Ransomware Detection Using Portable Executable Imports

Tanatswa Ruramai Dendere and Avinash Singh
University of Pretoria, South Africa

u19058927@tuks.co.za
asingh@cs.up.ac.za

Abstract: In recent years, there has been a substantial surge in ransomware attacks, wreaking havoc on both organizations and individuals. These attacks, driven by the lure of profits, particularly with the widespread use of cryptocurrencies, have prompted attackers to continuously develop innovative evasion techniques and obfuscation tactics to avoid detection. Ransomware, employing seemingly benign functions such as encryption and file-locking, poses a formidable challenge for detection as it evolves beyond traditional signature-based methods. Consequently, there is a growing need to identify previously unexplored and unstudied ransomware strains, necessitating the deployment of artificial intelligence (AI) to discern the unique characteristics and objectives of ransomware. The adoption of AI hinges on the prior selection of distinguishing features. Given that ransomware's intent fundamentally differs from that of benign files, there are variations in the structure of Portable Executables (PE) files. This study posits that the imports used by PE files can serve as a discriminating factor between ransomware and benign files. This research explored using machine learning models to detect ransomware by analysing and deriving insights from the PE Imports structure. To achieve this, the study trains seven machine learning classifiers, namely Random Forest, Logistic Regression, Naïve Bayes, Support Vector Machine, K-Nearest Neighbors, Gradient Boost, and Decision Tree. These models are trained on a dataset of carefully selected features derived from PE imports. The classifiers are benchmarked and ranked based on several evaluation metrics, including latency, accuracy, and confidence levels. For a model to be effective in ransomware detection, it should offer near real-time and highly confident accuracy. In other words, it should exhibit low latency, high accuracy, and strong AUC rates. Among the models, Logistic Regression emerges as the top performer, identifying ransomware programs with an impressive 98.5% accuracy and a confidence level of 98.6% within a mere 0.998-millisecond latency. This study conclusively affirms the efficacy of employing PE imports for ransomware detection.

Keywords: Ransomware detection, Portal executable, DLL, Machine learning, Feature selection

1. Introduction

There is an inherent need for robust cybersecurity techniques to deal with the emerging forms of cybercrime as the number of internet-enabled devices increases (A. Alissa et al., 2022; Nadir and Bakhshi, 2018). With the adoption of information technology came the development of malicious software, also known as malware (Manavi and Hamzeh, 2022). Ransomware is a form of malware that encrypts and/or restricts access to the data of a user and then requests a ransom payment to get the original data back (Manavi and Hamzeh, 2020). The evolution of cryptocurrencies has led to an increase in ransomware as payment traceability becomes difficult (Manavi and Hamzeh, 2022). This is due to the development of cryptocurrencies such as Bitcoin and Monero (Kok et al., 2022; Li et al., 2021). Furthermore, the development of Ransomware as a Service (Raas) makes ransomware appealing to cyber criminals because it has become easier to execute an attack and yields good profits (Razaulla et al., 2023; Sharmeen et al., 2020). There are two major types of ransomware (Masum et al., 2022), namely Crypto-ransomware and Locker-ransomware. Crypto-ransomware encrypts data that it deems would be important to the victim but does not interfere with the basic operating system functions (Razaulla et al., 2023). It uses strong encryption algorithms such as AES256 and RSA2048, which are difficult to break (Manavi and Hamzeh, 2020). Locker-ransomware has two different types of attacks. The first type is Screen Locker-ransomware, which locks a user out of their system and only displays the screen with payment instructions (Razaulla et al., 2023). Screen Locker-ransomware aims to disrupt basic computer functions whilst still allowing the Operating System (OS) to boot (Beaman et al., 2021). It is easy to resolve this type of ransomware by rebooting the infected device in safe mode or using a boot-time virus scanner (Beaman et al., 2021). The second type of attack is a Complete Locker-ransomware attack. This attack hinders the usage of the device it has infected by locking the bootstrapping procedure which disallows the OS to load. Research has shown many approaches for ransomware detection (Alqahtani and Sheldon, 2022; Aslan and Samet, 2020; Razaulla et al., 2023; Singh et al., 2022). The traditional approach for classification is the signature-based detection method (Manavi and Hamzeh, 2022; Rezaei Tina and Hamze Ali, 2020). With signature detection, the signature of a suspected file is used to compare to a set of known ransomware signatures to find a match (Rezaei Tina and Hamze Ali, 2020). If a match is found, the file is identified as ransomware, however, this only performs well against known ransomware(Aslan and Samet, 2020; Singh et al., 2019). Recent research explores machine learning techniques with the aim of having a robust method that can detect new ransomware that has not been evaluated or studied before (Ravi et al., 2022). In machine learning, classification models are trained with the extracted and selected

features in order to detect ransomware. The literature realises machine learning algorithms like Support Vector Machines, Random Forest, Logical Regression, Decision Tree, Naïve Bayes, K-Nearest Neighbour and Gradient Boost have potential efficacy for the classification and detection of ransomware (Masum et al., 2022; Razaulla et al., 2023).

