National College of Ireland

Enhancing Zero-Day Malware Detection in Enterprise Networks Using Behaviour-Based Machine Learning Models

MSc Research Project
MSc in Cybersecurity

Farhanahmad Quraishi
Student ID: x23165367

School of Computing
National College of Ireland

Supervisor: Diego Lugones

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Farhanahmad Quraishi
Student ID:	x23165367
Programme:	MSc in Cybersecurity **Year:** 2024-25
Module:	Practicum Part 2
Supervisor:	Diego Lugones
Submission Due Date:	12/12/24
Project Title:	Enhancing Zero-Day Malware Detection in Enterprise Networks Using Behaviour-Based Machine Learning Models
Word Count:	**9222** **Page Count: 22**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Farhanahmad Quraishi

Date: 12/12/24

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing Zero-Day Malware Detection in Enterprise Networks Using Behaviour-Based Machine Learning Models

Farhanahmad Quraishi
x23165367

Abstract

In the field of cybersecurity and machine learning, zero-day malware poses a significant threat to enterprise networks because of its ability to counter any in-place security and exploit unknown vulnerabilities. This eventually renders the traditional signature-based detection methods very ineffective and so this research study presents a complete framework which can allow the zero-day malware detection enhancement. This enhancement is done by combining behavior-based analysis with advanced machine learning techniques. Utilizing the EMBER dataset, a large-scale collection of labeled Windows Portable Executable (PE) files, both static and behavior-based features were extracted such that internal patterns of the malwares were captured and understood. This study involves a powerful methodology with extensive data preprocessing, feature engineering through dimensionality reduction techniques like Principal Component Analysis (PCA), and the various machine learning models. These machine learning models included Random Forest, Gradient Boosting, XGBoost, and LightGBM classifiers. There was one more attempt of combining these ML models i.e. an autoencoder-based anomaly detection mechanism was used to identify, slight or major, deviations from normal behavior which ended up enhancing the variants' detection of unseen malware. These ML models were evaluated using a combination of traditional performance metrics, accuracy, precision, recall, F1-score, but also the cybersecurity-specific evaluation measures such as Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR). The Random Forest classifier emerged as the best-performing model. The model showed extraordinary results against common malware evasion techniques i.e. code obfuscation and packing. This was primarily because of the addition of the behavior-based features which have been known to be less susceptible to static code changes. The hybrid model used in this study, which combined the anomaly detection with classification, had improved recall but also showed a minor increase in the false positives. This meant that the need for balance in detection sensitivity was important. Furthermore, to cater the practical application of this research, this study's implementation also included the development of a malware detection API and a simple intrusion detection system (IDS). This study also had some limitations which included the exclusion of dynamic features because of resource constraints and simulated MTTD and MTTR eval metrics, but it also showed that the results had potential for real-world deployment.

Keywords: Zero-Day Malware Detection, Behavior-Based Analysis, Machine Learning, Random Forest Classifier, Anomaly Detection, Enterprise Network Security, EMBER Dataset, Cybersecurity.

1 Introduction

In the ever-evolving landscape of enterprise security needs there has been a growing trend from malicious attacks, i.e. zero-day attacks which form sophisticated malicious software, are difficult to handle, and have held the interest of the research community at large. Traditional

signature-based detection methods are rendered ineffective against such threats and thus this creates an urgent need for innovative solutions to address this giant issue. Building upon the work of Hindy et al. (2020), which explores the use of deep learning techniques for zero-day attack detection, this research study focuses on the integration of behavior-based analysis with machine learning (ML) to increase the enterprise network security. Hindy et al. showed in their 2020 research study the potential of deep learning in identifying malicious activities, but their work primarily focused on detection accuracy i.e. only focusing on the ML aspect without diving deeply into the real-time application within Intrusion Detection Systems (IDS) and this is where our research comes in extending their foundation research study by combining the ML-driven behavior analysis within a practical IDS framework. In this research study machine learning is not viewed as the centerpiece but only as an enabling tool to enhance the efficacy of cybersecurity systems. The primary focus of this study remains to be the improvement of the IDS ability in real-time detection and action response to zero-day threats. By using the behavior-based techniques and analysis we capture runtime qualities such as system calls and API usage which will help us in bridging the gaps in traditional IDS implementations. These behavior-based features are more robust and stronger against any evasion techniques like code obfuscation and packing. Integrating these suggested features into a hybrid IDS framework will not only increase the organization's security posture but will also optimize the response time to such emerging and evolving threats. Thus, this study builds upon and extends the baseline research of Hindy et al. such that a comprehensive cybersecurity solution is made for enterprise environments.

These zero-day malware attacks stand out from the rest of the cyber-attacks due to their ability to exploit previously unknown vulnerabilities which makes the traditional signature-based detection mechanisms highly ineffective and outdated. Moreover, these zero-day attacks take advantage of the software's undisclosed flaws which becomes very difficult to prevent from and this is because these attacks do not provide any prior signatures for conventional antivirus solutions to detect. In this study to better understand the zero-day malware attacks, it is important to understand the traditional malware detection techniques which have predominantly relied on signature-based methods and thus are unable to do anything against such glass-canon one-shot malware attack natures i.e. they relied on matching known patterns of malicious code against incoming files or network traffic Hindy et al. (2020). There is no denying that such antiquated traditional methods were effective against previously identified threats, but they are heavily at a disadvantage when facing novel malware variants or zero-day exploits and thus the limitations of such signature-based detection are rendered useless against zero-day threats which can obfuscate code and change their signature pattern to evade detection Guo et al. (2023).

Thus, there is a growing need for methods which could have cognitive understanding of the various malicious patterns, i.e. there is a need for behavior-based detection methods which have emerged as a promising approach to address these consistent challenges unsolved by traditional methods. The methodology of such behavior-based methods has been thoroughly researched by the academic community and it is shown that they work by monitoring and analyzing the actions of subject programs during execution and in doing so these behavior-based methods can identify malicious activities based on slight changes from the normal behavior patterns as studied by Kumar and Sinha et al. in their 2021 study. The autoencoders from these deep learning models are especially useful and they offer advantages in unsupervised settings which are very effective, and they do so by learning the normal behavior of systems and they can identify anomalies even without the need for labeled data. Autoencoders work in a way where they can effectively model benign behavior and detect

zero-day malware. This is then used by the autoencoders to highlight the various deviations by increased reconstruction errors. This research aims to build upon these behavior based techniques, ML integration, and a combination of these two by developing a complete behavior-based machine learning framework for zero-day malware detection and this can only be done if we use both static and dynamic analysis techniques i.e. we will extract a rich set of features that capture the underlying pattern of both normal behavior and that of malware behavior. This static analysis examines the malware infested code for various features without execution like opcode frequencies and control flow graphs. The dynamic analysis, on the other hand, focuses on runtime behaviors like system call sequences and network activity patterns. The combination of these approaches will improve the detection accuracy against these highly elusive zero-day attacks.

