Machine Learning in Malware Detection for PE Files and URL's

By

SATHEISH A/L THIRUCHELVAM

TP060754

APD3F2402CS

A report submitted in partial fulfillment of the requirements for the degree of

B.Sc. (Hons) Computer Science

at Asia Pacific University of Technology and Innovation.

Supervised by Dr. Aitizaz Ali

2nd Marker: Ts. Umapathy Eaganathan

2024

Contents

Abstract

This project is to create a machine learning-based malware detection system that can detect malicious content in Portable Executable (PE) files and URLs. XGBoost, Random Forest, Support Vector Machine (SVM), Decision Tree, and Logistic Regression are some of the sophisticated techniques that the system uses to improve detection efficiency and accuracy. Tools like Anaconda, Jupyter Notebooks, and Python modules (Scikit-learn, NumPy, Pandas) are used for data preprocessing and feature extraction. The project employs agile methodology to ensure continuous development and iterative progress. The system's effectiveness is evaluated through extensive testing and validation, providing cybersecurity specialists with a helpful tool to lower malware risks. This research improves machine learning in cybersecurity overall and raises the detection rates of malware.

1.0 Chapter 1

1.1 Introduction

Throughout the modern digital world, malware is a constant threat that poses a serious risk to the integrity and security of digital systems across the globe. Malicious actors are getting better at tweaking and improving their methods every day, thus traditional cybersecurity solutions are becoming less and less effective at thwarting their sneaky attacks. The constant advancement of cyber threats poses a significant threat to sensitive information confidentiality, integrity, and availability. It also erodes the trust and reliability of digital infrastructures, which are vital to modern civilization.

Malware is quite common and has many different forms. It might appear as viruses, worms, trojans, ransomware, or spyware, each of which has a unique way of working and a malevolent purpose. Furthermore, the Internet of Things (IoT) landscape's exponential growth in linked devices has increased the attack surface, giving hostile actors a vast field on which to plan coordinated, large-scale cyberattacks (Kaspersky, 2021). These assaults cover a wide range of smart devices embedded in commonplace objects, from industrial gear to residential appliances, in addition to standard computing devices.

Traditional signature-based detection techniques are becoming less and less successful against zero-day and polymorphic malware variants that regularly change their properties to avoid detection as the digital landscape develops ((M. S.) Muhammad Shoaib Akhtar, 2022). As a result, the requirement for intelligent and adaptable defense mechanisms that can quickly and accurately recognize and neutralize new threats emerges. In this regard, the use of machine learning techniques shows promise as a paradigm for improving malware detection and classification effectiveness by utilizing data-driven algorithms' ability to identify minute patterns and abnormalities suggestive of malicious activity.

However, there are a number of formidable obstacles to overcome in the development of strong machine learning-based malware detection systems. These include the lack of labeled training data, the existence of adversarial evasion techniques used by sophisticated malware,

and the requirement for constant model adaptation to changing threat landscapes. A multidisciplinary strategy including knowledge in software engineering, data science, cybersecurity, and machine learning is needed to address these issues (Dunlea, 2024). Moreover, in order to guarantee the detection system's accuracy, dependability, and generalizability across a variety of contexts and attack scenarios, cutting-edge approaches for feature engineering, model training, and validation must be integrated.

As a response to these demands, our work aims to push the boundaries of malware detection by creating a novel machine learning-based tool designed especially for examining Portable Executable (PE) files and URLs. My objective is to develop a detection system that is both highly adaptive and robust. This system will be able to identify and neutralize malicious threats in real-time, strengthening the resilience of digital ecosystems against the constantly changing cyber threat landscape. We will achieve this by utilizing advanced algorithms and computational techniques.

1.2 Problem Background

Malicious software is changing faster than traditional signature-based antivirus technologies can keep up with, posing a growing threat to computer networks and systems throughout the world. Novel malware variants are difficult for these traditional methods to identify, particularly zero-day attacks and polymorphic malware, which is always changing to avoid detection. This deficiency underscores the critical need for more adaptable and powerful malware detection systems by putting businesses at increased risk of monetary losses, data breaches, and operational disruptions. The incapacity of current approaches to dynamically adjust to the shifting threat landscape reduces cybersecurity and heightens susceptibility to cyberattacks. ((F.A.) Faitouri A. Aboaoja, 2022).

Furthermore, the limits of behavioral-based detection systems significantly complicate real-time cyber threat mitigation. While these techniques seek to find deviations from a system's typical behavior in order to spot malicious activity, they frequently produce a huge number of false positives and use a large amount of resources, which limits their usefulness in large-scale settings. Furthermore, the limits of behavioral-based detection systems

significantly complicate real-time cyber threat mitigation. While these techniques seek to find deviations from a system's typical behavior in order to spot malicious activity, they frequently produce a huge number of false positives and use a large amount of resources, which limits their usefulness in large-scale settings. The quick development of malware and the resource-intensive nature of current detection techniques highlight a serious weakness in cybersecurity. The pressing need for creative solutions to improve cybersecurity resilience means that businesses must look for improved detection approaches that may minimize false positives, cut operational overhead, and effectively identify and mitigate new threats in real time (Gibert, 2024).

1.3 Project Aim

To develop the Hitler System using state-of-the-art machine learning techniques in Python to enhance the efficiency and accuracy of malware detection on personal laptops and within organizations.

1.4 Objectives

1. To build a Variety of Malware Datasets for Machine Learning-Powered URL and PE File Identification.

2. To provide Robust Metrics for Machine Learning Model Evaluation

3. To Iterative Improvement and Optimization of Detection Models

1.5 Scope

1.5.1 In-Scope

Data Collection and Preprocessing

The project's first major undertaking is to collect a variety of datasets that span both URL and Portable Executable (PE) file domains and include a broad range of malware samples and legitimate software (Ayuns Luz, 2024). To guarantee that the datasets appropriately represent the diverse landscape of digital risks, this procedure involves careful sourcing from reliable repositories and sources. The obtained datasets are next subjected to stringent preprocessing techniques with the goal of harmonizing formats, addressing discrepancies, and guaranteeing data consistency ((M. S.) Muhammad Shoaib Akhtar, 2022). Potential biases and inaccuracies are reduced by carefully selecting and preparing the datasets, providing a solid basis for further research and model building.

Feature Engineering

The project moves to feature engineering, which is a crucial stage in the creation of efficient malware detection models, after the painstaking data collection and preprocessing phase. The process of feature engineering includes extracting relevant attributes from the datasets, such as textual content, metadata, and file attributes (Abdulwahab Ali Almazroi, 2024). The research intends to extract significant patterns and insights from the raw data by utilizing sophisticated approaches like TF-IDF for textual data and specialized feature extraction methods catered to file properties. The models are enhanced in terms of overall effectiveness by providing them with the discriminative power required to distinguish between malware and benign software through careful feature creation (Fang Zhiyang, 2019).

Model Development and Training

The project moves forward to model building and training, a crucial stage where machine learning techniques are implemented to detect malware across URL and PE file domains, equipped with carefully selected datasets and engineered features. Using the pre-prepared datasets, a wide range of algorithms are built and trained, such as Decision Trees, Random Forest, Support Vector Machines (SVM), XGBoost, Logistic Regression, and TF-IDF (G, 2023). By means of iterative training iterations, the models acquire the ability to identify complex patterns and anomalies that are suggestive of malevolent conduct, thus endowing them with the capacity to precisely classify cases that remain unnoticed. The project's

foundation is the model building and training phase, which creates the framework for reliable and efficient malware detection systems.

Evaluation and Optimization

After training a model, a thorough assessment is carried out using reliable measures like accuracy, precision, recall, and F1-score to determine how well the trained models are doing. Subsequent optimization attempts are guided by the essential insights gained from this evaluation step regarding the models' strengths and flaws (Ananya Redhu, 2024). By utilizing sophisticated optimization methods including feature selection and hyperparameter tuning, the project iteratively improves the models' detection efficiency and accuracy. The iterative optimization process guarantees continuous enhancement of the model's performance, which is closely aligned with the project's main goal of creating extremely efficient malware detection systems that can successfully mitigate growing cyber threats.

Documentation and Reporting

The project concludes in the thorough documenting of all processes, including model building, optimization tactics, and data gathering methodologies. Thorough reports that describe the project's results, techniques, and ramifications are painstakingly written. This documentation acts as a knowledge base, promoting cooperation and the sharing of knowledge among cybersecurity professionals. Through the thorough documentation of the project's methods and results, malware detection techniques are better understood and advanced, which improves cybersecurity resilience throughout the community.

1.5.2 Out scope

Real time Deployment and monitoring

While the project focuses on developing and optimizing malware detection models, deploying these models in a real-time environment and setting up continuous monitoring systems is beyond the scope. Implementing and managing a live monitoring infrastructure

requires additional resources, including servers, network configurations, and ongoing maintenance, which are not covered in this project.

Advanced Threat Response and Mitigation

Rather than responding to and mitigating detected threats, the project's goal is to identify malware. The project's scope does not include creating automated response systems or combining the detection models with incident response tools to do tasks like quarantining files, blocking URLs, or notifying security teams. More intensive software engineering and integration work would be required for this.

Comprehensive Cybersecurity Training and Awareness Programs

It is outside the purview of the project to create comprehensive training courses to instruct individuals or groups on how to use the malware detection system efficiently. Although the project may involve basic documentation and guidelines, it does not include developing comprehensive training modules, holding workshops, or offering continuous support to users.

1.5.3 Constraints

Resource Limitations

The project is faced with intrinsic limits resulting from finite computer resources, including memory and processing power. The project's machine learning models' complexity and scalability may be greatly impacted by these limitations. Inadequate processing capacity can make it difficult to carry out computationally demanding tasks or successfully train complicated models, which will prevent the project from performing at its best. Similarly, a little amount of memory may limit the amount of data that can be processed at once or the complexity of models that can be trained. As such, the project needs to carefully manage these resource constraints by using approaches for prioritization and optimization to make the most out of the resources that are available. To lessen the effects of resource limitations, this can entail utilizing cloud-based computing resources, putting parallel processing strategies

into practice, or using resource-efficient algorithms. The project can optimize its productivity and efficiency while attempting to accomplish its goals within the allotted computational resources by skillfully handling resource constraints. The project needs to take proactive steps to find and select a variety of datasets from reliable sources and repositories in order to overcome this limitation. Working together with academic institutions, cybersecurity organizations, and industry partners may make it easier to gain access to private datasets and improve the representativeness and diversity of the data used in the project. The initiative can lessen the effects of data availability limits and encourage the creation of more reliable and efficient malware detection systems by giving priority to data collecting and curation efforts.

Data Availability

One of the main challenges the research faces is finding representative and varied datasets that can be used to test and train machine learning algorithms. The project's goals and breadth are seriously hampered by the lack of high-quality datasets that include a variety of malware samples and genuine applications. Insufficient data availability could limit the scope and depth of the study, which could jeopardize the findings' robustness and generalizability. Furthermore, the lack of various datasets could limit the effectiveness of the created detection models by impeding the project's ability to accurately capture the subtleties and intricacies of real-world cyber threats.

Time Contraints

The project works under the limitations set by the project schedule, which could put a lot of strain on the scope and depth of research as well as task completion. Due to time restrictions, effective resource allocation and task prioritization are necessary to guarantee the timely completion of project milestones and deliverables. The project timeline's limited duration might limit how deeply some tasks, such data collecting, preprocessing, and model optimization, can be investigated. As a result, the project needs to take a methodical and organized approach to project management that includes resource allocation, milestone monitoring, and efficient job prioritizing. Iterative development and incremental delivery are two examples of agile approaches that can help in adapting to changing project needs and

lessen the negative effects of time constraints on project results. Proactive risk management techniques can also assist in anticipating and reducing possible delays or setbacks, guaranteeing that the project stays on course to accomplish its goals within the allocated timeframe. Through prudent management of time limitations and efficient project procedures, the project can achieve optimal productivity and effectiveness while maintaining deadline adherence.

1.5.4 Inclusions and Exclusions

1.5.4.1 Inclusions

The project includes a wide range of activities designed to create and improve malware detection systems that are based on machine learning. As painstakingly detailed in the scope section, inclusions cover a range of phases of the project lifecycle, including data collection, preprocessing, feature engineering, model construction, evaluation, and optimization. Together, these goals serve as the project's main framework, allowing for the methodical investigation and use of machine learning strategies to improve cybersecurity resistance against dynamic cyberthreats.

The project aims to create strong and efficient detection models that can precisely identify harmful entities while reducing false positives and negatives by taking a comprehensive approach to malware detection.

1.5.4.1 Exclusions

On the other hand, the project streamlines its focus and objectives by clearly defining exclusions. The project does not cover tasks like malware cleanup, network security, or hardware-level security measures that are not related to the use of machine learning algorithms for malware detection. Furthermore, the project scope does not address cybersecurity threats or vulnerabilities unrelated to malware.

By outlining these exclusions, the project is able to allocate resources more effectively, stay in line with project goals, and keep a focused and clear focus on improving machine learning-based malware detection. By carefully balancing inclusions and exclusions, the

initiative hopes to produce significant results that advance cybersecurity practice and research.

1.6 Overview of IR

Cybersecurity is a broad field that aims to protect hardware, software, and data on internet-connected devices from cyberattacks. It is an essential procedure used by both individuals and businesses to reduce the dangers related to cyberattacks, data breaches, and illegal access.

CHAPTER 1: INTRODUCTION

Cybersecurity is a broad field that aims to protect hardware, software, and data on internet-connected devices from cyberattacks. It is an essential procedure used by both individuals and businesses to reduce the dangers related to cyberattacks, data breaches, and illegal access. A strong cybersecurity posture is built on a successful cybersecurity strategy that aims to block harmful efforts to access, modify, delete, destroy, or extort sensitive data and system resources. Cybersecurity aims to protect digital environments against a wide range of threats, such as ransomware, malware, phishing assaults, and advanced persistent threats, by putting in place a broad array of security measure.

CHAPTER 2: LITERATURE REVIEW

The defense-in-depth theory, which promotes the installation of numerous layers of protection across a variety of access points and attack surfaces, is fundamental to cybersecurity. Strong access controls, encryption methods, and intrusion detection systems support this tiered approach, which includes safeguards for data, software, hardware, and networked systems. Defense-in-depth, which promotes the implementation of many layers of security controls across multiple domains, including critical infrastructure, network security, endpoint security, application security, and cloud security, is the fundamental tenet of cybersecurity. Organizations may build a more adaptable and resilient security posture that can fend off a wide variety of cyberattacks by implementing a tiered approach to cybersecurity.

CHAPTER 3: METHODOLOGY

Moreover, application security, network security, data protection, disaster recovery planning, and user education are just a few of the many disciplines and activities that fall under the umbrella of cybersecurity. To strengthen an organization's overall security posture, mitigate vulnerabilities, and lessen the effect of cyber incidents, each of these elements is essential. The field of cybersecurity is dynamic and ever-evolving due to the ever-evolving cyber threats and technical breakthroughs. Cybersecurity professionals need to be proactive and adaptable in order to handle new threats and vulnerabilities. They can improve threat detection and response skills by utilizing cutting-edge technology like artificial intelligence, machine learning, and behavioral analytics.

CHAPTER 4: CONCLUSION

In the end, cybersecurity ensures the availability, integrity, and confidentiality of digital assets and infrastructure, acting as a vital defense against the numerous hazards presented by cyberattacks. In an increasingly connected digital world, organizations may protect their operations, reputation, and stakeholder trust by adopting a complete cybersecurity posture and skillfully navigating the complex threat landscape.

1.7 Project Plan (Gantt Chart)

Chapter 1: Introduction	03/22/24	04/08/24
Start writing Introduction.	03/22/24	03/22/24
Complete Introduction.	03/24/24	03/24/24
Start Problem Background.	03/25/24	03/25/24
Complete Problem Background.	03/28/24	03/28/24
Write Project Aim.	03/29/24	03/29/24
Complete Project Aim.	03/30/24	03/30/24
Start Objectives.	03/31/24	03/31/24
Complete Objectives.	04/01/24	04/01/24
Start Scope.	04/02/24	04/02/24
Complete Scope.	04/04/24	04/04/24
Write Overview of the IR.	04/05/24	04/05/24
Complete Overview of the IR.	04/06/24	04/06/24
Start Project Plan.	04/07/24	04/07/24
Complete Project Plan.	04/08/24	04/08/24

Figure 1 Gantt Chart Table

Chapter 2: Literature Review	04/09/24	04/25/24
Start Domain Research.	04/09/24	04/09/24
Complete Domain Research.	04/15/24	04/15/24
Write Characteristics of similar systems/works.	04/16/24	04/16/24
Complete Characteristics of similar systems/works.	04/17/24	04/17/24
Write Strength and weakness.	04/18/24	04/18/24
Complete Strength and weakness.	04/19/24	04/19/24
Write Conclusion for similar systems/works.	04/20/24	04/20/24
Start System Requirement Analysis.	04/21/24	04/21/24
Complete System Requirement Analysis.	04/25/24	04/25/24

Figure 2 Gantt Chart Table

Chapter 3: Methodology	04/26/24	05/15/24
Write Introduction to System Development Methodology.	04/26/24	04/26/24
Write Methodology choice and justification.	04/27/24	04/27/24
Describe activities in each phase.	04/28/24	04/28/24
Complete description of activities.	04/30/24	04/30/24
Start Data Gathering Design.	05/01/24	05/01/24
Complete Description of techniques.	05/03/24	05/03/24
Design questions/events/activities.	05/04/24	05/04/24
Complete design of questions/events/activities.	05/05/24	05/05/24
Get verification from the supervisor.	05/06/24	05/06/24
Conduct surveys.	05/07/24	05/07/24
Complete surveys.	05/08/24	05/08/24
Document analysis and observation.	05/09/24	05/09/24
Start analysis of data.	05/10/24	05/10/24
Complete analysis of data.	05/11/24	05/11/24
Summarize user requirements.	05/13/24	05/13/24
Complete summary of user requirements.	05/15/24	05/15/24

Figure 3 Gantt Chart Table

Chapter 4: Conclusion	05/16/24	05/22/24
Start discussing achievements.	05/16/24	05/16/24
Complete justification of investigation/research.	05/19/24	05/19/24
Identify gaps and areas for improvement.	05/20/24	05/20/24
Complete identification of gaps.	05/22/24	05/22/24

Figure 4 Gantt Chart Table

Final Review and Submission	05/23/24	05/28/24
Start final review.	05/23/24	05/23/24
Complete final review.	05/27/24	05/27/24
Submit the final assignment.	05/28/24	05/28/24

Figure 5 Gantt Chart Table

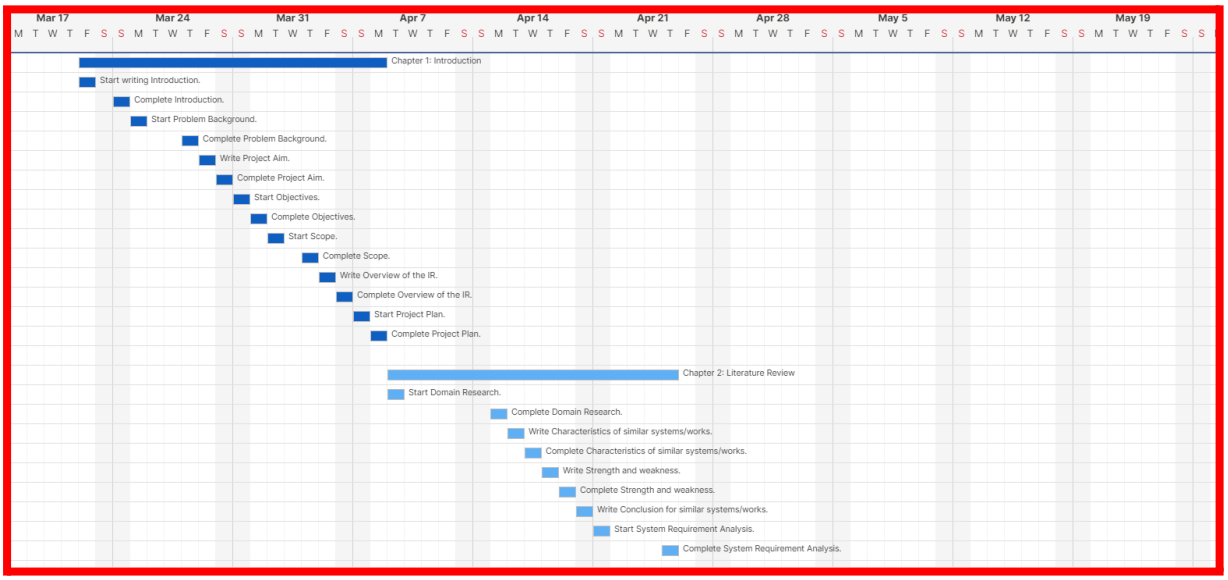


Figure 6 Gantt Chart

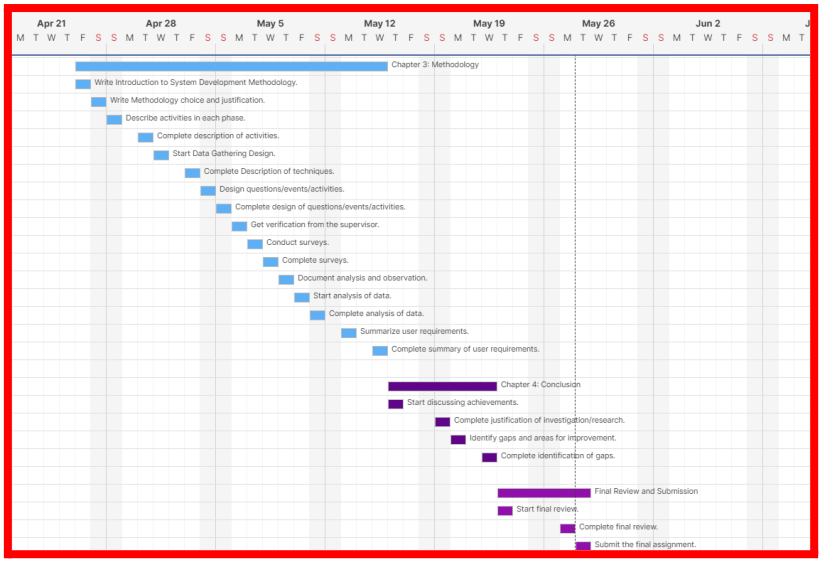


Figure 7 Gantt Chart

2.0 Chapter 2

2.1 Domain Research

2.1.1 An article on Malware Detection

An overview of malware detection methods with an emphasis on combining deep convolutional neural networks (CNNs) with image processing to improve malware detection and classification. Trojans, viruses, worms, spyware, rootkits, botnets, crimeware, and adware are only a few of the malware varieties that are covered, demonstrating the wide range of cyberthreats (Muhammad Azeem, Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches, 2024). It is highlighted that applying machine learning (ML), and especially deep learning models, is a potential way to deal with the increasing volume and complexity of malware.

Numerous cybersecurity applications, such as phishing detection, intrusion detection, malware detection, and anomaly detection, have demonstrated the effectiveness of machine learning techniques. However, polymorphic malware can evade conventional methods that rely on signatures. The paper emphasizes how crucial it is to keep ML models up to date with new datasets in order to improve their accuracy and defense against malicious attacks.

Also detection modules are defined as essential parts that analyze gathered data and train machine learning models to efficiently identify security issues. By using machine learning, cybersecurity measures are improved as systems are able to recognize underlying trends and gradually increase forecast accuracy (Tao Feng M. S., 2022).

Well highlighting how disruptive it is to computerized systems and how it might jeopardize private data or grant illegal access. It divides malware into three categories viruses, worms, and Trojans and highlights the differences between each type of malware and how it spreads. While worms can propagate independently across network connections and infect multiple systems, viruses require a host application to proliferate (Muchammad Naseer, 2021). In contrast, Trojan horses pose as normal software in order to carry out nefarious tasks.

Even with the availability of numerous security measures, such as encryption techniques and antivirus software, there are still issues in offering sustained defense against malicious attacks that change over time (Muchammad Naseer, 2021). It emphasizes how protective measures must be updated and improved constantly in order to effectively counteract harmful activity.

The outcomes of the experiments show how well ML classifiers including Random Forest (RF), nnMLP, Extra Tree (ET), KNN, Logistic Regression (LR), and Decision Trees (DT)identify and categorize malware. The suggested solution outperforms earlier approaches with a high accuracy score of 99.98% on the ET classifier by utilizing feature selection techniques and removing unnecessary information.

Overall, the paper emphasizes how important it is to combine domain-specific expertise with cutting-edge machine learning approaches in cybersecurity to improve malware detection and classification skills. To further enhance prediction models and fortify cybersecurity safeguards, future research paths will include hidden Markov models and deep learning techniques.

2.1.2 Algorithms in Machine Learning

2.1.3 Machine learning using SVM for Malware Detection

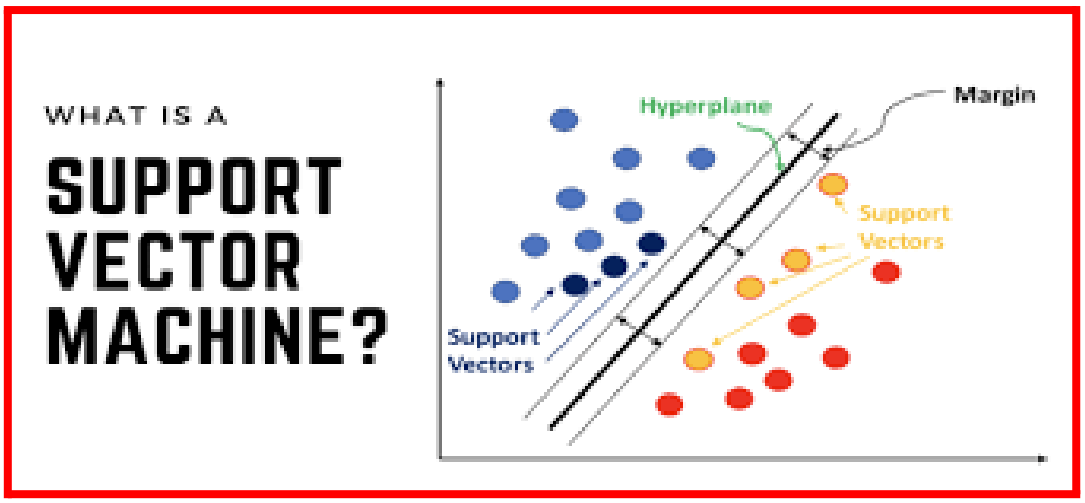


Figure 8 SVM (Smita Ranveer, n.d.)

