Journal of Theoretical and Applied Information Technology
15th May 2024. Vol.102. No 9
© Little Lion Scientific

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

A HYBRID MALWARE DETECTION FRAMEWORK WITH DRIFT ADAPTATION FOR TIMESTAMPED DATA

HARINADH VARIKUTI[1] , VALLI KUMARI VATSAVAYI[2]
[1,2]Department of Computer Science and Systems Engineering, Andhra University college of Engineering, Visakhapatnam, India
E-mail: [1]varikutiharinadh@gmail.com, [2]vallikumari@gmail.com

ABSTRACT

As technology is growing rapidly, new malware variants are evolving. Attackers are generating new malware patterns using code obfuscation and mutation techniques to escape from anti-malware engines and traditional models. Models built over historical malware data cannot perform well in detecting or classifying dynamic or real-time malware data. The Evolution of new malware patterns may lead to concept drift. Static models built over historical data trained from scratch whenever new data arrives which is time-consuming, so models with adaptive nature are used. Adaptive machine learning models are used to identify the drift in data and perform appropriate changes dynamically to the models over time. In this paper, a hybrid model is proposed which is the combination of Leveraging bagging and AdaBoost methods for incremental ensemble learning. Dynamic weights are assigned to the models used in the ensemble learning to increase the adaptivity. The use of boosting and bagging methods adds error adoption and diversity to the model respectively. Experiments also show the proposed model outperforms all the state-of-the-art drift adaption methods such as ADWIN Bagging, ADWIN Boosting, SRP Classifier, etc.

Keywords: *Concept drift, Machine learning, Malware, drift detection, Bagging, Boosting*

1. INTRODUCTION

Every year anti-malware models are created to detect malicious programs, but the changing nature of malware patterns escape the detection. Attackers use malware programs to perform unethical operations such as stealing sensitive data, manipulating system data, privacy violations, encrypting resources, denial of service, etc. The cyber security research community is working rigorously to strengthen security and avoid malicious attacks. On the other side, attackers use artificial intelligence, code obfuscation techniques, encryption, and various methods to create new malware family variants to perform attacks. There are various types of malware families like viruses, trojan horses, ransomware, backdoors, worms, botnets, and rootkits. Every malware attack depends on the core functionality of its family.

In the war situation between Israeli and Hamas groups (1) , Distributed Denial of Service (DDoS) cyber-attacks were carried out. Hamas performed a DDoS attack on Israeli websites to stop the alert news of rocket attacks. According to Statista (2), ransomware attacks affect over 72% of businesses worldwide in 2023. Around 17% of attacks increased from 2018-2023. According to the AV test report (3), every day 450,000 new malicious programs are detected. Feature patterns are extracted from these new malware files and saved for future detection purposes. As per the malware statistics and incidents happening around the world, malware operations have grown exponentially over the years. According to Malware analysis market statistics (4), the global malware analysis market is expected to reach $24.15 billion by 2026. Every human has to be aware of malware attacks to safeguard the things around them. Due to the rapid increase of technology, attackers use few technical tools to create malware programs in less time and inject them into the target systems. Sometimes victim are not aware of the background malware activities such as data theft, unauthorized access, and keylogging happening on their devices.

A few years ago, anti-malware engines have been used to perform malware detection. These engines used signature and heuristic-based techniques to detect the malware. In the signature-based method, known malware is identified by matching the signatures present in the anti-malware database. Unknown malware is not detected by the engines which limits its capability. The heuristic-based method is the process of investigating the source code for apprehensive action. Polymorphic malware and zero-day malware have been detected using the heuristic methods. However, there is a problem of getting false positives and false negatives with this method. False positive, which identifies benign files as malware and doesn't affect costly, whereas false negatives identify malware files as benign which destructs badly to the system.

Malware analysis is performed by using two approaches static approach and dynamic approach. In static analysis, significant features are extracted without executing the original malware file. In dynamic analysis, initially, an isolated and protected environment is set up. Original malware files are executed in an isolated environment to extract the behavioral features. These features give information about the malware file's interaction with the computer system. In this work, static features are considered to identify malware patterns. Static features such as byte histogram, byte entropy, imported functions, exported functions, general information, section information, string information, and header information are considered for malware detection.