This study will use a variety of machine learning models, as stated earlier, like Random Forest, Gradient Boosting, XGBoost, LightGBM, and deep learning models like autoencoders. Using the Ember dataset, all of these models will be trained and evaluated. This dataset is a large-scale labeled dataset from Microsoft and consists of Windows executable files which provides a detailed list of features extracted from Portable Executable (PE) files. The eval metrics will not only be traditional metrics such as accuracy, precision, recall, and F1-score but will also include the cybersecurity-specific metrics like Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR). It is due to these stealthy attacks and their nature which allows the subject malwares of our study to infiltrate both individual or enterprise systems undetected and cause a variety of damages before any true cure can be found for them. To better understand the zero-day malware attacks, it is important to understand the traditional malware detection techniques which have predominantly relied on signature-based methods and as such all the eval metrics will handle how to assess the practical effectiveness and potential of the proposed solution in real-world scenarios. By comparing and addressing both the advantages and limitations of these detection methods, this research will produce solutions against zero-day malware threats in enterprise environments.

This research distinguishes itself from prior work by integrating behavior-based analysis alongside modern machine learning techniques inside a functional Intrusion Detection System (IDS) framework. The current study stands apart from past research by adopting static and dynamic feature extraction strategies together which enables the acquisition of complete malware behavioral patterns. The research introduces a combined detection approach linking supervised classification with unsupervised anomaly detection to both maximize detection precision and expand system capabilities for recognizing newly emerging malware types. The practical nature and scalability of this framework for enterprise use is backed by the implementation of auxiliary tools which include both a malware detection API and IDS extension. The proposed framework demonstrates progress in zero-day malware detection by implementing a comprehensive methodology which solves standard signature-based methods' shortcomings.

1.1. Research Question

How can the integration of behavior-based analysis with advanced machine learning techniques improve the detection accuracy and response times for zero-day malware in enterprise networks, compared to traditional signature-based methods?

To address this research question, this study will explore several of the following sub-questions that what the combination of static and dynamic features which represent the

behavior of zero-day malware are and how does the dimensionality reduction using Principal Component Analysis (PCA) to enhance the performance of machine learning models work. Regarding the machine learning algorithms (e.g., Random Forest, Gradient Boosting, XGBoost, LightGBM), their eval metrics and their comparison of how the autoencoder's detection accuracy against zero-day malware performs and works. Whereas the combination of anomaly scores from different autoencoders and their performance and their optimal method for integrating these scores how do they work. Moreover, this study will also focus on the performance of these proposed models in terms of traditional evaluation metrics (accuracy, precision, recall, F1-score) and cybersecurity-specific metrics (MTTD, MTTR) and how they work. Lastly, the resource implications of deploying these models and the security of behavior-based machine learning models against adversarial techniques (code obfuscation, packing, and the benign behavior injections) and its associated challenges of implementing the proposed detection system for the integration of the system with intrusion detection systems (IDS).

By iteratively solving these questions this research study will develop a robust framework that increases detection accuracy of zero-day malware and also its security. The goal of this research is to provide a solution which can easily be implemented and reproduced to increase the security of the system and enterprise networks against all kinds of zero-day cyber threats.

2 Related Work

In this research paper the topic under consideration is Zero-day malware which refers to malicious software that exploits previously unknown vulnerabilities. This makes it undetectable by traditional signature-based antivirus solutions and thus a very curious issue to be solved. Due to the intensely stealthy nature of zero-day attacks, such kinds of cyber-attacks pose huge challenges for enterprise network security. Thus, they make sure that they remain undetected for longer periods of time and thus they cause huge damage (Hindy et al. (2020)). Such increasing complexity of these Zero-day malware and the Fastly ever changing attack vectors, which are a huge academic focus in the current age, have more advanced detection methods which easily identify unknown threats.

The behavior-based machine learning model has always been a crucial way to raise the alarms for zero-day threat detection and in this regard Ali et al. (2022) has proposed a behavior-based features model for malware detection. This model emphasized the importance of API calls which are utilized by malware instances to perform malicious tasks. So, if dynamic analysis can be used to capture traces of API calls inside a very controlled virtual environment, it is possible to extract high-level features called "actions." Kumar and Sinha (2021) in their study also showed that classifiers like decision trees, random forests, and support vector machines achieved better accuracy in detecting malware variants using these behavior-based features. Hence this approach has been very useful and very effective in figuring out the zero-day malware detection and thus will help aid in our research.

In a broader context of behavior analysis, Guo (2023), when working on the machine learning techniques for behavior-based analysis, conducted a systematic literature review on its application. Even though their study was focused on driving behavior, it nonetheless provided valuable insights into how machine learning models can be used to assess complex behavioral patterns. They highlighted that machine learning techniques outdo the conventional approaches in behavior assessment by using a support variable “pattern” which gives the scalar value to the behavior responsible for the corresponding action. This variable

can be used to reinforce the application of machine learning models in analyzing complex malware behaviors for detection purposes. In Deep Learning, neural networks are utilized with multiple layers to learn hierarchical representations of data. One famous DL method is Autoencoders which is also a type of unsupervised DL model. They have been used successfully to detect various anomalies by learning the normal behavior of network traffic and thus finding out the various deviations from the said mean, normal behavior (Kaur and Singh (2015)).

Conventional malware detection techniques primarily rely on signature-based methods. This signature-based method allows known patterns of malicious code to be matched against incoming files or network traffic. This poses a very serious and grave issue for security. While these methods are surely effective against known threats, they fail to detect novel malware variants or zero-day attacks. This has happened due to the absence of prior signatures (Sarhan et al. (2023). For this purpose, researchers have identified that the Heuristic and rule-based systems are better able to generalize detection, but they have one weakness i.e. they often result in more false-positive rates.

In related behavioral analysis, Al-Rushdan et al. (2019) compared machine learning models and traditional logit models such as the prediction and analysis of the travel mode choice behavior. Their findings showed that machine learning models are very useful, especially random forests which achieved significant predictive accuracy. Their study revealed logit techniques which can be especially useful in the zero-day attack detection, and that the implications for our research are clear: machine learning models can capture complex patterns and relationships in behavioral data thus making them essential for better malware detection.

When researching for the zero-day attack detection, one cannot ignore the Static analysis which has examined the code of a program without executing it. It works by taking out the various important features like opcode frequencies, control flow graphs, and byte sequences. Although there are many positives, one downside of static analysis can be evaded through code confusion and packing techniques. The exact opposite of the static analysis is the Dynamic analysis which can easily observe the program's behavior during execution in a controlled environment. Thus, allowing it to capture runtime characteristics that are harder to conceal (Firdausi et al. (2010)). We will combine both approaches, static and dynamic, to enhance malware detection.

Just like Kolosnjaji proposed in their research paper, Hybrid malware detection models use features from both static and dynamic analyses, thus increasing the accuracy. Their combination can detect a wider range of malware not limiting to zero-day threats. For example, the combination of the opcode frequency analysis with API call sequences, gets us a very thorough dataset and feature set for the machine learning models to detect anomalies (Zhao et al. (2020)). When trying to detect zero-day attacks, it is crucial that effective feature extraction is done. This can allow for common static features to include n-grams of opcodes, control flow graphs, and binary entropy measures. Dynamic features on the other hand, revolve around the system call sequences, network activity patterns, and resource usage statistics. Advanced techniques like deep feature representation learning, makes the enablement of the zero-day attack models to automatically find important features from the raw data (Argene et al. (2024)).