A strong method for classifying data, Support Vector Machine (SVM) works especially well when the data shows non-linear relationships and non-regular distributions. Since SVMs lessen the dominance of attributes with wider ranges over those with smaller ones, they are effective at managing datasets whose attributes have changing numeric ranges, in contrast to several other classification techniques. By optimizing the margin, or distance, between various classes in the feature space, this is accomplished.

Training and testing are the two separate stages in which the SVM algorithm functions. A collection of input points classified as belonging to one of two categories and represented in the Attribute Relation File Format (ARFF) are fed to the SVM during the training phase. During this stage, the SVM's objective is to build a model that successfully divides the input points of various categories by a distinct margin in the feature space. Maximizing this margin and making sure it is as broad as feasible are the goals (Smita Ranveer, n.d.).

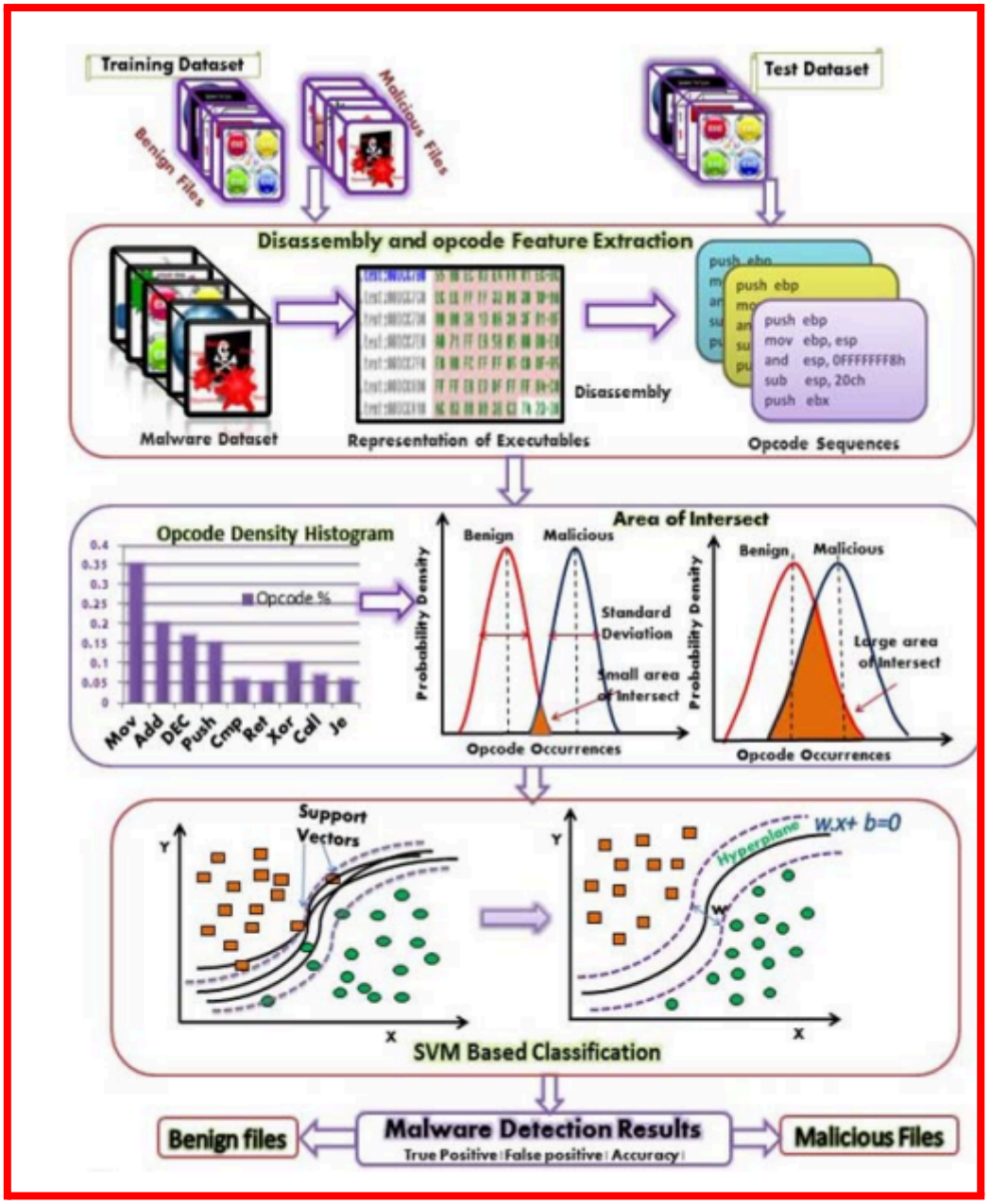


Figure 9 Architecture of SVM based effective malware detection system (Smita Ranveer, n.d.)

Well from the article I read new data points can be classified by the SVM model after the training phase is over. The SVM predicts a data point's category based on which side of the gap, or hyperplane, it falls on after mapping it into the same feature space that was used for training. When using a linear support vector machine (SVM) model, a hyperplane is used to divide data points into distinct categories. The hyperplane maximizes the distance between each of its closest data points on both sides.

But because to a method called the kernel trick, SVMs can also manage non-linear

relationships between data points. By using this method, the SVM can implicitly convert the feature space into a higher-dimensional space where a hyperplane can efficiently capture non-linear correlations (Smita Ranveer, n.d.). Different kernel functions, including polynomial or radial basis function (RBF) kernels, can be used to teach the Support Vector Machine (SVM) to recognize intricate decision boundaries that divide various data classes.

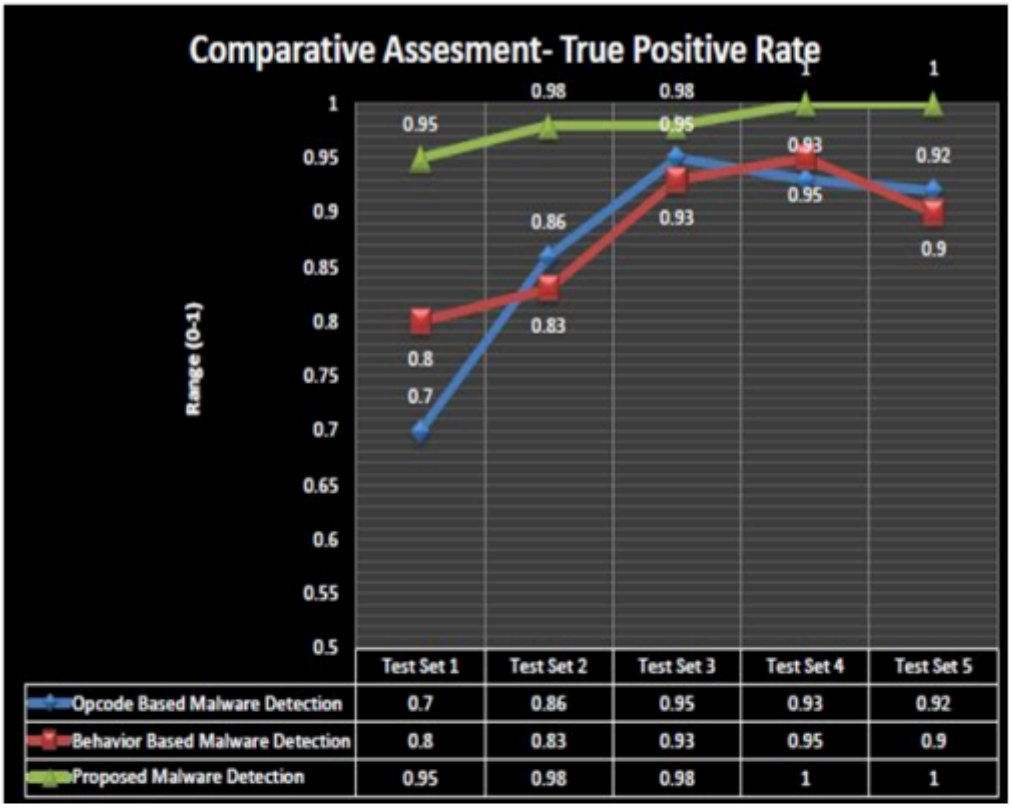


Fig. 2. Performance evaluation in terms of TPR

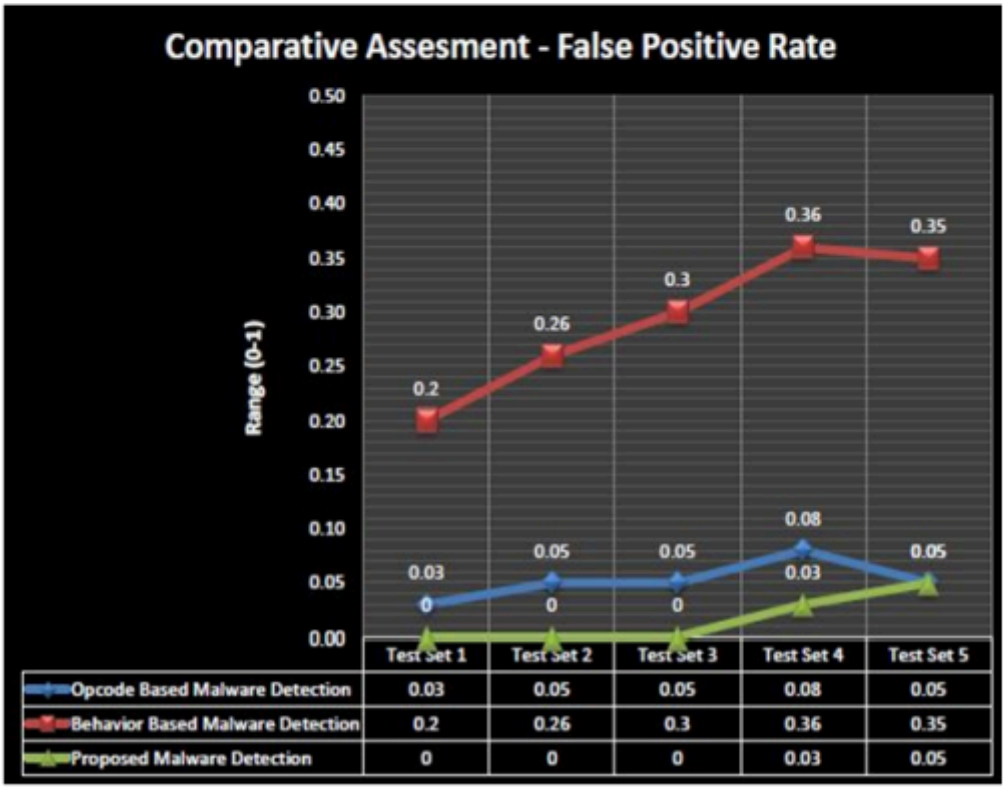


Fig. 3. Performance evaluation in terms of FPR

e

Figure 10 Performance evaluation in terms of TPR and FPR (Smita Ranveer, n.d.)

So by dividing groups of benign and dangerous files from new datasets, the SVM tests its efficacy during the testing phase. It uses the model that was learned in the training phase to distinguish between various classes according to the feature representations of those classes. SVMs can be set up to identify files containing system calls that behave differently from typical benign files and files containing opcodes that have a major influence on the differentiation between dangerous and benign software when it comes to malware identificat

So in conclusion, SVMs are strong and adaptable classifiers that can manage non-linear relationships and complicated data distributions. They are extensively employed in many domains, such as biology, image recognition, and cybersecurity, because of their capacity to efficiently divide data into distinct classes by optimizing margins in feature spaces.

2.1.4 Article on Machine learning using Decision Tree for Malware Detection

Basically, decision trees are essential tools in cybersecurity that are used for a wide range of activities, including malware classification, intrusion detection, and behavioral analysis. With the use of these trees, data is categorized into discrete classes by iteratively segmenting the feature space according to feature values. By examining network traffic data, decision trees are useful tools in the field of intrusion detection. The nodes in this tree represent several aspects of the network traffic, such as protocol types and source IP addresses. Decision trees are highly skilled in identifying patterns that correspond to both benign and malevolent network activity through a series of splits based on these features. In turn, the leaf nodes provide the anticipated class labels, like "normal" or "intrusion detected," giving security analysts practical information about possible dangers (What Is a Decision Tree? | Master's in Data Science, 2024).

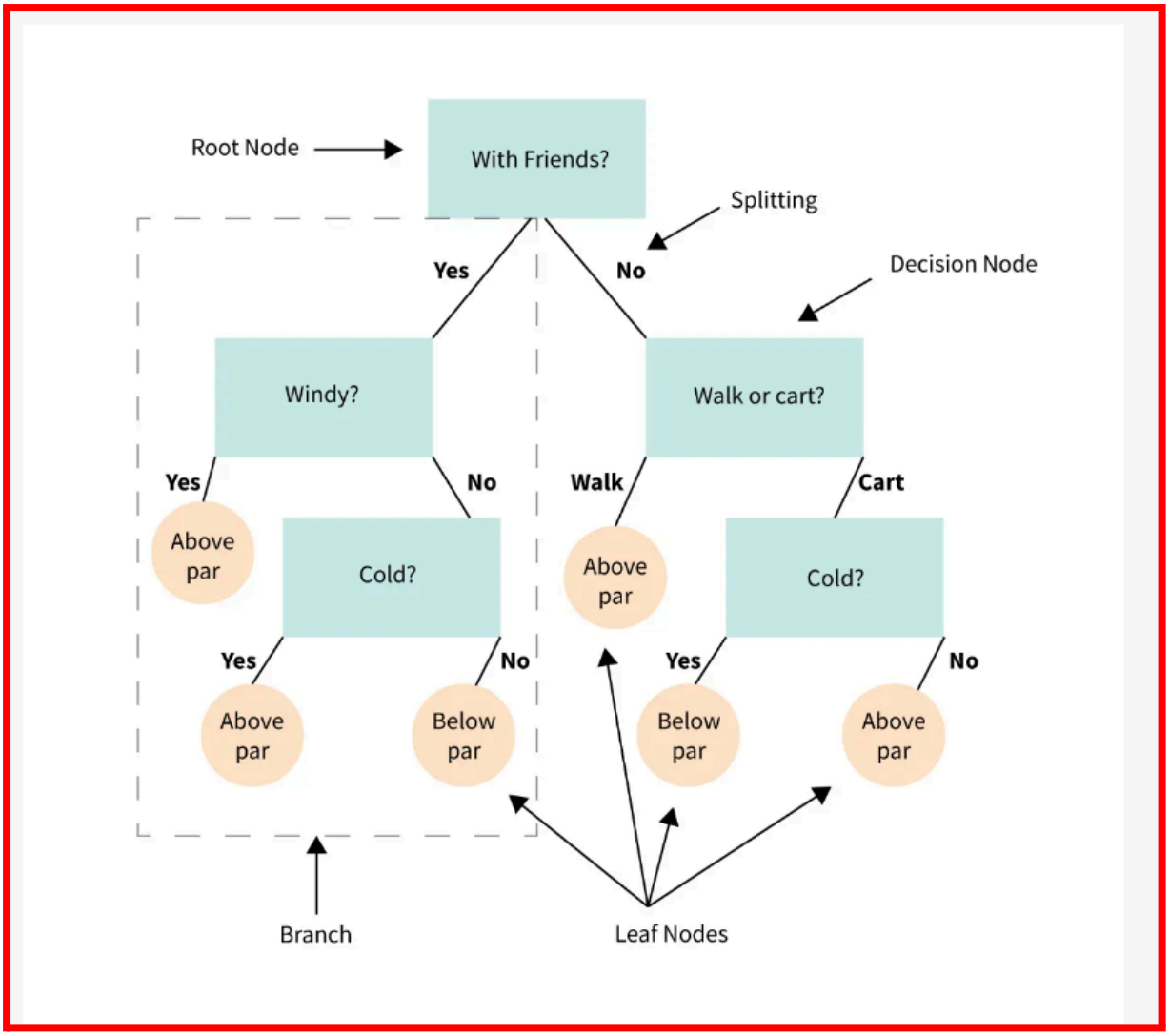


Figure 11 Decision Tree (What Is a Decision Tree? | Master's in Data Science, 2024)

Decision trees also play a crucial role in malware classification, since their capacity to interpret file properties like as size, type, and embedded opcodes makes it easier to classify files into harmful and benign categories. By utilizing recursive partitioning, decision trees methodically separate the data, allowing them to distinguish between files that are harmful and those that are not (What Is a Decision Tree? | Master's in Data Science, 2024). As a result, leaf nodes outline the anticipated class labels, enabling security professionals to quickly detect and neutralize such dangers that may be present in their systems.

Decision trees are very useful in behavioral analysis, especially when examining user or system actions for signs of unusual activity. Decision trees examine a variety of behavioral characteristics, including login timings, access patterns, and resource usage, in order to identify behavioral anomalies that might indicate illegal access or malevolent intent. The

interpretability of decision trees, which enables security analysts to quickly identify and address any dangers, further emphasizes this analytical prowess.

Most importantly, decision trees provide information on the significance of features, illuminating which features have the strongest discriminating ability when it comes to categorizing data. Security professionals can concentrate their attention on the most prominent signs of cyber dangers by using this information, which is essential for driving feature selection and prioritization activities.

To sum up, decision trees are an essential part of cybersecurity technologies, providing a powerful combination of ease of use, readability, and effectiveness when analyzing complicated cyber data. Their adaptability makes them indispensable for a variety of cybersecurity activities, such as behavioral analysis and intrusion detection, strengthening enterprises against a constantly changing landscape of cyber threats.

2.1.5 Article on Machine learning using Random Forest for Malware Detection

Random Forest is a machine learning technique with great potential for malware detection in networked computer systems. Accurate malware variant detection and categorization is critical as cybercrime operations continue to constitute a growing danger, especially in developed nations with developed information and communications infrastructures. Spyware, bots, rootkits, Trojan horses, and viruses are examples of malware that can perform a wide range of destructive tasks, from stealing private information to disrupting services.

The primary difficulty in battling malware is its capacity to evade detection and change its code, making signature-based anti-malware programs ineffectual. Each malware variant's obfuscation and morphing produce a number of data points, making identification and categorization more difficult. Nataraj et al. presented an innovative solution to this problem by using the intricate visual patterns present in malware signatures to visualize malware binaries as picture files. These picture representations, which preserve important structural features even after code obfuscation, show promise in the classification of malware variants.

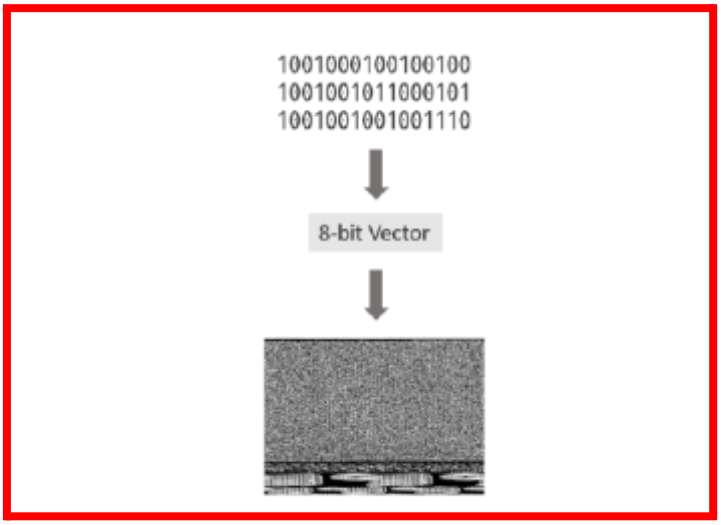


Figure 12 Conversion of Malware binaries into gray scale images (Felan Carlo C. Garcia, 2016)

From the article which I read, the emphasis is shifted to using Random Forest and malware images as feature vectors for efficient malware family classification and segregation. This method's evaluation is based on the Malimg Dataset, which consists of 9,342 malware samples from 25 distinct families. The dataset is notably unbalanced, which means that considerable thought must be given to minimizing overfitting and undergeneralization while creating a training set (Felan Carlo C. Garcia, 2016). Stratified sampling of dataset populations becomes essential to guarantee strong model performance across various malware strains.

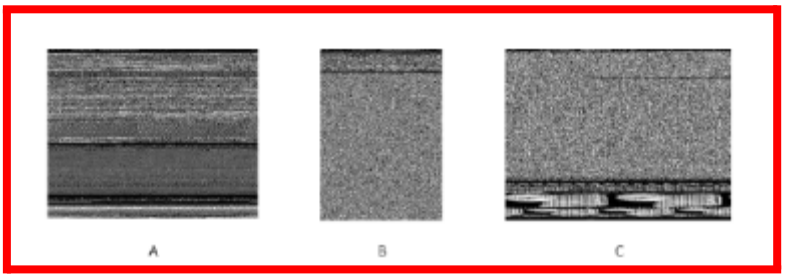


Figure 13 Resulting Malware images from the binary files (Felan Carlo C. Garcia, 2016)

Renowned for its capacity to manage high-dimensional data and reduce overfitting, Random Forest shows promise as a virus detection tool. Random Forest efficiently captures the intricate linkages present in malware images by building an ensemble of decision trees and

combining their predictions, making it possible to classify and distinguish malware families with accuracy. Because of its innate scalability and interpretability, it's a good fit for managing the complexities of malware detection in actual cybersecurity settings.

To put it briefly, combining Random Forest with malware photos as feature vectors offers a strong method for improving malware detection abilities. This methodology has the potential to enhance conventional signature-based techniques by utilizing the visual signatures present in malware binaries, hence providing enhanced resistance against dynamic cyber threats (Felan Carlo C. Garcia, 2016).

2.1.6 Article on Machine learning using XGBOOST for Malware Detection

When it comes to virus detection in networked computer systems, XGBoost (Extreme Gradient Boosting) is a very effective method. Robust detection approaches are becoming increasingly necessary as cyber threats multiply and are equipped with complex malware variants. Malware presents a variety of difficulties since it obfuscates code and transforms to avoid detection by conventional methods. It includes spyware, bots, rootkits, Trojan horses, and viruses. It takes a careful combination of features to identify this kind of malware, from file format properties to file-independent data like byte histograms, byte entropy, and strings inserted in programs.

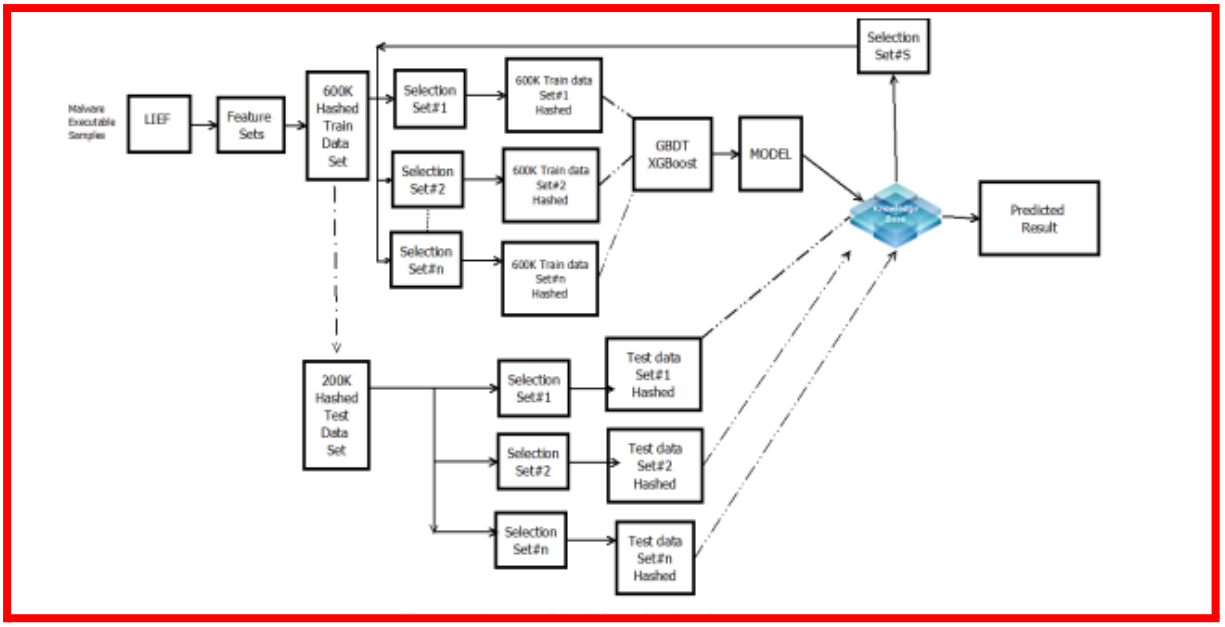


Figure 14 System Block for Selection Data (Rajesh Kumar, 2022)

An essential component of malware detection is the examination of Portable Executable (PE) files, which are the common file format used for Windows binaries. PE files are composed in an organized manner; they contain section tables, optional headers, COFF headers, and data directories, all of which provide vital information needed for malware research. Methods such as the LIEF (Library for Instrumenting Executable Files) facilitate the feature engineering process for malware detection by helping to extract relevant information from PE files.

Malware detection techniques are often divided into two categories: static and dynamic. Static detection uses structural information from file formats to identify malware; it works pre-execution. However, more sophisticated methods are required due to the severe hurdles posed by malware variants that are polymorphic and metamorphic to signature-based static detection. In contrast, dynamic detection involves running the virus in a controlled environment in order to watch how it behaves. Although dynamic analysis offers more profound understanding of malware activity, it is computationally demanding and needs specific conditions to reduce infection risks (Rajesh Kumar, 2022).

This is where XGBoost's effectiveness in malware detection rests. XGBoost is a scalable ensemble decision tree technique that performs exceptionally well when working with big

datasets that have a lot of features. XGBoost iteratively improves model performance by reducing classification mistakes and raising detection accuracy by utilizing the power of gradient boosting. Because of this, it's ideally equipped to handle the complexity of malware detection, especially when dealing with malware variants that exhibit polymorphism and metamorphism.

Furthermore, XGBoost provides notable computational benefits in comparison to conventional techniques such as Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN). Because of its efficiency and scalability, it is especially good at managing the high-dimensional feature spaces that are typical of malware detection jobs (Rajesh Kumar, 2022). Furthermore, developments in XGBoost implementations such as LightGBM significantly augment its efficacy, indicating significant gains in detection accuracies and training procedures.