Concept drift is one of the major challenges in machine learning models. Models built on historical malware data suffer from performance degradation when testing with the new variants. This happens due to the data distributional changes that occur with time. It is challenging to identify the evolving nature of malware patterns. Mostly static machine learning models suffer from performance degradation due to concept drift. Concept drift is the situation where the statistical properties of class variables vary as time changes in a machine learning model. There are different ways in which concept drift may occur. They are gradual drift, sudden drift, and recurring drift as explained below:

- **Gradual drift**: It happens due to the changes that occur slowly or gradually in the target class over time. These types of changes occur in situations such as changes in interest rates, the launch of alternative products, etc.
- **Sudden drift:** It happens due to a sudden change in the target class distribution. Here detecting the drift and adapting the model is crucial to maintain the effectiveness of the model. A typical example is covid-19 pandemic which has suddenly changed daily human life.
- **Recurring drift:** This type of drift happens as seasonal changes over time. Old concepts reoccur after some time. Situations such as business patterns on weekdays and weekends happen alternatively imitating the recurring drift.

The drifting nature of the data degrades the model performance and its applicability in malware detection. if there is any drift identified, sometimes it leads to the detection of malware files as benign. These mistakenly detected benign files destroy the system with its malicious operations. To overcome these situations, drift adaptation techniques are used. Nowadays machine learning models are applied to various timestamped datasets. The data is divided into time frames like yearly and monthly. Models are trained on initial time data points and testing is performed with later data points. Malware samples used in testing change feature patterns and hence escape detection. Identification of drift gives information that the attacker changes the malware code pattern using various techniques such as code obfuscation and mutation methods. Drift detection is identified by using various algorithms such as ADWIN (5), KSWIN, DDM, and Page Hinkley.

2. RELATED WORK

This section discusses the research work published in the field of malware detection and how concept drift affects model performance. As anti-malware engines and traditional models identify the known malware files effectively, malware authors changed the malicious code patterns to evade detection. These timely changing malware variants cause concept drift. The occurrence of concept drift degrades the performance of the model. Concept drift detection methods are classified into three categories (6). Error-rate-based drift detection algorithms are one of the categories that give an alarm if there is a statistical change in error rate, either it increased or decreased at some time series data points. Data distribution-based drift detection comes under the second category, drift occurs due to distributional change of an input variable data over the timestamped data. Multiple hypothesis test drift detection is the third category which uses variant hypothesis tests to detect concept drift. J.Lu et.al (7) suggested that the concept drift challenge can be resolved by using adaption techniques and ensemble models. The retraining technique slowed down model performance.

Drift detection (8) can be performed by popular state-of-the-art detection techniques such as ADWIN (9), KSWIN, DDM, and Page Hinkley. ADWIN is an Adaptive Window-based drift detection method (10). It is used to identify the drift and give drift warnings, then perform the adaption on machine learning models. The algorithm considers an adaptive sliding window (11) and the window grows as long as no concept drift is detected. Once the drift is detected, it cuts the older data samples present in the sliding window. ADWIN effectively identifies the gradual drift, because of dynamic sliding window size. DDM (12) is also a popular drift detection method that defines two thresholds such as warning level and drift level. The significant increase in model error rate and variance indicates the occurrence of concept drift. DDM is not efficient in identifying the gradual drifts, because long-term gradual drifts require a greater number of data samples to store (13). Page Hinkley is a statistical technique used to detect concept drift (14). Page Hinkley doesn't give drift warnings, it only changes the detections. It computes the mean value for the observed values up to the current moment. If the mean is greater than the threshold value, concept drift is detected. The Kolmogorov-Smirnov windowing (KSWIN) (15) drift detection method is performed by using the Kolmogorov-Smirnov statistical test. It is used to compare the data distribution of the samples to identify the drift.