Feature extraction and selection is the most important aspect of classification models for the detection of ransomware (Manavi and Hamzeh, 2020). Features are input values that contain the relevant information for solving a particular problem (Khalid et al., 2014). Feature extraction and selection is a step to select data that can provide the most insight to increase the learning ability of a model. Without it, the input will be redundant, misleading, or irrelevant which will increase the search space and inhibit the learning process of a model (Khalid et al., 2014). Extracting meaningful features that can help identify ransomware is vital for detection (Beaman et al., 2021). Static and dynamic ransomware analysis techniques provide meaningful information that can be used for detection. With static analysis, the application does not need to be run to determine whether the file is malicious or not (Manavi and Hamzeh, 2020). Some of the possible static analysis features are found in the Portable Executable (PE) file, PE header and parts of the PE header such as PE imports and PE sections, bytes, and opcodes (A. Alissa et al., 2022; Ravi et al., 2022; Vidyarthi et al., 2019). On the other hand, the dynamic analysis method is based on the behaviour of the program and requires it to be executed. Examples of dynamic analysis features include logs of the activities of the registry, the file system, and the Dynamic Link Libraries (DLL) (Manavi and Hamzeh, 2022). A hybrid analysis method utilises aspects of both static and dynamic methods (Manavi and Hamzeh, 2020). Static features are fast and less costly to extract, whereas dynamic features take more time and require computational resources to extract information. Though they may be more effective than static features, they are not as viable for near-real-time detection (Manavi and Hamzeh, 2020). Dynamic features can also be easily avoided if the malware applies a time delay, or anti-debugging mechanisms (Rezaei Tina and Hamze Ali, 2020). Presenting a real-time and fast detection system is critical in ransomware detection (Rezaei Tina and Hamze Ali, 2020). This requires a detection tool that is based on static features to have a high detection rate and low false positives. Ransomware authors are constantly innovating to evade detection by adding to the structure with code obfuscation techniques (Sharmeen et al., 2020).

This research proposes a detection method that is based on the external functions utilised by PE files, known as PE imports. Ransomware is commonly packaged and distributed as a PE file specifically targeting Microsoft Windows OS (Beaman et al., 2021; Singh et al., 2019). A PE file can easily be executed on an operating system because it sums up the information necessary for the operating system to manage and load it in memory, without installation. To allow this, PE files make use of Dynamic Link Libraries (DLL) to help perform specific functions and make the PE file portable and have a smaller file size. Therefore, a PE file contains a list of libraries and functions on which the application relies to function. This research proposes to analyse the PE imports in order to discriminate a ransomware file from a benign one.

The rest of the paper is organised as follows: Section 2 presents a review of some of the recent works in ransomware detection. Section 3 presents the proposed method of ransomware detection and its results. In Section 4, provides an analysis of the proposed method and discussion. Finally, in Section 5 the conclusion and future works are presented.

2. Related Work

A literature study was conducted to understand the scope, possibilities and limitations of ransomware detection. In the work by Manavi and Hamzeh (2022), ransomware detection is conducted by using the header section of the PE files. This header provides information about executable types that do not require any preprocessing or additional extraction of specific features. The authors created a graph using the PE header, which maps to the feature space using a "power iteration" method (Manavi and Hamzeh, 2022). This map converts PE headers into feature vectors for training a Random Forest (RF) classifier. With this approach, small changes in the header bytes cause a small number of edge weights in the graph to change, while leaving many edges unchanged. This enables the classifier to be able to identify new variations of malware. Ransomware has become fast-changing, and the variation in the header will increase exponentially as this malware family evolves (Sharmeen et al., 2020). The number of changes in the edges of the graph will be more consequential causing this classifier to require increasingly frequent updates in order to stay accurate in detecting ransomware. Similarly, in other work by Manavi and Hamzeh (2020), ransomware is identified using the header of executable files, by constructing an image. To classify the images, a Convolutional Neural Network (CNN) is employed to extract features. The authors accomplished this by extracting the headers from each executable file, resulting in a 1024-component vector. Each element within this vector contains values ranging from 0 to 255, which were subsequently

employed to create a grayscale image. This use of CNN deep learning on a PE header produced a 93.33% accuracy. The authors prioritised choosing features that had low computational complexity and time overhead, which can be a major drawback. Their method does not do feature reduction in order to get the most effective features in determining the label of the files. Therefore, the method resulted in a substandard accuracy percentage.