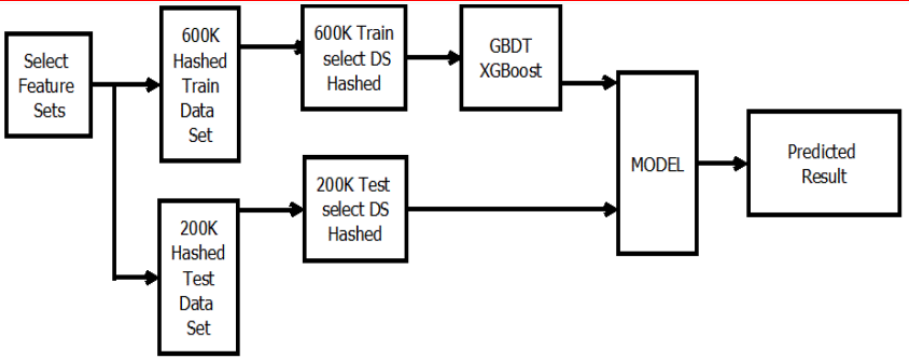


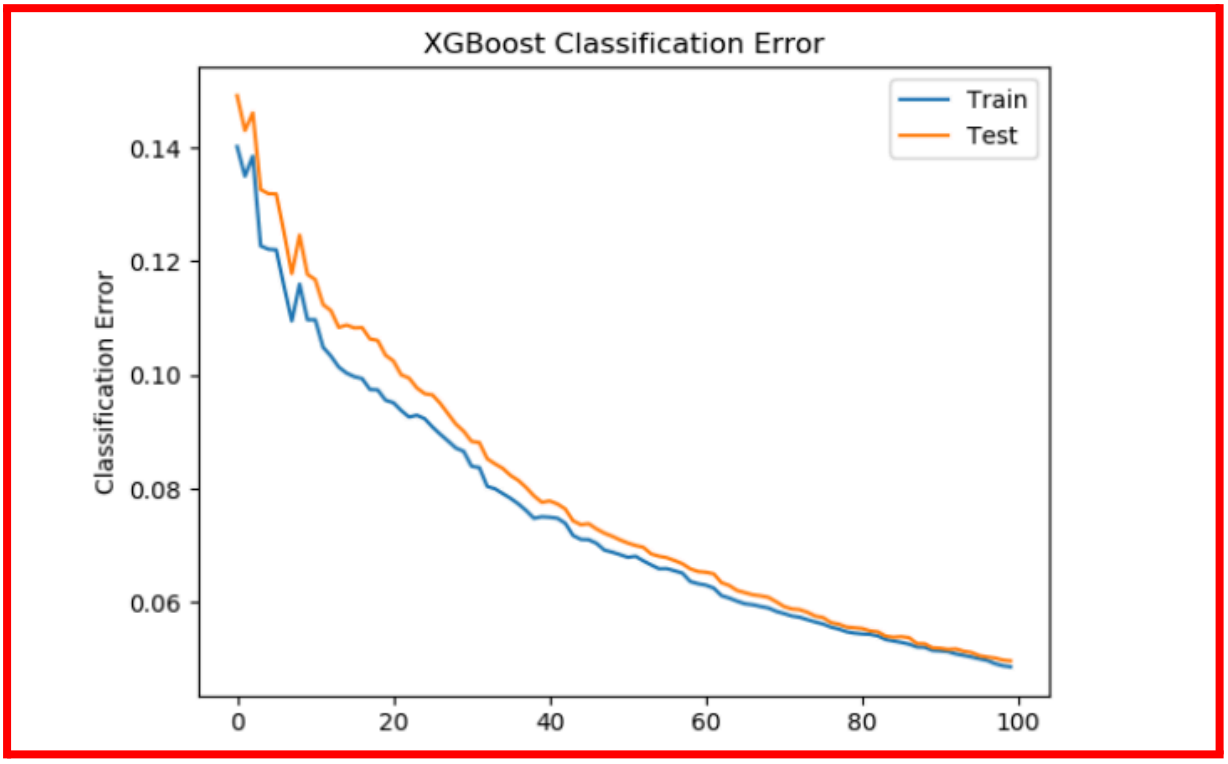
Figure 2: Model Block diagram using XGBoost feature importance

Table 4: Index of Selected 276 Import Features

Description	Number of features	Index number of features
Raw byte histogram	66	1, 2, 3, 6, 10, 15, 19, 21, 22, 27, 29, 31, 32, 33, 35, 37, 40, 41, 43, 45, 46, 50, 54, 63, 64, 65, 66, 68, 69, 74, 76, 88, 89, 90, 93, 95, 105, 106, 107, 111, 112, 113, 116, 123, 125, 128, 131, 133, 135, 181, 182, 196, 198, 199, 204, 209, 214, 217, 218, 219, 223, 231, 239, 245, 253, 255
Byte Entropy	46	256, 303, 312, 318, 326, 328, 335, 338, 357, 360, 362, 372, 373, 375, 377, 378, 385, 386, 388, 389, 392, 396, 399, 401, 410, 411, 413, 417, 426, 431, 447, 456, 457, 464, 466, 470, 472, 494, 498, 499, 502, 504, 508, 509, 510, 511
Strings	34	512, 515, 521, 529, 530, 531, 532, 533, 534, 543, 551, 552, 553, 554, 558, 562, 570, 578, 583, 588, 589, 596, 597, 600, 601, 602, 604, 605, 606, 611, 612, 613, 614, 615
General (COFF)	6	616(size), 617(v size), 618(has debug), 619(exports), 620(imports), 623(has signature)
Optional Header,	16	626(Time Stamp),632(Machine), 637(characteristics), 640, 654(subsystem), 655, 658(DLL characteristics), 660, 677, 678, 679, 680, 681, 682, 683, 685,
Sections	27	688(name of section),689(size), 691(v size), 692(properties), 693(section size), 707, 712, 734, 736, 748(section entropy hash), 770, 771, 775, 784, 785, 786, 797(section v size hash), 798, 803, 825, 827, 834, 836, 843, 906, 930, 940
Imports of API with DLL	81	951, 954, 986, 994, 1011, 1043, 1060, 1073, 1162, 1190, 1197, 1225, 1254, 1263, 1303, 1309, 1312, 1316, 1342, 1343, 1360, 1362, 1366, 1377, 1387, 1388, 1396, 1399, 1404, 1445, 1451, 1455, 1476, 1482, 1484, 1505, 1526, 1545, 1546, 1597, 1629, 1656, 1663, 1685, 1689, 1693, 1704, 1712, 1724, 1756, 1773, 1775, 1799, 1807, 1815, 1836, 1886, 1892, 1901, 1949, 1969, 1973, 1991, 2004, 2006, 2018, 2034, 2047, 2052, 2078, 2083, 2097, 2110, 2114, 2125, 2140, 2159, 2180, 2184, 2188, 2210
Exports of API	0	

Figure 15 Model Block Diagram using XGBoost feature importance (Rajesh Kumar, 2022)

All things considered, XGBoost proves to be a strong ally in the ongoing war against cyber dangers. Its crucial significance in strengthening cybersecurity defenses against changing malware landscapes is highlighted by its capacity to handle enormous datasets, extract complicated patterns from feature spaces, and provide exceptional detection performance.



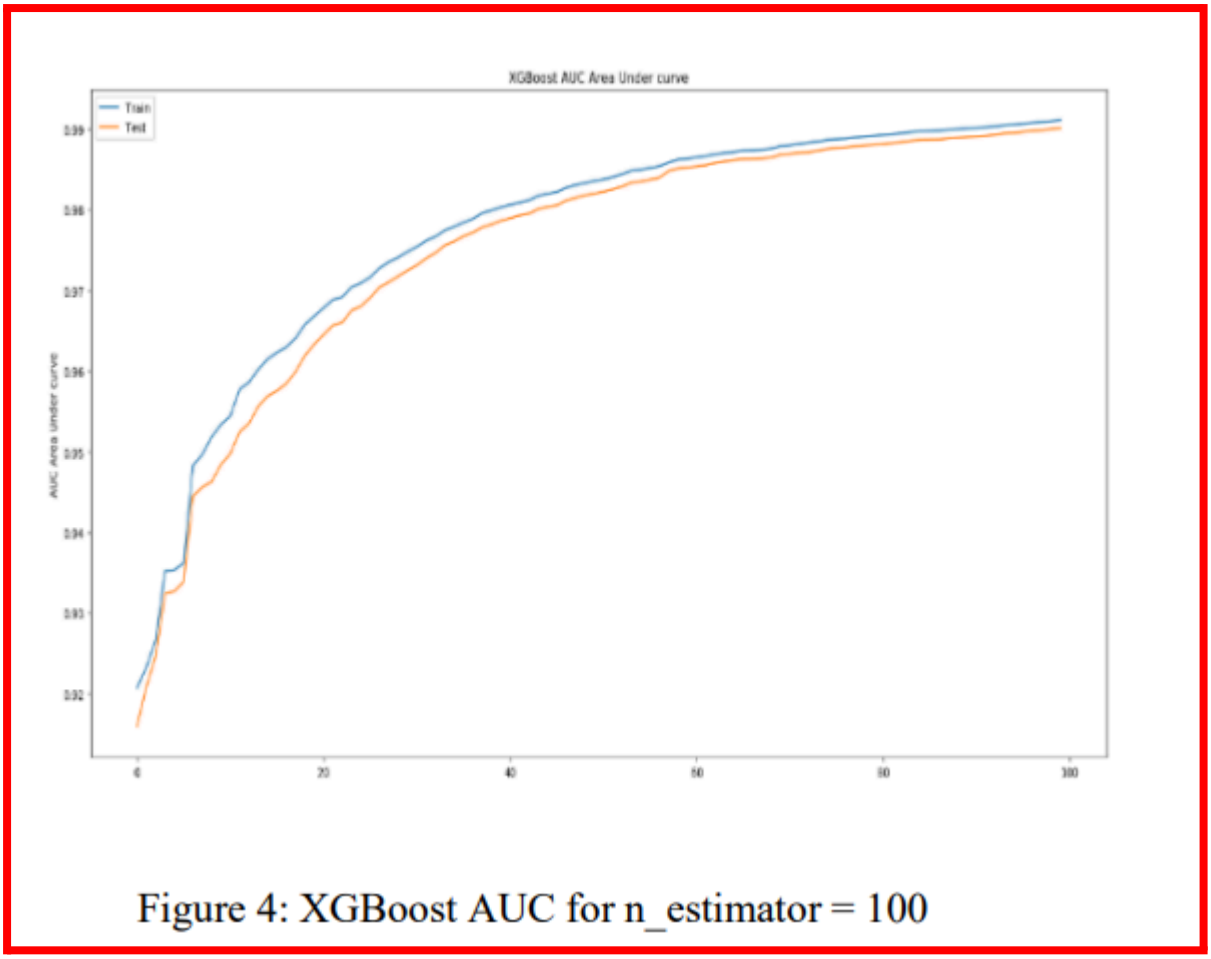


Figure 4: XGBoost AUC for n_estimator = 100

Figure 16 Classification Error example (Rajesh Kumar, 2022)

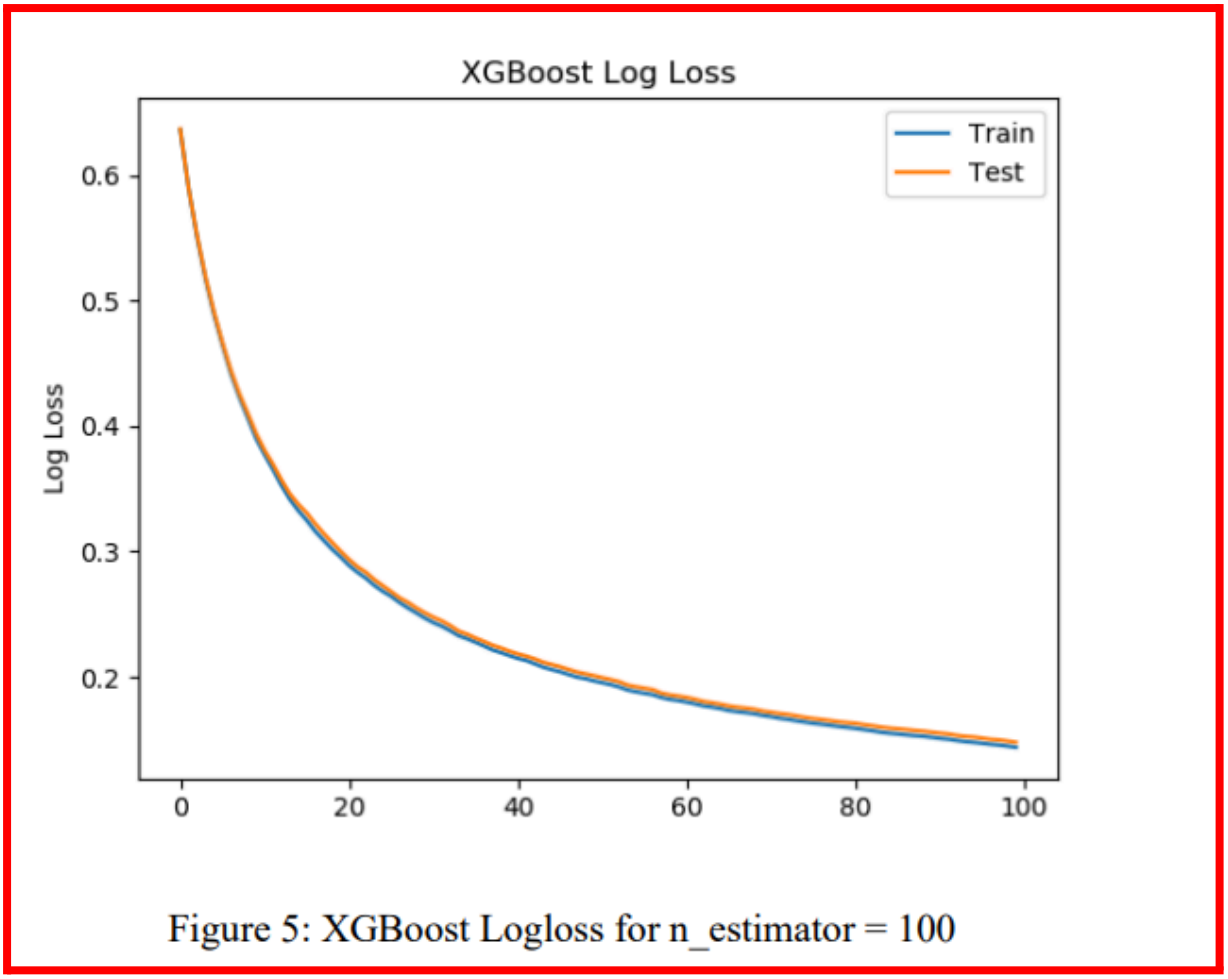


Figure 17 Classification Error Example (Rajesh Kumar, 2022)

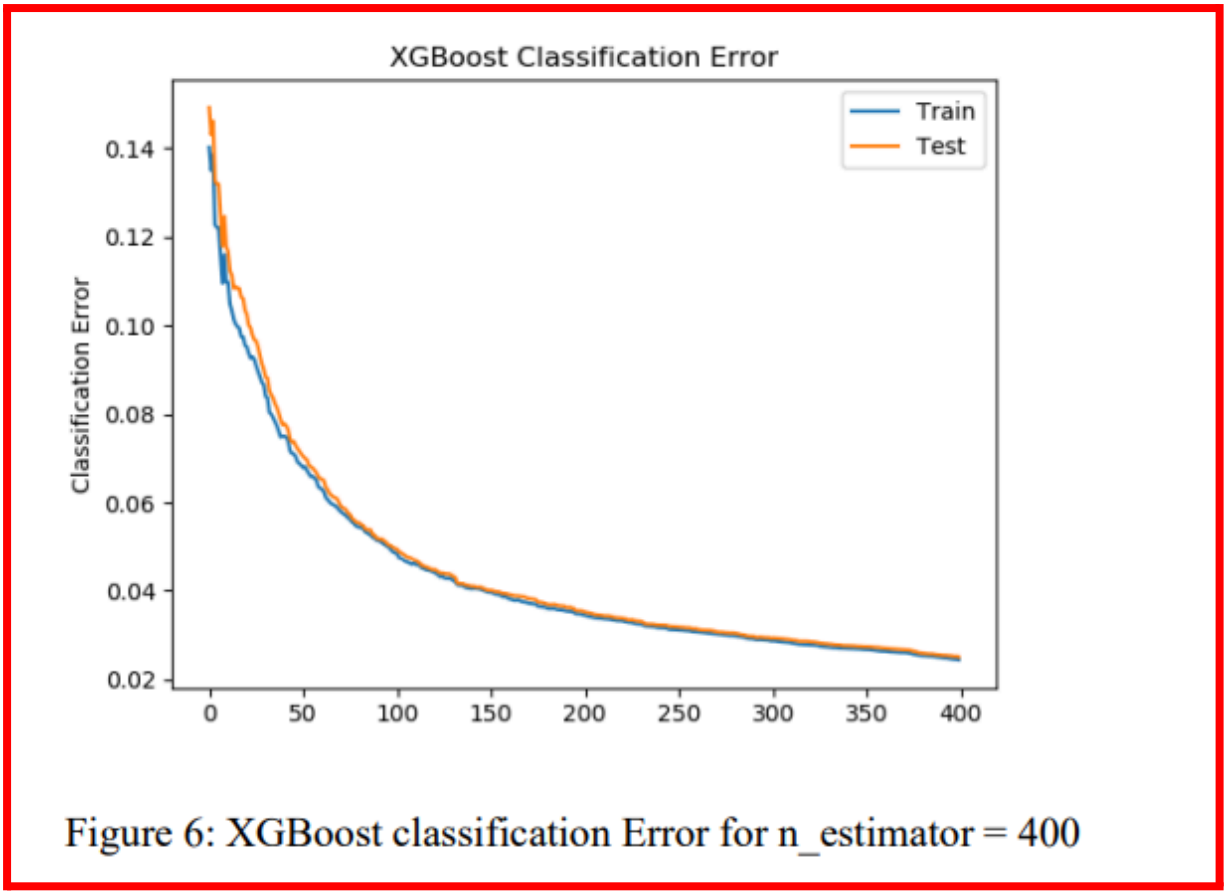


Figure 18 Classification Error Example (Rajesh Kumar, 2022)

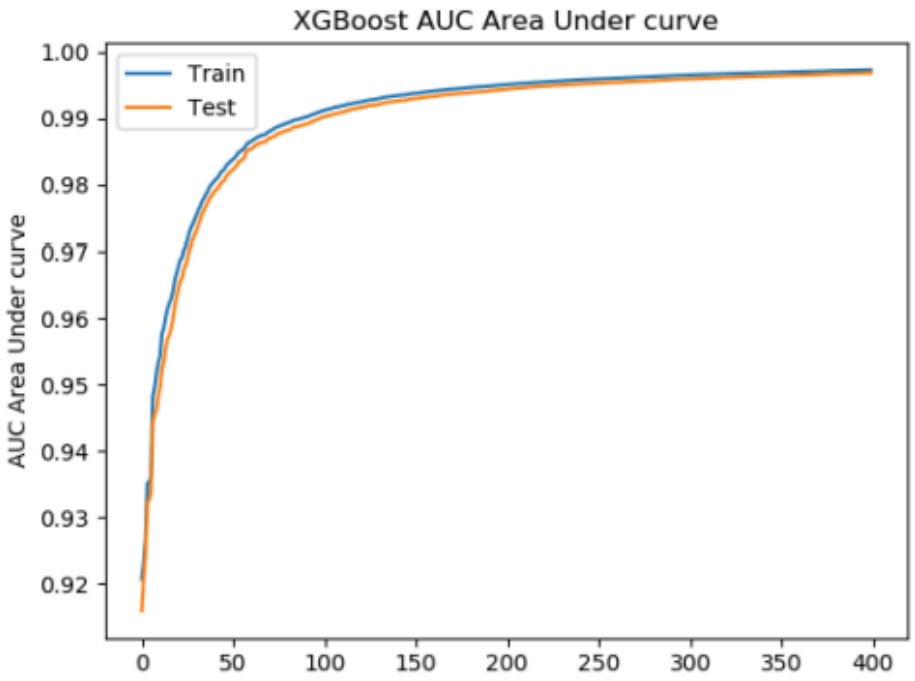


Figure 7: XGBoost AUC for n_estimator = 400

Figure 19 Classification Error Example (Rajesh Kumar, 2022)

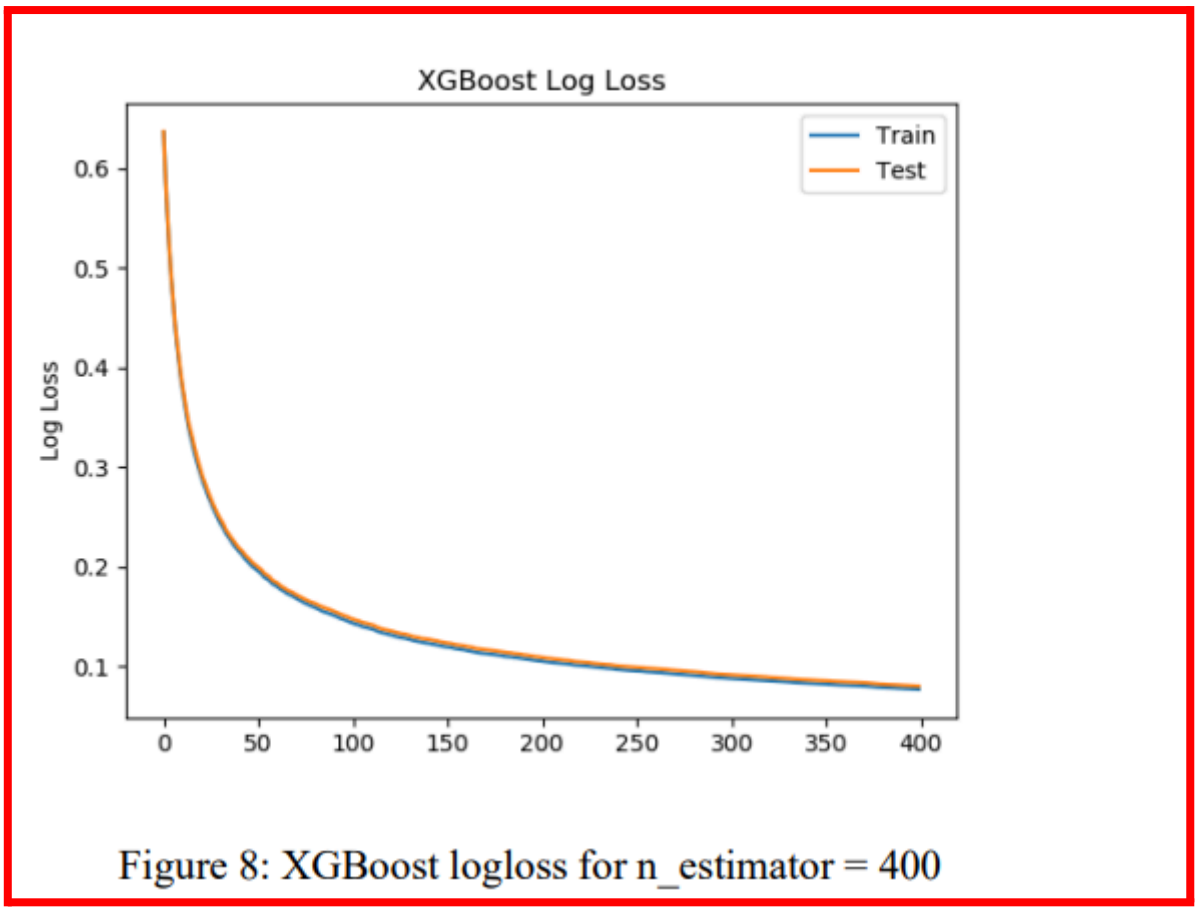


Figure 20 Classification Error Example (Rajesh Kumar, 2022)

2.1.6 Article on Machine learning using Logistic Regression for binary classification, with feature extraction using TF-IDF for Malware Detection

Another popular machine learning approach for binary classification problems is logistic regression. It simulates a binary outcome's likelihood depending on one or more predictor variables. The binary result may indicate whether a file is malicious or benign in the context of malware detection (Jadia, 2023).

The process of feature extraction is essential to getting the data ready for categorization. Text documents are transformed into numerical feature vectors through the use of tools like CountVectorizer and TF-IDF Vectorizer, which extract features from text data for sentiment analysis. Comparably, characteristics of the malware files, such as file format properties, byte histograms, or strings encoded in the programs, can be used to extract features for malware identification.

Term Frequency-Inverse Document Frequency, or TF-IDF The vectorizer is especially helpful for extracting features from textual data. By giving terms weights determined by how frequently they appear in the document and throughout the corpus, it can be used to highlight key terms and minimize less significant ones. This method can be modified to extract features from malware samples, where specific traits might point to malevolent activity.

After the features are retrieved, the data is used to train Logistic Regression, which determines the correlation between the characteristics and the binary result (malware or benign). Based on feature representations, the trained model may then determine whether additional samples fall into the benign or harmful class (Jadia, 2023).

In the sentiment analysis study, metrics such as accuracy, precision, recall, F1-score, and ROC curves are used to evaluate the Logistic Regression models' performance. Comparably, model performance in malware detection can be assessed using comparable metrics to determine how well it separates files that are harmful from those that are benign.

In conclusion, TF-IDF feature extraction combined with logistic regression can be a potent method for binary classification applications, such as cybersecurity malware detection. This

methodology can improve cybersecurity defenses against emerging threats by accurately and reliably detecting malware through the appropriate utilization of text data properties.

2.2 Similar systems

2.2.1 VirusTotal

Figure 21 VirusTotal (BERNARDO.QUINTERO, 2023)

Figure 22 VirusTotal Website (BERNARDO.QUINTERO, 2023)

An online application called VirusTotal gathers many antivirus engines and technologies to examine files and URLs for possible security risks. Researchers, cybersecurity experts, and regular users all use it extensively to find viruses, worms, trojans, and other harmful stuff.

Through the integration of data from more than 70 antivirus scanners, URL/domain blacklisting services, and additional technologies, VirusTotal offers a thorough evaluation of the possible risks associated with a certain file or URL.

VirusTotal's capacity to run concurrent scans with a wide range of antivirus engines is one of its key features. By combining the advantages and detection powers of multiple manufacturers, this multi-engine method dramatically raises the probability of finding malware (BERNARDO.QUINTERO, 2023). To find threats, each antivirus engine may employ a different method. These methods may include behavior analysis, heuristic analysis, and signature-based detection. VirusTotal can therefore provide a more comprehensive examination than just utilizing one antivirus program.

Another well-known feature of VirusTotal is its enormous collection of known malware behaviors and signatures. New definitions and threat intelligence from collaborating antivirus companies are regularly added to this database. This guarantees that the service is able to identify the most recent threats and offer current malware information. Furthermore, by submitting new files and URLs, users of VirusTotal add to the database, aiding in the continuous detection and cataloging of novel and emerging dangers.