Drift can be identified by using the above drift detection algorithms, now we have to implement drift adaption algorithms to maintain a more effective model. Drift adaption algorithms are classified into two categories such as incremental learning models and ensemble learning models (16). In incremental learning, the decision tree algorithm called the Hoeffding tree is used for adapting the data streams. Hoeffding bound is used to find the number of samples used to select the split node in the decision tree construction. whenever new samples come, the tree updates its node by adapting to the newly arrived data. There exists another algorithm called Extremely Fast Decision Tree (EFDT) (17) which splits the node as it reaches the confidence threshold value. It is more adaptive to concept drifts when compared to the Hoeffding tree.

In ensemble models, there are two categories such as block based and online ensemble learners. While using block-based models, initially data is divided into fixed size blocks and trained using base learners using each block. Every time when new block arrives, the base model is evaluated and updated. The limitation in block-based model is fixed block size. Algorithms such as Streaming Ensemble Algorithm (SEA), Accuracy Weighted Ensemble (AWE), and Accuracy Updated Ensemble (AUE) come under block-based methods and among them, AUE gives better results (18). Online ensemble models such as Adaptive Random Forest (ARF), and Streaming Random Patches (SRP) integrate incremental learning models such as the Hoeffding tree for better performance. H. M. Gomes et.al (19) proposed ARF with the Hoeffding tree as a base learner and ADWIN as a drift detector. As it resembles a random forest algorithm, ARF gives better performance with its adaptability to various types of drifts. SRP (20) uses a similar strategy as ARF, like detecting the drifts and its adaptability to drifts. SRP considers global subspace randomization where as ARF uses local subspace randomization which improves the diversity of base learners.

3. SYSTEM OVERVIEW

In this work, we considered the BODMAS benchmark dataset (21) which is timestamped data to perform malware detection. Generally, malware files can be detected by using anti-malware models, to evade this detection attackers continuously change the pattern. New malware patterns cause concept drift while changing the time. To study the concept drift and implement a better drift adaption algorithm, this timestamped benchmark dataset is considered.

3.1 BODMAS Dataset

BODMAS stands for Blue Hexagon Open Dataset for Malware Analysis. The Researchers performed joint work with Blue Hexagon to produce timestamped malware data. In this dataset, timestamped malware file data along with its related malware family information is provided by the researchers. The dataset also provides meta information on malware files and 2,351 features with values extracted from all the files. The feature types and their counts are given in Table 1. The dataset contains eight feature groups, each of which is described in more detail below.

Header Information: It includes COFF header information such as timestamp, target machine, and characteristics. Optional information includes subsystem, DLL_Characteristics, file_magic, major and minor image versions, linker versions, major and minor operating system versions, sub-system versions, and the header, code and commit sizes.

Byte Histogram: It contains byte value features from 00 to FF (a total of 256 features). it counts the frequency of each byte value in the file.

Byte Entropy Histogram: These features give the frequency distribution of byte entropy values over the file. Entropy measures the randomness of each byte value in the file.

Imported functions: These functions are called from external modules to perform various actions on the victim's system such as executing specific tasks, accessing the resources, and interacting with the system.

Exported Functions: These functions are present in the malware binaries and used by the other modules.

Section Information: It contains properties about each section such as name, size, entropy, virtual size, and string characteristics of each section.

String Information: It includes printable characters that are at least five characters long. It also contains strings that indicate file paths, URL links, HKEY_ which indicates a registry key, and the occurrences of MZ string that represents the Windows executable file.

Table 1. Feature Categories present in the BODMAS dataset

Feature Categories	No of features
Header Information	62
General file Information	10
Byte Histogram	256
Byte Entropy Histogram	256
Imported functions	1280
Exported functions	128
String Information	104
Section Information	255
Total	2351

The BODMAS data set consists of 1,34,435 timestamped data samples from the year 2007 to 2020, which includes both benign and malware samples. However, we considered only 2019 and 2020 years data to perform the malware drift analysis i.e, 1,23,793 samples. The frequency distribution of the data is shown in the Figure 1.

