FeSAD Ransomware Detection Framework with Machine Learning using Adaption to Concept Drift

Damien Warren Fernando, Nikos Komninos

Department of Computer Science, School of Mathematics, Computer Science and Engineering, City, University of London, UK

Abstract

This paper proposes FeSAD, a framework that will allow a machine learning classifier to detect evolutionary ransomware. Ransomware is a critical player in the malware space that causes hundreds of millions of dollars of damage globally and evolves quickly. The evolution of ransomware in machine learning classifiers is often calculated as concept drift. Concept drift is dangerous as changes in the behaviour of ransomware can easily lead to misclassifications, and misclassification can harm individuals and businesses. Our proposed framework consists of a feature selection layer, drift calibration layer and drift decision layer that allows a machine learning classifier to detect and classify concept drift samples reliably. We evaluate the FeSAD framework in various concept drift scenarios and observe its ability to detect drifting samples effectively. The FeSAD framework is also evaluated on its ability to extend the lifespan of a classifier. The results obtained by this research show that FeSAD can successfully and reliably classify ransomware and benign samples while under concept drift and can extend the time between retraining.

Keywords: Ransomware detection, Machine Learning, Concept Drift, Malware Evolution, Genetic Algorithm

1. Introduction

Ransomware is a malware type that restricts a user's access to their files by either locking the computing device or encrypting its files. According to Symantec, the cost of ransomware attacks globally amounts to damages running into the hundreds of millions of dollars [2], and BlackBlaze reports that the average cost of a ransomware attack on a business is 1.85 million dollars [3]. Modern ransomware is particularly difficult to deal with due to the usage of sophisticated encryption schemes that keep encryption and decryption keys on a remote command and control server [4]. Modern ransomware users use techniques beyond encryption to persuade victims to pay by threatening to dox victims and organizations that they successfully attack [5]. An attacker will access sensitive information related to victims and use this information to leverage victims into paying ransoms under the threat of having this information made public. The damaging effects of ransomware and its rate of evolution and growth create the need for more robust detection solutions. Machine learning ransomware detection systems are a popular solution to ransomware detection; however, machine learning algorithms have limitations despite their strengths. This research focuses on Crypto-Ransomware, which encrypts user files and asks for a ransom. [1]. We define machine learning algorithms as fit for purpose as long as most samples classified benign or ransomware do not show abnormal levels of drift. We define the lifespan of a machine-learning algorithm as the period between training and the threshold of abnormal drifting samples being exceeded. The drift calibration layer defines the abnormal drift threshold, and the drift of every incoming test sample is recorded. Once the number of samples showing an abnormal drift exceeds the set threshold, the algorithm is considered at the end of its lifespan, and the machine-learning algorithm must be retrained. A high number of samples showing abnormal drift would indicate that the machine learning algorithm is becoming less confident of its classifications and that a high proportion of samples' statistical properties are different from expected. The FeSAD framework aims to increase the lifespan of a classifier by preparing it to classify samples that show drift and calibrating its drift thresholds to account for the drift commonly seen from ransomware families from different distributions.

1.1. Motivation

The primary motivation for this research is the ransomware evolution rate that can lead to unsuccessful detection [6]. Zero-day attacks are always on the rise [7], which are attacks for which there is no signature; therefore, they will be unknown to detection systems. There are solutions for ransomware detection using machine learning systems such as Ransomwall [8] and the adaptive system [9]. Ransomwall aims to maintain a high detection rate using a multi-layer approach for detecting zero-day ransomware, and the adaptive system described in [9] takes high rates of ransomware evolution into account when developing a pre-encryption detection system for ransomware. The system proposed by Min et al. in [10] describes how to defend a Solid State Drive (SSD) from ransomware encryption and uses a machine-learning system. Machine learning systems have proven effective in detecting ransomware and achieving positive results on zero-day variants; however, machine learning systems can be prone to concept drift. Concept drift implies that the statistical properties that determine a sample class have changed, and this phenomenon can represent ransomware evolution. Experiments show that the machine-learning approaches used in prominent ransomware detection solutions are prone to degrade when faced with concept drift. The FeSA system [15] discusses that zero-day ransomware does not necessarily equate to evolved ransomware due to the multitude of infection vectors constituting a zero-day sample. Ransomware that is evolved behaves outside the scope of what a classifier deems to be ransomware behavior, and the FeSAD framework aims to solve it. In the context of the API call features we use; this implies that the ransomware's API-based behavior has shifted in a way that would make it more difficult for a classifier to identify it as ransomware correctly. In some cases, classifiers may be able to identify zero-day samples because the sample's behavior matches the behavioral profile defined by the classifier as ransomware; however, the issues arise when ransomware samples appear that do not match the classifier's behavioral profile of ransomware.

1.2. Paper Contribution

The main contribution of this paper is the FeSAD framework, the FeSAD framework can detect ransomware under concept drift and effectively extend the lifespan of a machine learning detection system.

- **FeSA Feature Selection:** The FeSA layer generates feature sets using the FeSA feature selection algorithm. The FeSA feature selection approach is proven to produce robust feature sets that show a degree of resistance to concept drift.
- **Drift Detection & Classification:** The FeSAD framework derives a numeric value to quantify concept drift and makes reliable classifications of samples showing concept drift.

1.3. Paper Organisation

The remainder of this paper is as follows: Section 2 covers related work. Section 3 covers our proposal. Section 4 describes our experiments, and section 5 discusses the results of the experiments. Section 6 concludes and expands our work.

2. Background & Related Work

This section covers the related work of our research. We have explored related work covering concept drift, ransomware detection, and malware detection with concept drift.

2.1. Ransomware Detection with Machine Learning

We observe that the systems designed in [11], [12], [13] and [14] provide machine learning-based detection for ransomware, with Almoussa et al. and Qin et al. providing machine learning and (Application programming interfaces) API solutions. It can be observed that API-based approaches can be considered effective as both of these systems provide strong ransomware detection results. The API-based solution provided by Sgandurra et al. has proven effective in detecting zero-day ransomware and early detection. Poudel et al. developed a framework for the Analysis Framework that performs multi-level analysis on ransomware and benign. The multi-level analysis involves analyzing binaries, assembly code, and function calls such as API calls. The analysis framework uses a combination of binary, assembly, and API features to make an informed decision on ransomware detection; the framework uses different machine learning classifiers. This research has achieved strong results with detection rates over 90% for almost all machine learning algorithms. Abbassi et al. propose an automated feature selection method for ransomware detection with machine learning algorithms. The system proposed uses particle swarm optimization for a behavioral-based detection system; this means the features used for ransomware detection are dynamic and taken during execution. The approach proposed groups features together and performs feature selection based on the significance of a group. The research by Ayub et al. uses I/O request packets to detect ransomware; I/O request packets are a low-level I/O system operation that is relatively new in the ransomware research space (Ayub et al., 2020). I/O requests are requests between the user and the kernel mode. If the user requests a file to open, the I/O service in the kernel mode carries out the task. The I/O logs are used with an ANN (Artificial Neural Network) to detect ransomware. The ANN is tested with three different configurations to optimize the detection rates for ransomware. The AI-based ransomware detection approach proposed by Alvee et al. attempts to detect ransomware by converting ransomware binaries into 2-D greyscale images and using a convolutional neural network to classify benign and ransomware images. This research discusses ransomware detection in a substation setting and considers infection of this substation through three infection vectors. The infection vectors are remote access, physical intrusion, devices, and the substation network. The main issue with all of the research in ransomware detection with machine learning is that they need to address the detection and adaption to concept drift. Concept drift in a malware detection system can be particularly damaging due to the consequences of a ransomware attack. The omission of concept drift adaption and detection is a weakness in the mentioned systems.

2.2. Concept Drift

Concept drift is best described as changes in $P(x, y)$. Since it is composed of the feature probability component $P(x)$ and class label conditional probability $P(y|x)$, the change of the joint probability can be better understood via the changes in either of these two components. As a concept can be split into components, the reasoning behind concept drift can be divided into four different types.

- **No Change** In this scenario, both $P(x)$ and $P(y|x)$ remain the same and have not changed regarding how they were when the model was initially trained. It might be useful to retrain on the most up-to-date data regardless; however, this indicates no concept drift, and the model is stable.
- **Feature Change** In this scenario, $P(x)$ has changed, but $P(y|x)$ has remained the same; this means features that may not have been important have now become important. The model can be reconstructed in a scenario to fit these feature changes.
- **Conditional Change** In this scenario, $P(x)$ has not changed, but $P(y|x)$ has changed. In this scenario, an expected error could go either way, so it is necessary to reconstruct the model. The issue with this is that there may not be enough new data to reconstruct the model, leading to high variance. It is possible to solve this problem through the weighted combination of new and old data.
- **Dual Change** Both $P(x)$ and $P(y|x)$ change. The expected error could increase, decrease or stay the same depending on the combination of $P(x)$ and $P(y|x)$. It is necessary to retrain in this instance.

We have reviewed prominent work in malware detection with concept drift and have identified strengths that we can use and weaknesses that we can improve on. Tsymbal et al. use a Heterogeneous Euclidean overlap metric (HEOM) to detect concept drift in machine learning systems; this approach is highly adaptable due to the Heterogeneous Euclidean overlap metric (HEOM) metric being able to handle categorical and continuous data when measuring levels of concept drift [16]. The transcend system shown in [17] explicitly addresses concept drift and the effect on machine learning malware solutions. The adaptive system proposal in [19] demonstrates how ensemble machine learning systems can be useful for detecting drift; this study also explores how different classifiers react to concept drift. Wilson et al. explore the Heterogeneous Euclidean overlap metric (HEOM) and its effectiveness when measuring statistical differences between data instances. The research proposed by Wilson et al. is used in [16] and is flexible and effective at determining whether a sample does not fit a distribution. Tan et al. produced a system that uses the Wilcoxon Rank Sum test [21] to determine whether a sample is displaying concept drift; this is an effective approach; however, it relies on retraining once concept drift is detected. Singhal et al. use Heterogeneous Euclidean overlap metric (HEOM) and gradient-boosted trees, Random Forests, and Deep Learning to detect concept drift in the malicious website detection space; they ensure the system can be retrained quickly in the presence of concept drift [22]. The transcendent system [23] builds on Transcend [17] and uses nonconformity values to detect drifting samples, and this achieves promising results on malware and benign datasets spanning five years; however, the solution to drifting samples is still to either pass the drifting sample to another machine learning algorithm or hand over the sample to a specialist for manual analysis. EACD: Evolutionary adaption to concept drift [24] proposes a method of coping with concept drift in non-stationary systems using evolutionary algorithms; however, it uses a window and relies on immediate retraining. The work presented in [25] breaks concept drift into three types: gradual, cyclical, and abrupt; not all research addressing concept drift focuses on one type of concept drift. and standard deviation [29]. Nishida et al. propose a Statistical test of equal proportions (STEPD)[30], which detects concept drift when there is a significant difference between data in two different windows. The research proposed by Biffet et al. attempts to detect drift using a sliding window that reduces in size when a drift is detected to capture the new concept more efficiently [31].