VirusTotal's user-friendly online interface and API, which make it accessible and simple to use for both individuals and enterprises, are another important feature. Users can simply upload files or submit URLs for scanning using the web interface, and the API offers sophisticated functionality for integrating VirusTotal's scanning capabilities into security systems and automated workflows. Because of its adaptability, VirusTotal is a useful tool for both automatic and human threat detection procedures.

Moreover, VirusTotal offers thorough reports on every file or URL that is scanned. These reports include details on any risks that are identified, the identity of the malware (if any), and the antivirus engines that found the infection. In addition, these reports contain metadata that might be helpful for obtaining threat intelligence and for additional analysis, such as file size, type, and hash values. Because these reports are transparent, users can comprehend the types of dangers and the reasoning for their identification.

In conclusion, VirusTotal is a strong and adaptable malware detection service that makes use of the combined strengths of multiple antivirus engines. With its extensive database, frequent updates, intuitive interface, and thorough reporting, it's a vital resource for researchers and cybersecurity experts who want to effectively find and evaluate harmful content.

2.2.1.1 Strength of VirusTotal

The primary strength of VirusTotal is its extensive detection capabilities, which is attained by utilizing more than 70 distinct antivirus engines. This multi-engine strategy integrates the various detection techniques and signature databases of multiple vendors, greatly increasing the probability of detecting a broad variety of malware. As such, using VirusTotal ensures a wider detection spectrum and provides a more comprehensive and trustworthy analysis than depending solely on one antivirus program (S, 2023). Additionally, VirusTotal maintains up-to-date protection in the rapidly changing cybersecurity scene thanks to its regularly updated database, which is fed by partner antivirus companies and user contributions.

VirusTotal's user-friendly interface and strong API, which serve both individual users and organizations, are two of its other key strengths. Users with varied degrees of technical expertise can easily upload files and submit URLs for scanning thanks to the user-friendly online interface. To improve corporate threat detection and response capabilities, the API provides sophisticated capability for incorporating VirusTotal's capabilities into automated security workflows and systems. Moreover, VirusTotal offers comprehensive and transparent reports that include details on threats that have been detected as well as pertinent metadata, helping users make well-informed decisions and gaining a deeper understanding of possible risks. The adaptability in managing different file formats and URLs highlights VirusTotal's extensive use in cybersecurity analysis (VIRUSTotals 2021 Malware Trends report, 2022).

2.2.1.2 Weaknesses of VirusTotal

VirusTotal provides a lot of advantages, but it also has some significant disadvantages. The possibility of false positives and false negatives as a result of the different integrated antivirus engines' detection techniques is one important drawback. Results can be inconsistent since different antivirus programs may classify different files as dangerous based on their own signature databases and algorithms (Gigahoo, 2017). Users attempting to ascertain the true threat level of a file or URL may become confused as a result of this. Moreover, users are left to manually interpret the disparate responses from the various engines, which could result in mistakes in threat assessment, as VirusTotal does not offer a consensus or weighted analysis of the data.

VirusTotal's deficiency in dynamic testing and in-depth behavioral analysis is another flaw. It performs excellent static analysis by employing several antivirus engines to scan files and URLs; but, it is not capable of performing comprehensive dynamic analysis, which entails running the questionable files in a safe environment to watch how they behave. Because of this restriction, VirusTotal may be unable to identify some sophisticated malware strains that elude static detection by using strategies like code obfuscation, metamorphism, or environment detection. Additionally, users must upload potentially sensitive files to an external server, which may not be appropriate for all organizations, especially those with strict data privacy and security standards. Reliance on a web-based interface and cloud-based processing can raise privacy concerns in this regard (Arntz, 2022).

2.2.2 Microsoft Defender Advanced Threat Protection (ATP)

Figure 23 ATP (India, 2023)

Microsoft Defender Advanced Threat Protection (ATP) is made to offer end-to-end defense, detection, investigation, and response in business environments. Its extensive connection with the Microsoft ecosystem which includes Windows 10, Azure, and Office 365—is one of its most notable features. For businesses that already use Microsoft products, this integration makes deployment and management simple (India, 2023). Microsoft Defender ATP is able to conduct real-time threat detection and offer insights that are continuously updated based on the most recent threat information by utilizing the massive volumes of telemetry data gathered from millions of devices. Through this collaboration, enterprises may use Microsoft's cloud infrastructure for threat hunting and advanced analytics.

The strong detection capabilities of Microsoft Defender ATP are yet another important feature. To find possible risks, the platform combines threat intelligence, cloud security analytics, and behavioral sensors. For example, Windows 10's behavioral sensors gather and process behavioral signals from the operating system, which are subsequently subjected to advanced analytics methods on the cloud. This strategy assists in identifying sophisticated threats that conventional signature-based detection techniques could miss. Furthermore, the platform's ability to adjust to new and emerging threats is ensured by the ongoing updating of threat intelligence, offering a dynamic defense against cyberattacks.

When it comes to threat analysis and response capabilities, Microsoft Defender ATP shines. It provides security experts with a range of cutting-edge tools to carry out thorough investigations and proactively look for dangers. The effort on security teams is greatly decreased by features like automated investigation and remediation, which analyze warnings automatically and take predetermined action to eliminate dangers. Microsoft's comprehensive threat intelligence and machine learning algorithms steer these automated procedures, guaranteeing prompt and efficient responses. Additionally, the platform offers comprehensive threat assessments that include details on the type of threat, the attackers' tactics, methods, and procedures (TTPs), and suggested countermeasures. This comprehensive data improves the organization's overall security posture and helps in the prompt response of problems (What is Microsoft Defender Advanced Thread Protection? | Chorus, 2023).

Finally, a uniform security administration interface is provided by Microsoft Defender ATP, which is essential for efficient security operations. The centralized view of security events, warnings, and incidents provided by this interface makes it possible for security teams to plan ahead and react more quickly. Through the use of APIs, the platform's integration capabilities go beyond the Microsoft ecosystem and provide compatibility with a wide range of third-party security products and systems. Because of this adaptability, businesses may incorporate Microsoft Defender ATP into their current security systems and tailor their protection plans to suit particular requirements. Furthermore, the platform is compatible with multiple operating systems, including iOS, Android, Linux, and Windows, and it offers cross-platform protection, making it an adaptable option for a variety of system contexts.

2.2.2.1 Weaknesses of Microsoft Defender Advanced Threat Protection (ATP)

Microsoft Defender ATP's reliance on the Microsoft ecosystem is one of its obvious weaknesses, which might limit its usefulness and usability for businesses that utilize a wide range of non-Microsoft products. Enterprises that depend on a wider range of operating systems and software providers may encounter difficulties with the deep integration with Windows 10, Azure, and Office 365, despite providing flawless operation for those operating within the Microsoft environment. While Microsoft Defender ATP is compatible with macOS, Linux, Android, and iOS, its full potential and key features are frequently best seen in a system that is predominately Microsoft. This may result in decreased efficiency or more

difficult deployment procedures in IT environments that are heterogeneous and frequently use non-Microsoft technologies (Aviiko, 2021).

The potential for significant expenses related to extensive use of Microsoft Defender ATP is another flaw. Significant license costs are frequently associated with the platform's vast capabilities and advanced features, especially for bigger companies that need extensive coverage across several endpoints. The requirement for qualified staff to efficiently manage and optimize the system might also raise the overall cost. The financial barrier to entry for Microsoft Defender ATP may be fairly high, even though it delivers advanced analytics and automated remediation for smaller enterprises or those with tighter IT budgets. This might force businesses to balance the costs and benefits, which could restrict their ability to utilize Microsoft Defender ATP's whole feature set.

2.2.2.2 Strengths of Microsoft Defender Advanced Threat Protection (ATP)

Microsoft Defender ATP's tight integration within the Microsoft ecosystem, which offers unmatched synergy and seamless functionality across Microsoft products, is one of its main advantages. This connection improves the platform's capacity to quickly identify and address threats by enabling extensive data sharing and real-time threat intelligence updates. Microsoft Defender ATP provides a holistic security solution that leverages current infrastructure and investments for businesses that already use Windows 10, Azure, and Office 365. This eliminates the complexity and expense of implementing and operating different security products (Siosulli, 2024). A proactive security posture and more accurate threat detection are made possible by the capacity to leverage vast amounts of telemetry data from the Microsoft network. This guarantees that possible threats are promptly discovered and countered.

The platform's sophisticated threat hunting and automated response features are yet another important asset. By providing security teams with advanced capabilities for thorough analysis and proactive threat hunting, Microsoft Defender ATP helps them find and eliminate threats before they have a chance to do serious damage. By automatically assessing alarms and implementing remedial measures based on specified criteria and comprehensive threat intelligence, automated investigation and remediation features lessen the workload on

security staff. This automation guarantees consistent and prompt reactions to security issues in addition to improving operational efficiency. Additionally, enterprises may improve their entire security strategy and resilience against future attacks by utilizing the platform's comprehensive threat reports and analytics, which offer actionable insights into the nature and impact of threats (Sandhu, 2023).

2.2.3 CylancePROTECT

Figure 24 CyclaneProtect (CylanceProtect overview, 2022)

CylancePROTECT is a next-generation antivirus and endpoint protection solution that offers enhanced threat prevention features. Via the application of predictive AI models, CylancePROTECT employs real-time malware identification and blocking in contrast to conventional antivirus solutions that depend on signature-based detection techniques. The software can identify and neutralize threats, including as polymorphic malware and zero-day exploits, before they have a chance to trigger thanks to this technique. Without frequent updates or internet connectivity, the AI-driven model learns and adapts continuously, improving its capacity to recognize novel and emerging threats. This makes it incredibly effective offline.

One of CylancePROTECT's distinguishing features is its small size and low influence on system efficiency. Conventional antivirus programs frequently use a large amount of system resources, which causes decreased productivity and poorer performance. On the other hand, CylancePROTECT is built to function effectively with the least amount of resources, guaranteeing that endpoints continue to be responsive and functional. This efficiency improves user experience and lessens overall stress on IT infrastructure, which is especially helpful for enterprises with a high endpoint count. It is also the best option for settings where

sustaining high performance is essential, such the financial services or healthcare industries, due to its minimal resource consumption (Key features of CylancePROTECT Desktop, n.d.).

CylancePROTECT can identify and thwart threats at different phases of the attack lifecycle, hence demonstrating strong threat prevention capabilities. The system offers thorough defense against a variety of cyberthreats by recognizing and eliminating risks at the pre-, execution, and post-execution phases. This multi-layered strategy makes sure that even in the unlikely event that a threat gets past one line of security, it will probably be detected by another. Moreover, CylancePROTECT is appropriate for endpoint protection in isolated or secure situations where continuous connectivity is impractical due to its ability to operate efficiently even in the absence of internet connectivity (Admin, 2024).

Lastly, to further strengthen the endpoint protection package, CylancePROTECT offers options for script management and device control. Device control features help stop data leakage and illegal data transfers by enabling businesses to regulate and limit the use of peripheral devices like USB drives and external storage. Script-based attacks, which are frequently used to get beyond conventional security measures, can be monitored and controlled with the help of script management tools. CylancePROTECT lowers the risk of insider attacks and unintentional data breaches by helping enterprises implement security rules and maintain a safe endpoint environment by giving them granular control over these areas.

2.2.3.1 Strengths on CyclanePROTECT

CylancePROTECT is a strong option for endpoint security because of its many advantages. Its proactive and anticipatory approach to threat prevention is one of its main advantages. By utilizing the power of artificial intelligence and machine learning, CylancePROTECT can identify and prevent both known and new threats in real-time, without depending on traditional signature-based detection approaches (Pros and Cons of BlackBerry Protect (CylancePROTECT) 2024, 2023). This gives enterprises a robust defense against new cyberthreats since it allows for the prevention of even zero-day assaults and malware versions that have never been seen before. Without affecting system performance, the solution's unparalleled speed and scale of file and behavior analysis guarantees that endpoints are kept safe from a variety of malicious activities.

CylancePROTECT is a strong option for endpoint security because of its many advantages. Its proactive and anticipatory approach to threat prevention is one of its main advantages. By utilizing the power of artificial intelligence and machine learning, CylancePROTECT can identify and prevent both known and new threats in real-time, without depending on traditional signature-based detection approaches. This gives enterprises a robust defense against new cyberthreats since it allows for the prevention of even zero-day assaults and malware versions that have never been seen before. Without affecting system performance, the solution's unparalleled speed and scale of file and behavior analysis guarantees that endpoints are kept safe from a variety of malicious activities.

2.2.1.3 Weaknesses on CyclanePROTECT

Well CylancePROTECT has many advantages, there are also some drawbacks that businesses need to be aware of. One significant drawback is that it is vulnerable to evasion tactics used by sophisticated adversaries. The AI-driven method of the solution does a great job of recognizing known and unknown malware based on static properties, but it might have trouble identifying threats that use more complex evasion techniques, such variations of polymorphic or fileless malware. In order to avoid being discovered by conventional endpoint security solutions, these kinds of threats can alter their behavior or appearance, which could allow them to get past CylancePROTECT's safeguards and infiltrate endpoints. Therefore, to reduce the chance that advanced threats would go unnoticed, enterprises might need to add to CylancePROTECT with further security measures like network-based detection or behavioral analysis.

CylancePROTECT's dependence on cloud-based threat intelligence updates and management is another flaw in the system. Although cloud-based security systems have several advantages, such as centralized management and real-time threat updates, they also come with dependence on cloud infrastructure and internet access. Updating threat intelligence and interacting with the central administration console may provide difficulties for CylancePROTECT in settings with spotty or nonexistent internet connectivity, such as isolated or air-gapped networks (Jr, 2021). Reliance on cloud infrastructure also raises possible privacy and compliance issues because it may be necessary for enterprises to give sensitive endpoint data to unaffiliated cloud providers. These factors might restrict

CylancePROTECT's suitability in some settings or sectors where data sovereignty regulations or connection limitations are stringent (CylanceProtect overview, 2022).

Table for the systems Comparison

Feature	VirusTotal	Microsoft Defender ATP	CyclanePROTECT
Detection Method	Multi-AV scanning, file analysis	AI-driven threat detection, behavioral analysis	AI-driven threat prevention, ML models
Integration	Web-based interface, API access	Deep integration with Microsoft ecosystem	Standalone solution, cloud-based management
Resource Usage	Minimal system impact	Moderate system impact, optimized for Windows 10	Minimal system impact, lightweight design
Threat Prevention	Reactive, based on existing signatures	Proactive, identifies and blocks unknown threats	Proactive, stops both known and unknown threats
Deployment	Cloud-based, accessible via web interface	Integrated with Windows 10, Azure, Office 365	Cloud-based, centralized management
Support	Limited support, community-driven	Comprehensive support, enterprise-grade support options	Comprehensive support, enterprise-grade

Scalability	Scalable, suitable for small to medium-sized organizations	Scalable, suitable for large enterprises	Scalable, suitable for diverse environments
Network Connectivity	Requires internet access for cloud-based scanning	Cloud-based, requires internet connectivity	Requires internet access for updates
Advanced Features	Limited advanced features, primarily focused on file analysis	Advanced threat hunting, automated response	Advanced AI-driven threat prevention

Summary

Three well-known endpoint protection systems VirusTotal, Microsoft Defender ATP, and CylancePROTECT can be compared, and different features stand out in terms of usefulness and implementation. Although it has a simple online interface and API access, VirusTotal, which is mainly recognized for its multi-AV scanning and file analysis capabilities, mainly depends on pre-existing signatures for threat identification. Conversely, Microsoft Defender ATP distinguishes itself by offering sophisticated threat detection via AI-driven techniques and behavioral analysis, all while being deeply integrated into the Microsoft ecosystem. Its Windows 10 environment-optimized design and mild system effect make it an attractive option for businesses using Microsoft's product line. CylancePROTECT, on the other hand, has a lightweight design and no influence on the system because of its proactive approach to threat avoidance, which is powered by AI and machine learning models. It can be used in a variety of situations because of its cloud-based management, which provides centralized control even though it functions as a standalone solution.

CylancePROTECT, on the other hand, has a lightweight design and no influence on the system because of its proactive approach to threat avoidance, which is powered by AI and

machine learning models. It can be used in a variety of situations because of its cloud-based management, which provides centralized control even though it functions as a standalone solution. That being said, Microsoft Defender ATP runs in the cloud and also needs internet access for updates, unlike VirusTotal and CylancePROTECT, which also require it. All things considered, even if every endpoint security system has pros and cons, businesses can choose an endpoint protection solution that best suits their needs and preferences—whether that be advanced threat prevention capabilities, resource efficiency, or seamless interaction with current infrastructure.

2.3 Technical research

2.3.1 Hardware

Figure 25 Hardware

Using a Gaming desktop PC and a Alienware laptop together will provide your machine learning project plenty of processing power and flexibility. Renowned for its sturdy performance and gaming-grade hardware, the Alienware laptop provides portability and top-tier features ideal for effectively executing machine learning and data analysis algorithms. Having an Intel Core i7 or other multi-core processor and at least 16GB of RAM or more, this laptop can quickly compute sophisticated machine learning models even while dealing with enormous datasets.

Furthermore, the Alienware laptop's solid-state drive (SSD) storage enables quicker program execution and data access, improving overall efficiency during the development and

experimentation stages (Dell Alienware m15 R7: full specs, tests and user reviews, n.d.). The Alienware laptop offers a portable platform for developing, testing, and optimizing machine learning algorithms thanks to its elegant appearance and robust hardware setup.

In addition to the laptop, the desktop PC can be relied upon as a dependable workstation for computationally demanding activities like machine learning. Resource-intensive algorithms run smoothly on a desktop PC outfitted with comparable or better specs, such as a multi-core CPU such as an AMD Ryzen 7 or an Intel Core i7, enough of RAM (such as 16GB or more), and roomy storage. The desktop PC's bigger form size makes it possible to modify and customize it to fit changing project requirements, which makes it a great option for extended data analysis and model training sessions.

Combining Alienware laptop and desktop PCs yields a flexible configuration for your machine learning projects that offers a balance between portability and performance, as well as adaptability for different computing demands and development circumstances.

2.3.2 Software

An operating system that is suitable, such Windows, macOS, or other Linux versions, is necessary to let my machine learning project work. The foundation required to operate the relevant software tools and libraries smoothly is provided by these operating systems. Any operating system provides a suitable setting for data analysis and machine learning tasks, regardless of preference: Windows' intuitive interface, macOS's dependability and performance, or Linux's versatility and customization possibilities.

There are two main Integrated Development Environments (IDEs) available for coding and development: Jupyter Notebook (IPYNB) (Team, 2015) and Visual Studio Code (VS Code). Microsoft created VS Code, a small yet effective code editor with extensive extensions for data analysis and machine learning as well as built-in support for Python programming (Staff, 2024). Because of its user-friendly interface, wealth of customization possibilities, and integrated terminal, it's a well-liked solution for developers looking to write and debug Python code quickly.

However, Jupyter Notebook offers an interactive computing environment that is perfect for recording research findings, building machine learning models, and conducting exploratory data analysis (Driscoll, n.d.) Jupyter Notebook facilitates collaboration and reproducibility in projects by enabling the creation and sharing of documents with live code, equations, visualizations, and narrative text. Its web-based interface and compatibility with other computer languages, such as R, Julia, and Python, make it an effective tool for academics on my final year project.

The advantages of either IDE can be utilized by integrating Visual Studio Code and Jupyter Notebook into your machine learning process (Code, 2021). To write and debug code, manage project files, and access a variety of extensions for Python development, use Visual Studio Code. In the interim, experiment with machine learning techniques, explore data interactively using Jupyter Notebook, and produce educational reports that blend code, graphics, and textual explanations. When combined, these software solutions offer an all-encompassing and effective setting for working on any machine learning project.

2.3.2.1 Visual Studio

Figure 26 VSCode (Code, 2021)

I have decided to utilize Visual Studio (VS) as my main integrated development environment (IDE) for my Final Year Project (FYP), which is centered on creating a reliable malware detection system utilizing machine learning techniques. Microsoft's Visual Studio is an all-inclusive integrated development environment (IDE) that supports numerous programming languages, including Python, which is essential to my project. With capabilities like syntax highlighting, code restructuring, and IntelliSense for code completion, the IDE

provides a robust code editor. These features will guarantee a smooth and effective development process by greatly increasing my productivity and lowering the possibility of coding errors.

Debugging and testing the machine learning algorithms used to detect malware in Portable Executable (PE) files and URLs is one of the most important parts of my research. Because of its sophisticated testing and debugging capabilities, Visual Studio shines in this domain. To efficiently identify problems, I can watch variables, set breakpoints, and examine the call stack. Furthermore, to confirm that my detection algorithms behave as intended under a variety of circumstances, I need to validate their functionality and accuracy, which can only be done with Visual Studio's support for unit testing. This will assist me in guaranteeing my system's dependability and efficiency.

Another essential part of handling the source code for my project is version control. Git and Visual Studio work together seamlessly, letting me manage branches, keep track of changes, and work with others when needed. This connection makes it possible for me to streamline the development process and have an orderly codebase. In addition, the Visual Studio Marketplace's extensibility enables me to expand my development environment by adding new tools and plugins designed specifically for Python, data science, and machine learning. I will have access to extra resources and features thanks to these extensions, which will be quite helpful for the duration of the project.

Working together with friends is essential, particularly when seeking advice or collaborating with peers. Multiple users can modify and debug code simultaneously during real-time collaborative coding sessions thanks to Visual Studio's Live Share functionality. Having quick access to peer and instructor support and feedback will be made possible by this feature. Furthermore, strong cloud services are available because to Visual Studio's interface with Azure, Microsoft's cloud platform. My malware detection system will perform better when I use Azure's machine learning services, data storage, and computational resources. This will allow me to process large amounts of data and do complex machine learning tasks with ease. With this integration, I'll be able to take advantage of cloud resources to increase the performance and scalability of my project.

To sum up, Visual Studio offers an extensive feature set and a wide range of tools that are ideal for my FYP. It's a great option because of its powerful code editor, sophisticated testing and debugging features, smooth version control integration, and flexibility via plugins. Additionally, Azure's cloud capabilities and real-time collaboration greatly improve its usefulness for creating an advanced malware detection system. By streamlining my development process and ensuring that my project is finished successfully and efficiently, using Visual Studio will ultimately help my FYP succeed.

2.3.2.2 Jupyter Notebook

Figure 27 Jupyter Notebook (Driscoll, n.d.)

Furthermore Using the Anaconda distribution, Jupyter Notebook is my main tool of choice for constructing a strong malware detection system for my Final Year Project (FYP), which is centered on machine learning approaches. I can create and share documents with live code, equations, graphics, and narrative text using the open-source web tool Jupyter Notebook. For my project, which entails in-depth data analysis, preprocessing, and the use of machine learning techniques to identify malware in Portable Executable (PE) files and URLs, this interaction is essential.

Figure 28 Anaconda Prompt (Anaconda, 2018)

The capacity of Jupyter Notebook to offer an interactive computing environment is one of its main features. For iterative development and experimentation with various machine learning models, including XGBoost, Random Forest, Support Vector Machine (SVM), Decision Tree, and Logistic Regression, this functionality is especially helpful. Because Jupyter Notebooks are interactive, I can test and see the results of my code instantly, which helps me better understand how changes to the code effect the results. This speed facilitates algorithm optimization and debugging, which eventually results in more precise and effective malware detection models.

An extensive collection of tools and libraries for data science and machine learning are provided by Jupyter Notebook's connection with Anaconda, which increases its usefulness. The Python and R programming languages are available for free and open-source under the Anaconda distribution, which has more than 1,500 packages appropriate for scientific computing. Installing necessary libraries like Scikit-learn, NumPy, Pandas, Matplotlib, and Seaborn is made simple by the Anaconda Navigator, which also streamlines package management and deployment. These libraries are essential for statistical analysis, data processing, and the development of intricate malware detection process visualizations. My development environment is consistent and all dependencies are efficiently managed because I use Anaconda.

The ability to fully document my code and findings thanks to Jupyter Notebook's support for LaTeX and markdown is another important advantage. This ability is crucial to my FYP since it will enable me to properly communicate my research techniques and findings through detailed and comprehensive documentation. Jupyter Notebook is a great tool for producing

intricate project reports and presentations since it can integrate code, graphics, and narrative text into a single document. It guarantees that my work is well structured and comprehensible for myself as well as any potential reviewers.

Jupyter Notebooks also make it simple to exchange and publish content, which promotes feedback and teamwork. This feature is especially helpful for academic work because the development process heavily relies on instructor comments and peer review. Sharing my notebooks allows me to get helpful feedback and ideas to make my work better. Furthermore, Jupyter Notebooks are adaptable to a range of presentation requirements, as they may be transformed into HTML, PDF, and slideshow formats. This adaptability guarantees that I can convey my results to various audiences in the most efficient manner.

In conclusion, Jupyter Notebook, which serves as the foundation for Anaconda, offers a strong and adaptable environment for my FYP on machine learning-based malware detection. Complex machine learning projects may be developed and presented with ease thanks to its interactive computing capabilities, extensive data science toolkit, and first-rate documentation features. The easy management of all required libraries and dependencies is made possible by the integration with Anaconda, which promotes a productive and seamless development process. This combination will help my FYP succeed overall by greatly improving my capacity to create a complex and reliable malware detection system.

2.3.2.3 Python

Figure 29 Python (Yegulalp, 2023)

Data analysis, machine learning, and the creation of prediction models are all based on Python, a flexible and popular computer language. An increasing number of data scientists, academics, and developers now use Python due to its straightforward syntax, wide library

support, and vibrant community. Compatibility with contemporary development processes and libraries is ensured by the many enhancements and new capabilities offered by Python 3.x, the most recent version of the language (Yegulalp, 2023).

The Anaconda Distribution offers a complete solution to simplify the creation and management of Python environments for data science and machine learning projects. Anaconda makes it easier to install, manage, and update Python packages and dependencies by acting as both an environment manager and a package manager. More than 1,500 open-source tools designed with data science, machine learning, and scientific computing in mind are included in this carefully curated collection. These packages include all of the necessary libraries, including NumPy, Pandas, Matplotlib, Scikit-learn, and TensorFlow, so users can do a number of different kinds of activities, like data visualization and manipulation as well as model training and deployment (Online, 2023).

Additionally, Anaconda provides a graphical user interface called Anaconda Navigator that makes it easy to browse, install, and run Python packages and applications (Anaconda, 2018). Additionally, it offers tools for setting up and maintaining virtual environments, enabling users to separate dependencies within projects and prevent conflicts between various versions and packages. Python users can increase productivity, improve workflow efficiency, and concentrate on resolving challenging data-related problems instead of worrying about dependency management or incompatibilities by utilizing the Anaconda Distribution.

