Year	Classifier	Features	Dataset	Accuracy (%)
[26]	SVM	Naïve APIs	18/72	94.40
[28]	NN	pefiles	1000	98.85
[29]	KNN	opcodes	3100/3100	91.09
[29]	C45	opcodes/n-gram	-------	95.54
[16]	RF, DNN	Opcode freq	11688/2819	99.78
[30]	GB,DT,NB,RF	Pefiles/APIs	5774/500	96.00
[24]	CNN	pefiles/opcodes	1500	95.00
[15]	RF,DNN,XGB	opcode Freq	7000/300	96.3
[4]	RF	APIs/pefiles	....	99.4
[5]	RF/DT	APKs	8750/250	95.00/80
Our Approach	**XGBoost**	**APIs,/Opcodes/ Strings**	**11678/3963**	**99.87/98.8**

Consequently, our proposed technique outperformed the cited literature, gave a true measure of the models as we have included the SMOTE to handle imbalances, demonstrated resilience to malware obfuscation, and used a relatively larger dataset in our classification. Thus, to the best of our knowledge, they demonstrate considerable novelty or value and therefore present a huge potential for malware classification in general and enhancing static techniques in particular.

4. CONCLUSION AND FUTURE WORKS.

As the volumes, variety and complexity of malware increases exponentially, the efficiency and efficacy of signature-based detectors is compromised requiring the use of automated Artificial Intelligence and ML techniques. However, poor malware features, poor classification accuracy due to the class imbalances and the use of obfuscated malware techniques to evade detection remains a major challenge in malware detection systems. We have demonstrated an efficient hybrid malware features based made up of only 'fine-grained' features and used an ensemble technique with SMOTE data augmentation method to overcome malware obfuscation and improve malware variant detection. Specifically, we have demonstrated the efficiency of the hybrid technique in classifying malware and variants, overcome the 'accuracy paradox' usually encountered with imbalanced data using the SMOTE Technique, and finally, demonstrated the resilience of the technique against malware obfuscation. We therefore conclude that on the bases of the achieved objectives, the proposed technique provided efficient features that improved the performance accuracy of the models in classifying unknown malware, demonstrated resilience and robust against obfuscated malware and variants of previously unknown and known malware, including zero-day malware. Thus, the proposed approach provides an improved malware variant detection approach and presents huge promise and potential for malware detection including variants of known and unknown malware.

Even though our approach achieved relatively high performance, it has some limitations such as not using the AUC and F1-Score, not using a rigorous feature dimensionality reduction technique and the use of ensemble not deep learning methods. in future, we will use AUC and F1-Score as a measure instead of the accuracy to see the effect. Also, we shall apply Principal Component Analysis (PCA) as a feature dimensionality reduction technique to observe the performance of the models and finally, apply deep learning techniques such as convolutional neural networks (CNN) and or Generative Adversarial Networks (GANs) to observe the performance of the models in detecting known, unknown and variants of known malware including zero-day malware.

CONFLICT OF INTEREST: There is no Known Conflict

AUTHOR CONTRIBUTION: **Azaabi Cletus. Main author.** Worked on the experiments, analysis, and writing up of the manuscript.

ALEX AKWASI OPOKU: Review of equations and structure of the paper.

BENJAMIN ASUBAM WEYORI: Review of the paper for experimental accuracy of the results

FUNDING : No External Funding.

REFERENCES

[1] Rajesh Kumar, Geetha, S. Effective Malware Detection using Shapely Boosting Algorithm. *(IJACSA) International Journal of Advanced Computer Science and Application, Vol. 13, No.1, 2022.*

[2] AV-Test Institute. Annual Malware statistics" Malware Statictics. Retrieved online at www.av-test.org/en/statistics/malware. 26th February 2023.

[3] Bassett, G., Hylender, C.D., Langloise, P., Pinto, A., and Widup, S. (DBIR Team), " Verizon Data Breach Investigations Report 2022".Results and Analysis. https://www.verizon.com/business/resources/report. Retrieved 26th February, 2023.

[4] Chen, Z, Brophy, E, and Ward, T .Malware Classification using Static Disassembly and Machine Learning. *CEUR-WS.org, 2021.*

[5] Qussai M. Yaseen, Esraa Odat and Batool Alazzam. Detecting Malware Families and Sub-families using Machine-learning Algorithms: An Empirical Study. *(IJACSA) International Journal of Advanced Computer Science and Application, Vol. 13, No.2, 2022.*

[6] Rosli, N., Warusia, Y., Faisal, M. A, Salamat, S. R . Clustering Analysis for Malware Behavioral Detection using Registry Data. *(IJACSA) International Journal of Advanced Computer Science and Application, Vol. 10, No.12, 2019.*

[7] Chritopher Hadnagy. Social Engineering: The Science of Human Hacking. Wiley, Indianapolise. *URL: www.wiley.com.*

[8] Ravi Kiran Vedrma, PLN Raju, K V Suba Raju, Akhila Kalidindi . Feature Selection and performance Improvement of Malware Detection System using Cuckoo Sear Optimization and Rough Sets. *(IJACSA) International Journal of Advanced Computer Science and Application, Vol. 11, No.5, 2020.*

[9] Saleh Alyahyan .Machine learning ensemble methods for classifying multi-media data. A thesis submitted for the degree of doctor of philosophy at the University of East Anglia, September 2020 (*A dissertation).*

[10] Javier Bermejo Higuera, Carlos, Abad, Aramburu, Juan-Ramon Bermejo Hieguera, Miguel Angel Sicilia Urban and Juan Antonio sicilia Montalvo. Systematic approach to malware analysis (SAMA). *Applied Science, 2021 MDPI. https://doi.org/10.3390/app10041360*.