Razaulla et al. (2023) provides a comprehensive survey on state-of-the-art research in ransomware detection. The paper surveys recent works from 125 ransomware research publications published between 2016 to 2022. It gives a summary of machine learning-based and non-machine learning-based ransomware detection contributions. The survey identifies that research is heading towards using machine learning models as detection mechanisms for ransomware. According to their readings, the authors of the survey outline that the identified techniques fall under machine learning detection. However, the effectiveness of this approach is somewhat constrained due to the occurrence of concept drift(Razaulla et al., 2023). Concept drift is a phenomenon in the field of machine learning that arises when the statistical properties of the target variable, which the model aims to predict, change over time. Consequently, the semantics of input data initially used to train the model, have undergone significant transformations over time, potentially rendering it irrelevant to new or current data. This can lead to inaccurate or suboptimal predictions by the model, specifically in its ability to classify or detect ransomware. The authors propose that further investigation into reverse engineering the most destructive ransomware from recent years could serve as a potential solution to address the challenges associated with concept drift.

Azeez et al. (2021) study puts forward an ensemble learning-based method for malware detection using features of the entire PE file. The ability of ensemble learning-based systems to learn and generalise enables the development of intelligent information protection systems. The paper employed five machine learning algorithms namely: Gradient Boosting, Decision Tree, Random Forest, AdaBoost, and Naïve Bayes. It employed the Multilayer Perceptron and the One-Dimensional (1D) CNN model as the deep learning models. The authors trained the baseline machine learning methods and noted the results achieved. They implemented an ablation study to find the best model by investigating the models with various settings of their hyperparameters, to measure their performance. The highest-performing models were chosen as the first-stage classifiers. The experiments were made on the Windows Portable Executable (PE) malware dataset. The best results were achieved through the ensemble of seven neural networks. The ExtraTrees classifier was the final-stage classifier, and it achieved a 100% accuracy rate for their database.

Previous researchers have utilised static analysis features for ransomware detection with machine learning algorithms which led to rapid detection. However, the approaches were found to have a weak impact on detection accuracy (Manavi and Hamzeh, 2020). Through an ensemble learning-based method, static analysis features can be effectively utilised in ransomware detection (Azeez et al., 2021). Ensemble learning leverages a combination of different learning algorithms to attain enhanced predictive performance surpassing what each individual algorithm can achieve on its own. Azeez et al. (2021) have shown this method improves accuracy to 100% accuracy in discriminating ransomware from benign files. However, this paper utilised features of the whole PE file. This is an expensive manual process which would be difficult to recreate for new variants as soon as they are released, leaving a gap for the possibility of new infections (Sharmeen et al., 2020). In order to address these limitations a more robust and swift detection mechanism is inherently needed that stems from the PE information to provide near real-time detection. Therefore, this research proposes to explore using only the PE imports to train a combination of machine learning models as a ransomware detection mechanism. This approach extracts a low number of features which will reduce the computational costs and allow for quick model training, making it viable to be used in real-time ransomware detection.

3. Ransomware Detection Using PE Imports

The proposed method of ransomware detection entailed determining how ransomware differs from benign files with regard to the utilisation of PE imports. Figure 1 demonstrates the overall process this research has explored. The process starts with conducting research on ransomware properties with regard to the PE Imports ransomware utilises to carry out its objective, in order to find out what features distinguish ransomware from benign files. Thereafter, with the information gained from experimentation, a malware feature engineering generation platform called MalFe (Singh et al., 2023) was used to build a custom dataset that consist of six features that were extracted from the PE imports. Since the PE imports are raw categorical data, this research encapsulates the data into numerical features that can be processed by Machine Learning (ML) algorithms. Once the data has been collected, the next step is ML exploration. Thereafter, this study explores and benchmarks

various common ML techniques for classification. The ML classifiers explored are the Support Vector Machine, Decision Tree, Logistic Regression, Random Forest, Gradient Boost, Naïve Bayes and K-Nearest Neighbours. These algorithms were tested to see which provides the best accuracy with the greatest confidence rate in the least amount of time. The following subsections expand on how the features are selected, the data acquisition and preprocessing for the machine learning exploration.