2.3.2.4 Machine Learning Libraries

In order for data scientists and machine learning practitioners to create reliable and efficient models for a variety of applications, machine learning libraries are essential. Scikit-learn is a library of this type that is notable for its ease of use and effectiveness in data mining and analysis. Scikit-learn offers a wide range of techniques that can be used for dimensionality reduction, regression, clustering, and classification. Both novices and seasoned practitioners will find its user-friendly interface and copious documentation to be highly beneficial, enabling them to confidently and easily deploy machine learning solutions.

XGBoost is a highly optimized distributed gradient boosting package that is well-known for its speed and performance, in addition to Scikit-learn. XGBoost is a popular tool for both regression and classification problems. It is particularly good at managing big datasets and producing cutting-edge predictive modeling outcomes. Because of its sophisticated algorithms and effective implementation, it is the go-to option for real-world and competitive applications where efficiency and accuracy are crucial. Through the use of XGBoost, professionals can create machine learning models that are dependable and incredibly accurate by utilizing ensemble learning approaches.

Also, Deep learning frameworks like TensorFlow (Shivanandhan, 2023) or PyTorch provide unmatched flexibility and scalability for more complex machine learning jobs (Simplelearn, 2023). These frameworks give developers access to a wide range of building and training tools and APIs for neural networks, empowering them to take on challenging tasks in domains including reinforcement learning, natural language processing, and image recognition. While TensorFlow has a thriving community and broad support for production deployment, PyTorch is a preferred option for researchers and developers because it provides a more dynamic and intuitive approach to model construction. By using PyTorch or TensorFlow (Deb, 2017), practitioners can push the limits of machine learning and fully utilize the promise of deep learning (Bourke, 2023).

Figure 30 PyTorch (Simplelearn, 2023)

Figure 31 TensorFlow (Shivanandhan, 2023)

2.3.2.5 Additional Libraries

From my research, NumPy (Developers, 2022) is a key library for Python that offers crucial functionality for matrices and multidimensional arrays. NumPy is a valuable tool for data manipulation and computing in many scientific disciplines, including machine learning. It allows users to efficiently perform numerical operations, manage arrays, and build mathematical functions.

Pandas is a robust data analysis and manipulation package designed specifically for working with structured data, which is a great addition to NumPy (W3.CSS, 2024). Pandas offers a wide range of functions and methods for working with numerical tables and time series data, as well as user-friendly data structures like DataFrame and Series. Pandas makes data wrangling simpler and insights production faster, whether you're cleaning up messy data, aggregating data, or doing exploratory data analysis.

Figure 32 seaborn (geeksforgeeks, 2023)

Figure 33 Numpy (Developers, 2022)

Figure 34 pandas (W3.CSS, 2024)

The most popular Python charting libraries for data visualization are Matplotlib and Seaborn, which let users generate a variety of static, interactive, and publication-quality representations (geeksforgeeks, 2023). Users may construct a wide range of plots with Matplotlib (schools, 2024), which is well-known for its adaptability and customization possibilities. These include line plots, scatter plots, histograms, and more. Seaborn, which is based on Matplotlib, is perfect for creating intricate and visually appealing plots with less code since it provides a higher-level interface and more features for statistical visualization.

Last but not least, JupyterLab offers an optional but very helpful interactive development environment for Jupyter Notebooks, which are commonly utilized in workflows related to data science. JupyterLab improves upon the classic Jupyter Notebook experience by offering a more feature-rich interface and better usability. Code, graphics, and descriptive prose can all be easily combined in an interactive document with JupyterLab, which makes data science projects more collaborative, repeatable, and iterative to produce. When utilized independently or in combination with additional libraries, JupyterLab provides an adaptable setting for investigating data, analyzing it, and creating models (Jupyter, 2024).

XGBoost, Random Forest, SVM, Decision Tree, and Logistic Regression methods can be used to efficiently construct and implement machine learning models for my project by utilizing the hardware and software resources mentioned above, in addition to the designated libraries and tools.

3.0 Chapter 3

3.1 Methodology

3.1.1 Agile Methodology

Figure 35 Agile Metho (Laoyan, 2024).

Throughout the course of a project, agile methodology fosters continuous improvement and is a dynamic and collaborative approach to software development that can adjust to changing requirements. Agile's primary focus is on customer-centricity, flexibility, and responsiveness with the goal of quickly and iteratively delivering value to stakeholders. Using Agile techniques can be quite beneficial for a Final Year Project (FYP) in terms of efficiently managing the development process and guaranteeing alignment with changing project goals.

I can break down projects into smaller, more manageable tasks or user stories, each of which represents a distinct functional unit or deliverable, by adopting Agile principles. Teams are free to concentrate on delivering the most valuable features early in the development cycle since these tasks are prioritized according to their significance to project goals and customer needs. Agile also promotes regular teamwork and communication with stakeholders and end users, which helps to create a common understanding of project needs and speed up feedback loops (Laoyan, 2024).

Agile work iteratively to create incremental updates or prototypes through iterative development cycles known as sprints. These are then reviewed and rated by stakeholders. Teams may spot any possible problems or changes in requirements early on thanks to this iterative strategy, which enables early and frequent feedback gathering. Agile teams can adjust to changing project dynamics, enhance their workflows, and produce high-caliber software solutions that successfully satisfy stakeholders by embracing a culture of continuous improvement. In the end, FYP teams can successfully traverse complexity, reduce risks, and accomplish my project goals by implementing Agile approach (Sacolick, 2024).

3.1.2 Justification

The implementation of agile methodology in software development projects such as full retrospective projects is supported by multiple arguments. First of all, Agile's gradual and iterative methodology fosters adaptability and flexibility, enabling me to successfully respond to shifting priorities and objectives. Within an FYP context, where project goals and

requirements may change over time, Agile offers a framework for effectively handling these adjustments. Agile helps me to deliver value gradually by dividing the project into smaller, more manageable iterations, which reduces the risks involved with extensive development projects.

Second, Agile places a strong emphasis on communication and teamwork between me, stakeholders, and end users. During the development process, Agile promotes ongoing participation and input by cultivating a culture of transparency and shared ownership. This collaborative approach allows for fast modifications to the project scope, priorities, and deliverables while guaranteeing that the project stays in line with my expectations. The focus on cooperation that Agile places on projects can improve project visibility and stakeholder satisfaction in an FYP setting, where stakeholders may include academic advisers, project supervisors, and possibly end users or beneficiaries (kissflow, 2023).

Agile also encourages the early and regular release of functional software, which helps me to verify hypotheses, get input, and adjust course as necessary. Its iterative delivery strategy finds problems early in the development process and allows for quick response and adaption, which lowers the risk of project failure. Agile's emphasis on producing measurable results iteratively can assist assure project success and reduce the possibility of last-minute surprises or setbacks for a fixed-fee project (FFP), when project durations may be constrained and stakes are high.

Ultimately, the Agile methodology fosters a culture of ongoing learning and progress in my process of growth. I may find areas for growth, try out new strategies, and gradually improve my practices by routinely reflecting on my actions and results. My dedication to ongoing enhancement cultivates a feeling of responsibility and authority, empowering me to produce ever-better results. Agile's emphasis on learning and adaptation can help my professional and personal growth in an FYP setting, as I am frequently navigating new ground and gaining new skills, ultimately increasing the project's value and effect (James, n.d.).

3.1.3 Phases

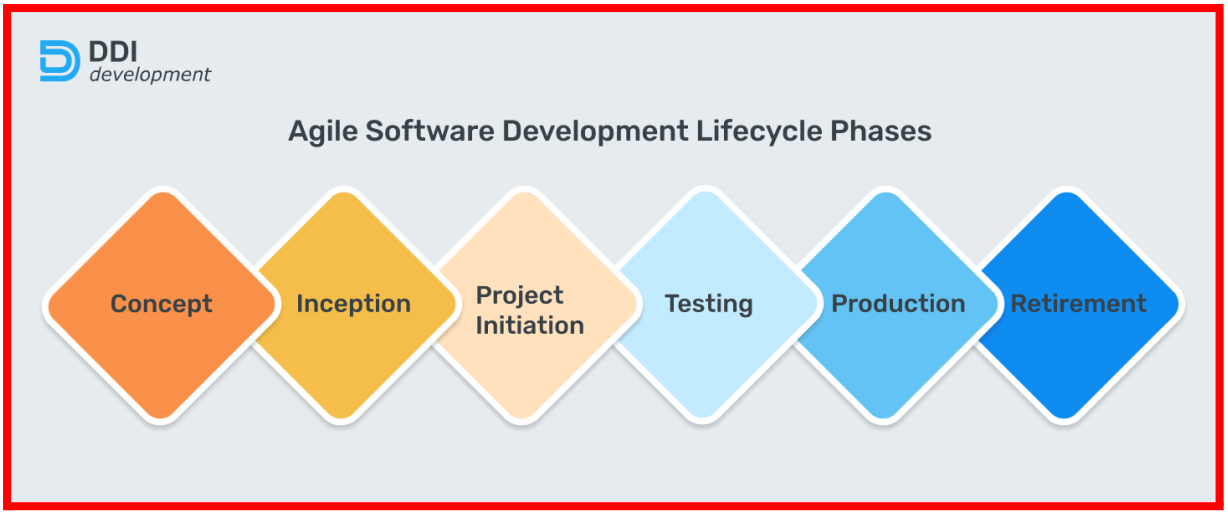


Figure 36 Phases of Agile (Dziuba, 2023).

Traditionally, software development projects are broken up into discrete phases, each with their own set of tasks and protocols. The specification of the project's objectives, parameters, and functional requirements is done during the requirements collecting phase. Interviews with stakeholders, user feedback, and comprehensive requirement documentation are all part of this phase's tasks.

The system's database architecture, user interface layouts, and architecture are designed and documented during the Design phase, which comes next. In this stage, stakeholder engagements and design reviews make sure that the project goals are in line.

The Implementation phase, which comes after Design, is when the real coding and programming work is done. Developers translate design specifications into executable code by following coding standards and best practices. Unit testing and code reviews are crucial responsibilities at this phase to find and fix bugs early (Dziuba, 2023).

The project moves onto the Testing phase after it is implemented. The system undergoes extensive testing to make sure it satisfies all requirements and performs as planned. Various testing procedures, including functional, integration, and system testing, are used in this phase to confirm the functionality and dependability of the system.

After testing is completed successfully, the software is either published to end users or put into a production environment during the deployment phase of the project. The purpose of

this phase is to provide a smooth transition to the new system through user training, data migration, and final acceptance testing.

The last phase is called Maintenance and assistance, and it consists of continuing work like bug fixes, maintenance, and user assistance. This stage guarantees the software product's long-term usage and stability (Comeau, 2023).

On the other hand, iterative and incremental development are prioritized in contemporary software development approaches like Agile. Agile methods use shorter development cycles, known as sprints, in which cross-functional teams work together to create usable software releases. The team and stakeholders establish the project's goals, parameters, and priorities during the first step of the process, known as planning. Tasks like sprint planning, user story definition, and backlog grooming for the next sprint are included in this phase.

The project moves onto the Execution phase after the Planning phase. In this stage, several sprints of iterative development work each lasting two to four weeks are completed, potentially resulting in a shippable product increment. Together, team members complete coding, testing, and code reviews in order to provide value to stakeholders. Daily stand-up meetings and regular sprint reviews are used to monitor development progress, address problems, and get input.

The project enters the Review and Retrospective phase following each sprint, during which the team assesses its performance, talks about lessons learned, and pinpoints areas in need of improvement. During this phase, stakeholders will be presented with the completed product increment at sprint review meetings. Additionally, retrospective sessions will be held to evaluate what went well, what may be improved, and how to streamline processes for upcoming sprints. Over time, the team's strategy may be continuously improved and adjusted thanks to these feedback loops, which raises output and quality.

Agile approaches, in general, place a high importance on adaptation, flexibility, and teamwork. This allows teams to deliver value gradually and quickly adjust to changing requirements as a project progresses

3.2 Scrum Framework

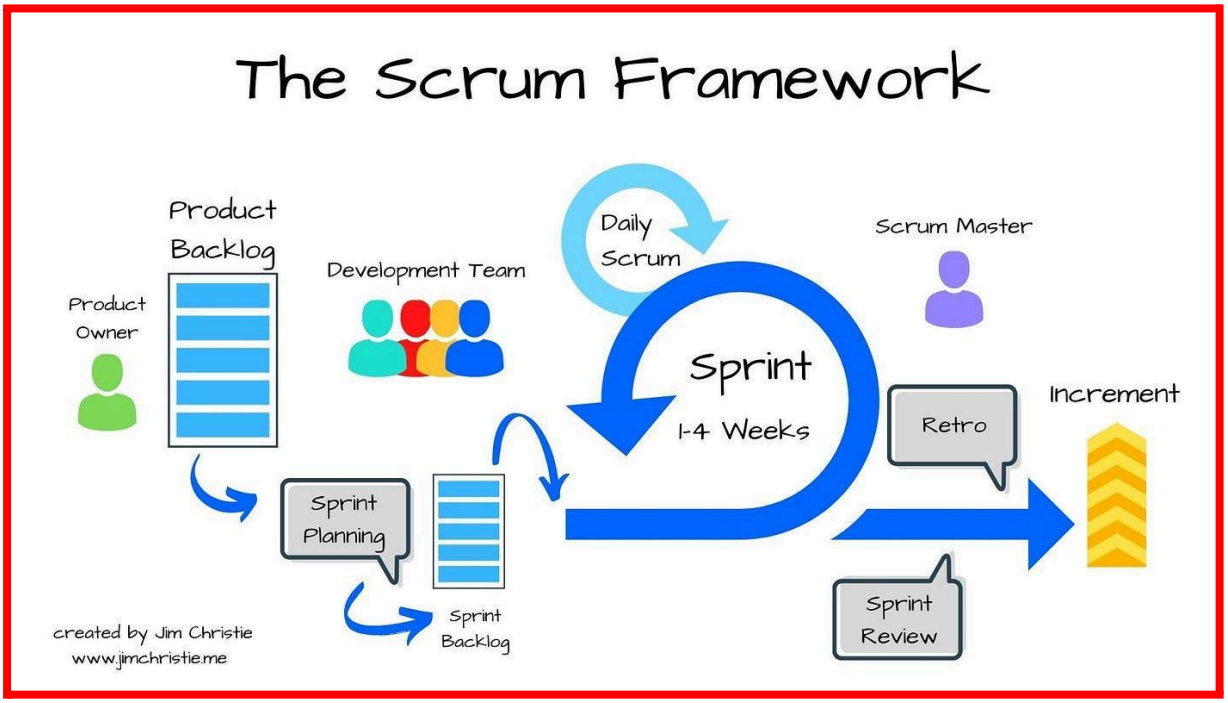


Figure 37 Scrum (DRUMOND, n.d.).

The Scrum framework places a strong emphasis on iterative development cycles, or sprints, which are usually one to four weeks long. The development team prioritizes a set of features or user stories from the project backlog and collaborates to generate a potentially shippable product increment throughout each sprint. By using an iterative approach, teams can react

swiftly to evolving requirements and feedback, providing stakeholders with frequent and incremental value updates.

Sprints, which are iterative development cycles of one to four weeks, are a fundamental component of the Scrum framework. Under the guidance of a prioritized set of features or user stories from the project backlog, the development team collaborates to deliver a potentially shippable product increment throughout each sprint. Teams may react swiftly to evolving requirements and feedback by using this iterative method, which enables them to provide stakeholders with small, regular benefits (DRUMOND, n.d.).

Scrum's emphasis on self-organizing, cross-functional teams is another essential element. With Scrum, teams are given the freedom to self-organize and make choices, and each member of the team contributes their own talents and knowledge to help the project reach its objectives. The team members' sense of accountability and ownership is fostered by this cooperative and decentralized approach, which boosts motivation, creativity, and productivity.

Moreover, Scrum uses a variety of rituals and artifacts to encourage openness and visibility across the project lifecycle. The main Scrum ceremonies that facilitate communication, teamwork, and feedback include daily stand-up meetings, sprint planning sessions, sprint reviews, and retrospectives. Scrum artifacts like burndown charts, sprint backlogs, and product backlogs also assist stakeholders in managing priorities, keeping track of work, and deciding how best to proceed with the project (Ram Srinivasan, n.d.).

All things considered, the Scrum framework offers an organized but adaptable method for developing software, allowing teams to produce value iteratively and quickly respond to shifting requirements. Scrum helps teams to maximize their efficiency and effectiveness by adopting values like transparency, cooperation, and continuous improvement. This eventually results in the successful delivery of high-quality software solutions.

3.2.1 Justification

There are various reasons to implement the Scrum framework in software development initiatives. First and foremost, Scrum encourages flexibility and responsiveness to change, which is especially important in the fast-paced and dynamic corporate climate of today. Scrum helps teams to quickly resolve new issues, reorganize work into shorter, time-boxed iterations called sprints, and incorporate stakeholder feedback. By using an iterative approach, teams may continuously provide valuable software increments while reducing the risks associated with uncertainty and changing needs.

Second, Scrum encourages cross-functional cooperation and collaboration, both of which are critical for optimizing productivity and innovation during the development process. Scrum gives self-organizing teams the freedom to decide for themselves, utilizing the variety of viewpoints and abilities among team members to efficiently tackle challenging issues. Team members become more engaged and motivated as a result of this collaborative culture's promotion of creativity, information sharing, and a feeling of shared ownership (SCRUMstudy, 2022).

Scrum also improves visibility and transparency over the course of a project, giving stakeholders information about the status, difficulties, and results of the development work. Regular practices like sprint planning, daily stand-ups, sprint reviews, and retrospectives help Scrum guarantee that the development team and stakeholders are in sync and that there is open communication. Scrum artifacts like burndown charts, sprint backlogs, and product backlogs also give stakeholders access to up-to-date project status information, facilitating resource allocation and decision-making.

Scrum also encourages a culture of learning and continual growth among the development team, which spurs constant innovation and excellence. Teams can find areas for improvement, try out novel strategies, and modify their procedures to better suit the requirements of the project and its stakeholders by routinely commenting on their methods and results (KAISPE, 2024). Teams are empowered to develop high-quality software

solutions that meet or beyond customer expectations when they are dedicated to continuous improvement, which in turn fosters an excellence culture.

So this Scrum framework provides a solid and tested method for developing software that is marked by flexibility, teamwork, openness, and constant improvement. Organizations can improve their agility, resilience, and capacity to provide value to customers in a world that is changing quickly by adopting these concepts (Nos, 2015).

3.2.2 Phases

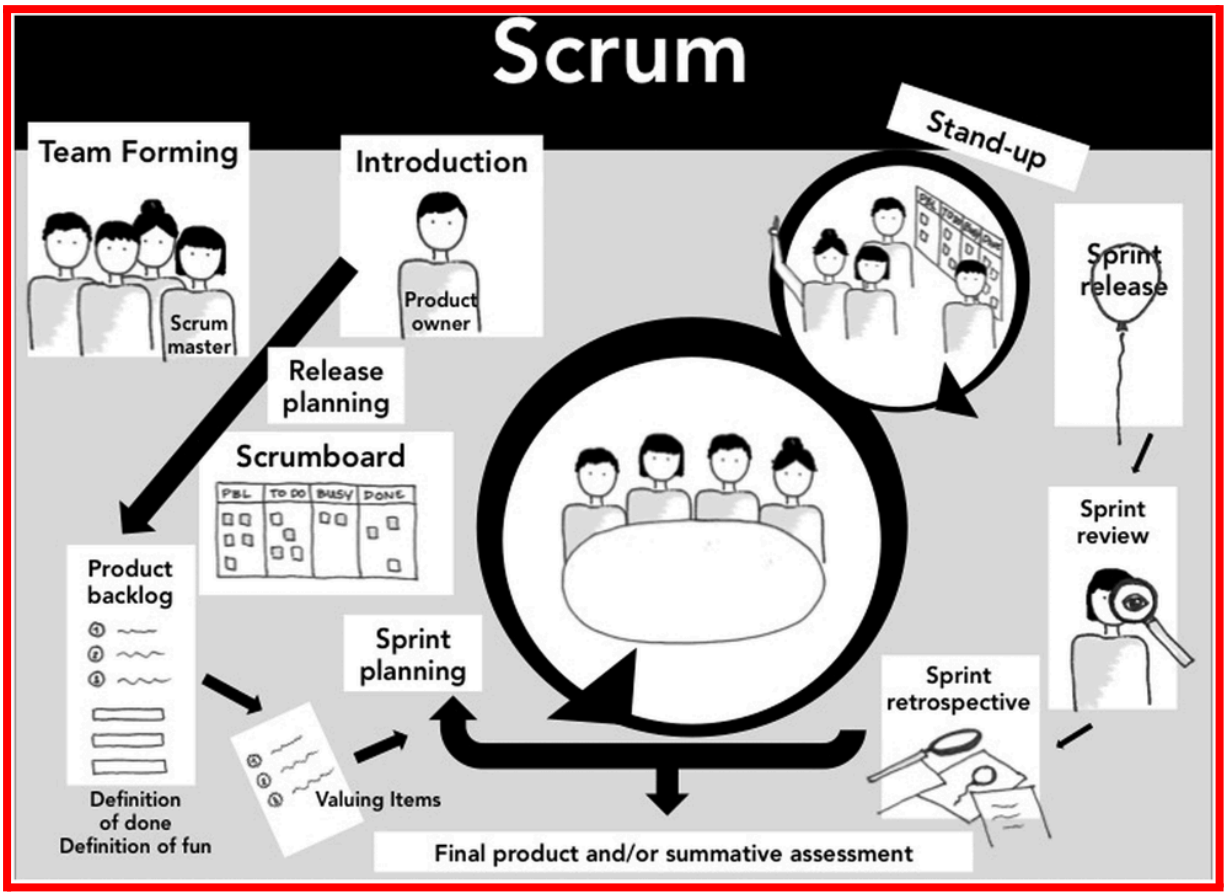


Figure 38 Phases of Scrum (Donato, 2023).

The project lifecycle is divided into multiple phases under the Scrum framework, each with its own set of tasks and procedures meant to produce valuable software increments quickly, regardless of whether it is used in a conventional or modern setting. The product owner, development team, and stakeholders work together to define the project's goals, scope, and priorities during the first phase, which is called planning. During this stage, tasks including prioritizing user stories, building a product backlog, and calculating the amount of work needed for each task are all done. To choose the things from the product backlog that will be included in the next sprint, sprint planning sessions are also held (Donato, 2023).

The project moves onto the Execution phase, which comprises of sprints—iterative development cycles, after the Planning phase. The development team collaborates to implement the chosen user stories and produce a potentially shippable product increment throughout each sprint. Every day, stand-up meetings are conducted to coordinate efforts, talk

about the status, and pinpoint any obstacles or difficulties. The team's major objectives are to do the assigned tasks, stick to the sprint target, and work at a sustainable pace throughout the sprint.

The project enters the Review and Retrospective phase when the sprint's development work is finished. The team presents the finished work to stakeholders during the sprint review meeting, gets their input, and talks about any modifications or additions that should be made to the product backlog. This feedback loop guarantees that the product is in line with stakeholder expectations and permits ongoing validation of the product. Simultaneously, the team holds a sprint retrospective meeting to evaluate their performance, pinpoint areas that need work, and suggest concrete adjustments to improve subsequent sprints (SCRUMstudy, Scrum Phases and Processes, n.d.).

The project culminates with the phase of Deployment and Monitoring. The finished product increment is either made accessible to end users for testing and validation at this phase, or it is delivered to the production environment. User acceptability testing can be carried out to get input and make sure the product satisfies the requirements. In order to acquire information for upcoming revisions and improvements, the product owner and stakeholders also keep an eye on the product's functionality and user happiness. Regardless matter whether they follow classic or current interpretations of the Scrum methodology, teams may deliver high-quality software products efficiently and effectively by using this incremental and iterative approach to development together with ongoing feedback and modification.

3.3 Devops Practice

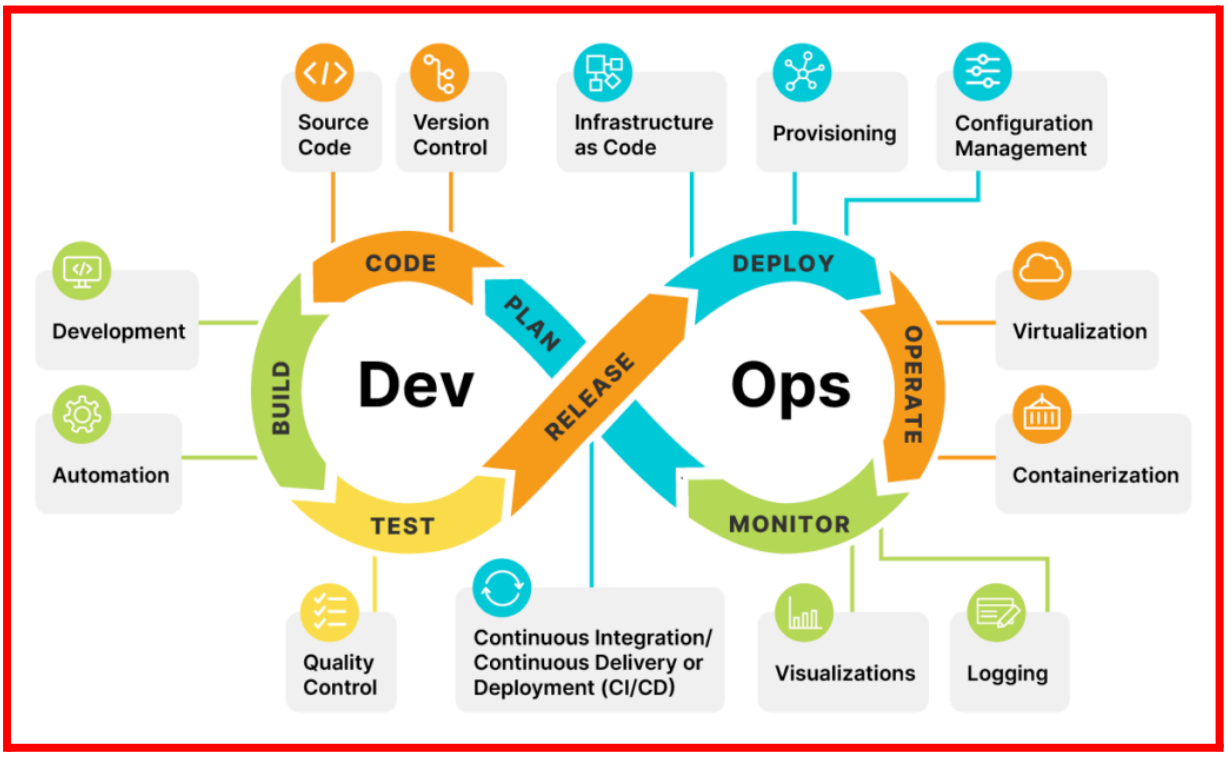


Figure 39 Devops Practice (Mohanan, 2021).