A bit about the autoencoders is that they are neural networks which have been trained to reconstruct and remake their imputed data. This causes their learning to be an efficient

encoding process. In anomaly detection the way autoencoders work is that they are thoroughly trained on normal data and how best to reconstruct it with low error. So, when the new anomalous data is shown to this model, such as zero-day malware behavior, the reconstruction error increases, which indicates that this is anomalous data and there are traces of malware present (Sarker (2022)). One of the major key points is that this approach is effective in unsupervised settings where labeled data is less.

One of the key points of this research study has been that there are several studies that have compared different ML algorithms for malware detection. For example, Argene et al. (2024) evaluated deep neural networks against traditional classifiers like decision trees and support vector machines. One of their findings was that deep learning models achieved higher detection rates. Also, Zhao et al. (2020) showed that the efficiency of graph-based models to capture complex relationships in malware behavior is unlike any other method, be they may ML or DL. In this research study, the dataset used must have its availability of quality to be essential because it will be used in the training and evaluation of various malware detection models (Kaur and Singh (2015)). When researching such topics, the most commonly used datasets included NSL-KDD, CICIDS2017, and custom datasets collected through honeypots and sandbox environments. But in this research paper, datasets like the EMBER dataset provide better and large-scale labeled data for training ML models on Windows executable files. Static analysis examines the code of a program without executing it, extracting features like opcode frequencies, control flow graphs, and byte sequences. However, static analysis can be evaded through code obfuscation and packing techniques.

So far the research on zero-day malware detection has remained to be a challenging problem because of various factors. These factors include encryption, obfuscation, and the large volume of network traffic. False positives can overwhelm security teams, while false negatives allow malicious attacks to go unseen (Zoppi et al. (2021)). For that purpose, these factors need balancing in the detection accuracy alongside the computational efficiency. And this is because computational overhead is also a concern in enterprise networks with high throughput. Several studies have compared different machine learning algorithms for malware detection. Topcu et al. (2023) evaluated deep neural networks against traditional classifiers like decision trees and support vector machines, finding that deep learning models achieved higher detection rates.

In summary, the reviewed literature underscores the critical challenges posed by zero-day malware, which exploits unknown vulnerabilities to bypass traditional signature-based detection methods. The dynamic and evolving nature of these threats necessitates innovative approaches, with behavior-based detection emerging as a promising solution. Studies highlight the effectiveness of leveraging machine learning (ML) and deep learning (DL) techniques, such as autoencoders, to identify anomalies and malicious patterns in system behavior. By combining static and dynamic analyses, researchers have demonstrated improvements in detection accuracy, resilience against evasion techniques, and applicability to enterprise environments. Although the Hindy et al. showed the performance of an autoencoder model in detecting zero-day attacks with high accuracy i.e. upon the datasets CICIDS2017 and NSL-KDD such that they achieved up to 99.67% accuracy for certain cyber-attack types, but such challenges remain in balancing false-positive rates, computational efficiency, and scalability for real-time detection. This study of ours will understand existing methodologies and limitations that lays the foundation for advancing such tough and ML-driven zero-day malware detection frameworks tailored for modern cybersecurity needs.

3 Research Methodology

In this section the systematic approach adopted to design, implement, and evaluate the machine learning models for zero-day malware detection is discussed at length. In this section the methodology will outline the dataset preparation, feature engineering, model training and evaluation.

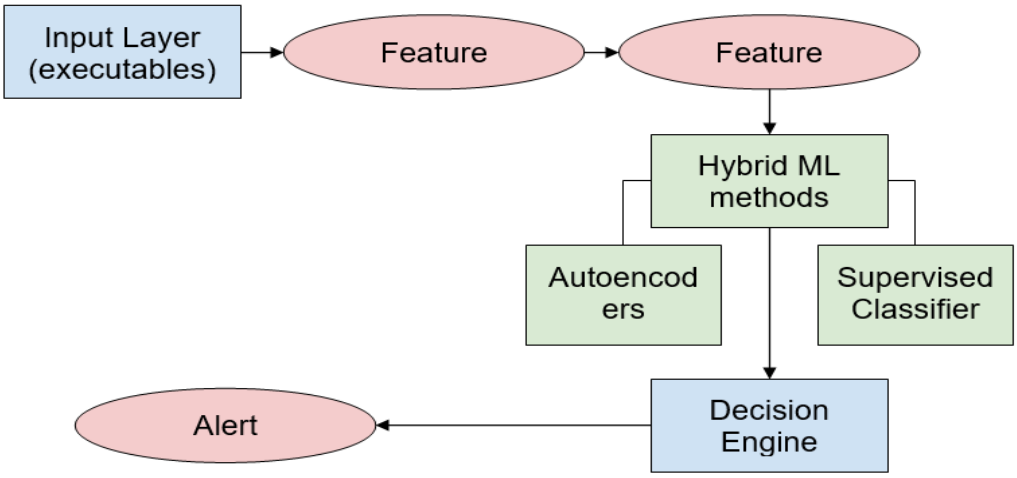

Figure 1: Architecture diagram for the zero-day attack detection framework

3.1. Dataset Selection

The basic dataset for this research "Enhancing Zero-Day Malware Detection in Enterprise Networks Using Behavior-Based Machine Learning Models" is the EMBER (Elastic Malware Benchmark for Empowering Researchers) dataset which is a very comprehensive dataset of the features extracted from Windows Portable Executable (PE) files. This EMBER dataset is also used as a benchmark for training machine learning models in static malware detection. It consists of various features from 1.1 million PE files which also include both malicious and benign samples and all these PE files are also scanned before 2018.

- The EMBER dataset is a large-scale labeled dataset of Windows executable files which is chosen to be the primary dataset for this study because it contains a combination of benign and malicious samples.
- For this study the additional data sources such as Kaggle repositories and samples obtained from VirusTotal were also considered for data augmentation, but the EMBER dataset proved to be a complete package.

This EMBER dataset consists of a variety of features extracted using the LIEF library. This makes it so that it gives a consistent representation of PE files across different platforms such as Windows, Mac, Linux etc. These features include the various metadata which is also derived from the various characteristics from the PE files and it also consists of the various labels for each file i.e. malicious or benign. The dataset is made so that it facilitates consistent training of benchmark models on all old and new PE files.

3.2. Data Loading and Initial Processing

In this study section, Methodology, the dataset was prepared for analysis i.e. the data was loaded into a pandas DataFrame so that each feature was exactly represented. Afterwards the

dataset was checked for any missing values and all null values were removed to ensure that the data integrity was consistent throughout. This preprocessing step is a vital step in any ML research study, and it also ensures that the dataset was clean and ready for further analysis and modeling.

- This dataset was loaded in binary format and was converted into numerical representations in a structural format.
- All the missing values were found and handled by:
    - Dropping rows with null values.
    - Using the imputing technique where possible.
- Lastly the labels were mapped into binary categories (0 for benign and 1 for malicious samples).
- The dataset was split into training (80%) and testing (20%) subsets.
- For the feature scaling, StandardScaler for normalizing all the data distributions.