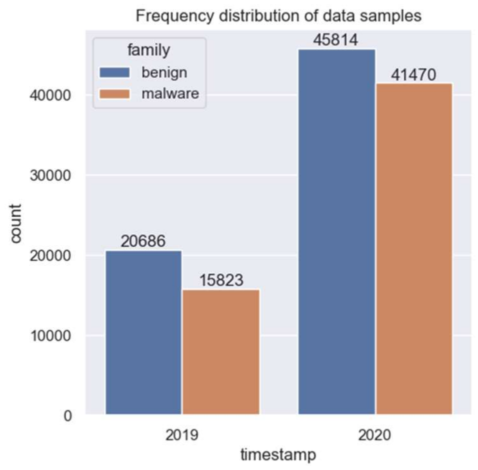

Figure 1 Frequency Distribution of data samples

4. PROPOSED METHODOLOGY

In this paper, drift adaption algorithms have been implemented on the bodmas malware dataset, and handling the concept drifts in timestamped data or data streams is the main contribution in this paper. Among the various drift detection and adaption algorithms, In this analysis, ADWIN Boosting, ADWIN Bagging, Leveraging Bagging, SRP classifier, and Adaboost algorithms have been used to perform malware detection. To Identify the drifts, drift detection methods such as ADWIN, KSWIN, and Page Hinkley are used as discussed in the related work section.

While building models, we considered base learners to better understand the drift points and handle it. The base learner is the one referred to as a predictive model trained on the initial data. while giving the initial data to the baseline model, it monitors when the accuracy of the model starts to degrade representing the drift nature. When new data is applied to the base model, there is a chance of occurring deviation in outcomes. Drift detection methods perform the statistical difference function on predictions and original outcomes, and gives an alert if significant drift is detected. when drift is detected, perform drift adaption techniques which involve retraining the base model on the most recent data and analysing that the model is efficient in handling the drifts. Based on the severity of the drift, update the model by using online incremental algorithms for adopting the changes in the data. base learner gives the drift analysis in the data which helps build the robust models. In this work, the Hoeffding tree is considered as a base classifier for all the models used.

4.1 Models Used in Drift Detection

Hoeffding tree: hoeffding tree is an incremental decision tree algorithm used for learning the data streams. It uses Hoeffding bound for constructing and analyzing the tree. It also assumes that the distribution of data doesn't change over time. Information gain has been used to select the best attributes for the tree construction. after selecting the best attribute, a test is performed using new data whether the selected attribute gives better results than other attributes. The attribute that is effective in the tests used for splitting the node for the growth of the tree.

ADWIN Boosting: It is an ensemble incremental learning algorithm used to adapt the drifts that occur in the data. It uses ADWIN as a drift detection algorithm. The concept of boosting is to build a robust model from the number of weak models. Combining boosting with ADWIN forms an algorithm that can adapt to distributional changes occurring in the data. initially, a model is built with some data and checks the efficiency. The misclassification that occurred in the first model is corrected and build the second model. Continue the creation of new models until all the training data samples are predicted correctly. Boosting improves the accuracy by combining all the weak models by averaging or voting method. It reduces the overfitting issues because of the high weightage given to the misclassified data. imbalanced dataset can also be handled effectively by boosting technique.

ADWIN Bagging: Bagging or Bootstrap Aggregation is also an ensemble learning algorithm that uses multiple models constructed by different subsets considered from the training data. integrating the ADWIN drift detection and Bagging method improves the model performance and adapting to the drift i.e., if any distributional changes occur in the data. In the bagging technique, random subsets of data are selected with replacements from the training data, and generate a model with every data subset. In this method, parallel training of models happens independently and generates an outcome. Now aggregation is performed such as majority or average function applied on all the outcomes.

SRP Classifier: Streaming Random Patches (SRP) Classifier is a batch learning ensemble method that considers random samples and random feature subspaces. These combinations of random samples and feature subspaces are named random patches. Here bagging method is used for batch training where whereas online streaming data online bagging method is used. Online bagging is robust to concept drift and adopting the distributional changes that occurred to the model.

Leveraging Bagging: It uses the bagging method to select the data and leverage the performance with two improvements such as increasing resampling and using output detection codes. Poisson distribution is used to perform the resampling with replacement and assign weights to the samples.

AdaBoost: Adaboost is an ensemble learning algorithm to combines predictions of multiple learning models as a robust and accurate model. It performs various iterations to assign weights to the samples in training the model. The initial weak learner is analyzed and samples that are misclassified are given a high weightage in the successive learning model.