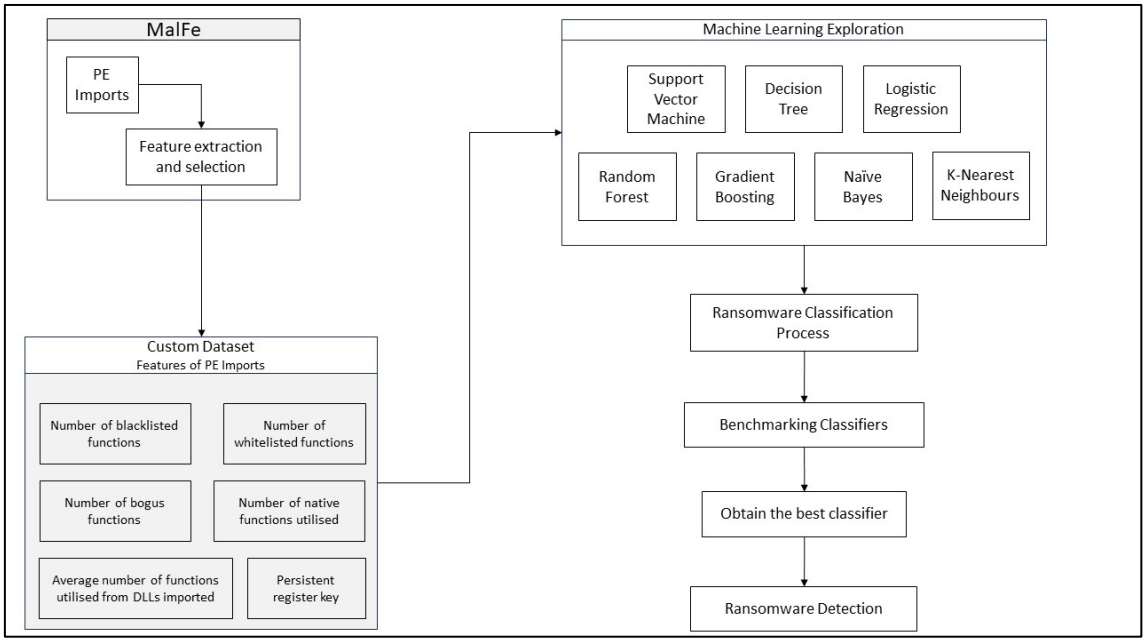

Figure 1: Ransomware detection using PE imports overview process model

3.1 Dataset Creation

The performance of a model and its capacity to generalise is directly impacted by the quality, quantity, and diversity of the data used for training the model. Therefore, feature selection and extraction are a crucial aspect of machine learning. Sufficient data collection ensures the model has access to a balanced representation of different classes in order to make neutral differentiations. For this reason, the classes chosen for detection are malicious (M) and benign (B). The data was extracted and acquired by making use of MalFe platform (Singh et al., 2023). The platform allows security researchers to use cuckoo reports of analysed malware samples to build their own datasets. MalFe builds the datasets from benign samples collected across different categories of executables. The dataset for this research contains a total of 137 benign samples and 1694 samples of ransomware chosen from different ransomware families. From these samples, this research used MalFe to extract and select features from PE imports. The extracted features are further discussed.

- The average number of functions utilised from DLLs imported

The analysis of ransomware and benign file properties showed that benign files typically have a more uniform number of functions utilised from each DLL imported. In contrast, ransomware can be more sporadic in its utilisation of the functions. Therefore, this research calculates the average number of functions utilised per each DLL imported. In order to standardise the comparison across the PE files, this study calculated the average as a ratio of the total number of DLLs imported in the PE file.

- The number of seemingly bogus functions present in the PE imports

Further analysis showed that one of the obfuscation techniques utilised by ransomware to evade detection is adding functions that do not exist into the PE imports. The ransomware authors do this in order to change the signature of the ransomware without adding extra functionality. They typically do this by adding non-alphanumeric characters to names of existing PE import function names. Therefore, this research counts how many functions in the PE imports are non-alphanumeric.

- The number of blacklisted functions present in the PE imports

Since the goal of ransomware is different from the objective of benign files, there will ultimately be a list of functions that show that distinction. This research compiled a list of PE import functions found to be prevalent

in ransomware as well as existing studies (Arabo et al., 2020; Hampton et al., 2018). The functions are WriteConsoleW, Process32NextW, Process32FirstW, CreateToolhelp32Snapshot, CoInitializeSecurity, MoveFileWithProgressW, CryptEncrypt, CryptExportKey, CryptGenKey, CryptDeriveKey, CryptDecodeObject, CryptImportPublicKeyInfo, socket, DrawTextExW, GetForegroundWindow.

- The number of whitelisted functions present in the PE imports

Similarly, since benign and ransomware programs serve different purposes, this research compiled a list of functions that are most likely to be invoked more in benign files than in ransomware (Karanam, 2023). The functions are DeviceIOControl, SetFileTime, SHGetFolderPathW. DeviceIOControl is employed for communication with I/O devices by interfacing with device drivers and utilizing control codes for specific functions. This method is predominantly used by legitimate software to facilitate the execution of their intended functions. In contrast, ransomware typically seeks to disrupt I/O devices. SetFileTime is a utility employed to establish and modify the timestamps indicating the last access, modification, and creation of a particular file or directory. On the other hand, SHGetFolderPathW is utilized for retrieving folder paths. Ransomware creators tend to avoid the use of these functions to evade detection by obscuring traceability.

- An indicator of a persistent registry key

A distinguishing characteristic of ransomware is its ability to maintain registry keys even after a computer has been rebooted. Ransomware authors achieve this by intentionally preserving registry keys that were either opened or created by the PE file. Alternatively, in some instances, they may delete a different registry key to obfuscate detection efforts. To underscore this differentiation, this research employs a methodology that involves subtracting the number of calls made to functions responsible for opening registry keys from the number of calls made to functions tasked with deleting and closing registry keys. The study has revealed a notable pattern for benign files, where the difference between these counts predominantly leans towards -1, further emphasizing the contrasting behaviour of benign and malicious files in terms of registry key handling.