Having reviewed prominent machine-learning ransomware detection and fundamental concept drift research, we have identified key limitations and why they exist. Most current ransomware detection research fails to address the effects of concept drift, and considering the speed at which new ransomware emerges, it is highly probable that the evolution of ransomware will affect machine-learning detection solutions in the form of concept drift. Some current ransomware studies do include zero-day detection; however, the work produced in [15] shows that a zero-day threat does not necessarily mean a sample that shows evolved behavior; therefore, it is essential to distinguish zero-day and true behavioral evolution. Concept drift in a ransomware detection system would suggest that the behavior patterns of ransomware have changed, making it harder to classify ransomware and benign samples correctly. The current solutions to concept drift in malware detection acknowledge the importance of concept drift detection and identifying when a model is showing concept drift; however, these solutions focus on retraining and manual inspection of samples that behave differently than expected. Current malware concept drift solutions do not acknowledge a window of vulnerability between concept drift happening and the machine learning system being retrained, which leaves a system entirely vulnerable to concept drift. In the context of ransomware and malware, the system will encounter evolved samples and misclassify them due to their behavior being different from what is expected. The limitations of current concept drift malware research allow too much room for misclassification.

3. FeSAD Framework Design

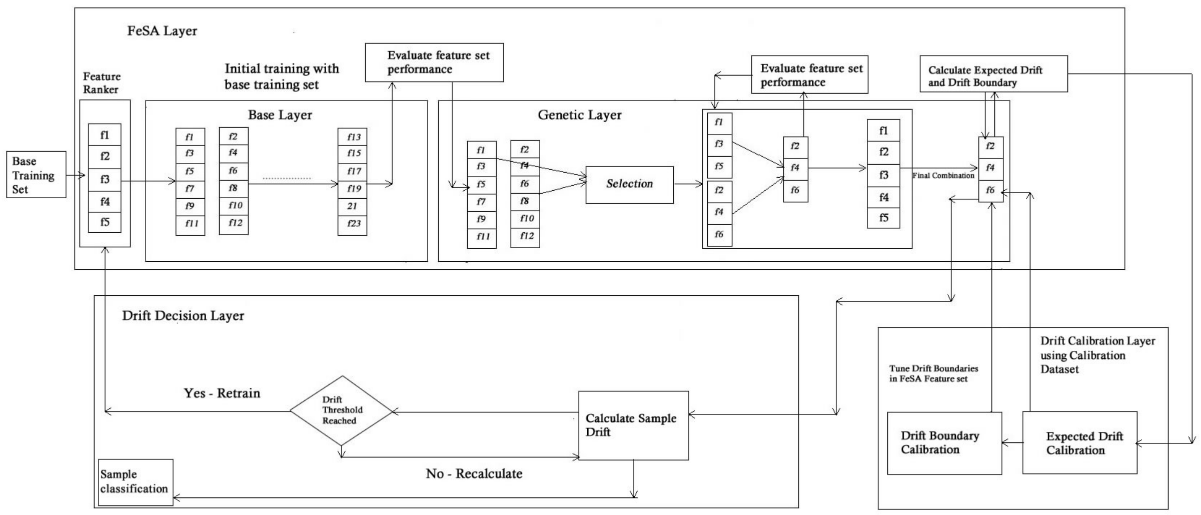
FeSA Layer
Feature Ranker
Initial training with base training set
Evaluate feature set performance
Base Layer
Genetic Layer
Selection
Evaluate feature set performance
Calculate Expected Drift and Drift Boundary
Final Combination
Base Training Set
Drift Decision Layer
Yes - Retrain
Drift Threshold Reached
Calculate Sample Drift
No - Recalculate
Sample classification
Drift Calibration Layer using Calibration Dataset
Tune Drift Boundaries in FeSA Feature set
Drift Boundary Calibration
Expected Drift Calibration

Figure 1: FeSAD framework

The FeSAD framework, shown in Figure 1, will help a machine learning detection system for ransomware to make accurate predictions under concept drift and is designed to prolong the period a detection system can go without being retrained by taking concept drift into account. The FeSAD framework consists of three central components: the FeSA layer, the Drift Calibration Layer, and the Drift Decision Layer. The functions and flows of Figure 1 are described below.

- **FeSA Layer:** The FeSA (Feature Selection) layer generates feature sets using the FeSA feature selection algorithm, and the best-performing feature set is identified for use in the following two framework layers. The FeSA feature selection approach is proven to produce robust feature sets that show a degree of resistance to concept drift. The base layer generates feature sets that flow onto the evaluation. The genetic layer in Figure 1 takes the top-performing feature sets from the base layer, and this flows to the selection process where pairs of strong feature sets are bred together; this process is repeated until the performance of the generated offspring is no longer improving. The drift boundaries and values are calculated for the optimal and stored in the genetic layer. The drift values calculated are fed into the Drift Calibration Layer. The robustness of the feature sets is judged by their ability to reduce the performance degradation a classifier experiences due to concept drift. The FeSA layer is modified in the FeSAD framework to use a modified similarity metric that identifies potentially drifting samples in the training data set. The FeSA layer calculates the initial thresholds for abnormal and normal samples based on the training data set.The FeSAD framework is equipped to use different similarity metrics combined with prediction probabilities to measure concept drift; this approach is different from something like the Nonconformity measure (NCM) used in Transcend as it uses metrics derived from the underlying algorithm and statistical measures outside the underlying algorithm. Approaches like the NCM (Jordaney et al., 2017) are effective; however, there is room to add reliability, seeing as the underlying algorithm is degrading due to concept drift; therefore, it would be logical not to derive all measures to combat drift from a degrading machine learning classifier as the decisions made by the algorithms will inevitably be affected by the drift.

- **Drift Calibration Layer:** The drift calibration layer uses a calibration dataset to measure the shifts in drift between the training and calibration set. Figure 1 shows the drift values that are calculated flow to the drift calibration layer and the change in the drift between the training and calibration data. The changes in drift are fed back to the optimal feature set and stored in the framework. The updated drift values are then fed into the Drift Decision Layer. The calibration dataset is a dataset that contains a different distribution of ransomware and benign samples; the ransomware in the calibration set is expected to evolve from the training ransomware and, therefore, should present concept drift when compared to the training data. The calibration layer analyses the drift observed in the calibration data compared to the training set and adjusts the thresholds for abnormal drift. The updated thresholds for drift are sent to the FeSA layer, and these thresholds are used in the drift decision layer. The main difference between the Drift Calibration Layer and other current work is that the Drift Calibration Layer aims to observe the changes in drift across different distributions to anticipate future drift; this allows for the theoretical expansion of the lifespan of the classifier used by the FeSAD Framework.

- **Drift Decision Layer:** Figure 1 shows the drift decision layer uses the drift thresholds determined using the drift calibration layer and the feature set produced by the FeSA layer to make decisions on incoming test samples. The drift decision layer calculates whether a sample shows drift beyond what is acceptable for benign or ransomware files. There is a user-set threshold for the proportion of samples displaying abnormal or unacceptable drift the system can encounter before retraining must be carried out.

3.1. Data & Features

The data used was the Application Programming Interface (API) call data of ransomware and benign files from 2013 to 2021. API calls are interfaces that allow the program to interact with an OS, and the API call structure we use allows FeSAD to determine ransomware behavior at the early stages of execution. Once an infection is detected, the system can be isolated, and the suspicious processes can be halted, preventing the spread of the ransomware infection through the network. Each dataset was split by year; each dataset's year was unimportant, only that the data from each dataset were from different distributions. The datasets were structured so that different prominent ransomware from different eras appeared in each dataset; this meant that there would be a clear concept drift in every dataset relative to the dataset the machine learning algorithm was trained on. The features are API-call features, the same as the API-call features used in the FeSA layer [15]. The idea behind using API calls is that ransomware's infection vectors, and pre-infection behavior are likely to be changed; however, the experiments in [15] indicate that API-call patterns can be identified as present in ransomware across different periods. API- calls can also be used as an effective last resort if other vectors for detection have failed, as ransomware will have to interact with the OS during its execution. API calls have proven effective in detecting ransomware infections in [14], [15] and [19]. We split data into training, calibration, and testing sets. The training sets will initially train the underlying machine-learning algorithm. The calibration datasets contain new ransomware and benign files outside the training distribution that will display concept drift; this data set calibrates how much concept drift the system should expect. The testing set contains benign and ransomware samples that display concept drift, and the system's overall effectiveness is tested with the test set. The hooked API calls used in this system are found in [41], which lists the Windows API calls used in Cuckoo Sandbox. Our dataset contains API call and registry behavior data taken from ransomware and benign files from 2013 to 2021; this dataset was created to capture the behavioral data for ransomware across nearly a decade. This dataset would capture ransomware and its evolution from its early stages; this is the perfect way to simulate concept drift and gives us an ideal way of testing the FeSAD framework. Our dataset contains 720 ransomware samples and 2000 benign file samples. The data set is split with 460 ransomware samples from 2013-2015, 65 from 2016-17, 65 from 2018, 65 from 2019 and 65 from 2020-2021. There are 900 benign files from 2013-2015 and 1100 from 2016-2021, split into the same yearly batches as the ransomware files but with 275 benign files in each batch. Some test set configurations will contain evolved versions of ransomware families, which will also appear in the training data. Table 1 shows a breakdown of the ransomware samples used in this dataset.