[11] Hemant Dhamija, Ajay, K. Dhamija. Malware detection using Machine Learning Classification Algorithms. *International Journal of Computational Intelligence Research. Research India Publications, 2021.*

[12] Jagsir Singh and Jaswinder Singh. Challenges of malware Analysis: Obfuscation Techniques. *International journal of information security science, vol.7. No. 3, 2018.*

[13] Heena, A., and Mehtre, B., M. Advances in malware detection-An overview. Institute for Development and Research in Banking Technology. arXiv:2104.01835v2[cs.CR]8May 2021.

[14] Kang, J., Won, Y. A study on Variant Malware Detection Techniques using Static and Dynamic Features. *Journal of Information Processing Systems , 2020 Https//:doi.org/10.3745/jips.03.0145.*

[16] Rathore, H., Agarwal, S., Sahey, S.K., Sewak, M. Malware Detection using Machine Learning and Deep Learning. *Journal of Information Systems, 2020.*

[17]] Hussain, S.J., Ahmed, U., Liaquat, H., Mir, S., Jhanjhi, N.Z., and Humayu, M. IMIAD: Inteeleleigent Malware Identification for Android platform, in proceedings of the international conference on computer and information sciences. Pp.1-6, Sakaka, Saudi Arabia, April, 2019.

[18] ABM. Adam et al. Performance Analysis of Machine Learning Classifiers for Detecting PE Malware. *(IJACSA) International Journal of Advanced Computer Science and Application, Vol. 11, No.1, 2020.*

[19] Yunus, K. B. M., and Ngah, S. B. Review of hybrid analysis technique for malware detection. *IOP Conf. Ser. Mter.Sci.Eng. 769-012075, 2022.*

[20] Arora, A., Peddoju, S.K., and Permpair, M.C. Android malware detection using permission pairs" IEEE Transactions on Information Forensics and Security, vol. 15, pp.1968-1982, 2020.

[21] F. Ali, N.B. Anuar, R.Salleh, G. Suarez-Tengil and S. Funell " AndroiDialysis: Analysis of the android intent effectiveness in security and communication networks. Computers & security, vol.65.pp.121-134, 2017.

[22] Karbab, E.B., Debbabi, A., and Maldozer, D.M. Automatic Framework for android malware

detection using deep learning. Digital Investigation, vol. 24, no. S48-S59, 2018.

[23] Kim, T., Kang, B., Rho, M., Sezer, S., and IM, G.E. A multimodal deep learning method for android malware detection using various features, IEEE Transactions on Information, Forensics and security, vol.14. no. 3, pp.773-778, 2019.

[24] Wang, W., Zhao, M., and Wang, J. Effective android malware detection with a hybrid modal based on deep autoencoder and convolutional neural network. Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 8. Pp.3035-3043, 2019.

[25]] Rami Sihwail, Khairuddin Omar, Khairul Akram Zaninol Ari Sanad Al-Afghani· Malware Detection Approach Based on Artefacts in Memory Image and Dynamic Analysis. *Applied Sciences. MDPI, 2019. https://doi.org/10.3390/app9183680 .*

[26] Monnappa, K. A. Learning malware analysis: explore the concepts, tools and the techniques. *www.packt.com,* 2018. Birmingham, Mumbai.

[27] Alexsandro Parisi. Hands-on artificial intelligence for cybersecurity. Implement smart AI System for preventing cyber-attacks and detecting threats and network anomalies, *Packt. www.packt.com,* 2020.

[28] Danny Kim. Improving Existing Static and Dynamic Malware Detection Techniques with Intrusion-level Behaviour. *(Dissertation,* 2019).

[29] Darabian et al. Detecting cryptomining Malware: a Deep learning approach for static and dynamic analysis. *J.Grid Computing, 2020, https://doi.org/10.1007/s10723-020-09510-6. .*

[30] Ravi, Diwakar. Handling imbalance data with imbalance-learn in python. Published, may 30, 2020. Data Science Blogathon. Retried 3rd march, 2023.

[31] Soma, Halder & Sinan Ozdemir "Hands-on machine learning for cybersecurity. Safeguard your systems but making your machines intelligent using the python ecosystem". *Packt. Birmingham-Mumbai. www.packt.com. 2018.*

[32] Pablo Duboue. The art of feature engineering: Essentials for machine learning. Textualization software Ltd, 2020. https://doi.org/10.1017/9781108671682