- The number of native functions utilised in the PE imports

The native API serves as the highest-level functionality in kernel mode and, at the same time, as the lowest-level functionality in user mode. In the Win32 API, native API functionality is mostly undocumented. This has attracted the attention of various sorts of ransomware authors. They avoid the documented Win32 APIs with the aim of carrying out their dubious tasks while evading detection. The native API functions are prefixed with 'Nt' or 'Zw' (Chappell, 2023). This research counted the number of utilised functions in the PE imports that are native functions as a discriminatory feature of ransomware.

The process of feature selection has resulted in the creation of a refined dataset, as shown by a sample snippet in Table 1. This dataset has been carefully curated to showcase certain key attributes that are pivotal for distinguishing between benign files and ransomware files. Notably, the dataset sample illustrates the average utilization of functions imported from Dynamic Link Libraries (DLLs). For benign files, this average falls within a relatively narrow range of 3.3 to 3.4, providing a sense of uniformity. In contrast, ransomware files exhibit a less constant pattern in the number of functions they utilize, reflecting the inherent variability in their behaviour. Additionally, a noteworthy observation in this dataset is the presence of what appear to be spurious or seemingly bogus functions, which tend to average around three functions per PE file. However, for ransomware files, the count of such functions displays a more erratic pattern. The full dataset can be found at https://malfe.cs.up.ac.za/datasets/view/Ransomware_Detection_Using_Features_of_PE_Imports_2/22.

Table 1: Dataset Sample

label	ave_functions_ utilised_from_ dlls_imported	bogus_ functions	num_ blacklisted_ functions	num_ whitelisted_ functions	persistent_ reg_key	num_ native_ functions
B	3,3469	3	0	1	-1	0
B	3,3673	3	0	1	-1	0
B	3,3469	3	0	1	-1	0
M	2,0938	4	7	0	0	0
M	2,9444	6	1	0	1	1
M	2,6250	2	2	1	0	1

3.2 Machine Learning Exploration

This research chose seven ML classifiers to be benchmarked on this dataset. It explored the classifiers with a training/validation/testing ratio of 70/15/15. To optimise the performance of the models, the research used hyper-parameter tuning to avoid overfitting, with 5-fold cross-validation to ensure the model is robust and to minimise data leakage. This study evaluated the models by comparing the time taken for detection and their Area Under the Curve (AUC), Recall, Precision, F1- Score and Accuracy. AUC measures the confidence of the prediction of a model regardless of the classification threshold. The Recall rate is a measure of the ability of the model to identify all the relevant cases within the dataset. It is an important metric if missing a positive case would be costly. Precision measures the ratio of correctly predicted positive instances to all instances predicted as positive. The F1-Score provides a balanced measure between Recall and Precision. Lastly, Accuracy measures the proportion of accurately classified instances.

4. Results and Discussion

To evaluate and benchmark the classifiers, this research ranked the evaluation metrics according to their relevance. In this research, the latency of the ML model, its accuracy and confidence level are the most important metrics. However, due to the imbalance in the classes of this dataset, the accuracy result may be skewed. Therefore, the metrics were ranked as follows: Latency, AUC, Recall, Precision, F1- Score and Accuracy. The training results of the models are shown in Table 2.

Table 2: Training Results

Classifier	Latency (milliseconds)	AUC	Recall	Precision	F1 Score	Accuracy
Decision Tree	1.003	0.9978	0.9975	0.9966	0.9970	0.9945
Gradient Boosting	3.989	0.9981	0.9983	0.9992	0.9987	0.9977
K Nearest Neighbours	56.463	0.9944	0.9983	0.9966	0.9975	0.9953
Logistic Regression	1.982	0.9938	0.9975	0.9966	0.9970	0.9945
Naïve Bayes	1.072	0.9761	0.9924	0.9966	0.9945	0.9899
Random Forest	6.968	0.9969	0.9975	0.9966	0.9970	0.9945
Support Vector Machine	30.429	0.9921	0.9983	0.9966	0.9975	0.9953

In order to pick the top three ML models, the models were ranked for each metric. The Decision Tree and the Naive Bayes algorithm had the quickest detection time. They both had a latency of approximately one millisecond. However, Naive Bayes ranked last in all the other metrics partly because of the low number of features and the interrelated nature of these features. The worst-performing models in terms of latency were the Support Vector Machine and K-Nearest Neighbours algorithms which took 30.4 milliseconds and 56.5 milliseconds respectively. After evaluating the latency, with regard to the other metrics, we chose the Decision Tree algorithm and eliminated the Naive Bayes, Support Vector Machine and K Nearest Neighbours. The next metric to evaluate was the AUC, which is a measure of confidence in the prediction of a model. The results ranged from 99.2% to 99.8% for all models, except for Naive Bayes which produced 97.61%. However, the Gradient Boost algorithm had the highest AUC rate at 99.8% followed by the Decision Tree at 99.78% and the Random Forest at 99.69%. Figure 2 shows the Receiver Operating Characteristic(ROC) Curve of this evaluation, which is a plot of all the AUC rates for the seven ML models.