A revolutionary approach to software development and IT operations, DevOps is a portmanteau of "development" and "operations," with the goals of improving cooperation, streamlining procedures, and quickening delivery cycles. DevOps is fundamentally a cultural movement inside companies that promotes more cooperation and coordination between development teams, which make software, and operations teams, which handle its deployment and upkeep. By putting an emphasis on shared objectives, group ownership, and ongoing communication, this collaborative attitude dismantles silos and advances an integrated perspective of the software delivery lifecycle (Mohanan, 2021).

Apart from its cultural components, DevOps comprises a collection of guidelines, procedures, and instruments intended to mechanize and optimize the complete software delivery process. These procedures cover every stage of the lifecycle, from testing and code development to deployment and monitoring, and they help businesses provide value to clients faster and more consistently. Two important DevOps techniques are continuous delivery (CD), which focuses on automating the deployment process to guarantee that software can be released to production promptly and safely, and continuous integration (CI), which automatically integrates code changes into a shared repository and runs automated tests to validate them.

In addition, DevOps promotes the use of configuration management and infrastructure as code (IaC), which let businesses handle their infrastructure and configuration settings programmatically and as code. This methodology guarantees uniformity, reproducibility, and expandability in the operations of infrastructure provisioning and deployment, diminishing the likelihood of mistakes and enhancing the general dependability of software systems. DevOps emphasizes monitoring and observability to gain insights into system performance, user behavior, and business consequences. This allows teams to continuously iterate and improve. Moreover, it fosters a culture of experimentation, learning, and feedback (AWS, n.d.).

In order to deliver value to consumers more quickly and reliably, DevOps emphasizes collaboration, automation, and continuous improvement. As a whole, this marks a fundamental transformation in how organizations approach software development and operations. Organizations can get increased agility, resilience, and efficiency by adopting DevOps methods and concepts. This will empower them to innovate and contend successfully in the current dynamic and fast-paced commercial landscape (mijacobs, 2023).

3.3.1 Justification

Adopting DevOps principles provides firms looking to improve overall company agility and software delivery capabilities with a number of strong arguments. First off, by automating and optimizing the whole software delivery process, DevOps promotes a quicker time-to-market for software features and products. Organizations may adapt more quickly to shifting customer preferences and market demands by releasing new products and updates

more regularly and consistently thanks to DevOps, which integrates development, testing, deployment, and operations processes (BUCHANAN, n.d.).

Second, by dismantling conventional silos and fostering cross-functional cooperation, DevOps encourages cooperation and alignment between development and operations teams. Throughout the software development lifecycle, from planning and design to deployment and maintenance, DevOps encourages teams to work closely together by establishing a culture of shared accountability, mutual respect, and open communication. This collaborative approach promotes a sense of ownership and accountability among team members, which in turn spurs ongoing innovation and development while also increasing efficiency and productivity.

DevOps approaches also encourage automation, consistency, and repetition in deployment procedures, which helps to reduce the risks related to software updates and deployments. Organizations can lower the risk of mistakes, downtime, and service interruptions by utilizing automation tools and infrastructure as code (IaC) methods to ensure that deployments are carried out consistently and successfully across various environments. DevOps also promotes the use of feedback loops and continuous monitoring, which help teams identify problems early and take proactive measures to fix them, thus increasing the robustness and reliability of software systems (AppDynamics, n.d.).

Furthermore, by utilizing cloud computing, containerization, and microservices designs, DevOps helps enterprises to achieve higher scalability and efficiency in resource consumption. Organizations may deploy and manage applications more dynamically and efficiently by utilizing cloud-native technologies and container orchestration systems like Kubernetes, which allow resources to be scaled up or down in response to changing demand. Organizations are able to maximize resource utilization, save expenses, and provide value to customers more successfully thanks to this adaptability and agility.

To sum up, DevOps approaches provide a strong argument for businesses looking to boost creativity, increase teamwork, and improve the scalability and dependability of their software delivery procedures. Organizations can get increased agility, efficiency, and resilience by

adopting DevOps principles and utilizing automation, collaboration, and feedback systems. This will help them prosper in the quickly changing digital landscape of today.

3.3.2 Phases

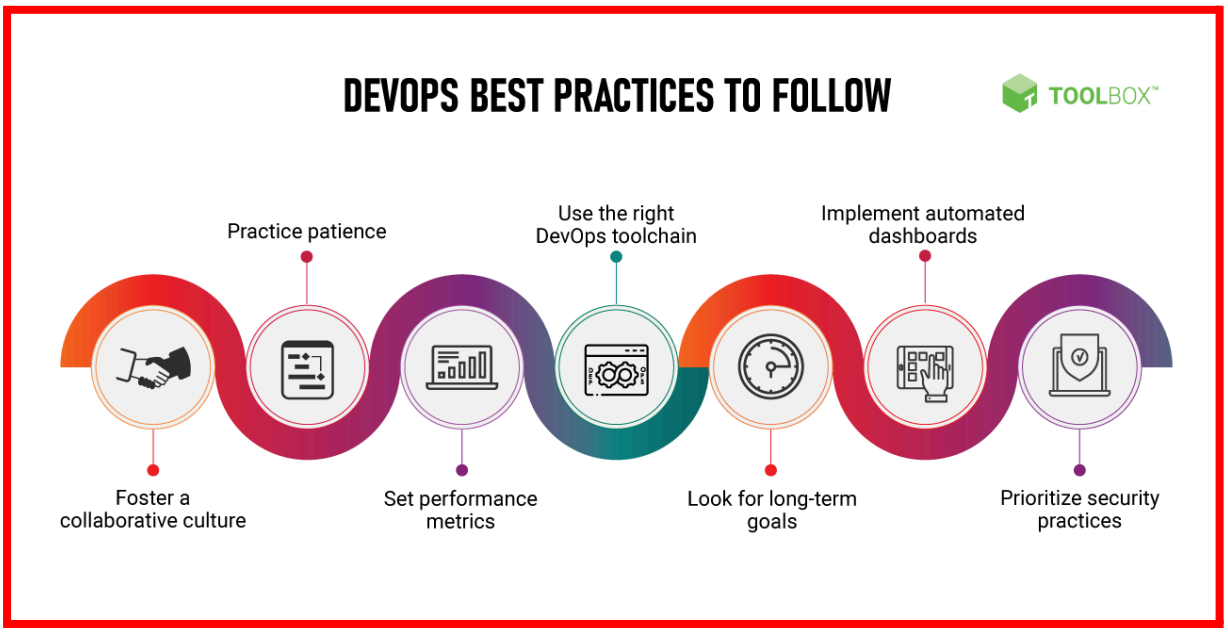


Figure 40 Phases of Devops (Ziegler, n.d.).

The software development lifecycle is split into multiple phases in the traditional DevOps technique, each with its own set of procedures and activities meant to foster cooperation, automation, and ongoing improvement. Usually, the first stage is called planning, during which stakeholders work together to determine the project's objectives, specifications, and schedule. Gathering requirements, scoping the project, allocating resources, and other tasks are part of this phase, which lays the foundation for later development and deployment operations (Ziegler, n.d.).

The project enters the Development phase after the Planning phase, during which developers write code, put features into practice, and run unit tests. Developers and operations teams collaborate closely in conventional DevOps to make sure that code modifications are consistent with the deployment procedures and infrastructure in place. Code quality is maintained throughout the development process by using continuous integration (CI)

techniques, which continuously merge code changes into a shared repository and run automated tests to evaluate them (Technologies, 2024).

After development work is finished, the project moves onto the testing phase, during which extensive testing is done to find and fix bugs, faults, and performance problems. Functional testing, regression testing, performance testing, and security testing are a few examples of testing activities. In this stage, automation is essential because it helps teams run tests fast and reliably, find problems early in the software development process, and make sure the program is up to par before it is deployed.

The project moves on to the Deployment phase, where code modifications are deployed to production or staging environments, after testing is completed successfully. The deployment process is automated through the use of continuous delivery (CD) techniques, guaranteeing that code changes may be put into production fast, consistently, and with no need for human interaction. By automating the provisioning and configuration of infrastructure resources, infrastructure as code (IaC) concepts help to ensure consistency and repeatability in deployment procedures.

The project culminates in the Monitoring and Feedback phase, which involves ongoing monitoring of the deployed apps' stability and performance. Through the use of monitoring tools and metrics, teams are able to proactively identify and address problems by gaining insights into system performance, user behavior, and business outcomes. In order to get input from end users and stakeholders for upcoming software product iterations and enhancements, feedback mechanisms like user surveys, bug reports, and stakeholder reviews are used. Organizations can continuously improve and hone their software delivery processes using this iterative, feedback-driven strategy, which fosters innovation and value creation (Bhasin, 2024).

3.4 Comparison of Methodology

Aspect	Agile Methodology	Scrum Framework	DevOps Practice
Approach	Iterative and collaborative approach	Iterative and incremental framework	Collaborative and integrated approach
Focus	Customer collaboration and flexibility	Team collaboration and sprint-based approach	Collaboration, automation, and continuous improvement
Roles and Responsibilities	Cross-functional teams with shared ownership	Defined roles (Product Owner, Scrum Master, Development Team)	Cross-functional teams with shared responsibilities
Iterations	Short iterations (sprints)	Fixed-length iterations (sprints)	Continuous iteration and improvement
Planning	Adaptive planning based on changing requirements	Sprint planning with defined goals and backlog refinement	Continuous planning and adaptation based on feedback
Feedback	Regular feedback and adaptation	Sprint reviews and retrospectives	Continuous feedback and monitoring
Tools and Techniques	User stories, kanban boards, burndown charts	Product backlog, sprint backlog, daily scrums	Automation tools, CI/CD pipelines, monitoring tools
Flexibility	Highly flexible and adaptable	Structured framework with	Flexible and adaptable, but requires cultural and

		defined roles and events	organizational changes
Complexity Management	Emphasizes simplicity and iterative development	Addresses complexity through sprint planning and backlog refinement	Manages complexity through automation, monitoring, and feedback loops

Summary

Three well-liked software development methodologies Agile Methodology, Scrum Framework, and DevOps Practice are thoroughly summarized in the comparison table. A number of important factors are taken into consideration when evaluating each methodology, such as approach, focus, roles and responsibilities, planning, feedback, iterations, tools and techniques, adaptability, complexity management, and general fit for various project scenarios.

Agile methodology emphasizes flexibility and client collaboration through an iterative and collaborative approach. It includes short iterations (sprints), adaptive planning based on changing needs, frequent feedback and modification, and the use of tools like user stories, kanban boards, and burndown charts. It also involves cross-functional teams with shared ownership. Agile is incredibly adaptive and flexible, putting an emphasis on iterative development and simplicity in order to successfully handle complexity.

On the other side, the Scrum Framework provides a more organized strategy with well defined roles and events. It entails sprints, which are set-length iterations with predetermined targets and backlog refinement, sprint reviews, and retrospectives. Product owners, scrum masters, and development teams are the usual members of scrum teams. They manage the project with the help of daily scrums, sprint backlogs, and product backlogs. Scrum offers an organized framework for handling complexity and providing value gradually, but it is less adaptable than Agile (Sosa, 2023).

The DevOps Practice centers on continual improvement, automation, and cooperation between the development and operations teams. It places a major emphasis on automation, monitoring, and feedback loops while highlighting ongoing iteration and development. Software application delivery and maintenance are shared duties by cross-functional DevOps teams. They streamline the development, deployment, and operations processes through the use of automation tools, CI/CD pipelines, and monitoring tools, resulting in faster software delivery and improved quality.

A number of criteria, including team dynamics, corporate culture, and project objectives, will determine which methodology is appropriate for your FYP. Agile Methodology can be the best choice for my project if regular adaptability to shifting requirements and close stakeholder collaboration are top priorities. The Scrum Framework, on the other hand, can work well for you if you want a more organized framework with well defined responsibilities and events. On the other hand, DevOps Practice would be the best option if my project intends for smooth cooperation, automation, and continuous improvement across development and operations. The technique that best fits my project goals and team dynamics will ultimately allow me to provide value in an effective and efficient manner.

3.2 Data Gathering Design (Survey)

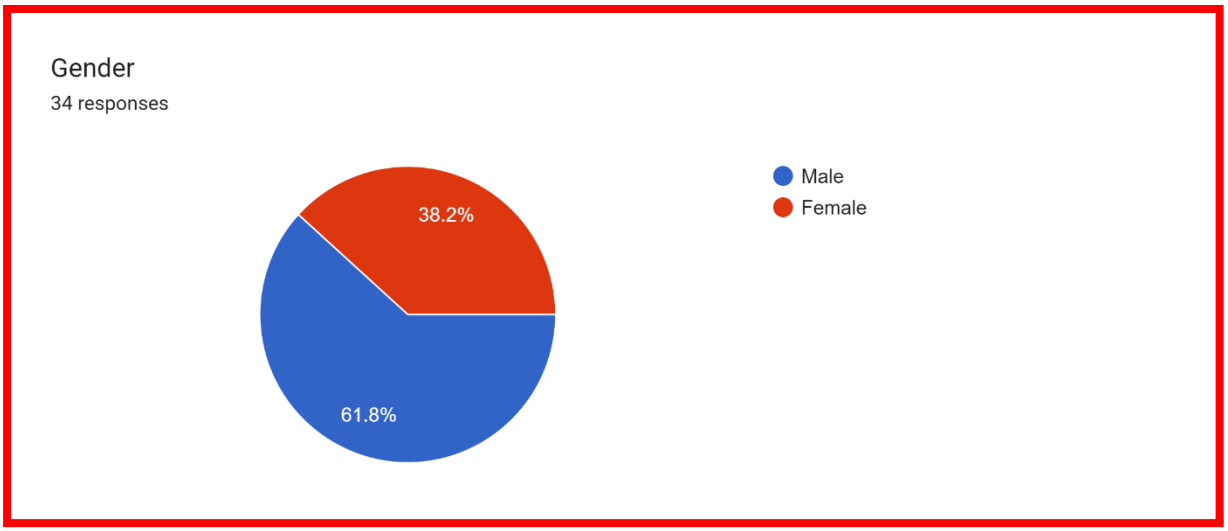


Figure 41 Gender

The distribution of respondents' genders is shown in the pie chart. Of those that took part, 38.2% identified as female and 61.8% as male. This distribution helps put the viewpoints and experiences expressed in the survey responses in context and sheds light on how gender is represented in the population that was polled. Even if men make up the majority of responders, it's vital to acknowledge the participation of women, whose viewpoints add to a more inclusive and diverse understanding of the topic.

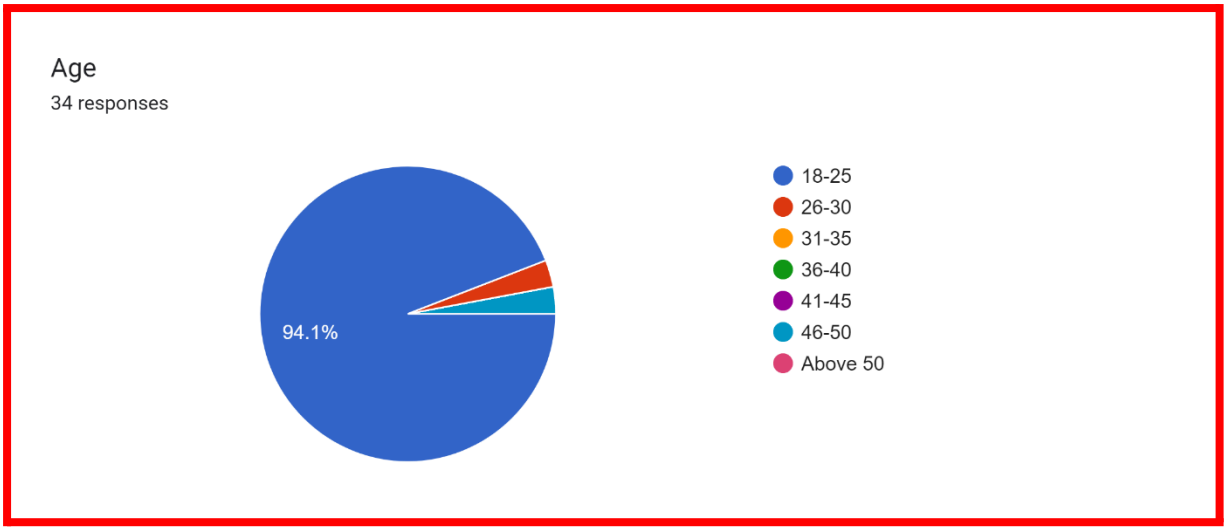


Figure 42 Age

The distribution of survey respondents by age group is shown graphically in the pie chart. With 94.1% of the respondents in this chart, the bulk of respondents are between the ages of 18 and 25. The majority of poll respondents are in this age range, which suggests that young individuals are very interested in and engaged with the topic of malware detection with machine learning techniques. On the other hand, a smaller percentage of the remaining responders are between the ages of 26 and 30. Even though this group makes up a small portion of survey respondents, their viewpoints and insights are nonetheless important for comprehending the wider demographic landscape and any differences in beliefs or experiences with malware detection.

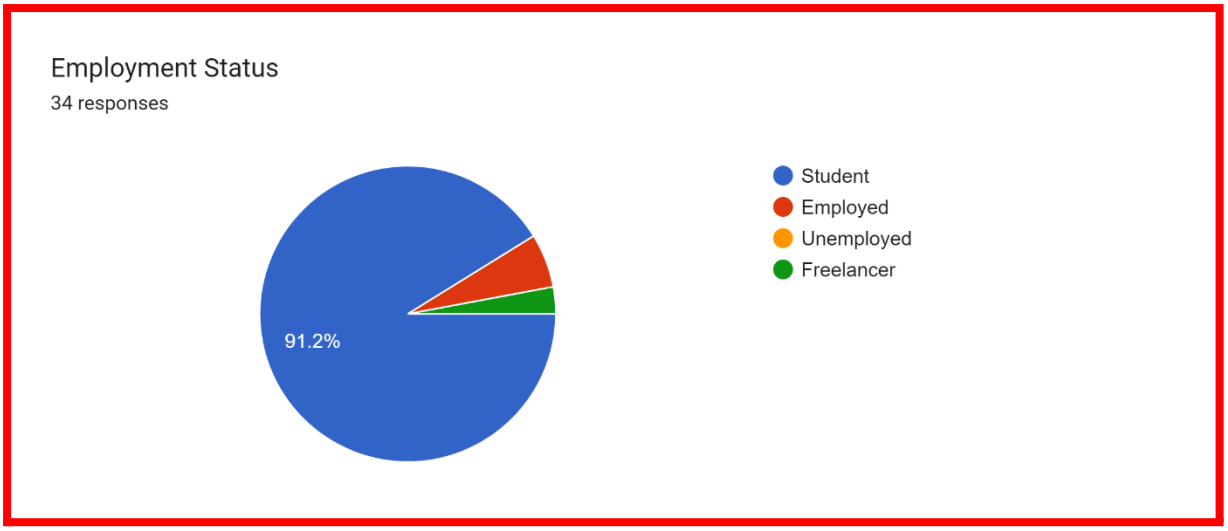


Figure 43 Employment Status

The poll respondents' employment status is represented by a pie chart, wherein 91.2% of the respondents identify as students, a lesser fraction as employed individuals, and a minority as freelancers. This distribution makes clear that most of the respondents are actively enrolled in school, suggesting that the poll is primarily intended for people who are pursuing academic or educational goals. It is important to comprehend the participants' work status since it offers insights into their backgrounds and experiences, which may affect how they see and use machine learning for malware detection.

Figure 44 What is your current fiekd or area of expertise?

The pie chart illustrating the current field or area of expertise of the survey respondents reveals a diverse range of specializations. The majority, 41.2%, are concentrated in cybersecurity, reflecting the survey's focus on malware detection using machine learning. Other fields represented include finance (11.8%), information technology (8.8%), and software engineering (8.8%). This diversity ensures comprehensive feedback and insights, enhancing the project's development by incorporating varied perspectives from different professional backgrounds.

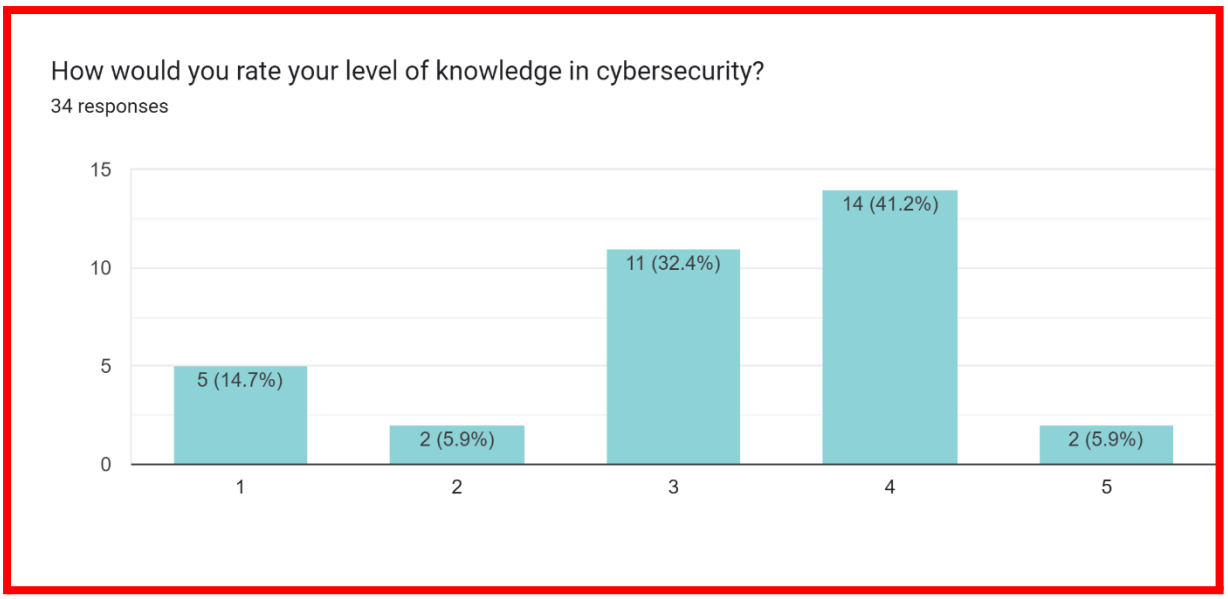


Figure 45 How would you rate your level of knowledge in cybersecurity?

The replies to the question "How would you rate your cybersecurity knowledge?" are displayed as a bar chart, which shows that participants' degrees of experience are not evenly distributed. Five people gave themselves a level 1 rating, which denotes a rudimentary comprehension. Level 2 ratings from two participants indicate a little bit greater familiarity. Eleven respondents, making up the largest group, gave their expertise a level 3 rating, indicating a moderate understanding. Two participants evaluated themselves at level 5, demonstrating advanced competency in cybersecurity, while fourteen participants rated their knowledge at level 4, showing a good degree of skill. This wide range of expertise levels offers a fair assessment of the practicality and usability of the suggested malware detection method.

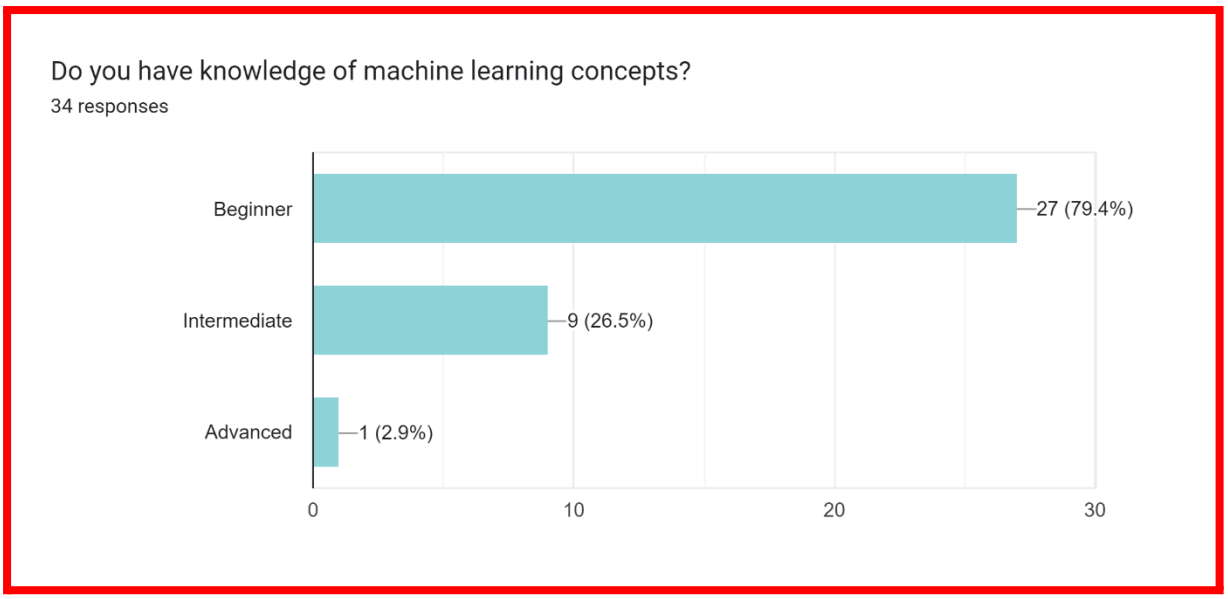


Figure 46 Do you have knowledge of machine learning concepts?

With 27 participants identifying as novices, the chart displaying answers to the question "Do you have knowledge of machine learning concepts?" reveals a notable skew towards the beginner level. This suggests that the majority of responders only have a rudimentary comprehension of machine learning ideas. Nine respondents gave their expertise an intermediate rating, indicating a mediocre degree of familiarity and comprehension. Among the responders, only one person thought they were advanced in machine learning, indicating how uncommon deep competence is. This distribution indicates that while the majority of participants are somewhat knowledgeable about machine learning, there is still much space for the group's expertise to expand in terms of depth and breadth.