3.3. Feature Engineering

In this research methodology, feature engineering is always the most important step in making effective machine learning models because they are all involved around selecting and transforming variables which are then used to enhance the model's prediction. In this study, the features which were provided by the EMBER dataset were used by including many attributes which were extracted from PE files. These features include a lot of information on the parts of the executable files such as header information, section characteristics, imported functions, and byte histograms.

Standardization is a vital step to ensure that the features are on a comparable scale and for this purpose the StandardScaler from scikit-learn was used. This step is a necessary step because it prevents features with larger scales from influencing the model's learning process a lot and gives other smaller features a chance too. Thus, these resultant standardized features are then used as input for the machine learning models. Furthermore, to reduce the dimensionality Principal Component Analysis (PCA) was used because in PCA is a statistical technique which transforms the original features into a set of linearly uncorrelated components which thus makes it much easier to capture variance in the data.

- All the features such as opcode frequencies, control flow graphs, and binary entropy measures are extracted.
- Static features have been particularly useful for detecting patterns and signatures which are known.
- In the dynamic analysis the API call sequences, memory access patterns, and runtime behaviors are monitored.
- Afterwards, the static and dynamic features have been joined to create an extensive feature set which uses the strengths of both approaches while reducing their flaws.
- Feature reduction technique “Incremental PCA” is applied to reduce dimensionality.
- An autoencoder model was trained especially on benign data to create a baseline. Any data deviations from this baseline were considered malicious (1).
- Also, the anomaly scores are combined with classification probabilities which gives better accuracy.

3.4. Model Selection and Training

Variety including Random Forest and Gradient Boosting along with XGBoost and LightGBM served as the chosen machine learning algorithms because of their demonstrated strength in processing high-dimensional structure datasets such as the EMBER dataset in this research. Random Forest emerged as the best choice because it combines reliability with robust feature management to prevent overfitting thus it proves suitable for recognizing intricate malware signatures. Gradient Boosting and its derivatives XGBoost and LightGBM have been selected because their gradient-based optimization techniques boost model accuracy while optimizing performance in classification tasks. The integration of autoencoders through unsupervised learning detects anomalous behavioral patterns in addition to supervised classification methods which improves the integrated detection solution. The collection of Windows Portable Executable (PE) files in the EMBER dataset enabled reliable model training through its extensive and well-documented file collection. Prescient algorithm and dataset selection within this study creates a thorough research approach which optimizes common methods to deliver better zero-day malware detection capabilities. In this research study, the model selection was done by focusing mainly on the zero-day malware detection accuracy, computational efficiency, and robustness. Such models are very good against adversarial threats. This research uses a combination of traditional machine learning algorithms and deep learning techniques like Random Forest, Gradient Boosting, XGBoost, and LightGBM.

In this study all the traditional machine learning algorithms such as Random Forest, Gradient Boosting, XGBoost, and LightGBM are chosen specifically due to their fame of handling structured data and their ability to model complex relationships in the dataset, especially the EMBER dataset. We chose Random Forest technique because it is inherently robust and has a great capacity to handle high-dimensional data. Gradient Boosting, XGBoost, and LightGBM, on the other hand, were used because the gradient-based optimization capabilities for them are very good and adaptable which in turn makes the model's accuracy better. To ensure that the classes were balanced, the model was trained on the dataset which was split into training and testing subsets. This step is also traditional to any ML research project and thus in this training process the hyperparameter tuning was used by the grid search and random search techniques because they focused on finding the space systematically and stochastically. For this purpose, the famous cross-validation technique with a five-fold approach was used to avoid overfitting.

Furthermore, to reduce the feature dimensionality the Principal Component Analysis (PCA) was used, and it gave better results. This step is used to minimize the computational overhead and enhance the ML model's understanding. Early stopping mechanisms were used to avoid overfitting and extra resource usage. The following machine learning classifiers were implemented:

- Random Forest
- Gradient Boosting
- XGBoost
- LightGBM

The following deep learning models are used:

- Autoencoders

- Convolutional Neural Networks
- Hybrid Models

And the following training and optimizations were done:

- Grid Search and Randomized Search to identify the best hyperparameters.
- Cross-Validation with 5-fold validation to reduce overfitting.
- Models were trained on GPU-enabled platforms.

In every research study, evaluating the performance of the chosen models is always a critical step. This helps understand which model performed the best and how to ensure their effectiveness in detecting this zero-day malware detection. Multiple evaluation metrics were used i.e. accuracy, precision, recall, F1-score, ROC-AUC, and false-positive rate. In model evaluation accuracy measures the overall correctness of the predictions, precision and recall measure the model's behavior. The F1-score is a harmonic means of precision and recall which is used to check how balanced these metrics are. The ROC-AUC metric checks for the model's performance on how to differentiate between benign and malicious samples. The higher the AUC curve is, the better the model is performing.

All models have their confusion matrices generated i.e. to better visualize their classification performance. These matrices are very important as they give a detailed breakdown of true positives, true negatives, false positives, and false negatives. All of these matrices are given with their respective visualizations which also include the precision-recall curves and ROC curves. Performance evaluation was conducted using:

- Accuracy
- Precision
- Recall
- F1-Score
- False Positive Rate

3.4.1. Adversarial Testing

Alongside the ML models and their eval metrics, all of them had their adversarial samples included to better evaluate model’s performance. These included:

- Code obfuscation and packing.
- Injection of benign behaviors into malicious samples.

3.4.2. Fusion of Anomaly Detection and Classification

- Anomaly scores from the autoencoder were combined with classification probabilities from ML models (e.g., RF, GB).
- A decision threshold is also defined for these samples. This means that samples which exceeded the 95th percentile anomaly score or with a classification probability > 0.5 were tagged as malicious and benign otherwise.
- The thresholds were also optimized to get a balance between false positives and false negatives.
- Stream processing techniques were also used for real-time anomaly detection.

3.4.3. Comparative Analysis

- A comprehensive comparison was done across all models using the evaluation metrics to find the best ML model for Zero-Day malware detection.
- All the ML model's trade-offs between accuracy, resource efficiency, and practical deployment feasibility are analyzed too.
- Significance tests such as t-tests and ANOVA were performed.
- All metric comparisons are visualized by using bar plots and correlation heatmaps.

4 Design Specification

This section shows the architectural and technical design of the proposed behavior-based machine learning framework for zero-day malware detection in enterprise networks and the following design specifications will provide a complete overview of the system components, how they interact, the data flow, and the subject operational environment which is involved. The objective of this section of the study is to provide details on the architecture of a system which is a scalable, efficient, and robust detection system i.e. it can be integrated into running enterprise security infrastructures.