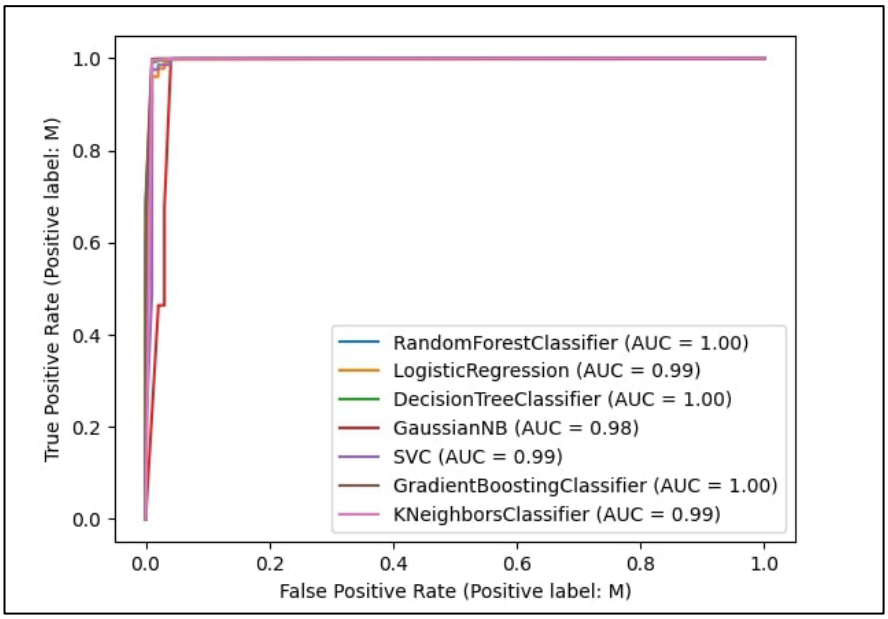

Figure 2: Area Under the ROC curve of the explored classifiers

The ranking of the ML models is identical for the rest of the metrics. Gradient Boost is first, followed by K-Nearest Neighbours and Support Vector Machine tied for second. The Decision Tree, Random Forest and Logistic Regression are tied at third place and as previously mentioned, Naive Bayes is last. With the exception of the recall rate, where Gradient Boost is tied first with K-Nearest Neighbours and Support Vector Machine, these results left the choice of the final top 3 models to either Random Forest or Logistic Regression. Logistic Regression was chosen over Random Forest because its latency was 1.98 milliseconds while the Random Forest was 6.97 milliseconds. Therefore, after this evaluation, the top three models are the Decision Tree, the Gradient Boost and the Logistic Regression algorithms. These three algorithms are run through the validation set and their results are compared to choose the best ML classifier to test with on this dataset. The results of the validation are shown in Table 3.

Table 3: Validation results of top three classifiers

Classifier	Latency (milliseconds)	AUC	Recall	Precision	F1 Score	Accuracy
Gradient Boosting	1.996	0.9732	0.9961	0.9961	0.9961	0.9927
Decision Tree	2.069	0.9675	1.0000	0.9961	0.9980	0.9964
Logistic Regression	0.997	0.9891	0.9882	0.9960	0.9921	0.9855

The top classifier is chosen by examining the latency, the AUC and the accuracy of the models. Logistic Regression is the fastest detection model by approximately 1 millisecond. Figure 3 demonstrates the clear distinction in latency of the three models. Logistic Regression is also the model with the highest confidence level. Though the Decision Tree has an accuracy rate of 99.6%, which is the highest among the three models, this accuracy rate only differs by approximately 1% from 98.6% which is the accuracy rate of the Logistic Regression model. Figure 4 shows the accuracy and AUC of the top three algorithms. Since the dataset has classes that are imbalanced, the accuracy metric cannot weigh as much as the latency and AUC metrics. Therefore, the Logistic Regression was chosen as the classifier to test the dataset. With a latency of 0.9978 milliseconds, the Logistic Regression model can detect ransomware with 98.55% accuracy at a confidence level of 98.57%. This model, utilising PE imports, was able to learn the generalised structure of ransomware and be able to detect it in near real time.