Table 1: Ransomware Families

Ransomware	**Year of Variant Origin**	**Number of Samples**
PGPCoder	2012	3
Reveton	2013	4
Kollah	2013	9
CryptoLocker	2013	15
Locker	2013	3
Critroni	2014	8
Kovter	2014	5
Matsnu	2014	11
CryptoWall	2014	15
TeslaCrypt	2015	29
Trojan-Ransom	2015	6
Locky	2016	30
Petya	2016	26
Chimera	2016	25
BadRabbit	2017	19
Cerber	2017	40
WannaCry	2017	19
GandCrab	2018	35
Ryuk	2018	28
Scarab	2018	24
WannaCry	2019	19
Stop	2019	29
SamSam	2019	31
Virlock	2019	17
Rakhni	2019	19
Maze	2020	23
REvil	2020	32
Conti	2020	27
SunCrypt	2020	14
GandCrab	2020	32
Sodinokibi	2020	21
Dharma	2021	36
MedusaLocker	2021	25
Avaddon	2021	30

3.2. FeSA Layer

The FeSA layer [15] generates the feature sets for the FeSAD framework. The FeSA layer uses a modified genetic algorithm to generate feature sets that are optimized to be effective in machine learning systems that display concept drift. FeSA identifies the top features from a set of features that present the highest information gain; FeSA includes these features in every feature set produced, meaning the optimal feature set is attained faster. FeSA is shown to effectively maintain the accuracy of ransomware detection systems when concept drift is present [15]. FeSA is an effective base for the FeSAD framework and consists of the feature ranker, base layer, genetic layer, and FeSA updates.

3.2.1. Feature Ranker

The feature ranker uses information gain to identify the features that provide the distinguishable difference between the two classes, ransomware and benign. The FeSA layer enforces these features in every feature set the feature set generation layer produces [15]. The idea behind the feature ranker is to ensure the feature sets will be built on a solid base that can help achieve the highest accuracy and detection rate; this approach also reduces the number of generations needed to reach an optimal solution.

3.2.2. Base Layer

The base layer generates the first set of feature sets. The base layer will generate a predefined number of feature sets using features from a feature pool and the top features identified by the feature ranker. The base layer will measure the average ransomware accuracy of all the feature sets generated; the feature sets passed onto the genetic selection layer are the feature sets that yield detection and accuracy rates above the measured average.

3.2.3. Genetic Layer

The genetic layer acts as a breeding mechanism and will breed high-performing feature sets together. The parent feature sets are the highest-performing feature sets from the base layer. The child feature sets are a combination of the high-performing feature sets. The breeding function uses uniform crossover to select features from each parent feature set [15] and ensure the high information gain features are in every feature set. The genetic selection layer will provide its optimal solution based on the feature sets' performance and the features' overall feature gain in each feature set. The feature set with the combined highest average detection, accuracy, and information gain is chosen as the optimal solution.

3.2.4. FeSA Updates

The FeSA layer yielded positive results shown in [15]. Using our genetic algorithm improved ransomware detection rates quite dramatically, and these detection rates were higher than that of all the feature selection algorithms FeSA was compared with. The main issue with the FeSA layer was that it tended to produce feature sets that yielded a high false-positive rate despite maintaining a solid detection rate when faced with concept drift. The FeSA layer was also a proactive measure, and despite being effective, it did not allow the FeSAD framework to react to concept drift and relied on retraining once the system produced too many incorrect predictions. FeSA could not react to concept drift, which meant it was not a complete solution to concept drift in the ransomware detection space. Ultimately, there is a need to guarantee the lowest possible false-negative rates; however, the high false-positive rate FeSA produced must also be addressed. The solution to the false positives was found by tweaking the fitness function FeSA used, which was detection rate and accuracy; this was altered to consider feature sets with above-average accuracy, detection rate, and overall information gain of the feature set. The altered fitness function ensures that the features included in the dataset will not introduce ambiguity. The updated feature ranker also eliminates features that provide no use from the breeding pool; this means features that provide an information gain below a certain threshold. By tuning the fitness function and feature ranker in the FeSA layer, we eliminated the high proportion of false positives; however, the FeSA layer still needed to react and adapt to concept drift and further extend the classifier's lifespan, creating the need for FeSAD. As part of this update, FeSAD also measures the drift for ransomware and benign in the feature set it generates to define thresholds for acceptable and unacceptable drift. The FeSA layer interacts with the Drift Calibration layer to tune its drift thresholds for greater longevity and higher detection rates over time. The measurements for drift are explained in the following section.

3.3. Drift Calibration Layer

We apply drift to measure the similarity between a sample and the samples of the class it is classified into. The drift calculations are measured using a combination of a Heterogeneous Euclidean Overlap Metric and the prediction probability of a sample. The drift calibration layer measures the drift in the base training data and the increase in drift between the training data and calibration data to predict the increase in drift the FeSAD framework can expect. The expected drift metrics produced in the drift calibration layer are used to tune the expected drift and drift boundaries in the classifier produced in the FeSA layer.

3.3.1. HEOM (Heterogeneous Euclidean Overlap Metric)

The Heterogeneous Euclidean overlap metric (HEOM) is a distance metric designed to measure distances between two data samples with both continuous and nominal attributes. The Heterogeneous Euclidean overlap metric (HEOM) is used because it gives an insight into how close or far away a sample is from the other samples it has been classified with, giving insight into how reliable a prediction may be. The flexibility of HEOM allows the feature set to use continuous and nominal data if necessary [16]; however, it currently uses only continuous features. The HEOM metric measures the distance between two values x and y given an attribute a. The FeSAD framework uses a modified implementation of the Heterogeneous Euclidean overlap metric (HEOM) metric to measure concept drift per sample and overall concept drift in a system. The equation for the standard Heterogeneous Euclidean overlap metric (HEOM) calculations is shown in Equation 1 and Equation 2.

$$d_{heom}(x_1, x_2) = \sqrt{\sum_{a=1}^{m} heom_a^2(x_1, x_2)} \tag{1}$$

Where

$$heom_a(x_1, x_2) = \frac{|x_{1a} - x_{2a}|}{range_a} \tag{2}$$

The standard Heterogeneous Euclidean overlap metric (HEOM) equation measures the distance between two samples x_1 and x_2 in relation to feature a, where m is the number of features. Equation 2 ensures that there are no discrepancies due to large variances between the values for each feature by normalizing them with $range_a$. The Heterogeneous Euclidean overlap metric HEOM measurements for each sample are taken as a ratio by measuring the HEOM distance of a sample from all other ransomware samples and all other benign samples and dividing these two values. Equation 3 shows how the HEOM distance from ransomware for a sample is calculated, and Equation 4 shows how the HEOM distance from benign samples is calculated. In both cases, n represents the number of benign or ransomware samples the sample is compared to during the calculation. The instance number is represented by i_r or i_b and is compared to another ransomware instance j.

3.3.2. HEOM ratio calculation

The calculations for generating the HEOM ratio value for a sample are shown in Equation 5. The HEOM ratio h_i will be much smaller for ransomware than for benign samples due to how the ratio calculation is constructed. The ratio value is calculated equally for both classes, so the HEOM ratio is relative to the distance to ransomware for both benign and ransomware classes.

$$heom_r = \frac{\sum_{i,j=1}^{n} d_{heom}(x_{i_r}, x_{j_r})}{n} \tag{3}$$

$$heom_b = \frac{\sum_{i,j=1}^{n} d_{heom}(x_{i_b}, x_{j_r})}{n} \tag{4}$$

$$h_i = \frac{heom_r}{heom_b} \tag{5}$$

3.3.3. Prediction Weights

A sample's HEOM ratio can tell us how far or close to its classification, but it cannot provide 100% accuracy as with any metric. A weight is introduced to the HEOM values to balance any uncertainty the HEOM calculations introduce. The prediction probability has a strong rate of identifying incorrect samples, as does the HEOM metric. Combining two high-performing metrics will add a layer of certainty to the drift metric, with one of the metrics taking input from the underlying machine learning algorithm and the second metric being the objective of the underlying algorithm. The weight for benign predictions is calculated by dividing the probability of the benign prediction by the average probability of an incorrect benign prediction. The introduction of the weights eliminates any false positives introduced by using the HEOM calculations alone. The calculations used for generating the weights for ransomware, w_r, and benign predictions, w_b, are shown in equation 6 and equation 7, respectively. For the weights added to the HEOM ratio, p_i is the probability of the prediction, $p_{\bar{x}_r}$ and $p_{\bar{x}_b}$ being the average probability of incorrect ransomware or benign prediction respectively. The average incorrect prediction values are calculated by using the training set generated by FeSA.

$$w_r = \frac{p_i}{p_{\bar{x}_r}} \tag{6}$$

$$w_b = \frac{p_i}{p_{\bar{x}_b}} \tag{7}$$

3.3.4. Weighted HEOM Calculation

The weighted HEOM combines a sample's HEOM and prediction probability in our dataset. The weighted HEOM measures drift, the average drift in our dataset, the expected drift, and the drift of individual test samples. The calculation for the weighted HEOM for a ransomware prediction is shown in Equation 8, and the calculation for the weighted HEOM of a benign prediction is shown in Equation 9. A logarithmic value is used for both calculations to shrink the values into a distinguishable range. The calculations for both values differ as the desired effect of the weight is different for both classifications.

$$hw_{b_i} = \log(h_i \cdot w_b) \tag{8}$$

$$hw_{r_i} = \log(\frac{h_i}{w_r}) \tag{9}$$

Ransomware HEOMs are shrunk if the prediction weight is high; however, benign HEOMs increase if the prediction weight is high; this is because the HEOM ratio is measured based on how far each sample is from ransomware on average, meaning that benign weighted HEOMs hw_{b_i} being higher than ransomware hw_{r_i}

is desired. Low prediction weights would have the inverse effect on the HEOM values; ransomware HEOMs would increase, and benign HEOMs would decrease.

Algorithm 1 shows how the weighted HEOM calculations are made. The HEOM distance values from ransomware samples, $heom_r$, and benign samples $heom_b$, are calculated for each sample in the training set, and a ratio value is calculated to generate h_i. The weighted HEOM calculations are carried out for correct ransomware and benign classifications in the training set, and the average weighted HEOM for both correct benign and ransomware classifications are generated. Algorithm 1 has a computational complexity of $O(nlogn)$.