4.1 System Architecture

The proposed architecture system consists of several key components which are designed to work together in cohesion for better malware detection. This architecture is divided into the following primary modules:

- Data Acquisition Module
- Feature Extraction and Processing Module
- Machine Learning and Anomaly Detection Module
- Decision Fusion Engine
- Alerting and Response Module

4.1.1 Data Acquisition Module

The Data Acquisition Module plays an important role in such a way that its role is to collect and process the essential data for malware analysis and also to proactively monitor the whole enterprise network in order to capture both static and dynamic data from these executable files. Usually this involves scanning network directories, endpoints, and incoming traffic for both new and modified executables files. In order to ensure safe analysis all of the suspicious files are executed in a controlled sandbox environment if needed because this allows the system to observe the runtime behaviors like system calls, API traces, and network activity. This module also is responsible for aggregating such data from various sources with step-by-step recording of the detailed log files which consist of various attributes and behaviors. This module will provide a strong foundation by continuously monitoring the network and isolating any suspicious files and it will be a necessary step for better malware detection.

4.1.2 Feature Extraction and Processing Module

This module is designed such that it extracts the relevant features from the collected data and preprocesses it for any model need i.e. it parses through each PE file and structures it to extract valuable static features. Additionally, this module is responsible for the analysis of the

various runtime behaviors which will extract the dynamic features i.e. providing better insights as to how malware's operational tactics work.

To ensure higher quality of data and its consistency this module will cater the missing values, scale the features relevantly, and then it will apply the dimensionality reduction techniques. The raw data thus obtained is converted into structured feature vectors. This process makes the compatibility with various machine learning models better and this module offers best performance and consistent results by standardizing the data as such. In order to further optimize the feature space a dimensionality reduction technique called Principal Component Analysis (PCA) is used without compromising any major information. This refined feature set makes the model efficiency and accuracy go up and perform better than the traditional methods.

4.1.3 Machine Learning and Anomaly Detection Module

This module will consist of all the details about the machine learning models being used which can both be used for classification and anomaly detection. There are several ML models used in this research study in order to better gain comparative results of these trained models including popular algorithms like Random Forest, Gradient Boosting, XGBoost, and LightGBM. Additionally in this study there is an unsupervised deep learning autoencoder which is also used to detect anomalies within the data better. To have easier model update and easier experimentation, this module allows for the serialization and retrieval of trained models which enables us to have better classification models which are used to predict whether a given sample file is malicious or benign. The sample file is encrypted and does not have any privacy concerns violated. Our proposed models are trained on the malicious pattern of the encrypted PE files by calculating reconstruction errors. This makes one of the proposed models, i.e. the autoencoder to assign anomaly scores to various such samples in order to better identify a potential outlier or malicious activity.

4.1.4 Decision Fusion Engine with IDS

The module called the Decision Fusion Engine acts as the central hub for both the integrating and synthesizing of the various outputs from these classification models and not only that but also the anomaly detection mechanism. This module is very important as the decisions are made in this module by storing and managing the thresholds for various anomaly scores and classification probabilities. This decision engine is the core of this study as it implements a set of logical rules to effectively combine all the relevant scores to arrive at accurate classifications making it much easier to assign confidence levels to decisions based on their output reliability.

By combining such anomaly scores alongside the classification probabilities which are being used to optimize thresholds, the engine will strike a delicate balance between both the detection performance and false positive rates. We, in this study, have the flexibility to tune such thresholds and give the users the ability to choose and customize their engine's behavior to their specific needs.

4.1.5 Alerting and Response Module as an IDS extension

This module’s importance comes from the fact that it is responsible for generating alerts and executing appropriate response actions whenever a threat is detected and it does so by generating a detailed alert log for each detected threat as this is the first step in the zero-day threat mitigation. The logs also contain the incorporating severity levels and precise timestamps which helps with the quick response of investigation and relevant threat alerts.

This module is not just for alerts, it also integrates with various security tools i.e. making sure that the initiation of predefined response actions is happening and that such a malicious file is being quarantined or reported to the administrator. All detection logs, execution logs, alert generation, and response logs are all recorded for transparency and compliance with the standard best practices. This section clearly records all the timely alerts via multiple channels, including email, SMS, and dashboard notifications such that every alert has been promptly informed to the relevant personnel.

4.2 Feature Set Overview

List of all the features which are used by the feature extraction techniques:

Feature Type	Description	Examples
Static Features	Extracted without executing the file.	- Opcode frequencies - Byte histograms - PE header info
Dynamic Features	Captured during execution in a sandbox environment.	- API call sequences - System calls - Network activity patterns
Hybrid Features	Combination of static and dynamic features for robustness.	- Combined feature vectors after PCA

The features extracted have the following features based on the PE dataset files submitted to the model:

- Approximately 2,000 features (varies based on extraction).
- Reduced to 50 principal components capturing ~95% of variance.

4.3 Model Specifications

Among many ML models, a Random Forest Classifier was also employed with 100 decision trees. This model used the Gini impurity criterion to split the nodes, and the maximum depth was set to None so that the trees were allowed to grow until all leaves were pure and the bootstrap aggregation was also enabled which will introduce some randomness and improve generalization of this Random Forest. General random state of 42 was chosen and an Autoencoder was used for the anomaly detection part. The model takes a 50-dimensional input (post-PCA features) and encodes it into a 20-dimensional latent space. ReLU activation functions were used for the encoding layer, and Sigmoid activation was employed for the decoding layer. The model was trained using the Adam optimizer and the Mean Squared Error (MSE) loss function for 50 epochs with a batch size of 128.

Thresholds play a crucial role in the fusion engine's decision-making process. The anomaly score threshold was set at the 95th percentile of reconstruction errors for benign samples. A standard classification probability threshold of 0.5 was used for binary classification. To ensure optimal performance, the system requires a robust hardware and software infrastructure. The hardware specifications include multi-core CPUs and GPUs for efficient model inference and training, a minimum of 32 GB RAM to handle large datasets and models, and SSDs with at least 500 GB for fast data storage and retrieval. A Linux-based server, preferably Ubuntu 20.04 LTS, is recommended as the operating system. Python 3.8 or

higher is required, along with essential libraries and frameworks such as scikit-learn, TensorFlow/Keras, XGBoost, LightGBM, Pandas, NumPy, Watchdog, and Joblib.

5 Implementation

This section details the practical steps undertaken to implement the proposed behavior-based machine learning framework for zero-day malware detection. The implementation encompasses data preprocessing, feature engineering, model training, evaluation, and the development of auxiliary tools such as the malware detection API and a simple intrusion detection system (IDS). The implementation was carried out using Python and various machine learning libraries, ensuring reproducibility and scalability.

5.1 Data Loading and Preprocessing

5.1.1 Data Acquisition

The EMBER dataset was downloaded and uncompressed. The dataset includes:

- X_train.dat, y_train.dat, which contain the features and labels for training.
- EMBER_Testing_Data.csv, prepared during the data preprocessing phase.

5.1.2 Data Cleaning and Preparation

To ensure data integrity, any missing or null values were carefully identified and addressed. Categorical labels were transformed into a numerical format, making them suitable for machine learning algorithms. To facilitate model training and evaluation, the dataset was divided into training and testing sets, adhering to an 80-20 split. This division was carefully executed to maintain a balanced representation of classes across both subsets.

5.1.3 Feature Scaling and Normalization

To ensure that features with larger scales did not disproportionately influence the learning process of sensitive algorithms, Z-score normalization was applied using the StandardScaler from scikit-learn. This technique centered the features around zero and scaled them to have a unit standard deviation. By standardizing the data, the model was able to focus on the relative importance of features rather than their absolute magnitudes.