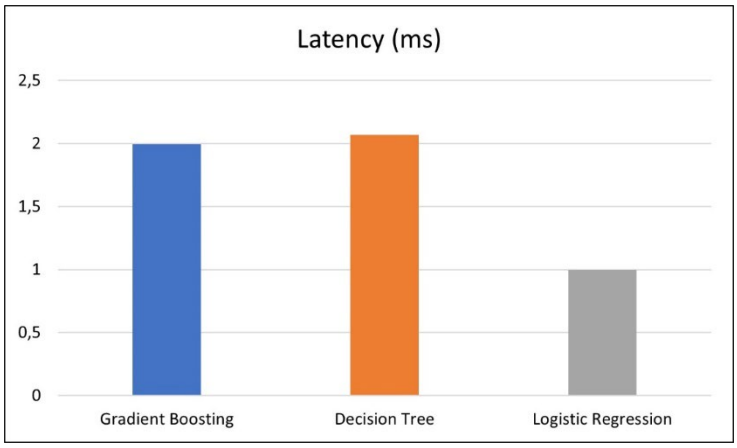

Figure 3: Latency of top three classifiers

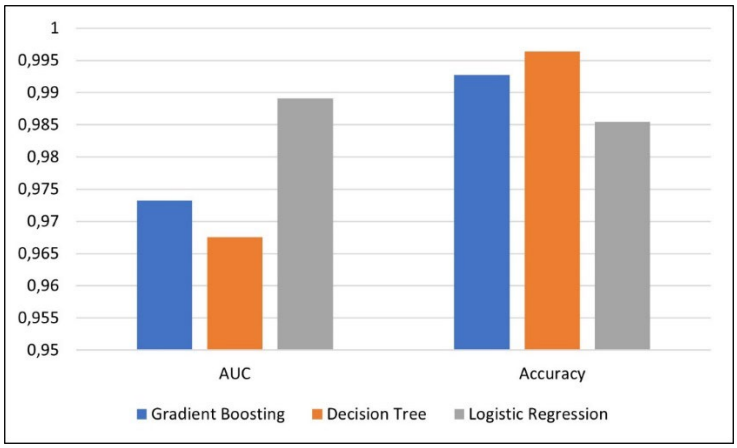

Figure 4: AUC and accuracy of top three classifiers

As compared to other detection models, the superiority of this proposed model is its fast detection times and impressive accuracy scores. The execution time for the proposed method in Manavi and Hamzeh (2020) was 3105.58 seconds for a dataset of 2000 executable files in PE format, averaging a latency of 1.5 seconds. For a similar dataset in Manavi and Hamzeh (2022) the testing was 62.84 seconds resulting in a latency of 30 milliseconds. The proposed methods of these papers had 93.33% and 93.30%, respectively. Compared to this paper, which utilised a small number of features solely from the PE imports, both of these papers used features from the entire PE file to detect ransomware. Thus, the proposed method had a latency of 0.9978 milliseconds with 98.5% accuracy for a dataset of 1831 samples.

5. Conclusion and Future Work

Ransomware authors make use of benign security functionality, such as encryption to lock and disable a user from using their data on their device. They also constantly evolve their software to evade detection using obfuscation techniques. This creates a need for ransomware detection mechanisms that can identify ransomware that has not been evaluated or seen. This paper employed PE imports in the context of ransomware detection, utilising a dataset that included 1831 records, consisting of 1694 malicious samples and 137 benign samples. The research findings highlighted that by harnessing various machine learning algorithms, PE imports can offer valuable insights for enhancing ransomware detection. The study achieved impressive accuracy scores ranging from 98.99% to 99.77%, with Logistic Regression demonstrating the best performance among the tested algorithms. By utilising the PE imports, this paper selected features that distinguish ransomware from benign files and are most likely to remain the same, even as ransomware evolves. This is a static analysis solution, meaning it does not need to run the program to extract the features. In addition, the proposed method extracts a low number of features. Therefore, this proposed solution is fast and optimal for detection. The best performing model, Logistic Regression, had a latency of 0.998 milliseconds, further showing that the first line of defence needs to be a static analysis solution that can detect ransomware before it is ever executed. However, the dataset of PE imports used in this paper is not sufficient as it had an imbalance between the malicious and benign classes. This can yield misleading performance metrics. For example, accuracy depends on the patterns

the model has learned during training. Therefore, future work would involve improving the dataset by balancing the classes of the samples. However, this paper makes a valuable contribution to the field of security by introducing innovative and rapid response methods for detecting ransomware using static analysis features.

References

A. Alissa, K., H. Elkamchouchi, D., Tarmissi, K., Yafoz, A., Alsini, R., Alghushairy, O., Mohamed, A., Al Duhayyim, M., 2022. Dwarf Mongoose Optimization with Machine-Learning-Driven Ransomware Detection in Internet of Things Environment. Applied Sciences (Switzerland) 12. https://doi.org/10.3390/app12199513

Alqahtani, A., Sheldon, F.T., 2022. A Survey of Crypto Ransomware Attack Detection Methodologies: An Evolving Outlook. Sensors. https://doi.org/10.3390/s22051837

Arabo, A., Dijoux, R., Poulain, T., Chevalier, G., 2020. Detecting ransomware using process behavior analysis, in: Procedia Computer Science. Elsevier B.V., pp. 289–296. https://doi.org/10.1016/j.procs.2020.02.249

Aslan, O., Samet, R., 2020. A Comprehensive Review on Malware Detection Approaches. IEEE Access. https://doi.org/10.1109/ACCESS.2019.2963724

Azeez, N.A., Odufuwa, O.E., Misra, S., Oluranti, J., Damaševičius, R., 2021. Windows PE malware detection using ensemble learning. Informatics 8. https://doi.org/10.3390/informatics8010010

Beaman, C., Barkworth, A., Akande, T.D., Hakak, S., Khan, M.K., 2021. Ransomware: Recent advances, analysis, challenges and future research directions. Comput Secur 111. https://doi.org/10.1016/j.cose.2021.102490

Chappell, G., 2023. Native API Functions [WWW Document]. Geoff Chappell, Software Analyst.