Algorithm 1 Weighted Calculations

```
Input: FeSA features $a_0, ..., a_i$,
Output: Average Weighted HEOMs $\bar{hw}_{r_c}$, $\bar{hw}_{b_c}$
 1: for Samples in training set $x_i....x_n$ do
 2:     Calculate HEOM value from Ransomware samples
 3:     $heom_r = \frac{\sum_{i,j=1}^{n} d_{heom}(x_i,x_j)}{n}$
 4:     Calculate HEOM value from Benign samples
 5:     $heom_b = \frac{\sum_{i,j=1}^{n} d_{heom}(x_i,x_j)}{n}$
 6:     Calculate HEOM ratio for samples
 7:     $h_i = \frac{heom_r}{heom_b}$
 8: end for
 9: for training ransomware samples $x_i...x_n$ do
10:     Calculate Weighted HEOM
11:     $hw_{r_i} = \log(\frac{h_i}{w_r})$
12: end for
13: for training benign samples do $x_i...x_n$
14:     Calculate Weighted HEOM $hw_{b_i} = \log(h_i \cdot w_b)$
15: end for
16: Calculate average correct benign weighted HEOM only using the weighted HEOM of correct benign
    predictions:
17: $\bar{hw}_{b_c} = \frac{\sum_{i=0}^{n} hw_{b_i}}{n}$
18: Calculate average correct ransomware weighted HEOM only using the weighted HEOM of correct ran-
    somware predictions:
19: $\bar{hw}_{r_c} = \frac{\sum_{i=0}^{n} hw_{r_i}}{n}$
20: return $\bar{hw}_{r_c}$ return $\bar{hw}_{b_c}$
```

3.3.5. Expected Drift Threshold Generation

The FeSAD framework uses drift calculations that formally measure how far from an expected point a value is; our drift values attempt to formalize and quantify concept drift in a machine learning detection system. FeSA calculates the average weighted HEOMs for correct benign and ransomware classifications during training. The average weighted HEOM for correct ransomware classifications is $\bar{hw}_{r_c}$ and the average weighted HEOM for correct benign predictions is $\bar{hw}_{b_c}$. The drift calculations measure the difference between the average weighted HEOM in the training set and the weighted HEOM of an individual sample. The weighted HEOM for a sample i is represented with hw_{r_i} for ransomware and hw_{b_i} for benign files. The calculation for drift in a sample classified as ransomware is shown in Equation 10, and the calculation for drift in a sample classified as benign is shown in Equation 11

$$D_{r_i} = hw_{r_i} - \bar{hw}_{r_c} \tag{10}$$

$$D_{b_i} = hw_{b_i} - \bar{hw}_{b_c} \tag{11}$$

The calibration set shown in Figure 1 is used to calculate the expected drift in the system; the expected drift is calculated to allow the system to know what changes in weighted HEOM values to expect. Expected drift D_e is calculated by taking the average drift in the calibration set and the average increase in drift f in datasets from different ransomware distributions. The calculation for the average drift for ransomware in the calibration set is shown in Equation 12, and the average benign drift for benign predictions is shown in Equation 13.

$$\bar{D_r} = \frac{\sum_{i=1}^{n} D_{r_i}}{n} \tag{12}$$

$$\bar{D_b} = \frac{\sum_{i=1}^{n} D_{b_i}}{n} \tag{13}$$

Expected drift is calculated by combining the average drift in the calibration set and the average increase in drift for ransomware f_r and for benign files f_b from different ransomware and benign distributions; this is shown in equation 14 and equation 15 for expected drift in ransomware and benign samples, respectively. The configurations for our calibration sets are shown in Table 4. The calibration dataset contains data from a different distribution to the training dataset; an example would be the training data from 2013-2015, a calibration set from 2018 data, and a test set of data from 2019.

$$D_{e_r} = \bar{D_r} \cdot f_r \tag{14}$$

$$D_{e_b} = \bar{D_b} \cdot f_b \tag{15}$$

Algorithm 2 shows how the drift calculations in the training and calibration sets are carried out. The drift for correct ransomware and benign classifications is calculated using the training set. The increase in drift, f_r, and f_b for both correctly classified ransomware and benign files is calculated using the calibration set. The expected drift for both benign and ransomware samples is calculated by combining the expected drift calculated in the training set and the increase in drift noted in the calibration dataset. Algorithm 2 has a computational complexity of $O(n)$.

3.3.6. Drift Thresholds

The drift thresholds represent levels of drift that require different courses of action; the drift calibration layer allows the FeSAD framework to define what is classified as normal and abnormal drift. Normal drift would require no action as it is within the levels of drift the framework expects. Abnormal drift is beyond the standard threshold for drift; however, it is not beyond the maximum threshold. Abnormally drifting samples are recorded, and too many abnormal samples suggest a need to retrain. The drift calibration layer also defines drift boundaries; drift beyond the drift boundary would suggest a misclassification.

Abnormal Drift. Abnormal drift is defined as a sample that is displaying drift beyond expected drift D_{e_r} or D_{e_b}. The FeSAD framework uses a drift factor metric that determines whether a test sample displays abnormal or normal drift; it is shown in Equation 16 and Equation 17.

$$D_{f_r} = \frac{D_{i_r}}{D_{e_r}} \tag{16}$$

Algorithm 2 Drift Calculations in FeSA & Drift Calibration Layer

```
Input:  Weighted HEOMs $\bar{hw}_{r_c}$, $\bar{hw}_{b_c}$, $f_r$, $f_b$
Output: Expected Drift, $D_{b_r}$, $D_{e_r}$
 1: for All correct classifications in the training set  do
 2:     Calculate drift for correct ransomware classifications:
 3:     $D_{r_i} = hw_{r_i} - \bar{hw}_{r_c}$
 4:     Calculate of drift for correct benign classifications:
 5:     $D_{b_i} = hw_{b_i} - \bar{hw}_{b_c}$
 6: end for
 7: Calculate average drift in training set for benign files:
 8: $\bar{D}_r = \frac{\sum_{i=1}^{n} D_{r_i}}{n}$:
 9: Calculate average drift in training set for ransomware files:
10: $\bar{D}_b = \frac{\sum_{i=1}^{n} D_{b_i}}{n}$
11: Change in ransomware drift calculated using the calibration set: $f_r$
12: Change in benign drift calculated using the calibration set: $f_b$
13: Calculate expected ransomware drift using: $D_{e_r} = \bar{D}_r \cdot f_r$
14: Calculate expected benign drift using: $D_{e_b} = \bar{D}_b \cdot f_b$
15: return $D_{e_r}$ return $D_{e_b}$
```

$$D_{f_b} = \frac{D_{i_b}}{D_{e_b}} \tag{17}$$

If the D_{f_r} value for a sample classified as ransomware is above 1, this would mean the sample is showing abnormal drift for a ransomware classification; this is also the case for D_{f_b} of a sample classified as benign. The abnormal drift threshold is decided by the expected drift value D_e. The expected drift value taking drift from many different distributions allows the abnormal drift threshold boundary to identify true outliers while not flagging samples that represent drift that can be expected. The threshold taking concept drift into account should extend the time between retraining while flagging samples that display abnormal drift. Once z number of samples have been identified as showing abnormal drift, the system needs to be retrained. The user defines the z value.

Maximum Drift. FeSAD calculates the drift thresholds using the training and calibration sets. D_{max} is a level of drift that a sample cannot exceed. Exceeding the D_{max} would suggest an incorrect prediction, so a benign sample displaying drift beyond D_{max} would be considered ransomware and vice versa for samples classified as ransomware. D_{max} is defined as the point at which a sample, according to the weighted HEOM metric, is closer to the opposite class it is classified as. An example of this scenario would be a sample classified as benign but using weighted HEOM; the sample would be closer to ransomware than being benign; therefore, its drift level should exceed D_{max}. D_{max} is calculated based on the training and incorrect classifications in the calibration set. D_{max} is calculated using the training set shown in equation 18. The midpoint is defined and is D_{max}. If the drift of a sample is beyond D_{max} then it will have a $D_{f_{max}}$ greater than 1; this is determined by equation 19.

$$D_{max} = \frac{\bar{hw}_{r_c} - \bar{hw}_{b_c}}{2} \tag{18}$$

$$D_{f_{max}} = \frac{D_i}{D_{max}} \tag{19}$$

3.4. Drift Decision Layer

The Drift Decision Layer uses the drift thresholds calculated using the FeSA and Drift Calibration Layer to accurately and reliably classify samples showing concept drift. The FeSAD framework uses its drift thresholds defined by the FeSA and calibration layer to identify samples that show abnormal drift beyond D_e or drift beyond the drifting boundary, $D_{f_{max}}$. The Drift Decision Layer keeps track of the abnormal samples and will recommend retraining once the threshold, z, for abnormal drifting samples has been reached. Algorithm 3 shows how the drift decision layer works when determining whether a classification shows unacceptable drift. The expected drift values from algorithm 2 are used to determine whether a sample is displaying abnormal drift or unacceptable drift. There is a defined threshold z for the amount of abnormal or unacceptable samples the system can encounter before retraining is advised. Algorithm 3 has a computational complexity of $O(n^2)$.

Algorithm 3 FeSAD Drift Decision Layer

```
Input: Expected Benign Drift $D_{e_b}$, Sample Drift $D_i$, Max Drift $D_{max}$, max abnormal samples $z$
Output: Drift Max Factor $D_{f_{max}}$
 1: Abnormal count $a$
 2: Drift Factor $D_{f_b} = \frac{D_{i_b}}{D_{e_b}}$
 3: if $D_{f_b} > 1$ then
 4:     $D_{i_b}$ = abnormal drift
 5:     $a = a + 1$
 6: end if
 7: Drift Factor Max $D_{f_{max}} = \frac{D_i}{D_{max}}$
 8: if $D_{f_{max}} > 1$ then
 9:     $D_{i_b}$ = unacceptable drift
10:     $a = a + 1$
11: end if
12: if $a > z$ then
13:     Retrain System
14: end if
15: return $D_{f_{max}}$
```

4. Experimental Setup

The FeSAD framework is designed to aid a machine learning ransomware detection to maintain a high level of detection rates under concept drift, extend the classification system's lifespan, and lengthen the time between retraining. The experiments are structured to simulate real-world concept drift in ransomware-, by testing with samples from new ransomware families that display new behaviors, considered to be zero-day to the system. Our results, shown in [15], demonstrate that our experiments use the random forest as the underlying algorithm; the API-call features work best with the Random Forest. Table 2 shows the performance metrics used throughout this paper.

4.1. Test Environment

Our test environment uses a Cuckoo Sandbox that allows for malware and behavioral analysis execution. Cuckoo Sandbox generated the API call data for the ransomware and benign samples used in this research. The WEKA API was used to carry out all machine learning-related experiments. WEKA is a machine-learning platform that can be used for data analysis [39]. The WEKA API allows us to use the machine learning algorithms on the WEKA platform and integrate them into our code. The WEKA API for Java allowed us to integrate Machine Learning algorithms with the FeSAD framework. The WEKA API allowed us to use the data and predictions taken from the machine learning algorithms and integrate them into the predictions made by the FeSAD framework.