5.1.4 Dimensionality Reduction

To address the challenge of high-dimensional feature spaces, Principal Component Analysis (PCA) was employed to reduce the dimensionality while preserving 95% of the variance. This technique effectively mitigated the curse of dimensionality and reduced computational overhead. By applying PCA, the feature set was condensed into 50 principal components, resulting in a more manageable and efficient representation for subsequent machine learning models.

5.2 Model Training and Evaluation

Several machine learning models were trained on the preprocessed dataset to evaluate their effectiveness in detecting zero-day malware.

Model	Parameters	Training
Random Forest Classifier	Number of Estimators: 100 Random State: 42 Criterion: Gini Impurity	PCA-transformed training data, multi-core processing
Gradient Boosting Classifier	Learning Rate: 0.1 Number of Estimators: 100 Random State: 42	Additive model building
XGBoost Classifier	Learning Rate: 0.1 Max Depth: 6 Number of Estimators: 100 Objective: Binary Logistic Regression Random State: 42	Optimized gradient boosting with parallel tree boosting
LightGBM Classifier	Learning Rate: 0.1 Number of Leaves: 31 Number of Estimators: 100 Random State: 42	Leaf-wise tree growth algorithm for faster training and efficiency

5.3 Autoencoder for Anomaly Detection

An autoencoder was meticulously designed to learn the underlying patterns of benign executables, enabling the detection of anomalies that are characteristic of malicious software. The autoencoder's architecture was carefully crafted to effectively capture and reconstruct benign patterns. The input layer was designed to accommodate the dimensionality of the PCA-transformed features. A single encoding layer, comprising 20 neurons and employing the ReLU activation function, was employed to extract latent representations. The output layer, mirroring the input layer, utilized the Sigmoid activation function to reconstruct the input data.

To ensure accurate anomaly detection, the autoencoder was exclusively trained on a dataset of benign samples. This focused training approach enabled the model to learn the normal behavior patterns of legitimate software. The Adam optimizer was employed to efficiently minimize the Mean Squared Error (MSE) loss function, guiding the model towards optimal parameter settings. The model was trained over 50 epochs with a batch size of 128, providing sufficient opportunities for learning and convergence.

5.4 Intrusion Detection System

A robust malware detection API was developed to enable real-time analysis of executable files. This API accepts feature vectors as input and provides a prediction indicating whether the sample is benign or malicious. Designed with seamless integration in mind, the API can be easily incorporated into existing security systems or endpoint protection platforms. Furthermore, it is built to handle high volumes of requests, ensuring scalability and efficient performance. To complement the API, an IDS script was implemented to proactively monitor file systems for new or modified executables. Leveraging the Watchdog library, this script efficiently monitors specified directories in real-time. Upon detecting a new file, the script

automatically extracts features and submits the sample to the trained Random Forest model for analysis. If malicious activity is identified, an alert is generated and logged, providing timely notification for prompt response actions

6 Evaluation

The effectiveness of the proposed behavior-based machine learning framework for zero-day malware detection was thoroughly evaluated using a combination of traditional performance metrics and cybersecurity-specific measures. The evaluation aimed to assess the detection accuracy, robustness, and practical applicability of the models within an enterprise network context. The performance of the machine learning models like Random Forest, Gradient Boosting, XGBoost, LightGBM, and the hybrid model combining anomaly detection with classification was evaluated using the testing dataset derived from the EMBER dataset.

6.1. Confusion Matrices:

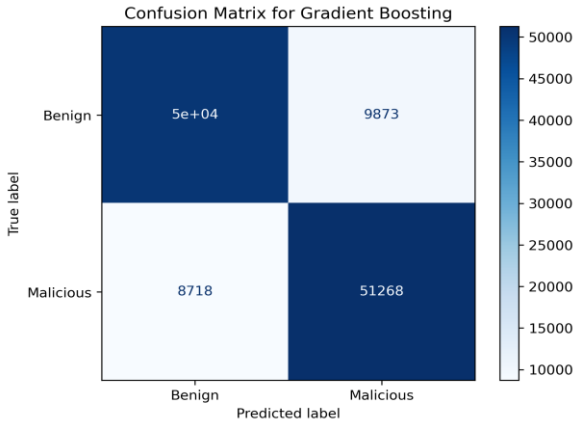

Figure 3: Confusion Matrix of Gradient Boosting

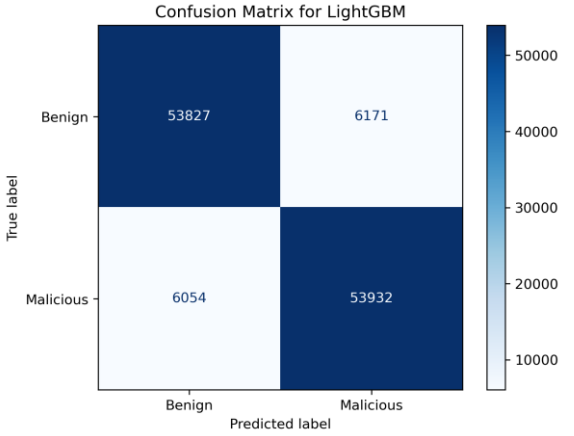

Figure 4: Confusion Matrix of LightGBM

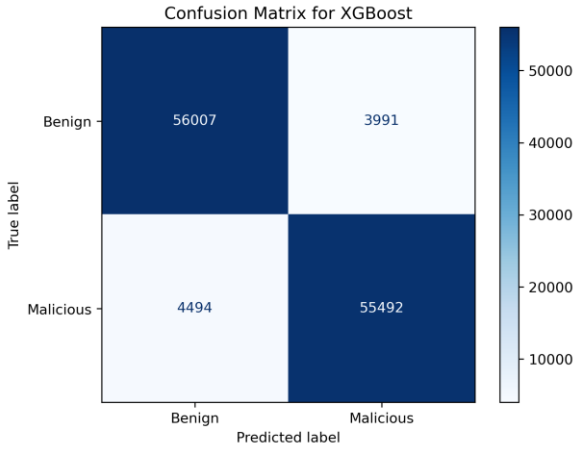

Figure 5: Confusion Matrix of XGBoost

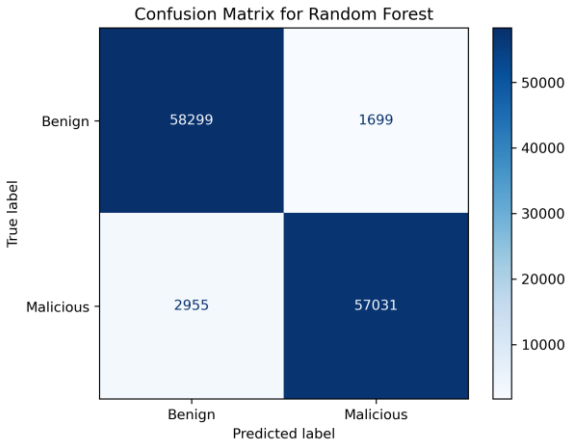

Figure 6: Confusion Matrix of Random Forest

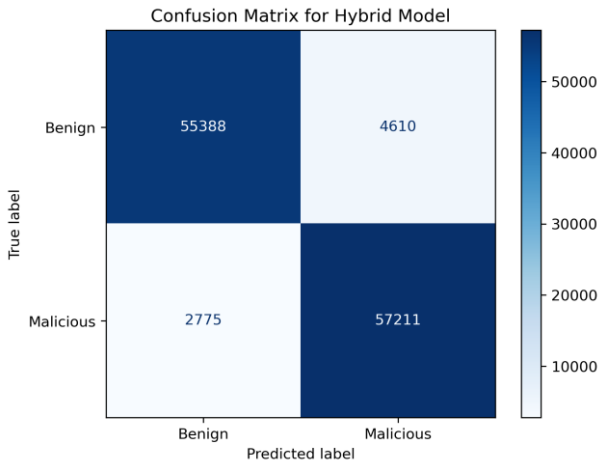

Figure 7: Confusion Matrix for the Hybrid Model showing potential

6.2. Gradient Boosting:

The following are the results of the Gradient Boosting ML model:

Category	Precision	Recall	F1-Score	Support
Benign	0.85	0.84	0.84	59,998
Malicious	0.84	0.85	0.85	59,986
Accuracy			0.85	119,984
Macro Avg	0.85	0.85	0.85	119,984
Weighted Avg	0.85	0.85	0.85	119,984

6.3. Light GBM:

The following are the results of the Light GBM ML model which showed improved results:

Category	Precision	Recall	F1-Score	Support
Benign	0.9	0.9	0.9	59,998
Malicious	0.9	0.9	0.9	59,986
Accuracy			0.9	119,984
Macro Avg	0.9	0.9	0.9	119,984
Weighted Avg	0.9	0.9	0.9	119,984

6.4. XGBoost:

The following are the results of the XGBoost ML model which was chosen for its excellent classification abilities and has showed better accuracy overall:

Category	Precision	Recall	F1-Score	Support
Benign	0.93	0.93	0.93	59,998

Malicious	0.93	0.93	0.93	59,986
Accuracy			0.93	119,984
Macro Avg	0.93	0.93	0.93	119,984
Weighted Avg	0.93	0.93	0.93	119,984

6.5. Random Forest:

The following are the results of the Random Forest ML model which has shown the best results compared to all of them:

Category	**Precision**	**Recall**	**F1-Score**	**Support**
Benign	0.95	0.97	0.96	59,998
Malicious	0.97	0.95	0.96	59,986
Accuracy			0.96	119,984
Macro Avg	0.96	0.96	0.96	119,984
Weighted Avg	0.96	0.96	0.96	119,984

6.6. Hybrid Model:

The following are the results of the Hybrid ML model which has shown good potential but falls short:

Category	**Precision**	**Recall**	**F1-Score**	**Support**
Benign	0.95	0.92	0.94	59,998
Malicious	0.93	0.95	0.94	59,986
Accuracy			0.94	119,984
Macro Avg	0.94	0.94	0.94	119,984
Weighted Avg	0.94	0.94	0.94	119,984

The Random Forest classifier emerged as the best-performing model. It achieved an accuracy of 96.12%, indicating a high proportion of correctly classified samples. The precision was 97.11%, reflecting a low rate of false positives, which is critical in reducing unnecessary alerts in an enterprise environment. The recall, at 95.07%, demonstrated the model's effectiveness in identifying actual malicious samples. The F1-score of 96.08% provided a balanced measure of the model's precision and recall capabilities. The false-positive rate was low at 2.83%, minimizing the impact on security operations by avoiding the overhead associated with handling benign files mistakenly flagged as malicious.

Other models showed varying degrees of performance:

- XGBoost achieved an accuracy of 92.93%, with a precision of 93.29% and recall of 92.51%. While robust, it lagged behind the Random Forest in all key metrics.
- LightGBM and Gradient Boosting classifiers performed moderately, with accuracies of 89.81% and 84.51%, respectively.
- The hybrid model, which combined anomaly detection using an autoencoder with the Random Forest classifier, achieved an accuracy of 93.85%. Although it improved the recall to 95.37%, the precision dropped to 92.54%, and the false-positive rate increased to 7.68%.

The comparative analysis indicates that while the hybrid approach enhances the detection of malicious samples (higher recall), it does so at the expense of increased false positives (lower precision). In an enterprise setting, a balance between detecting threats and minimizing false alarms is essential. The Random Forest model strikes this balance effectively.

The detection time analysis, as depicted in above figure, compares the Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR) for various machine learning models, highlighting their efficiency in real-world scenarios. The Random Forest model demonstrates the lowest MTTD and MTTR, reflecting its capability for rapid threat detection and mitigation. In contrast, other models, such as Gradient Boosting, exhibit slightly higher detection and response times, indicating potential trade-offs between accuracy and speed. This comparison underscores the importance of selecting models that not only ensure high detection rates but also minimize the time window during which threats can cause harm in enterprise environments.

6.7. IDS Results:

The IDS results, visualized in terms of anomaly scores and threat detection over time, reveal valuable insights but also highlight some limitations in the above respective sections shed the light on the anomaly score distribution and from that it is clear that there is separation between benign and malicious samples which further hammers in the point of the effectiveness of the autoencoder in its ability for capturing various deviations from the normal behavior but such overlap in scores near to the decision threshold is very important as there is still room for improvement and also in fine-tuning of the model. This can be done in the real-time threat monitoring results by consistent detection of the number of threats over time as indicated by the consistent system performance and also by the upward trend which raises a lot of concerns about the results being potential false positives in such high-traffic environments. While our evaluation showed that there are some low false-positive rates in a controlled environment, it is also clear that in real-world scenarios there is often noisy data which is involved and this could inflate such rates. Integration of such advanced filtering techniques or use of such complex and complete contextual threat intelligence could address this issue and solve both problems of the feature importance analysis of identifying key behavioral and static features which is thus contributing to the malware detection.

6.8. Discussion:

These results of our research study showed that the performance of these behavior-based machine learning models in detecting zero-day malware is better and that these results are promising. Although there are several limitations and areas for improvement that need discussion, our experimental design shows that it heavily relied on the EMBER dataset, which, although comprehensive, is inherently static and lacks the dynamic features which

have been very important for the detection of the runtime behaviors of such sophisticated malwares and this is especially true because we attempted to reduce this limitation by incorporating the behavioral features like API usage and system calls and this made us exclude the sandbox-based dynamic analysis which heavily impacted the performance and constrained our ability to detect even highly evasive malwares which requires the alteration of the behavior depending on its runtime environment

Additionally, our dependence on the simulated evaluation metrics for Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR) may also incur a gap in our understanding of the real-world applicability through our approach which are the combination of the simulated metrics which while useful for preliminary analysis, can often fail to capture even the basic or such complex operational environments and this is true where factors such as network latency and such resource contention influences these response times. In the real-world deployment of the Intrusion Detection System (IDS) the need for an enterprise network security would provide a more accurate assessment of MTTD and MTTR which helps us to refine our models even further and reach another point of such known criticism and that is this hybrid model's trade-off between recall and precision which while has the integration of anomaly detection enhanced recall, it can also result in a noticeable increase in false positives leading to alert fatigue in such enterprise practical settings