4.2 Hybrid Model

The hybrid model performs incremental learning in detecting the malware data samples with the adoption of identified drifts. Incremental learning is the way of training the model continuously over time and including the new data with the model by avoiding the retraining of the entire dataset from scratch. Incremental learning models are useful in analyzing time-stamped data, i.e., data that is continuously arriving over time. This approach is essential for malware detection because of the evolving nature of the malware. Attackers are continuously adapting new techniques to create new malware variants and escape traditional malware detection systems. The behavior and characteristics of new malware variants may cause drift. This drift causes various challenges in identifying the newly evolving malware using traditional models. Incremental learning models are used to mitigate drift efficiently. Periodically updating the models using new malware data reduces the drift challenge and increases the effectiveness of the model. Ensembling the models also reduces the drift challenge in malware detection.

In the proposed method, a hybrid model is constructed by ensembling the boosting and bagging techniques to perform malware detection on timestamped data (assumed as a data stream). AdaBoost and Leveraging Bagging algorithms are two state-of-the-art drift adaption methods considered as base learners in the model construction. ADWIN drift detection method is used in the base learners to detect the changes that occurred in the data. In the hybrid model, boosting focuses on error adaption on difficult instances whereas bagging gives diversity by training on various random data. This model is powerful and efficient in scenarios where diversity and adaptability are desired. Ensemble model construction aims to perform efficient drift detection and apply an adaptive nature to the model if the drift value exceeds the threshold. The proposed hybrid malware detection framework is shown in Figure 2.

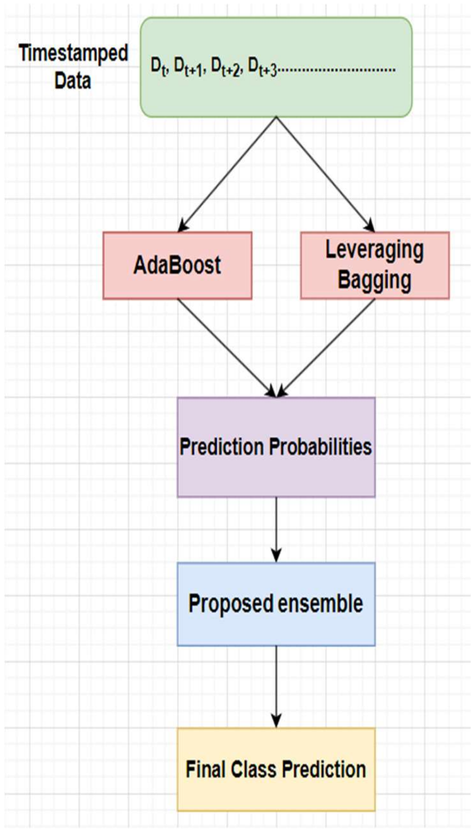

Figure 2 shows the proposed hybrid malware detection framework

Initially, the data set is divided into various bins, such as D_t, D_{t+1}, D_{t+2}, D_{t+3} .... based on timestamps. The data sample D_t is used to train the hybrid model. Then each data sample from the remaining bins (D_{t+1}, D_{t+2}, D_{t+3} ....….. etc.) is given to the hybrid model, which contains Adaboost and Leveraging bagging models. Individually, both models take the data sample and get the prediction probabilities of both the benign and malware classes. The Hoeffding tree is used as a base model for both models. Base models are considered foundation models in drift detection algorithms. These models understand the pattern changes identified by the drift detection methods and adopt them. To deal with the drifts, the ADWIN drift detection method is used to monitor the performance of the classifier on the upcoming data samples. If any drift occurs, a new classifier is replaced with the old model to perform prediction.