Table 2: Performance Metrics

Metric	**Calculation**
TPR (True Positive Rate) / Recall	$\frac{TP}{(TP+FN)}$
False Positive Rate (FPR)	$\frac{FP}{FP+TN}$
False Negative Rate (FNR)	1-TPR
Precision	$\frac{TP}{TP+FP}$
AUC-ROC	$AUC - ROC = \int_0^1 TPR(FPR)\, dFPR$

4.2. Dataset

Our experiments' ransomware and benign data were gathered from 2013 to 2021. Having a large range of data gives us data from different distributions that will simulate the effect of concept drift in a machine-learning system. The base for the dataset used in our experiments is the dataset used in [15]; this dataset was proven to contain concept drift due to the loss in average prediction probability when classifying samples outside the training distribution. The experiments carried out for FeSA [15] show that the average prediction probability for data trained and tested from 2013 to 15 was 0.94; however, when the Random Forest model was tested on ransomware data from 2016 to 2017, the average prediction probability for a classification fell to 0.74. The base dataset contained data from 2013 to 2019; we have added ransomware and benign data from 2020 and 2021 to the updated dataset used in these experiments. The ransomware samples used in the experiments were procured from hybrid analysis and virusshare.com.

4.3. Machine Learning Algorithm Experiments

We have tested prominent machine learning algorithms with the API call features to understand what algorithm works as a good starting point to run our experiment. The FeSAD framework is designed to work with any underlying machine learning algorithm; however, the algorithm's performance will also depend on the feature set provided. Our experiments showed that the Random Forest algorithm was the most stable, having tested its detection capabilities on ransomware spanning eight years. The underlying machine learning algorithms tested were Logistic Regression, J48 Decision Trees, SVM, Bayesian Networks, GTB, Multi-Layer Perceptrons, and Random Forests.

4.4. Concept Drift Experiments

4.4.1. Base Training

The FeSAD framework has been tested in scenarios that simulate concept drift, and its capabilities to predict under concept drift have been evaluated. The FeSAD framework aims to predict reliably under concept drift while prolonging the lifespan of a machine learning model. The focus of these experiments is not on the year the data is from, as the ransomware behavior does not change uniformly year to year; the changes in behavior seen could occur at any point, and they are structured to represent zero-day ransomware that has evolved beyond the scope of the training set. Using a calibration set gives the training a new dimension and allows the system to measure the drift that can occur and incorporate this into predictions of samples that show concept drift. Naturally, we expect a machine-learning algorithm to be uncertain of samples that display concept drift; however, incorporating drift with calibration allows the machine-learning system to predict drifting samples confidently. The FeSA component of the framework produces the optimal feature set, and the average weighted HEOM values in the training set are calculated.

4.4.2. Calibration

The calibration datasets are used to expose the system to data that may differ from what it has been trained on. An example of the calibration sets being used would be if the system is trained on the 2013-2015 data and calibrated using the 2016-2018 data. The variability observed in the calibration sets allows the system to determine the abnormal and unacceptable drift threshold. If the system is not exposed to the calibration sets, the drift values calculated from the training set alone will be limited and will not take drift caused by a different distribution into account; the thresholds for abnormal drift will be too low. We use the training and calibration sets to define the maximum tolerance for drift and the tolerance for abnormal drift.

4.4.3. Test

We test the system on data that the system has not been exposed to, for example, the 2019-2021 data. The system measures the drift of each sample in the test set and will determine whether they are showing abnormal or unacceptable drift. The framework is tested to determine whether the system can accurately classify samples showing concept drift and whether it extends the time between retraining. The system must be retrained once the threshold for abnormal drift samples is exceeded.

5. Results & Discussion

This section explains and discusses our results. The FeSAD framework is built using the WEKA API tool for Java, and an in-house developed concept drift measurement system that evaluates the drift of each sample the detection system classifies.

5.1. Machine Learning Experiments

We tested machine-learning approaches used by prominent ransomware research with the API call features to understand the most robust approaches against concept drift. The underlying machine learning algorithms tested were Logistic Regression, J48 Decision Trees, SVM, Bayesian Networks, GTB, Multi-Layer Perceptrons, and Random Forests. The FeSAD framework is designed to work with any underlying machine learning algorithm; however, the algorithm's performance will also depend on the feature set provided. Table 3 shows the results of the experiments that test the approaches of prominent ransomware research and how these algorithms react to concept drift. Table 3 shows the base performance of each algorithm when the system is not exposed to evolved samples that will not show concept drift, and the detection drop-off for each algorithm represents the average reduction in detection rate when the algorithm is exposed to ransomware outside the training distribution that shows concept drift. Our experiments showed that the Random Forest algorithm used by Poudel et al. was the most stable. The experiments show that the Random Forest [34] is the most consistent algorithm for ransomware detection with our chosen feature set. Random Forests also show the best overall accuracy and balance high detection rate and overall accuracy.

5.1.1. Machine Learning Algorithm Experimental Configuration

We used algorithms and approaches used by prominent ransomware research; however, we have also tried to adapt these approaches to the API-call feature set we use. This approach ensured that the most optimized algorithm would be used as the underlying algorithm for the FeSAD framework. We have trained each classifier on data from 2013-2015, and these classifiers have been tested against data from 2016-2017, 2018, 2019, and 2020-2021. We observe the average drop-off in detection rate across the test datasets, which is shown in Table 3. The drop-off in detection is not uniform as time progresses, and the results in Table 4 suggest that each new batch of ransomware displays a different degree of concept drift.

Random Forests: The random forests used an unlimited depth approach with 100 trees in the forest; the random forest achieved high detection rates of 95.9% on the same distribution test set with a low false-positive rate of 4.5%. The detection drop-off was 11% when exposed to ransomware and benign files displaying concept drift.

Logistic Regression: Logistic regression used conjugated descent and achieved a detection rate of 89.2%; its accuracy was low compared to the better-performing algorithms tested. When exposed to concept drift, the detection rate drop-off was 14%, which means initial detection and degradation over time makes this classifier unsuitable as a base classifier for FeSAD.

SVM: SVM could not achieve strong results; we attempted to use linear, polynomial, sigmoid, and radial basis kernels. None of the kernel configurations could achieve strong base or concept drift results.

J-48 Decision Trees: The J48 decision tree was used with a confidence factor of 0.25 and an unlimited depth. It achieved strong detection and accuracy results but had a relatively high false-positive rate of 7.2%. The detection drop-off was 12%, marginally higher than the random forests.

GTB: The gradient-boosted trees used a J-48 decision tree as its base algorithm and a Z-Max of 3.0. The gradient-boosted trees achieved the same drop-off in detection as the random forests but were weaker in initial and false-positive rates. The gradient-boosted tree attempts to improve the performance of the base tree over multiple iterations, and it does not provide enough additional performance over the J-48 decision tree to justify its use.

Bayesian Networks: The Bayesian network was configured with a simple estimator and a K2 search algorithm. The Bayesian network achieved a strong detection rate of 93.2%; however, its overall accuracy was weak, generating a large volume of false positives at 16.2%. The detection drop-off under concept drift was 11%, but the low overall accuracy made the Bayesian networks unsuitable.

Multi-Layer Perceptron 2 Hidden Layers: The multi-layer perceptron is a back-propagated neural network and uses two hidden layers. The Multi-Layer Perceptron achieved strong accuracy results; however, the ransomware detection 89.2% was not on par with the stronger classifiers tested. In addition to low detection rates, the drop-off in detection under concept drift was 18%.

Deep Learning: This approach used stochastic gradient descent, the Adam updater, and an SGD bias updater. The WEKA Deep Learning for Java library configuration contained various configuration options, and this combination yielded the best results. The number of hidden layers was 10. The deep learning approach was the most resource-intensive and seemed infeasible when using a breeding process like the FeSA layer unless the fitness function was changed, so detection and accuracy were not required. The deep learning approach achieved a low accuracy of 88.1% despite having the highest detection rate of 96%. The detection drop-off was 15%; therefore, the deep learning approach was not as well rounded as the Random Forests despite having a higher detection rate.

Table 3: Machine Learning Experiments

Algorithms	**Performance Statistics**
Random Forest [34]	**Detection:** 95.9% **Accuracy:** 95.5% **FPR:** 4.5% **Precision:** 0.955 **Recall:** 0.955 **Average Detection Drop-off:** 11%
Logistic Regression [14]	**Detection:** 89.6% **Accuracy:** 86.2% **FPR:** 13% **Precision:** 0.871 **Recall:** 0.869 **Average Detection Drop-off:** 14%
SVM [36]	**Detection:** 20.8% **Accuracy:** 60.9% **FPR:** 40.8% **Precision:** 0.716 **Recall:** 0.609 **Average Detection Drop-off:** 9%
J48 Decision Trees [37]	**Detection:** 92.3% **Accuracy:** 92.9% **FPR:** 7.2% **Precision:**0.928 **Recall:** 0.928 **Average Detection Drop-off:** 12%
Gradient Boosted Trees [8]	**Detection:** 93.5% **Accuracy:** 91.2% **FPR:** 8.7% **Precision:** 0.913 **Recall:** 0.912 **Average Detection Drop-off:** 12%
Bayesian Networks [38]	**Detection:** 92.9% **Accuracy:** 83.7% **FPR:** 16.0% **Precision:** 0.905 **Recall:** 0.905 **Average Detection Drop-off:** 11%
Multi-Layer Perceptron [32]	**Detection:** 89.2% **Accuracy:** 94.3% **FPR:** 10.0% **Precision:** 0.902 **Recall:** 0.898 **Average Detection Drop-off:** 18%
Deep Learning [33]	**Detection:** 96.0% **Accuracy:** 88.1% **FPR:** 11.6% **Precision:** 0.891 **Recall:** 0.881 **Average Detection Drop-off:** 15%