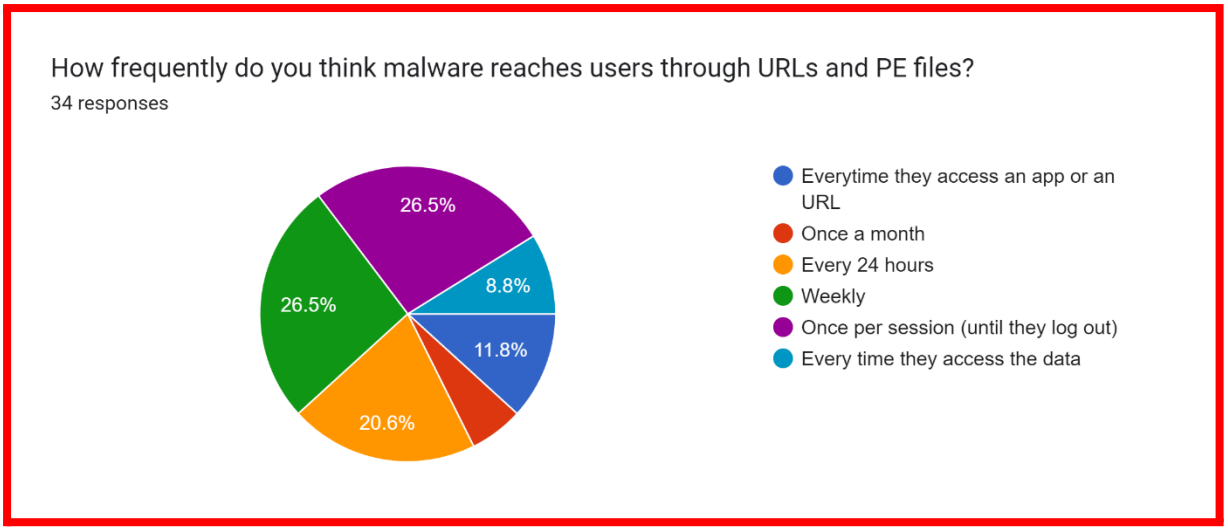


Figure 47 How frequently do you think malware reaches users through URL and PE files?

Participants' perspectives vary, as evidenced by the pie chart that shows answers to the question, "How frequently do you think malware reaches users through URLs and PE files?" Particularly concerning about ongoing exposure to dangers, 26.5% of respondents think malware only affects users once every session until they quit using their hardware. Furthermore, 8.8% believe that malware assaults happen each time data is accessed, and 11.8% link these dangers to each time a program or URL is accessed. Twenty-six percent of respondents believe that malware attacks occur every 24 hours, while another twenty-five percent believe that they occur weekly. The diversity of answers suggests that consumers' understanding and perceptions of danger about the frequency of malware exposure vary.

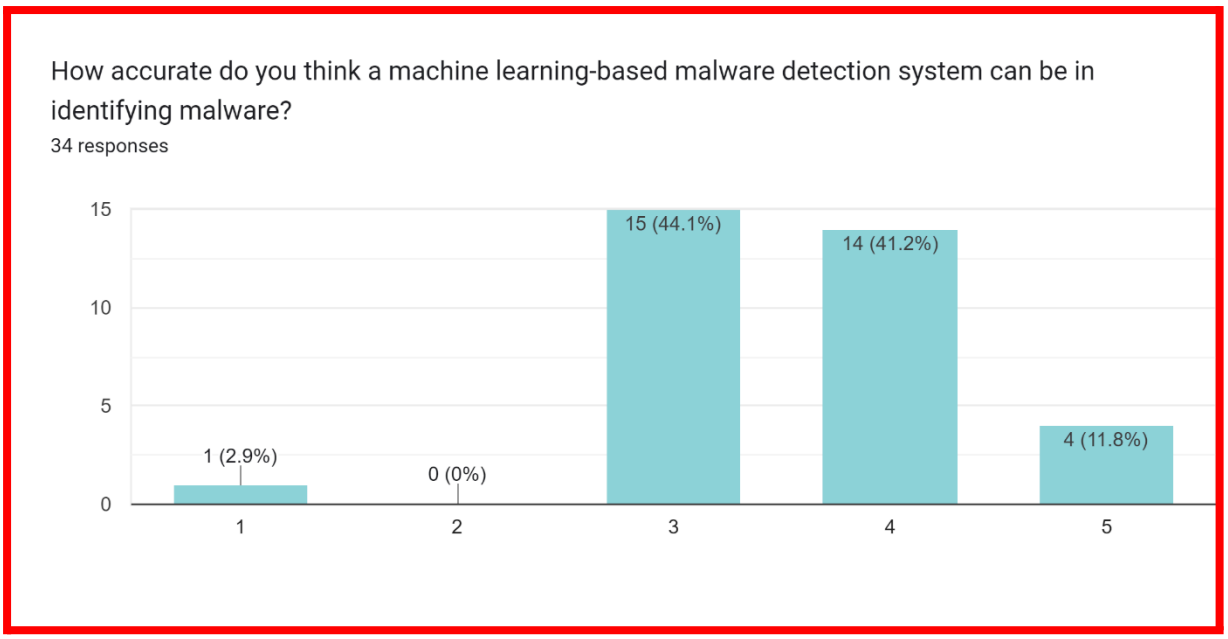


Figure 48 How accurate do you think a machine-learning based malware detection system can be in identifying malware?

The topic of the poll was "How accurate do you think a machine learning-based malware detection system can be in identifying malware?" a variety of answers, which are displayed in a bar chart. In particular, one responder gave the accuracy a level 1 rating, suggesting that they had poor hopes for the system's efficacy. The majority of responders, or 15, gave the accuracy a level 3 rating, indicating a moderate degree of confidence in machine learning's capacity to correctly identify malware. Furthermore, 14 respondents gave the accuracy a level 4 rating, indicating a higher degree of confidence in the correctness of the method. Last but not least, four respondents gave the accuracy a level 5 rating, suggesting a high degree of confidence in the efficacy of malware detection systems based on machine learning.

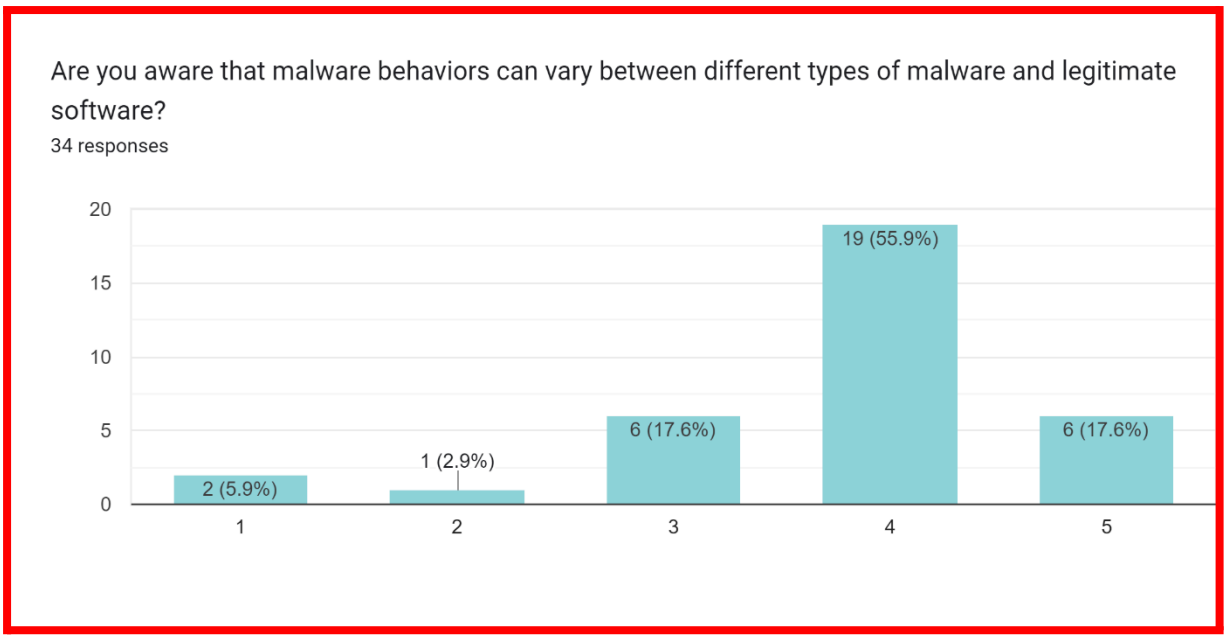


Figure 49 Are you aware thaat malware behaviours can vary between diffrent types of malware and legitimate software?

The topic of the poll was "How accurate do you think a machine learning-based malware detection system can be in identifying malware?" a variety of answers, which are displayed in a bar chart. In particular, one responder gave the accuracy a level 1 rating, suggesting that they had poor hopes for the system's efficacy. The majority of responders, or 15, gave the accuracy a level 3 rating, indicating a moderate degree of confidence in machine learning's capacity to correctly identify malware. Furthermore, 14 respondents gave the accuracy a level 4 rating, indicating a higher degree of confidence in the correctness of the method. Last but not least, four respondents gave the accuracy a level 5 rating, suggesting a high degree of confidence in the efficacy of malware detection systems based on machine learning.

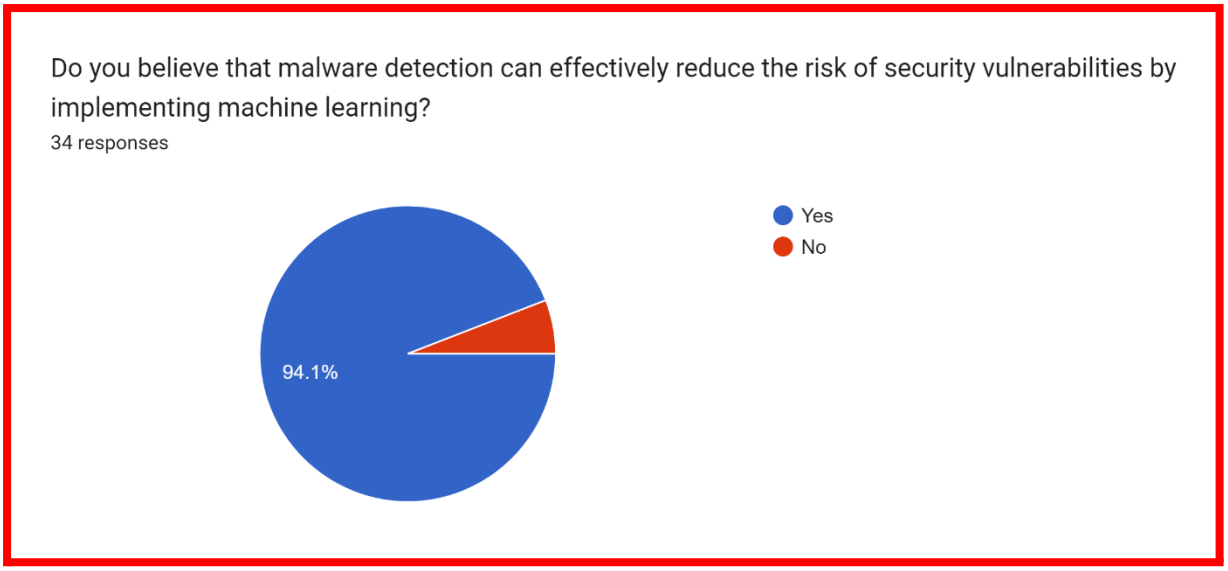


Figure 50 Do you believe that malware detection can effectively reduce the risk of security vulnerabilitites by implementing machine learning

The majority of respondents, or 94.1%, expressed confidence in the effectiveness of machine learning-based malware detection systems to mitigate security vulnerabilities in response to the question, "Do you believe that malware detection can effectively reduce the risk of security vulnerabilities by implementing machine learning?" The results of the survey were displayed in a bar chart. This high proportion suggests that participants strongly believe that machine learning techniques to detect malware can improve cybersecurity measures and lower the likelihood of security breaches brought on by malware threats.

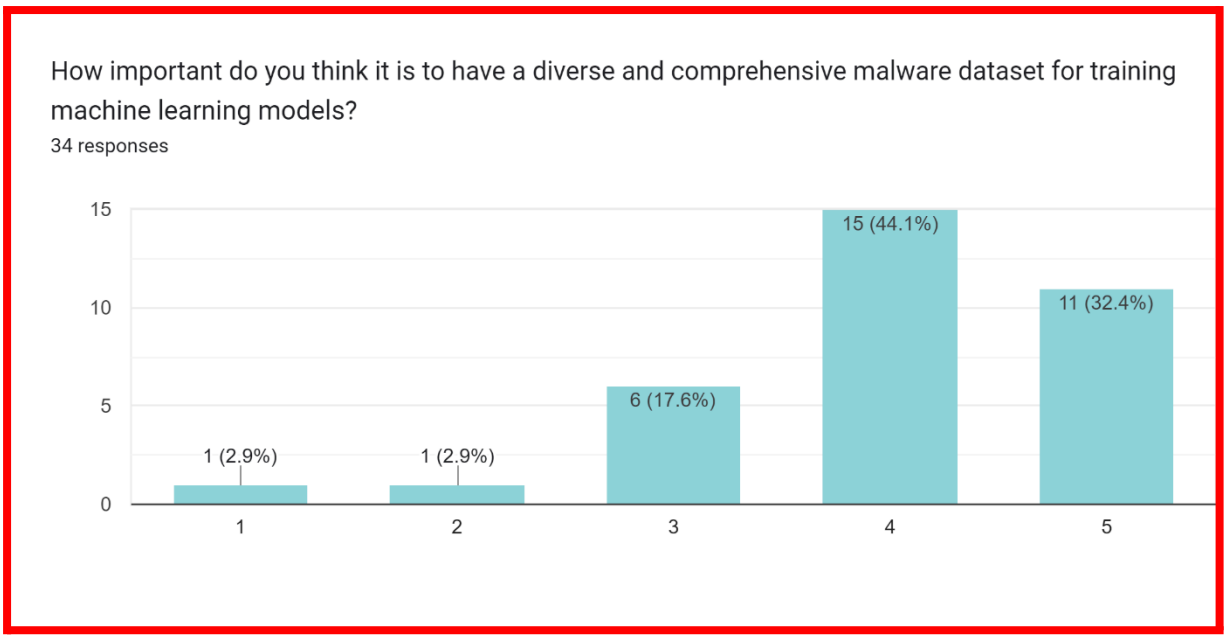


Figure 51 How important do you think it is to have a diverse and comprehensive malware dataset for training machine learning models?

The majority of respondents, or 15 people, stressed the critical importance of having a diverse and comprehensive malware dataset, rating it at level 5, in response to the question "How important do you think it is to have a diverse and comprehensive malware dataset for training machine learning models?" presented in a bar chart. Eleven more respondents shared a similar opinion, highlighting the importance of diverse datasets for efficient training of machine learning models. Six participants, or a lesser percentage, assessed the significance at level 3, indicating a moderate recognition of the significance of dataset variety. Remarkably, just one respondent per respondent, or a tiny minority, assessed the significance at level 1, suggesting that some participants do not view dataset diversity as being very important.

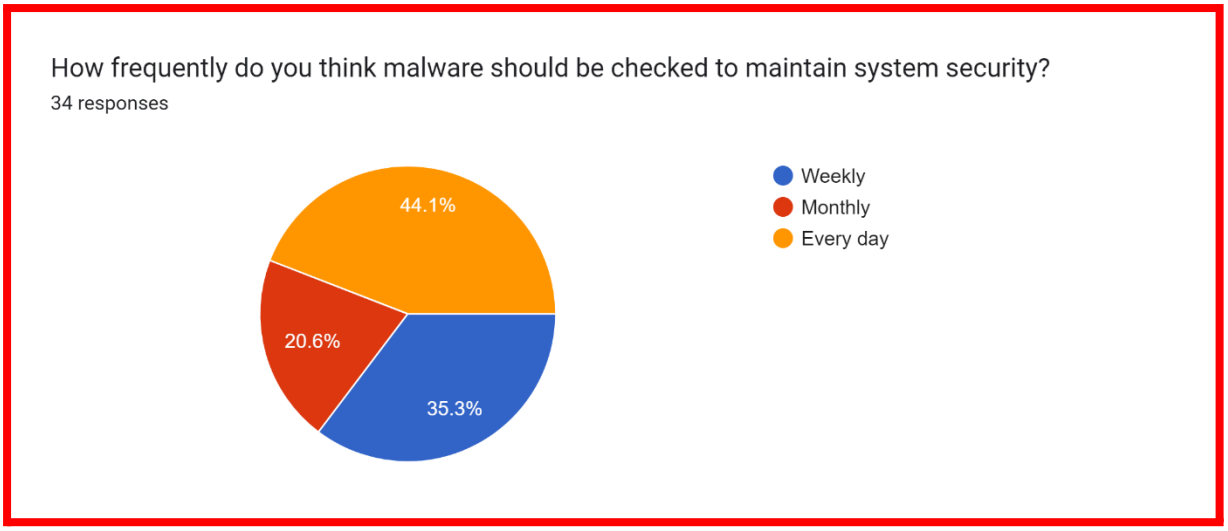


Figure 52 How frequently do you think malware should be checked to maintain system security?

The majority of participants, or 44.1%, support daily malware checks to maintain system security in response to the question "How frequently do you think malware should be checked to maintain system security?" as shown in a pie chart. Furthermore, a similar opinion is expressed by 25.3% of respondents, who stress the significance of performing daily malware checks. Nonetheless, a significant portion of responders (20.6%) recommend performing malware scans once a month, suggesting a belief in less regular intervals for monitoring. These results show that respondents generally favor routine malware monitoring, with a notable focus on daily inspections to guarantee strong system security.

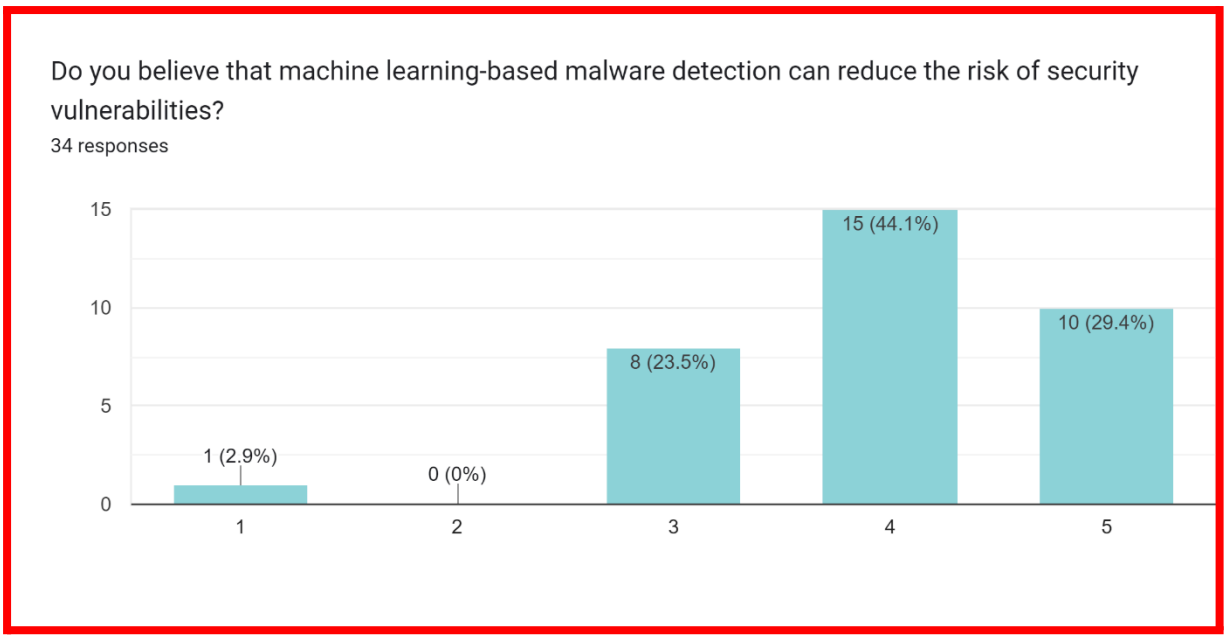


Figure 53 Do you believe that machine learning-based malware cann reduce the risk of security vulnerabilities?

The bar chart illustrates the distribution of responses to the question "Do you believe that machine learning-based malware detection can reduce the risk of security vulnerabilities?" The replies show differing levels of trust in the effectiveness of machine learning-based malware detection. The majority of participants exhibit moderate to high levels of confidence, with the minority of respondents—represented by one person—expressing poor confidence and giving a grade of level 1. In particular, 15 respondents show a high level of confidence (level 4), 10 people express the highest level of confidence (level 5), and 8 people show a moderate level of confidence (level 3). Together, these answers highlight a common perception among participants regarding the efficacy of machine learning-based malware detection as a workable method of lowering the likelihood of security flaws.

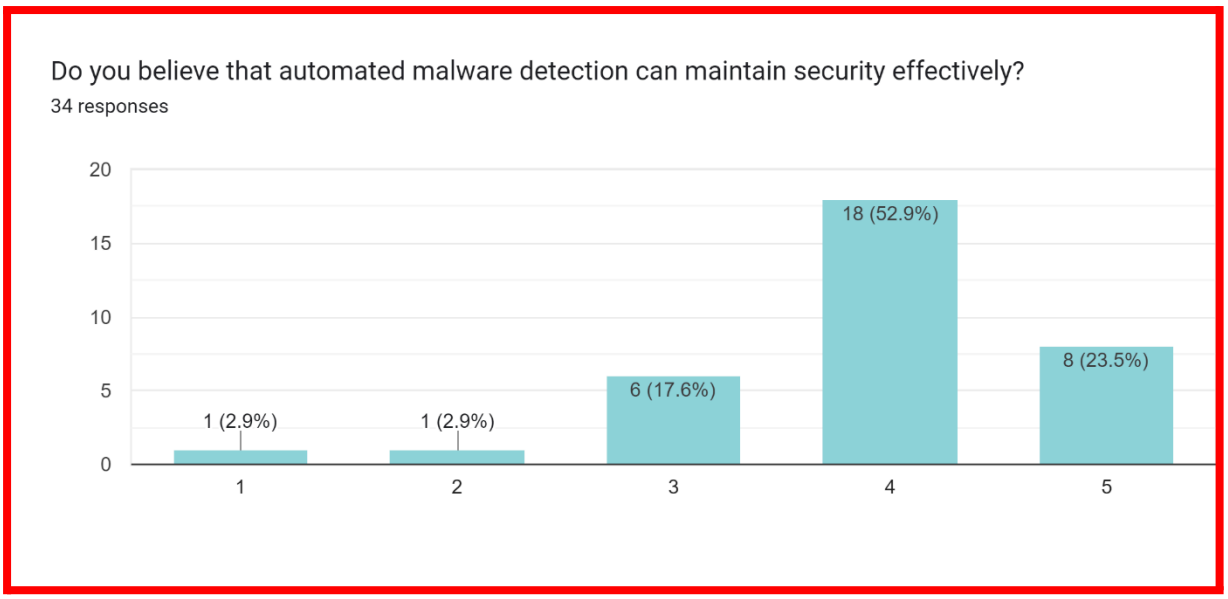


Figure 54 Do you believe that malware detection can maintain security effectively?

Respondents' opinions range across various levels of confidence when asked, "Do you believe that malware detection can maintain security effectively?" The results are displayed in a bar chart. A small minority of participants one person each express the lowest levels of confidence (levels 1 and 2), but most respondents exhibit moderate to high levels of confidence in the efficacy of malware detection for security maintenance. Six participants demonstrate a moderate level of confidence (level 3), signifying some hesitancy, while a significant proportion of participants eight in all express a high level of confidence (level 4). Moreover, eight responders express the maximum degree of confidence (level 5), indicating a firm conviction in the effectiveness of malware detection as a strategy for maintaining security. Overall, these answers show that participants generally agree about the significance and usefulness of malware detection in supporting cybersecurity efforts.

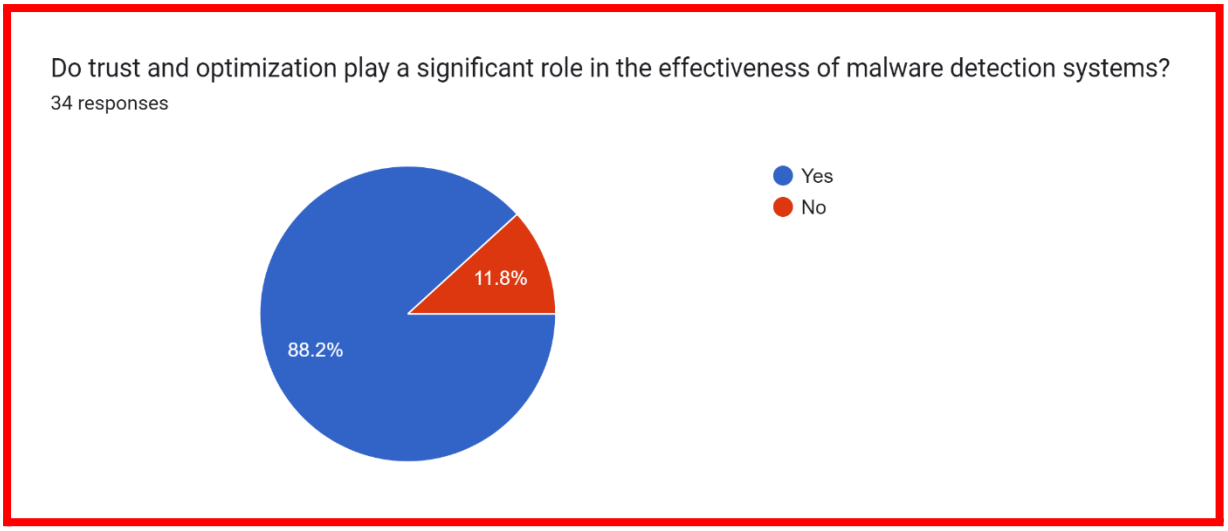


Figure 55 Do trust and optimization play a significant role in the effectiveness malware detection system?

Participants' opinions on the significance of trust and optimization in boosting malware detection system performance were shown in a pie chart style that asked, "Do trust and optimization play a significant role in the effectiveness of malware detection systems?" The vast majority of participants, or 88.2 percent, recognize the critical role that trust and optimization play in supporting the efficacy of these systems. This overwhelming confirmation highlights the understanding that reliability and optimization are important components that enhance the general effectiveness of malware detection programs. On the other hand, a smaller percentage of individuals express a different opinion, indicating that trust and optimization are not as important in determining how successful malware detection systems are. These answers, taken together, highlight the general agreement among participants on the importance of trust and optimization in improving the effectiveness and dependability of malware detection systems.