Dynamic weights are assigned to the two base learners Adaboost and Leveraging bagging instead of static weights. Considering the data $M=\{(x_1,y_1),(x_2,y_2)\ldots..(x_n,y_n)\}$ where $x_1,x_2,\ldots..x_n$ represents the data sample with features and $y_1,y_2,\ldots\ldots y_n$ represents the respective class labels The target class estimated by the hybrid model for the data sample x is given by following equation:

$$y = argmax_{k\in\{0,1\}} \frac{\sum_{n=1}^{2} h_n \; p_n\left(\frac{y=k}{x}\right)}{2} \qquad (1)$$

Where n represents the number of base learners used, n = 2 in the proposed hybrid model. $p_n\,(y = k \mid x)$ represents the predicted probability of a target class k by using the n_{th} base learner on the data sample x. h_n represents the weight of the n_{th} base learner. Malware detection has been performed here, so k = 2 and y ϵ {0,1}. Data samples are processed one by one as a stream to improve the model. After processing each data sample, the error rate is calculated by dividing the number of misclassified samples by total number of samples taken. The weight of base classifiers h_n is calculated by following the equation:

$$h_n = \frac{1}{E_b + \epsilon} \qquad (2)$$

E_b represents the real-time error rate. To avoid the denominator value being equal to zero, a small constant value is represented with the notation ϵ.

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

Compared to other state-of-the-art ensemble models used to mitigate drifts in malware detection problems, the hybrid model performed better in handling false positives. Fasle positive means the model is predicting a benign sample as malware. Due to this, various challenges arise, such as disruption of benign file operations, computational time wasted in identifying these misclassified files, and loss of trust in the malware detection systems. The dynamic weight is considered in the proposed model for reducing the sample error rate when compared with static weights used by the ensemble models. It performs better to detect drifts present in the data and ease of adapting to the drifts that occurred.

5. RESULTS AND EVALUATION

The proposed framework was implemented by using Python 3.10 by extending the river package [5] on a system with an i7-10750H processor with 16GB memory. This section provides an experimental evaluation of the proposed hybrid model in the Bodmas malware dataset. The proposed model is an ensemble incremental learning that adopts the concept of drift effectively and performs malware detection.

Figure 3. shows the accuracy of the proposed hybrid model. The X-axis represents the number of samples and the Y-axis represents the accuracy percentage. The overall accuracy of the timestamped malware dataset used to perform malware detection is 98.6%.

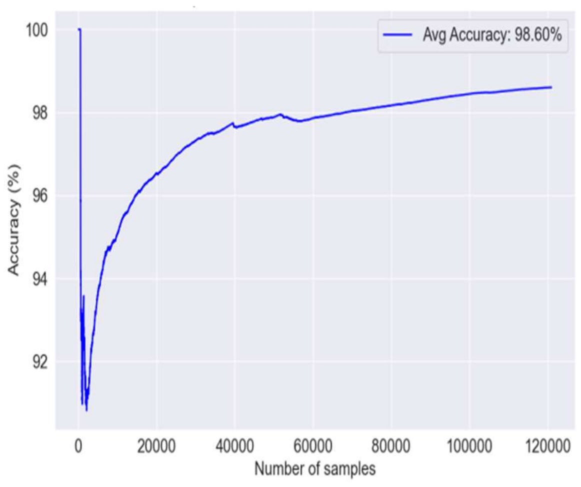

Figure 3 Accuracy of the Proposed hybrid model in BODMAS Malware dataset

5.1 Comparison with State-of-Art Models

In this work, hybrid model performance is compared with the other State-of-Art learning models and shows how it outperformed the all considered models such as ADWIN Boosting, ADWIN Bagging, SRP classifer, AdaBoost and Leveraging Bagging. Ensembling the various models gives a more efficient model by aggregating the model's decision on the data. Table 2 gives the data which shows the performance metrics such as Accuracy, Precision, Recall and F1-score for all the models studied. Formally accuracy, Precision, Recall and F1-score are defined as follows:

$$\text{accuracy} = \frac{\text{number of correct predictions}}{\text{total number of predictions}}$$

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + Fal\ \ \ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ negative}$$

$$\text{F1-score} = 2 \text{ x } \frac{Precision\ x\ Recall}{Precision + Recall}$$

Table 2 Performance Comparison of State-of-the-art models with the proposed model

Method	BODMAS Malware			
	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
ADWIN Bagging	93.75	93.84	93.65	93.72
ADWIN Boosting	92.79	92.9	92.67	92.75
SRP Classifier	96.93	96.94	96.91	96.92
AdaBoost	97.22	97.23	97.19	97.21
Leveraging Bagging	98.21	98.21	98.21	98.21
Proposed hybrid model	**98.6**	**98.64**	**98.4**	**98.52**

6. CONCLUSION AND FUTURE WORK

This paper presents a robust and effective model to perform malware detection in a time-stamped dataset. Malware programmers unethically create new patterns of malicious files, which affect the victims badly because of the escaping nature of the malware. This escaping nature occurs sometimes due to distributional changes, i.e., concept drift in the malware data.