Table 4: Experimental Results under Concept Drift

Test Batch Number	Setup	Detection Without FeSAD	Detection Using Only FeSA Layer	Detection with FeSAD Drift Calibration
1	**Train:** 2013-15 **Test:** 2016 **Calibration:** 2017-18	**Detection:** 73.6% **Detection Drop:** 22.7% **FPR:** 19.7% **Precision:** 0.788 **AUC:** 0.770	**Detection:** 93.2% **FPR:** 7.4% **Precision:** 0.926 **AUC:** 0.929	**Detection:** 97.3% **FPR:** 2.1% **Precision:** 0.978 **AUC:** 0.976 **Unexpected Drift Reduction:** 50%
2	**Train:** 2013-15 **Test:** 2018 **Calibration:** 2016-17	**Detection:** 90.6% **Detection Drop:**5% **FPR:** 8.0% **Precision:** 0.918 **AUC:** 0.913	**Detection:** 98.7% **FPR:** 0.6% **Precision:** 0.991 **AUC:** 0.991	**Detection:** 100.0% **FPR:** 0% **Precision:** 1.0 **AUC:** 1.0 **Unexpected Drift Reduction:** 41.5%
3	**Train:** 2013-15 **Test:** 2019 **Calibration:** 2017-18	**Detection:** 85.2% **Detection Drop:** 11.1% **FPR:** 12.0% **Precision:** 0.876 **AUC:** 0.866	**Detection:** 88.3% **FPR:** 8.3% **Precision:** 0.916 **AUC:** 0.901	**Detection:** 92.3% **FPR:** 3.4% **Precision:** 0.967 **AUC:** 0.945 **Unexpected Drift Reduction:** 52.3%
4	**Train:** 2013-15 **Test:** 2020-21 **Calibration:** 2018-19	**Detection:** 82.8% **Detection Drop:** 13.6% **FPR:** 8.0% **Precision:** 0.859 **AUC:** 0.874	**Detection:** 87.8% **FPR:** 10.2% **Precision:** 0.895 **AUC:** 0.888	**Detection:** 90.0% **FPR:** 5.0% **Precision:** 0.947 **AUC:** 0.925 **Unexpected Drift Reduction:** 62%
5	**Train:** 2013-18 **Test:** 2019 **Calibration:** 2020-21	**Detection:** 86.6% **Detection Drop:** 9.8% **FPR:** 12.0% **Precision:** 0.898 **AUC:** 0.873	**Detection:** 89.9% **FPR:** 9.2% **Precision:** 0.907 **Recall:** 0.904	**Detection:** 96.8% **FPR:** 7.1% **Precision:** 0.931 **AUC:** 0.949 **Unexpected Drift Reduction:** 50%
6	**Train:** 2013-18 **Test:** 2020-21 **Calibration:** 2019	**Detection:** 90.4% **Detection Drop:** 5.8% **FPR:** 5.0% **Precision:** 0.947 **AUC:** 0.923	**Detection:** 91.1% **FPR:** 7.1% **Precision:** 0.944 **AUC:** 0.912	**Detection:** 94.1% **FPR:** 5.1% **Precision:** 0.927 **AUC:** 0.945 **Unexpected Drift Reduction:** 48%

5.2. Concept Drift Experiments

The results in Table 4 demonstrate FeSAD performance on data displaying concept drift. Table 4 compares the detection rate achieved by the feature set chosen by FeSA to the detection rates FeSAD achieves by using the calibration set and considering concept drift. Table 4 also demonstrates the reduction rate in unexpected drift in the testing data. An abnormal sample is defined as a sample that demonstrates drift beyond what is expected if a sample does display concept drift. The structure of the experiments has the system initially observe the training data, define thresholds for abnormal and unacceptable drift, and then classify test samples. The initial experiments did not expose FeSAD to a calibration dataset outside the distribution of the training set. The parameters for drift are decided based on the training data alone, and we observe how many samples are classified as abnormal. The system must be retrained once the number of observed abnormal samples is reached. The number of abnormal observations or proportion of abnormal observations per defined batch of test samples determines whether the system is still considered fit for purpose. A large proportion of abnormal observations would mean a shift in concept; the model will be less certain in its predictions and will need retraining. Table 4 shows a reduction rate in abnormal observations; this metric is measured in the concept drift experiments. Section 2.6.1 shows the formulas used to apply the expected drift from calibration sets to the base expected drift from the training set. We observe that considering the drift from the calibration and training sets means the system has a wider understanding of the drift it can expect in test samples. With the abnormal drift parameters altered according to expected drift, the average reduction in abnormal observations across all experiments is 50.6%; this leads us to conclude that the FeSAD framework can help increase the periods between retraining as the lower the proportion of abnormal samples, the lesser the need to retrain.

Table 5: Navarra University Dataset with Concept Drift

	FeSAD with Random Forest	**Random Forest [34]**	**Logistic Regression [14]**	**SVM [36]**	**J48 Decision Trees [37]**	**GTB[8]**	**Bayesian Networks[38]**	**MLP[32]**	**Deep Learning[33]**
Tested on Training Distribution	**Detection:** 99.9% **FPR:** 0.4% **Precision:** 0.998 **AUC:** 0.997	**Detection:** 99.5% **FPR:** 0.4% **Precision:** 0.998 **AUC:** 0.995	**Detection:** 87.0% **FPR:** 10.4% **Precision:** 0.955 **AUC:** 0.883	**Detection:** 86.6% **FPR:** 10.8% **Precision:** 0.954 **AUC:** 0.881	**Detection:** 96.6% **FPR:** 2.8% **Precision:** 0.985 **AUC:** 0.969	**Detection:** 95.3% **FPR:** 3.8% **Precision:** 0.984 **AUC:** 0.958	**Detection:** 99.9% **FPR:** 0.2% **Precision:** 0.996 **AUC:** 0.999	**Detection:** 10% **FPR:** 69.3% **Precision:** 0.617 **AUC:** 0.2035	**Detection:** 88.9% **FPR:** 8.9% **Precision:** 0.952 **AUC:** 0.90
Tested on Zero-Day Distribution	**Detection:** 94.3% **FPR:** 0.8% **Precision:** 0.943 **AUC:** 0.968	**Detection:** 89.5% **FPR:** 10.5% **Precision:** 0.999 **AUC:** 0.895	**Detection:** 45.6% **FPR:** 54.3% **Precision:** 0.999 **AUC:** 0.467	**Detection:** 50.9% **FPR:** 49.1% **Precision:** 0.991 **AUC:** 0.509	**Detection:** 78.1% **FPR:** 21.9% **Precision:** 0.998 **AUC:** 0.781	**Detection:** 86.0% **FPR:** 14.0% **Precision:** 0.994 **AUC:** 0.860	**Detection:** 92.1% **FPR:** 0.9% **Precision:** 0.994 **AUC:** 0.956	**Detection:** 0.00% **FPR:** 99.9% **Precision:** 0.0 **AUC:** 0.01	**Detection:** 50.0% **FPR:** 0.5% **Precision:** 0.999 **AUC:** 0.748

The results presented in Table 5 are from a dataset built by researchers from the University of Navarra. Berrueta et al. have constructed a ransomware dataset from 70 different ransomware strains from 2015 to 2019; this dataset used network traffic based on a file-sharing system and I/O features for ransomware and benign files collected between 2015 and 2019. This dataset is conveniently set up for zero-day testing, with the testing set comprising exclusively of data the training set will not contain. The Navarra University contains network traffic data from 70 strains of ransomware and 2500 hours of network data from benign files. The Navarra University dataset is designed to evaluate ransomware detection systems and contains ransomware spanning four years, so it fits our purposes and tests the FeSAD framework with network data. The Navarra University dataset contains 3867 ransomware behavioral samples and 11159 benign behavioral in its training set and 114 ransomware behavioral samples and 14000 benign behavioral samples in its zero-day test set. We have used the algorithms used in prominent ransomware research to compare with FeSAD and how the detection metrics compare when the classifiers are presented with the zero-day data most likely to contain concept drift. We observe that most of the tested algorithms perform well through 10-fold cross-validation except the Multi-Layer Perceptron, which seems unable to classify ransomware based on the I/O and network features. Based on the initial training and test, the highest performers are the Random Forest, J48 Decision Tree, Gradient Tree Boosting, and the Bayesian Network; however, every algorithm faces a significant performance drop-off. The Logistic Regression, SVM, MLP, and Deep-Learning approaches experience a significant drop-off, with detection rates falling to around 50%. The Bayesian Network and Random Forest show the least degradation out of the tested algorithms besides FeSAD. The Random Forest without FeSAD achieves a detection rate of 89.5% on the zero-day data without FeSAD, and FeSAD combined with the Random Forest achieves a detection rate of 94.3%.

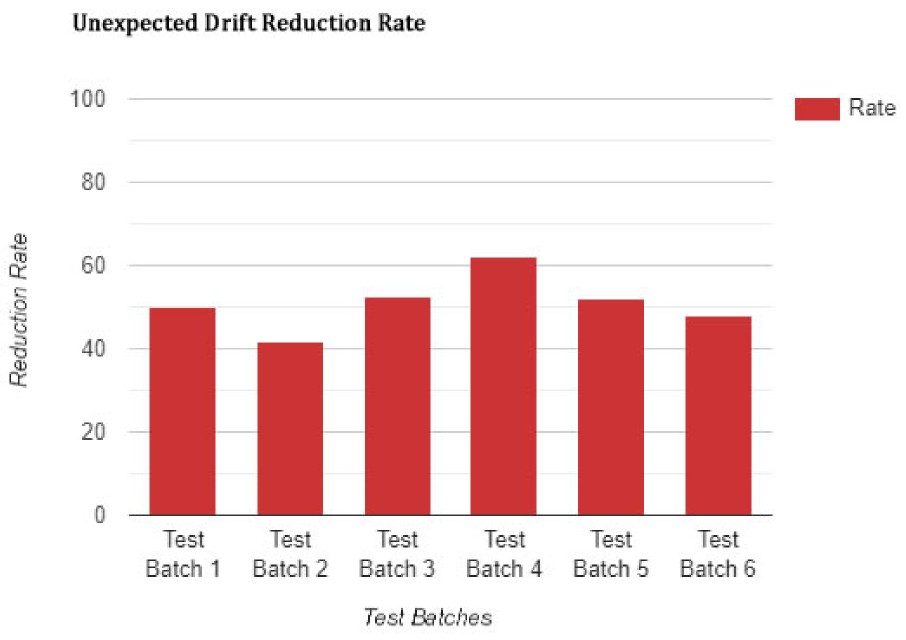

Figure 2: Unexpected Drift Drop-off Rates

5.3. Discussion

The results achieved by FeSA [15] have been improved, as shown in Table 4; the FeSA layer used a genetic algorithm to generate optimal feature sets for ransomware detection when concept drift was involved. The FeSAD framework's first objective was to improve the detection results obtained in the FeSA layer consistently; this has been achieved as shown in Figure 3; FeSAD demonstrates consistently stronger detection results than the base FeSA layer.

Our results show that the FeSAD framework can improve a machine learning system's detection rates under concept drift. Figure 3 shows that taking concept drift into account when making a prediction increases the rates of correct predictions for ransomware and, as Table 4 shows, a reduction in the false-positive rate. Figure 2 shows the reduction rate for abnormal instances detected during all of the test runs for the system. Test batches 1, 2, 3, and 4 shown in Table 4 give the best representation of FeSAD's ability to maintain a high detection rate despite being exposed to data showing concept drift. We trained the Random Forest on data from 2013-2015 and tested in separately on test data sets from 2016 to 2021. Using calibration sets, the system trained on 2013-2015 data to maintain detection rates above 90% for data from various distributions. Test batches 5 and 6 in Table 4 show that using calibration sets with an expanded training set keeps detection rates at a 95% average. The results presented in this section also indicate that API calls effectively detect ransomware from different distributions. API calls give an insight into the behavior of a program during early execution; evidently, ransomware consistently uses similar API call patterns over time despite showing concept drift.