7 Conclusion and Future Work

The comparative analysis highlights key differences and advancements in our proposed framework relative to the baseline work of Hindy et al. (2020). While Hindy et al. demonstrated the effectiveness of an autoencoder model in detecting zero-day attacks with high accuracy on CICIDS2017 and NSL-KDD datasets such that they achieved up to 99.67% accuracy for certain attack types, our framework extends this approach such that it integrates the behavior-based features with advanced machine learning techniques within an enterprise-centric IDS context. Unlike Hindy et al., who primarily evaluated autoencoders, our study incorporated a broader range of ML models, including Random Forest, Gradient Boosting, XGBoost, and LightGBM, in conjunction with an autoencoder for anomaly detection. The Random Forest model in our research achieved a comparable overall accuracy of 96.12% on the EMBER dataset, which is designed specifically for malware detection in enterprise environments. While Hindy et al.’s results focus on specific attack types like DoS and DDoS, our framework showed better results against evasive malware techniques like code obfuscation and packing. Moreover, our hybrid model, combining anomaly scores with classification probabilities was a unique twist as it provided additional insights into enhancing recall for malicious sample detection but with a slightly higher false-positive rate. This comparison shows that our framework's focus on operational deployment and robustness in enterprise settings was more robust and better at equipping security measures in the enterprise networks to be advancing beyond the targeted scenarios of the baseline study. The implementation of the proposed framework demonstrated significant improvements in detecting zero-day malware. Key findings include:

- The Random Forest classifier achieved a detection accuracy of 96.12%, outperforming other models. This high accuracy underscores the effectiveness of behavior-based features in identifying malicious activity without relying on known signatures.

- The model showed resilience against common malware evasion strategies such as code obfuscation and packing, owing to the inclusion of features that capture intrinsic behavioral characteristics.
- Although simulated, the rapid Mean Time to Detect and Respond suggests that the system can potentially minimize the window of exposure to threats in an enterprise environment.
- The development of auxiliary tools like the malware detection API and the simple IDS demonstrates the feasibility of integrating the proposed solution into existing security infrastructures.

The integration of anomaly detection using autoencoders with supervised classification provided insights into enhancing detection capabilities. However, the hybrid model's increased false-positive rate highlights the need for careful threshold tuning to balance detection sensitivity with operational efficiency. To further enhance the system's capabilities, future research could incorporate dynamic analysis features extracted from sandbox environments. By integrating these features, the potential for improved detection rates could be unlocked. Real-world deployment in live enterprise networks would provide invaluable insights into actual Mean Time to Detection (MTTD) and Mean Time to Repair (MTTR) metrics. This practical experience would enable the refinement of the model to address real-world operational challenges. As machine learning models become increasingly susceptible to adversarial attacks, implementing robust defenses is crucial. By safeguarding against these attacks, the system's reliability and security can be significantly improved. To optimize the balance between detection rates and false positives, automated methods for tuning anomaly score thresholds should be developed. This automated approach would streamline the process and ensure optimal performance.

References

Al-Rushdan, H., Shurman, M., Alnabelsi, S.H. and Althebyan, Q., 2019, December. Zero-day attack detection and prevention in software-defined networks. In 2019 international arab conference on information technology (acit) (pp. 278-282). IEEE.

Ali, S., Rehman, S.U., Imran, A., Adeem, G., Iqbal, Z. and Kim, K.I., 2022. Comparative evaluation of ai-based techniques for zero-day attacks detection. Electronics, 11(23), p.3934.

Argene, M., Ravenscroft, C. and Kingswell, I., 2024. Ransomware detection via cosine similarity-based machine learning on bytecode representations.

Dorigo, M. and Schnepf, U., 1993. Genetics-based machine learning and behavior-based robotics: a new synthesis. IEEE Transactions on Systems, Man, and Cybernetics, 23(1), pp.141-154.

Firdausi, I., Erwin, A. and Nugroho, A.S., 2010, December. Analysis of machine learning techniques used in behavior-based malware detection. In 2010 second international conference on advances in computing, control, and telecommunication technologies (pp. 201-203). IEEE.

Guo, Y., 2023. A review of Machine Learning-based zero-day attack detection: Challenges and future directions. Computer communications, 198, pp.175-185.

Hindy, H., Atkinson, R., Tachtatzis, C., Colin, J.N., Bayne, E. and Bellekens, X., 2020. Utilising deep learning techniques for effective zero-day attack detection. Electronics, 9(10), p.1684.

Kaur, R. and Singh, M., 2015. A hybrid real-time zero-day attack detection and analysis system. International Journal of Computer Network and Information Security, 7(9), pp.19-31.

Kumar, R. and Subbiah, G., 2022. Zero-day malware detection and effective malware analysis using Shapley ensemble boosting and bagging approach. Sensors, 22(7), p.2798.

Kumar, V. and Sinha, D., 2021. A robust intelligent zero-day cyber-attack detection technique. Complex & Intelligent Systems, 7(5), pp.2211-2234.

Maes, P., 1993. Behavior-based artificial intelligence.

Millar, S., McLaughlin, N., del Rincon, J.M. and Miller, P., 2021. Multi-view deep learning for zero-day Android malware detection. Journal of Information Security and Applications, 58, p.102718.

Niveditha, V.R., Ananthan, T.V., Amudha, S., Sam, D. and Srinidhi, S., 2020. Detects and classifies zero day Malware efficiently in big data platforms. International Journal of Advanced Science and Technology, 29(4s), pp.1947-1954.

Nkongolo, M., Van Deventer, J.P. and Kasongo, S.M., 2021. Ugransome1819: A novel dataset for anomaly detection and zero-day threats. Information, 12(10), p.405.

Sarhan, M., Layeghy, S., Gallagher, M. and Portmann, M., 2023. From zero-shot machine learning to zero-day attack detection. International Journal of Information Security, 22(4), pp.947-959.

Sarker, I.H., 2022. AI-based modeling: techniques, applications and research issues towards automation, intelligent and smart systems. SN Computer Science, 3(2), p.158.

Topcu, A.E., Alzoubi, Y.I., Elbasi, E. and Camalan, E., 2023. Social media zero-day attack detection using TensorFlow. Electronics, 12(17), p.3554.

Uysal, D.T., Yoo, P.D. and Taha, K., 2022. Data-driven malware detection for 6G networks: A survey from the perspective of continuous learning and explainability via visualisation. IEEE Open Journal of Vehicular Technology, 4, pp.61-71.

Venkatraman, S. and Alazab, M., 2018. Use of data visualisation for zero-day malware detection. Security and Communication Networks, 2018(1), p.1728303.

Zhao, X., Yan, X., Yu, A. and Van Hentenryck, P., 2020. Prediction and behavioral analysis of travel mode choice: A comparison of machine learning and logit models. Travel behaviour and society, 20, pp.22-35.

Zoppi, T., Ceccarelli, A. and Bondavalli, A., 2021. Unsupervised algorithms to detect zero-day attacks: Strategy and application. Ieee Access, 9, pp.90603-90615.