Using the drift adoption ensemble algorithms, these types of drifts can be easily identified and adopted into the learning models to detect malware samples in the future. According to the performance evaluation, the proposed model outperforms all the state-of-the-art models explained above with an accuracy of 98.6%. A future direction of research work is to use dynamic features to detect malware. Instead of depending exclusively on the static features, extracting the dynamic features by executing the malware files in an isolated environment is future research.

REFERENCES

[1] Omar Yoachimik. CloudFlare. [Online].; 2023 [cited 2024 January 5. Available from: https://blog.cloudflare.com/cyber-attacks-in-the-israel-hamas-war.

[3] Statista. [Online].; 2023 [cited 2024 January 6. Available from: https://www.statista.com/statistics/204457/businesses-ransomware-attack-rate/.

[5] AVTEST. [Online].; 2024 [cited 2024 January 6. Available from: https://www.av-test.org/en/statistics/malware/.

[7] Allied Market Research. [Online].; 2020 [cited 2024 January 7. Available from: https://www.alliedmarketresearch.com/malware-analysis-market-A05963.

[9] Riverml. [Online].; 2019 [cited 2024 January 6. Available from: https://riverml.xyz/latest/api/drift/ADWIN/.

[11] Mayaki MZAaRM. Autoregressive based Drift Detection Method. In 2022 International Joint Conference on Neural Networks (IJCNN); 2022.

[12] Zhang JLaALaFDaFGaJGaG. Learning under Concept Drift: A Review. IEEE Transactions on Knowledge and Data Engineering. 2019; 31: 2346-2363.

[13] Tracking concept drift in malware families. In ACM workshop on Security and artificial intelligence; 2012. p. 81-92.

[14] Bifet AaGR. Learning from Time-Changing Data with Adaptive Windowing. In 7th SIAM International Conference on Data Mining; 2007.

[15] Barddal JPaGHMaEF. Advances on Concept Drift Detection in Regression Tasks Using Social Networks Theory. In International Journal of Natural Computing Research; 2015. p. 26-41.

[16] 11. [22] Grulich PaSRaTJaBSaRTaMV. Scalable Detection of Concept Drifts on Data Streams with Parallel Adaptive Windowing. In ; 2018.

[17] Gama JaMPaCGaRP. Learning with Drift Detection. In Intelligent Data Analysis; 2004. p. 286-295.

[18] Wares SaIJaEE. Data stream mining: methods and challenges for handling concept drift. SN Applied Sciences. 2019; 1.

[19] K. Handling Concept Drift for Predictions in Business Process Mining. CoRR. 2020; abs/2005.05810.

[20] Schleif CRaMHaF}. Reactive Soft Prototype Computing for Concept Drift Streams. CoRR. 2020; abs/2007.05432.

[21] Yang LaMDMaSA. PWPAE: An Ensemble Framework for Concept Drift Adaptation in IoT Data Streams. In IEEE Global Communications Conference (GLOBECOM); 2021. p. 01-06.

[22] Chaitanya Manapragada and Webb. Extremely fast decision tree. In 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2018. p. 1953-1962.

[23] Sun Y WZLHDCYJ. Online Ensemble Using Adaptive Windowing for Data Streams with Concept Drift. International Journal of Distributed Sensor Networks. 2016.

[24] Gomes HMBARJBJPEFPBHGAT. Adaptive random forests for evolving data stream classification. Machine Learning. 2017.

[25] Gomes HMaRJaBA. Streaming Random Patches for Evolving Data Stream Classification. In ; 2019. p. 240-249.

[26] Wang LYaACaILaAAaG. BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware. In 2021 IEEE Symposium on Security and Privacy Workshops, SPW 2021; 2021. p. 78-84.