The results in Table 4 also show that the occurrence of abnormal samples reduces by 50% on average by using the FeSAD framework's approach of using calibration datasets outside the training set's distribution to prepare the system for drift. Abnormal samples are defined in section 5.2. The FeSAD framework calculates the initial drift boundary and expected drift using the training set; therefore, we compare the number of samples that show unexpected drift using the drift boundary and expected drift on the training set first. The FeSAD framework calibrates the expected drift threshold and drift boundary based on the calibration set. The reduction in the unexpected drift reduction rate is calculated by comparing the number of test samples that show unexpected drift or drift beyond the drift boundary using initial drift boundaries calculated by

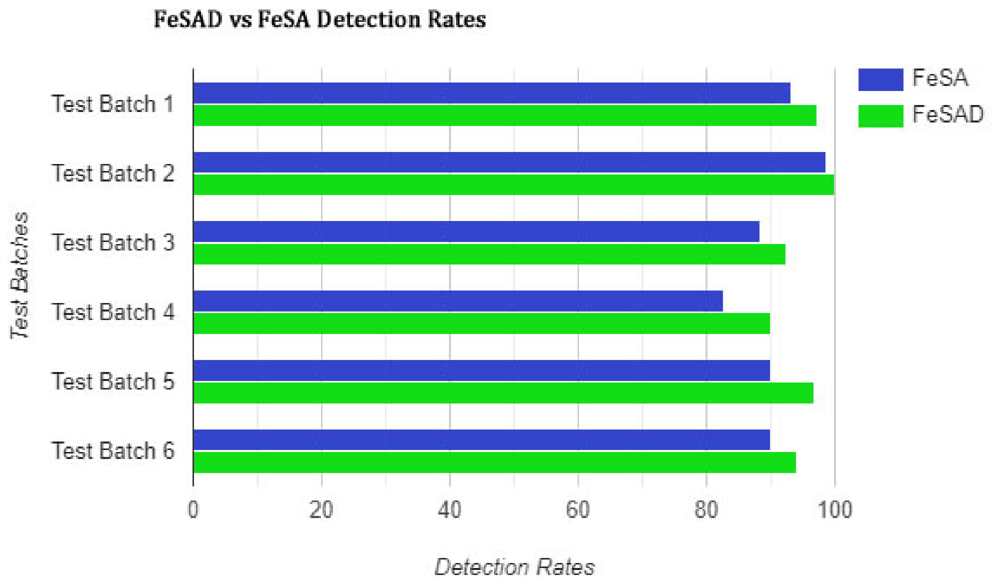

Figure 3: FeSA vs FeSAD Detection Rates

the FeSA layer and the test samples that show unexpected drift using the boundaries calculated by the drift calibration layer. The FeSAD framework defined in Figure 1 will retrain a classifier if too many samples drift beyond what is expected. The average reduction of 50% shows that a detection system's lifespan can be extended significantly with FeSAD. The FeSAD framework shows the ability to measure drift and compensate for it accurately; however, there will be cases where retraining will be unavoidable. Test batch 2 shows a situation where the reduction rate of abnormal samples is lower than the 50% average due to the high volume of ransomware samples showing abnormal levels of concept drift. Despite test batch 2 retaining a high detection rate, the classifier is less sure of its predictions, and samples show statistical drift at a higher-than-expected volume. It would be safest to retrain a system when there is a high level of drift to avoid a sudden drop-off in detection. The ransomware samples from test batch 2 are from 2018, and this batch of ransomware displayed more abnormal behavior than the other test batches despite maintaining high detection rates.

The structure of the experiments is designed to ensure that the FeSAD framework can withstand concept drift and the differing degrees of concept drift that ransomware could pose. Each dataset represents a level of concept drift that could co-occur, as our observations show that ransomware does not evolve linearly. The year of each test dataset does not mean that we would ideally realistically expect a detection system to go untrained for 2-3 years; however, the test structure is required to prove the robustness of the FeSAD framework and the fact that it can accurately classify ransomware from different eras. Test batches 1, 2, and 3 use training data from 2013-15, but with calibration datasets from different ransomware distributions, maintaining a detection rate above 90% for data up to ransomware from 2020-21. The results in Table 4 also show that without FeSAD, the underlying algorithms would need retraining faster; test batches 1, 3, and 4 show the drop-off in the detection rates of the classifier trained on 2013-15 data without FeSAD or any feature reduction. The results in Table 4 show that the lifespan of a classifier can be extended by the fact that, with FeSAD, a classifier maintains strong detection rates on ransomware data several years ahead of the training data; test batches 1 to 4 accurately demonstrate this. Without the FeSAD framework, the ransomware detection system trained in 2013-15 would not be effective on the ransomware data from 2016 onwards; however, with FeSAD, the drop-off in detection rate is slowed significantly. Based on the results shown in Table 4, using a calibration data set in addition to a training set allows the FeSAD framework to extend the lifespan of its underlying machine learning algorithm by factoring in the level of concept drift to expect from a different distribution of ransomware.

5.4. Ablation Study Results

Table 6: Ablation Study

Case Study	**Settings**	**Result**
Random Forest Depth	**Depth:** 5 **Depth:** 10 **Depth:** 15 **Depth:** Unlimited	**Detection:** 71.9% **Detection:** 93.0% **Detection:** 91.8% **Detection:** 90.6%
Feature Selection	**Algorithm:** FeSA **Algorithm:** Best First **Algorithm:** Evolution **Algorithm:** Genetic **Algorithm:** Harmony **Algorithm:** Greedy	**Detection:** 95.6% **Detection:** 87.4% **Detection:** 92.3% **Detection:** 86.7% **Detection:** 88.8% **Detection:** 73.7%
Drift Metric Altered	**Drift Metric Removed:** Prediction Probability **Drift Metric Removed:** HEOM Ratio **Drift Metric Removed:** HEOM Weights Removed	**Detection Loss:** 6.5% **False Positive:** +7.0% **Detection Loss:** 4.8% **False Positive:** +6.2% **Detection Loss:** 9.3% **False Positive:** +8.4%
FeSA Version	**FeSA Version: 2**	**Detection:** +1.2% **False Positive:** -6.0%

The results of the ablation study explore how altering or removing components of the FeSAD system affects system performance. The optimal depth of the random forest was 15, as going deeper than this figure only reduced detection and increased false positives. Increasing the depth beyond 10 harmed the detection rate as there was a drop in detection rate of 1.2% when increasing to a depth of 15; however, the false positive rate was reduced by close to 8% . The FeSAD system was expected to easily compensate for the reduction in detection rate by increasing the tree depth. We have extensively tested different feature selection algorithms against the FeSA layer, and the detailed results of these tests can be found in [15], where we justify the use of the genetic algorithm in the FeSA layer and why it is effective in a concept drift scenario. Table 6 also shows the results of removing different components of the drift calculations; the performance statistics for the FeSAD system with and without the drift calibration and decision layers are included in Table 5. Table 6 shows that removing any components for drift calculation negatively affects detection rates and false positives, reinforcing the necessity for each component. The HEOM and weights derived from the prediction probability give a partial picture when used independently; however, when combined, they give a complete picture of whether a sample is misclassified. The combined metrics used to derive drift help significantly reduce the rate of benign files classified as ransomware and maintain a high ransomware detection rate. The final case study explored in the ablation study analyses the effect of the FeSA updates; we observe a significant reduction in false positives with a slight increase in detection rate with the new FeSA layer configuration described in section 3.2.4.

6. Conclusion & Future Work

In conclusion, the FeSAD framework has proven effective in detecting ransomware that shows concept drift; the approach the system also takes successfully extends the lifespan of a machine learning system. Our experiments allowed us to test the system in the most realistic and versatile way possible. The system is also exposed to a range of benign files that display various behaviors, some resembling behavior patterns close to ransomware. The FeSAD framework provides valuable insight into ransomware behavioral patterns and the changes that occur in these behavioral patterns. The FeSAD framework could be used differently to integrate directly into its underlying algorithms. It may be useful to integrate the drift data taken from the FeSAD directly into machine learning algorithms as features and then measure the differences between performance; it would be helpful to view how a machine learning classifier processes drift data without FeSAD being used as an independent framework that aids the classifier. The theory behind the FeSAD framework could be expanded to be tested on other types of malware, not just ransomware. The FeSAD framework's primary application is for ransomware, and the feature sets and calibrations work mainly to maximize ransomware detection; however, it can be changed and tuned to work for other types of malware. Other types of malware could be detected if the approach to feature selection is altered and the type of features gathered are tuned to adapt to the malware type in question. The FeSAD framework's drift boundaries could be refined and tuned beyond the midpoint between the two distributions. The use of probabilities from the underlying classifiers could also be calibrated to ensure further reliability; this is in the context of tuning prediction probabilities instead of using default thresholds of 0.5 for binary classification. Overall, there are aspects of the conformal evaluator, such as its retrain and performance thresholds, which could be integrated into the FeSAD framework, and values like credibility and confidence could be used in place of a similarity metric according to the FeSAD framework.

References

[1] R. Richardson and M. M North, *Ransomware: Evolution*, mitigation and prevention," International Management Review, vol. 13, no. 1, p. 10, 2017

[2] Symantec Corporation. Internet security threat report, 2016

[3] N. Scaife, H. Carter, P. Traynor, and K. R. B. Butler, "CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data," in Proceedings of the International Conference on Distributed Computing Systems (ICDCS), June 2016

[4] M. Clancy, The true cost of Ransomware, 2021, [Accessed: 10/05/2022] Available at: https://www.backblaze.com/blog/the-true-cost-of-ransomware/

[5] P.Saxena, Breed of MBR Infecting Ransomware âĂŞ an analysis by Quick Heal Security Labs, 2018 [2018] [Accessed 12/05/2022] Available at: https://blogs.quickheal.com/breed-mbr-infecting-ransomware-analysis-quick-heal-security-labs/

[6] Y. A. Ahmed, B. Koçer, S. Huda, B. A. Saleh Al-rimy, and M. M. Hassan, "A system call refinement-based enhanced Minimum Redundancy Maximum Relevance method for ransomware early detection," Journal of Network and Computer Applications, vol. 167, p. 102753, 2020/10/01/ 2020

[7] H. Carson, Cyberhawk, 2007, [Accessed: 10/05/2022] Available at: www.kickstartnews.com/reviews/utilities/cyberhawkzeroday

[8] S. Shaukat, V. Ribeiro, RansomWall: A Layered Defence System against Cryptographic Ransomware Attacks using Machine Learning, Proc. 10th Int. Conf. Commun. Syst. Netw. (COMSNETS), Jan. 2018,pp. 356âĂŞ363.