Hampton, N., Baig, Z., Zeadally, S., 2018. Ransomware behavioural analysis on windows platforms. Journal of Information Security and Applications 40, 44–51. https://doi.org/10.1016/j.jisa.2018.02.008

Karanam, S., 2023. Ransomware Detection Using Windows API Calls and Machine Learning. Virginia Polytechnic Institute and State University, Virginia.

Khalid, S., Khalil, T., Nasreen, S., 2014. A survey of feature selection and feature extraction techniques in machine learning, in: Proceedings of 2014 Science and Information Conference, SAI 2014. Institute of Electrical and Electronics Engineers Inc., pp. 372–378. https://doi.org/10.1109/SAI.2014.6918213

Kok, S.H., Abdullah, A., Jhanjhi, N.Z., 2022. Early detection of crypto-ransomware using pre-encryption detection algorithm. Journal of King Saud University - Computer and Information Sciences 34, 1984–1999. https://doi.org/10.1016/j.jksuci.2020.06.012

Li, Y., Yang, G., Susilo, W., Yu, Y., Au, M.H., Liu, D., 2021. Traceable Monero: Anonymous Cryptocurrency with Enhanced Accountability. IEEE Trans Dependable Secure Comput 18, 679–691. https://doi.org/10.1109/TDSC.2019.2910058

Manavi, F., Hamzeh, A., 2022. A novel approach for ransomware detection based on PE header using graph embedding. Journal of Computer Virology and Hacking Techniques 18, 285–296. https://doi.org/10.1007/s11416-021-00414-x

Manavi, F., Hamzeh, A., 2020. A New Method for Ransomware Detection Based on PE Header Using Convolutional Neural Networks, in: Proceedings of 17th International ISC Conference on Information Security and Cryptology, ISCISC 2020. Institute of Electrical and Electronics Engineers Inc., pp. 82–87. https://doi.org/10.1109/ISCISC51277.2020.9261903

Masum, M., Jobair Hossain Faruk, M., Shahriar, H., Qian, K., Lo, D., Islam Adnan, M., 2022. Ransomware Classification and Detection With Machine Learning Algorithms. 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC) 316–322.

Nadir, I., Bakhshi, T., 2018. Contemporary cybercrime: A taxonomy of ransomware threats & mitigation techniques, in: 2018 International Conference on Computing, Mathematics and Engineering Technologies: Invent, Innovate and Integrate for Socioeconomic Development, ICoMET 2018 - Proceedings. Institute of Electrical and Electronics Engineers Inc., pp. 1–7. https://doi.org/10.1109/ICOMET.2018.8346329

Ravi, V., Alazab, M., Selvaganapathy, S., Chaganti, R., 2022. A Multi-View attention-based deep learning framework for malware detection in smart healthcare systems. Comput Commun 195, 73–81. https://doi.org/10.1016/j.comcom.2022.08.015

Razaulla, S., Fachkha, C., Markarian, C., Gawanmeh, A., Mansoor, W., Fung, B.C.M., Assi, C., 2023. The Age of Ransomware: A Survey on the Evolution, Taxonomy, and Research Directions. IEEE Access. https://doi.org/10.1109/ACCESS.2023.3268535

Rezaei Tina, Hamze Ali, 2020. An Efficient Approach For Malware Detection Using PE Header Specifications. International Conference on Web Research (ICWR) 234–239.

Sharmeen, S., Ahmed, Y.A., Huda, S., Kocer, B.S., Hassan, M.M., 2020. Avoiding Future Digital Extortion through Robust Protection against Ransomware Threats Using Deep Learning Based Adaptive Approaches. IEEE Access 8, 24522–24534. https://doi.org/10.1109/ACCESS.2020.2970466

Singh, A., Ikuesan, A.R., Venter, H.S., 2019. Digital Forensic Readiness Framework for Ransomware Investigation, in: Digital Forensics and Cyber Crime. pp. 91–105.

Singh, A., Ikuesan, R.A., Venter, H., 2023. MalFe—Malware Feature Engineering Generation Platform. Computers 12, 201. https://doi.org/10.3390/computers12100201

Singh, A., Ikuesan, R.A., Venter, H., 2022. Ransomware Detection using Process Memory, in: ICCWS 2022 17th International Conference on Cyber Warfare and Security. pp. 413–422.

Vidyarthi, D., Kumar, C., Rakshit, S., Chansarkar, S., 2019. Static Malware Analysis to Identify Ransomware Properties. IJCSI International Journal of Computer Science 16, 10–17. https://doi.org/10.5281/zenodo.3252963