[9] U. Urooj, M. A. B. Maarof and B. A. S. Al-rimy, "A proposed Adaptive Pre-Encryption Crypto-Ransomware Early Detection Model," 2021 3rd International Cyber Resilience Conference (CRC), 2021, pp. 1-6, doi: 10.1109/CRC50527.2021.9392548.

[10] D. Min, Y. Ko, R. Walker, J. Lee and Y. Kim, "A Content-based Ransomware Detection and Backup Solid-State Drive for Ransomware Defense," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, doi: 10.1109/TCAD.2021.3099084.

[11] M. Almousa, S. Basavaraju and M. Anwar, API-Based Ransomware Detection Using Machine Learning-Based Threat Detection Models, 2021 18th International Conference on Privacy, Security and Trust (PST), 2021, pp. 1-7.

[12] B. Qin, Y. Wang and C. Ma, "API Call Based Ransomware Dynamic Detection Approach Using TextCNN," 2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), 2020, pp. 162-166.

[13] X. Zhang, J. Wang and S. Zhu, "Dual Generative Adversarial Networks Based Unknown Encryption Ransomware Attack Detection," in IEEE Access, vol. 10, pp. 900-913, 2022

[14] D. Sgandurra & L. Munoz-Gonzalez & R. Mohsen & E. Lupu, Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection, 2016, CoRR abs/1609.03020

[15] D.W. Fernando and N. Komninos, *FeSA: Feature selection architecture for ransomware detection under concept drift*, Computers & Security Volume 116, 2022.

[16] A. Tsymbal, M. Pechenizkiy, P. Cunningham, S. Puuronen , *Dynamic integration of classifiers for handling concept drift*, Information Fusion, Volume 9, Issue 1, 2008,Pages 56-68

[17] R. Jordaney, K. Sharad, S.K Dash, Z. Wang, D. Papini, L. Cavallaro, Transcend: Detecting Concept Drift in Malware Classification Models, Proceedings of the 26th USENIX Security Symposium August 16âĂŞ18, 2017, Vancouver, 2018

[18] J.J. Cranford , T. Amey, I. Barak, CyberReason ThreatHunters vs Ransomware attack Simulation, 2022, [Accessed 10/04/22], Available at:https://www.cybereason.com/watch-on-demand-hunter-attack-sim

[19] I. Khamassi, M. Sayed Mouchaweh, M. Hammami, K. Ghedira, Ensemble classifiers for drift detection and monitoring in dynamical environments, Conference: annual conference of the prognostics and health management society 2013, New Orleans, USA, 2013.

[20] D.R. Wilson. , T.R.Martinez.: Improved heterogeneous distance functions, Journal of Artificial Intelligence Research 6, 1997 pp.1âĂŞ34.

[21] G.Tan, P.Zhang, Q.Liu, X.Liu, C.Zhu, F.Dou, Adaptive Malicious URL Detection: Learning in the Presence of Concept Drifts, 2018, 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering, New York, NY, USA

[22] S. Singhal, U. Chawla and R. Shorey, "Machine Learning & Concept Drift based Approach for Malicious Website Detection," 2020 International Conference on COMmunication Systems & NETworkS (COMSNETS), 2020, pp. 582-585

[23] F.Barbero, F.Pendlebury, F.Pierazzi, L.Cavallaro Transcending Transcend: Revisiting Malware Classification in the Presence of Concept Drift, 2022, IEEE Symposium on Security & Privacy (Oakland), 2022

[24] M.M.Gaber, and Y.Kovalchuk, "EACD: evolutionary adaptation to concept drifts in data streams", Data Mining and Knowledge Discovery (2019) 33:663-694, 2019

[25] A.Kantchelian, A.Afroz, S.Huang, A.Islam, B.Miller, M.Tschantz, R.Greenstadt, A.D.Joseph, J.D.Tygar, Approaches to adversarial drift. In AISec 13, Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security, Co-located with CCS 2013, Berlin, Germany, November 4, 2013, pp. 99âĂŞ110.

[26] S. S. Hussain, M. Hashmani, V. Uddin, T. Ansari and M. Jameel, A Novel Approach to Detect Concept Drift Using Machine Learning, 2021 International Conference on Computer & Information Sciences (ICCOINS), 2021, pp. 136-141.

[27] M.Baena-Garcia, J.Del Campo-Avila, R.Figaldo, A.Bifet, R.Gavalda, R.Morales-Bueno, Early Drift Detection Method, Early drift detection method. Fourth International Workshop on Knowledge Discovery from Data Streams. 2006.

[28] Roberto S.M. Barros, Danilo R.L. Cabral, Paulo M. Gonçalves, Silas G.T.C. Santos, RDDM: Reactive drift detection method, Expert Systems with Applications, Volume 90, 2017, Pages 344-355,

[29] G. J.Ross , N. M.Adams , D. K.Tasoulis , & D. J.Hand (2012). Exponentially weighted moving average charts for detecting concept drift. Pattern Recognition Letters, 33 (2), 191âĂŞ198 .

[30] K. Nishida, & K.Yamauchi, (2007). Detecting conceptdriftusing statistical testing. In V. Corruble, M. Takeda, & E. Suzuki (Eds.), Proceedings of the 10th international conference on discovery science (DS 07) . In LNCS: 4755 (pp. 264âĂŞ269). Springer .

[31] A.Bifet , & R.Gavaldà, (2007). Learning from time-changing data with adaptive windowing. In Proceedings of the 7th SIAM international conference on data mining (SDM 07),Minneapolis, MN, USA (pp. 4 43âĂŞ4 48).

[32] S.Maniath, A.Ahok, R.P.Poornach, V.G.Sujadev, A,U Prem Sankar, S.Jan, Deep Learning LSTM based Ransomware Detection, 2017. In Proceedings of the Recent Developments in Control, Automation & Power Engineering (RDCAPE), Noida, India, 26–27 October 2017; pp. 442–446.

[33] A.Tseng, Y.Chen, Y.Kao, T.Lin, Deep Learning for Ransomware Detection.*Aragorn 2016 DeepLF*, 2016.

[34] S.Poudel, P.Subedi, D.Dasgupta, A Framework for Analyzing Ransomware using Machine Learning, 2018. In Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI), Bengaluru, India, 18–21 November 2018.

[35] G.Cusack, O.Michel, E.Keller, Machine Learning-Based Detection of Ransomware Using SDN, 2018. In Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks &Network Function Virtualization, SDN-NFV Sec'18; Tempe, AZ, USA, 19–21 March; pp. 1–6.

[36] M.Hasan, M. Rahman, RansHunt: A Support Vector Machines Based Ransomware Analysis Framework with Integrated Feature Set,2017, In 20th International Conference of Computer and Information Technology (ICCIT); Dhaka, Bangladesh 22–24 December 2017, pp. 1–7.

[37] H.Daku, P.Zavarsky, Y.Malik, Behavioural-Based Classification and Identification of Ransomware Variants Using Machine Learning, 2018. In Proceedings of the 2018 17th IEEE International Conference On Trust, Security And Privacy, NY, USA, 1–3 August 2018; pp. 1560–1564.

[38] A.O.Almashhadani, M.Kaiiali, S.Sezer, P.A.O'Kane, Multi-Classifier Network-Based Crypto Ransomware Detection System: A Case Study of Locky Ransomware, 2019, *IEEE Access* 2019, *7*, 47053–47067.

[39] Weka 3: Machine Learning Software in Java. 2018. Available online:https://www.cs.waikato.ac.nz/ml/weka/index.html (accessed on 21 November 2022).

[40] E. Berrueta, D. Morato, E. Magaña and M. Izal, "Open Repository for the Evaluation of Ransomware Detection Tools," in IEEE Access, vol. 8, pp. 65658-65669, 2020, doi: 10.1109/ACCESS.2020.2984187.

[41] Cuckoo Sandbox Hooked API calls Available online:https://github.com/cuckoosandbox/cuckoo/wiki/Hooked-APIs-and-Categories (accessed on 15 December 2022).

[42] S.Poudel, P.Subedi, and D.Dasgupta, , A Framework for Analyzing Ransomware using Machine Learning, 2018. In Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI).

[43] S.Poudal and D.Dasgupta, AI-powered ransomware detection framework, 2020 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2020.

[44] M.S.Abbasi, H.Al-Sahaf, M.Mansoori, and I.Welch, 2022. Behavior-based ransomware classification: A particle swarm optimization wrapper-based approach for feature selection, Applied Soft Computing, Volume 121.

[45] M.A.Ayub, A.Continella, A.Siraj, 2020. An I/O Request Packet (IRP) Driven Effective Ransomware Detection Scheme using Artificial Neural Network,IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI), pp. 319-324

[46] S.R.B.Alvee, B.Ahn, T.Kim, Y.Su, Y.Youn, and M.Ryu, 2021. Ransomware Attack Modeling and Artificial Intelligence-Based Ransomware Detection for Digital Substations, 2021 6th IEEE Workshop on the Electronic Grid (eGRID), pp. 01-05

Damien Warren Fernando received a MSci in Computer Science and Cyber Security in 2017 from City, University of Lon-don. Having worked at City University of London as a Teaching assistant since late 2017, he is now a 4th year PhD student at City, University of London with an interest in researching Ran-somware. While being a part time employee of City, University of London as a Teaching Assistant to the Cyber Security course, Damien provides teaching support along with managing and up-grading the Cyber Security penetration testing environment used by students.

Dr Nikos Komninosreceived his PhD in 2003 from Lancaster University (UK) in Communication Systems with an Information Security focus. He is currently a Senior Lecturer (US System: Associate Professor) in Cyber Security in the Department of Computer Science at City University London. Part of his research has been patented and used in mobile phones by Telecommunication companies; in crypto-devices by Defense companies; and in healthcare applications by National Health Systems. Since 2000, he has participated, as a researcher or principal investigator, in many European and National R&D projects in information security, systems and network security. He has authored and co-authored more than 80 journal publications, book chapters and conference proceedings publications in his areas of interest. He has been invited to give talks at conferences and Governmental Departments and train employees in Greek and UK businesses.