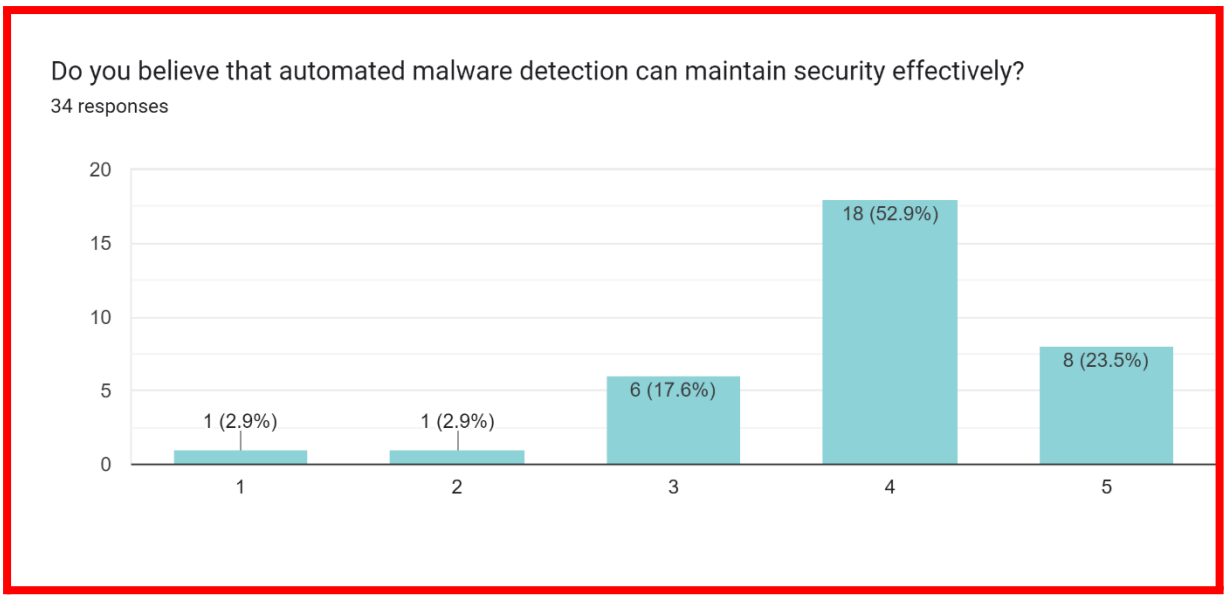


Figure 56 Do you believe that automated malware detection can maintain security effectively?

To evaluate respondents' opinions on the efficacy of malware detection in preserving security, the poll question "Do you believe that malware detection can maintain security effectively?" was shown in the form of a bar chart. The replies' distribution shows that participants had a variety of viewpoints. The respondent with the lowest level of confidence (level 1) in malware detection's ability to maintain security was one person, an insignificant percentage of respondents. In a similar vein, another responder (level 2), who likewise represented a small fraction, expressed a somewhat elevated but essentially unchanged degree of trust. On the other hand, the majority of respondents showed differing levels of confidence in their ability to detect malware; only six individuals (level 3), or a tiny minority, expressed moderate confidence. Moreover, a considerable proportion of participants, consisting of eighteen individuals (level 4), exhibited a markedly elevated degree of faith in the effectiveness of malware identification. Lastly, a sizable percentage of respondents eight at level 5 expressed the highest degree of trust, indicating a firm conviction in malware detection's capacity to successfully uphold security. This variety of answers highlights the many viewpoints that participants have on how useful malware detection is for maintaining security.

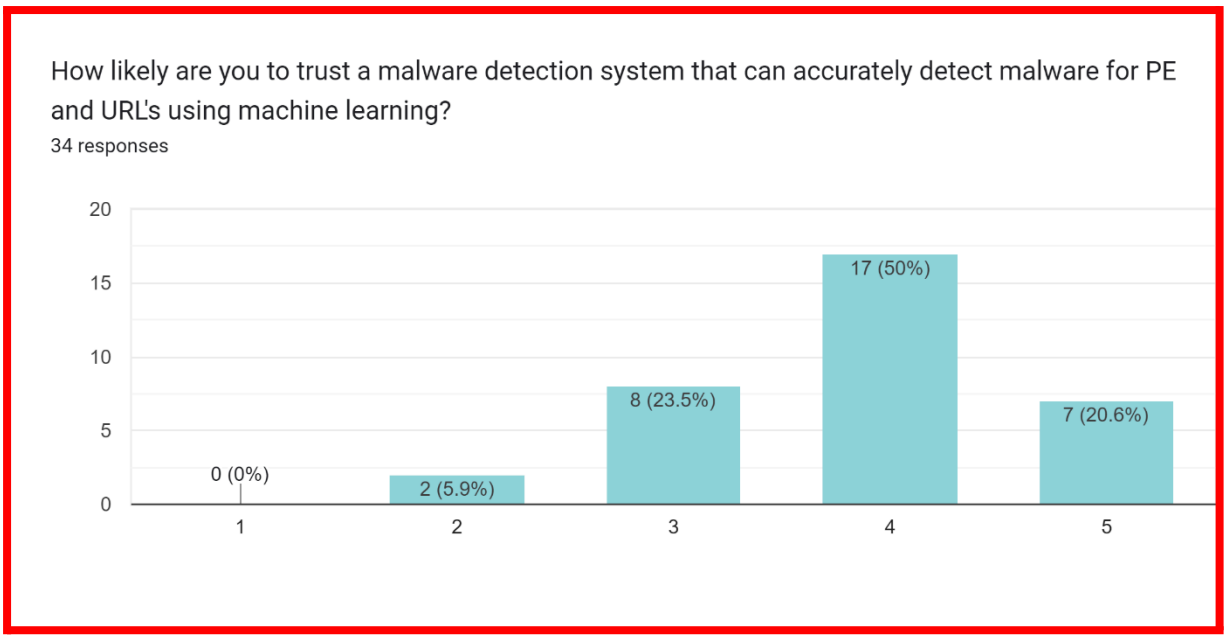


Figure 57 How likely are you to trust a malware detection system tha can accurately detect malware for PE and URL's using Machine Learning

To determine respondents' levels of trust in such a system, the poll question "How likely are you to trust a malware detection system that can accurately detect malware for PE and URLs using machine learning?" was displayed in a bar chart format. Participants' differing levels of trust are reflected in the distribution of replies. Just two respondents (level 2), or a negligible portion of the sample, indicated that they were unlikely to trust the system. In a similar vein, a tiny percentage of eight individuals (level 3) showed a somewhat higher but still moderate level of trust. On the other hand, a significant proportion of participants, consisting of 17 individuals (level 4), had a significantly greater propensity to have faith in the system, signifying a moderate to high degree of assurance. Furthermore, a substantial proportion of respondents, namely seven (level 5), indicated that they were most likely to trust the system, demonstrating a strong belief in its correctness and dependability. The diversity of opinions expressed by participants on whether or not to trust a malware detection system that uses machine learning for PE and URLs is highlighted by the range of responses.

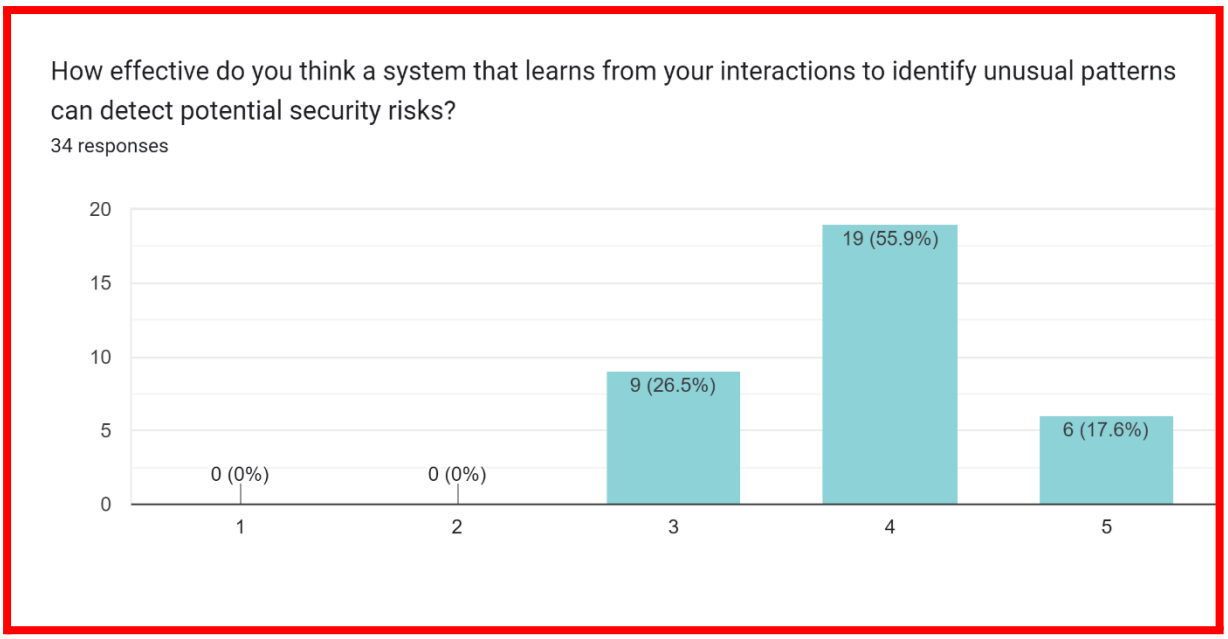


Figure 58 How effective do you think a system that learns from your interactions to identify unusual patterns can detect potential security risks?

A bar chart representing respondents' opinions of the efficacy of systems that use learning from interactions to identify unusual patterns for detecting potential security risks was used to present the question, "How effective do you think a system that learns from your interactions to detect unusual patterns can detect potential security risks?" Based on the distribution of answers, nine participants, or a significant fraction of the participants, indicated a reasonable level of confidence in such systems and rated their effectiveness at a level of three. Moreover, a larger group of responders 19 people in total showed increased confidence and gave these systems a level 4 grade for effectiveness. Furthermore, a noteworthy subgroup of participants, consisting of six individuals, exhibited an exceptionally elevated degree of confidence, designating the efficacy of these systems as level 5. All things considered, these answers point to a common belief among users about the significant efficacy of systems that use interaction learning to spot anomalous patterns in identifying possible security threats, and most of them are confident in their skills.

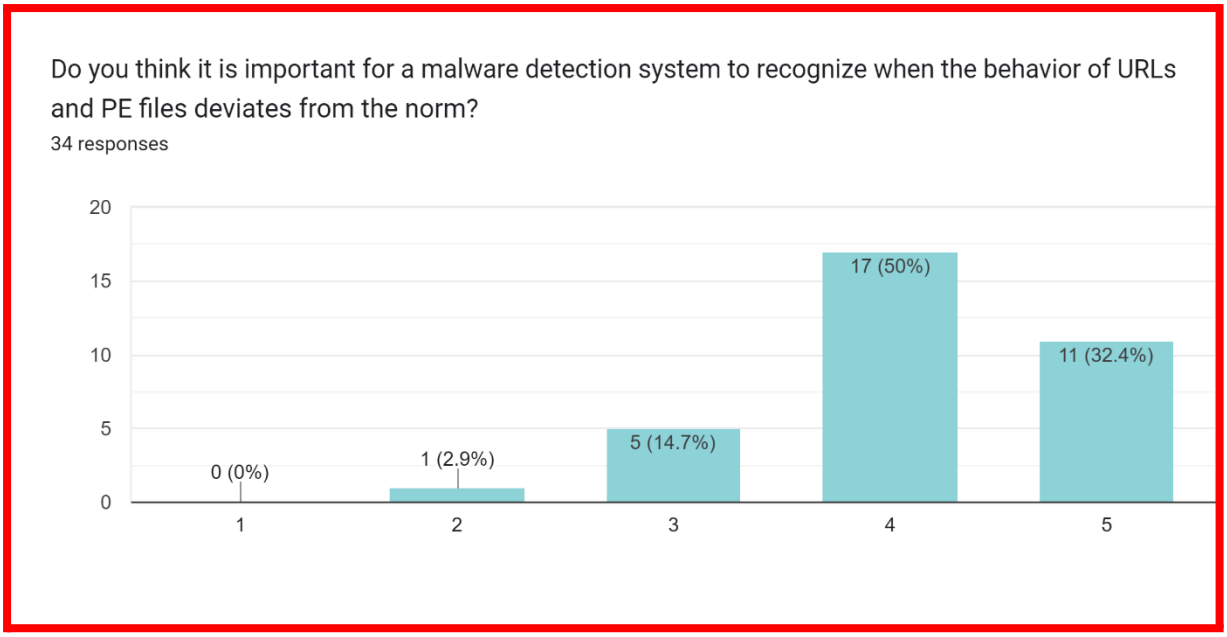


Figure 59 Do you think it is important for a malware detection system to recognize when the behaviour of URL's and PE Files deviates from the norm?

To illustrate respondents' opinions on the significance of anomaly detection in malware detection systems, the poll question "Do you think it is important for a malware detection system to recognize when the behavior of URLs and PE files deviates from the norm?" was displayed in the form of a bar chart. The responses' distribution shows that two participants, or a minority of the sample, rated the significance of anomaly detection as having a significant level of importance (level 2). Five respondents, a somewhat larger sample, rated the significance of anomaly detection at level 3, indicating a moderate-to-high level of importance. Furthermore, a significant proportion of respondents 17 people in total expressed a high degree of relevance, giving anomaly detection a level 4 grade. Furthermore, a noteworthy subgroup of participants, comprised of 11 individuals, exhibited an extraordinarily elevated degree of significance, assigning a level 5 rating to anomaly identification. All things considered, these answers demonstrate how widely participants recognize the vital role anomaly detection plays in malware detection systems, with most highlighting the role anomaly detection plays in detecting abnormalities in the behavior of URLs and PE files.

Office Record	Receipt
Date Received:	Student name: Satheish
Received by whom:	Student number: TP060754
	Received by:
	Date:

DRAFT PROJECT PROPOSAL FORM

Proposal ID :

SDG No : 9 Industry, Innovation and Infrastructure

Supervisor :
1) Dr. Aitizaz Ali - aitizaz.ali@apu.edu.my

Student Name : Satheish A/L Thiruchelvam

Student No : TP060754

Email Address : tp060754@mail.apu.edu.my

Programme Name : Bachelor of Science (Honours) in Computer Science (Cyber Security)

Title of project : Malware Detection using Deep learning

Please record which module(s) your topic is related to:

- **(CT119-3-3-VAPT)**

Vulnerability Assessment and Penetration Testing

Vulnerability Assessment and Penetration Testing

- **(CT050-3-3-PRMGT)**
  Project Management

- **(CT049-3-1-OSCA)**
  Operating Systems and Computer Architecture

- **(CT098-3-2-RMCT)**

Research Methods for Computing and Technology

Contents

Introduction

This research focuses on creating a state-of-the-art malware detection system utilizing Python and machine learning techniques including Random Forest, Support Vector Machine (SVM), and Decision Tree algorithms in response to the growing threat of malware in today's digital environment. To improve the accuracy and efficiency of malware detection, the system analyzes various feature sets that are taken from both benign software and malware samples. This allows the system to distinguish between genuine and dangerous apps. Unlike standard antivirus software, which frequently finds it difficult to keep up with the constantly changing world of malware strains, this approach overcomes such restrictions. In the end, the initiative seeks to strengthen cybersecurity protocols by offering a strong and proactive defense system against the enduring and always evolving threat that is provided by malicious software.

Project Aim

The goal of this project is to use Python machine learning techniques to create a reliable malware detection system. This method uses a variety of attributes taken from a wide dataset of malware samples and benign software in order to distinguish between dangerous software and normal apps. The project aims to improve malware detection accuracy and efficiency by utilizing machine learning. This approach aims to overcome the drawbacks of conventional signature-based antivirus systems and resource-intensive behavioral analysis techniques. Furthermore, this project is in line with Sustainable Development Goal 9 since it strengthens the resilience of digital infrastructure, advances sustainable industrialization through improved cybersecurity measures, and encourages innovation in the cybersecurity field.

In the end, the objective is to offer a dependable solution that can identify and lessen the risks caused by dynamic malware variations, thereby supporting international efforts to create resilient and safe digital environments.

Objectives

1. Gathering and Preparing a Wide Range of Malware Datasets for Machin e Learning-Powered URL and PE File Detection
2. Gather and prepare a variety of malware sample datasets for testing and training, together with legitimate software.
3. Assess the efficacy of the generated models in differentiating between legitimate software and malware by utilizing metrics like accuracy, precision, recall, and F1-score to evaluate their performance.

Problem statement

1. Malware is a serious threat to computer networks and systems.

The major challenge brought about by the quick evolution of malware and the shortcomings of conventional signature-based antivirus systems is the first issue that I would like to discuss. Signature-based methods find it difficult to keep up with the sheer number of new malware variants as cyber threats continue to multiply in sophistication and complexity (Daniel Gibert, 2020). These systems are ineffectual against zero-day assaults and polymorphic malware, which continuously mutates to avoid detection, because they rely on established patterns or signatures to identify known threats. As a result, companies are exposed to new risks and run the danger of experiencing operational disruptions, financial losses, and data breaches. Therefore, in order to mitigate the changing threat landscape and improve cybersecurity resilience, there is an urgent need for more efficient and adaptive malware detection algorithms that can reliably identify malicious software without depending simply on predetermined signatures.

2. The reliance of traditional signature-based antivirus systems on predefined patterns makes them unable to keep up with the continually changing threat field.

The second issue statement explores behavioural-based malware detection approaches' inefficiency and resource-intensiveness, which present major obstacles to real-time cyber threat detection and mitigation. Malicious activity can be difficult for traditional signature-based antivirus software to detect, but behavioural-based detection techniques try to spot it by looking for changes from the typical behaviour of the system. Unfortunately, these techniques frequently produce a huge number of false positives and demand a significant amount of computer power, which limits their applicability and efficacy in large-scale settings. (Saiiful Islam Raimon, 2022)Furthermore, the flexibility of behavioural detection systems to respond to new threats and zero-day assaults is restricted by their dependence on historical data to create baselines of typical behaviour. Organizations are consequently forced to deal with more response times and operational limitations, which makes it more difficult for them to identify and eliminate complex malware attacks before they can do harm. Thus, there is a critical need

for more effective and flexible malware detection methods that balance resource consumption and accuracy. These methods should also minimize operational overhead and allow enterprises to proactively protect against new cyberthreats.

3. It may not be feasible to identify malware in real time in large-scale environments using behavioral-based techniques due to their resource-intensive nature.

Amidst the ever-changing threat landscape, the third problem statement highlights the necessity for enhanced efficacious and efficient methods of detecting malware. The sophistication and evolution of cybercriminal tactics outpace the ability of standard malware detection techniques to detect it, leaving enterprises open to infection and exploitation. The detection of polymorphic malware variants and zero-day assaults is a limitation of signature-based antivirus systems, while being the cornerstone of cybersecurity defence. Comparably, albeit extremely resource-intensive and prone to producing false positives, behaviourally based detection techniques show promise. The problem is made worse by the increasing variety and volume of malware variants, which overwhelm security teams and call for more adaptable and scalable detection methods. As a result, there is a crucial vacuum in the cybersecurity ecosystem, and enterprises need sophisticated detection approaches that can minimize false positives and operational overhead while precisely identifying and managing emerging threats in real-time. It is critical to address this demand to effectively protect against the changing threat landscape and increase cyber resilience. (Marcus Botacin, 2021)

Literature review

1. A comprehensive survey on deep learning-based malware detection techniques

A major obstacle to modern cybersecurity is the spread of malware threats, since rogue software puts networks and digital devices at serious danger. The wide range of malware variations, which include ransomware and Trojan horses, attack both individual computers and large networks in an effort to steal money by taking advantage of security flaws. The dynamic threat landscape is difficult for traditional signature-based antivirus systems to keep up with as malware becomes more complex and cybercriminals' strategies change. As such, there is an urgent need for novel approaches to malware detection that can recognize and neutralize new threats. (M.Gopinath, 2023)

Due to the limitations of conventional detection techniques, scientists are now using cutting edge technologies like Deep Learning (DL) and Machine Learning (ML) to create more effective and flexible detection models. Researchers hope to develop models that can learn and adapt to new malware types, improving detection accuracy, by utilizing ML and DL techniques. Though ML and DL hold great potential, there are still difficulties in properly identifying next-generation malware, which uses sophisticated obfuscation methods to avoid detection by security systems. Therefore, in order to enhance malware detection capabilities, research is still needed to investigate novel methodologies that combine sophisticated algorithms with static and dynamic analytic techniques.

Advances in malware detection techniques and new trends like ransomware and advanced persistent threats (APTs) have been illuminated by recent surveys conducted in the area. There are still certain research gaps, nevertheless, especially when it comes to linking popular malware categories like file-less malware and APTs with ML and DL-based detection methods. Moreover, current studies frequently fall short of providing a thorough summary of the state of malware detection techniques today and how they relate to various malware kinds. Therefore, by providing a thorough analysis of DL-based malware detection methods and discussing their advantages, disadvantages, and application, this survey aims to close these gaps. In order to strengthen cybersecurity defenses in an increasingly hostile digital world, the study intends to

assist researchers in creating efficient mitigation solutions for both traditional and advanced malware threats through this exploration.

2. A novel deep learning-based approach for malware detection

The Internet, a vast network that links millions of computers and other devices, has grown to be a popular target for cybercriminals looking to take advantage of weaknesses for nefarious ends. Malware is one of the many cyberthreats that poses a serious threat to online safety. The term "malware," which was coined from the words "malicious" and "software," refers to a variety of code fragments that are covertly introduced into computer systems or networks with the goal of interfering with regular operations. Based on how they operate, viruses, worms, and Trojan horses are common categories of malware. Cybercriminals create malware variations using a variety of obfuscation techniques, which makes detection more difficult. Malware detectors are rendered useless by techniques like Exclusive-OR, base64 encoding, and dead code insertion, which include dangerous code into textual and binary data. The emergence of novel malware types presents a significant obstacle, since more than 450,000 new cases are documented every day. Additionally, dangerous patterns are hidden in data files on several systems, such as Windows, Linux, and Android.

Static and dynamic malware analysis approaches are the two main types. While dynamic analysis involves executing the application in a controlled virtual environment and observing its behavior, static analysis looks for dangerous patterns in the application without actually running it. Although static analysis offers valuable insights into structural features, malware variants might exploit it using their evasion techniques. Even though dynamic analysis works well, it takes a lot of time and resources, and it requires human interaction for in-depth study. Researchers have looked into hybrid techniques that combine static and dynamic analysis to improve malware detection skills in order to overcome these issues.

Conventional anti-malware systems have additional hurdles due to the exponential increase of malware varieties. Innovative methods for malware detection are required because these solutions have trouble identifying polymorphic and quickly changing malware strains. By learning from past attack patterns, machine learning (ML) has become a potential method for identifying evolving malware strains. However, polymorphic and novel malware variants are difficult to identify using ML-based methods, which has led researchers to look into other

The development of ideal ML architectures for malware detection can be automated with the

strategies. One such method is image-based malware conversion, which uses aspects that are apparent to overcome code obfuscation. Notwithstanding its potential, sophisticated texture feature extraction continues to present difficulties that impede detection processing. To address these issues, this research suggests a novel hybrid approach for malware detection that combines machine learning with deep

transfer learning. The suggested approach provides a scalable and effective way to identify malware by employing machine learning models as detectors and deep transfer learning to extract characteristics from pictures of Portable Executable (PE) files. The superiority of the suggested method over current methods is highlighted by thorough studies that use a variety of deep learning and machine learning models to show how effective it is. (Kamran Shaukat, 2023)

3. Automated machine learning for deep learning based malware detection

Innovative and advanced defense techniques are desperately needed to protect digital assets from malevolent intrusions. Depending on the use case and data availability, there are several malware analysis techniques, such as static, dynamic, and online analysis, each with unique benefits. By analyzing file properties without executing, static analysis, for example, offers speed and efficiency, while dynamic analysis offers insights into real-time activity but requires more computer resources. In contrast, online analysis provides a comprehensive view of system activity by capturing holistic operating system indicators in real-time.

According to its capacity to recognize unknown malware and acquire generalized patterns, machine learning (ML), and especially deep learning (DL), has become a viable tool for creating reliable malware detection systems. Several studies have looked into the use of ML models both conventional and deep learning-based for the detection of malware on a range of platforms and data sources. DL-based techniques do away with the need for time-consuming feature engineering by domain experts, as opposed to standard ML approaches, which frequently depend on learnt representations from raw data. But DL models especially the more intricate ones need to be fine-tuned and might not always match malware detection specifications exactly. One promising approach to improving malware detection capabilities is Automated Machine Learning (AutoML), which aims to automate the model tuning and selection process.

The development of ideal ML architectures for malware detection can be automated with the help of autoML, and these models can be improved over time as malware and data sources change. Even with all of its potential, AutoML is still in its infancy, and more research is necessary to fully understand its applicability particularly in the field of malware detection. However, the incorporation of this technology into the malware detection pipeline holds promise for democratizing the use of machine learning (ML), enabling domain experts with little background in data science to successfully leverage the capabilities of ML. AutoML is a useful tool in the continuous fight against emerging malware threats because of its capacity to decrease the number of human hours needed for model construction and tuning. (Austin L.Brown, 2024)

Deliverables

1. Utilize deep learning: models for malware detection, such as decision trees, random forests, recurrent neural networks (RNNs), convolutional neural networks (CNNs), and support vector machines (SVMs). Labelled datasets including both harmful and benign files are used to train and optimize these models.

2. Integration: "Dark Matter" program with Visual Studio Code in a fluid manner so that users may access the malware detection features right within the VS Code environment. Both the user experience and workflow productivity are improved by this integration.

3. User Guide and Documentation: Write an extensive user guide and documentation that includes installation instructions, usage guidelines, and troubleshooting advice for the "Dark Matter" application. Clearly describe the deep learning algorithms that are employed and how they contribute to malware detection.

4. Codebase and repository: Keep the "Dark Matter" application's codebase neatly structured while following coding standards and best practices. Create a version-controlled repository using GitHub or another platform to track code updates, work together with team members, and distribute projects.

5.Testing and Validation: To guarantee correctness, dependability, and robustness in a variety of scenarios, thoroughly test and validate the "Dark Matter" application. To confirm the application's functionality and performance, run unit, integration, and end-to-end tests.

6.Present and Showcase: Give a captivating presentation and showcase of the "Dark Matter" application, highlighting its functionalities, capabilities, and outcomes. Communicate the project's goals, processes, and results clearly by interacting with stakeholders, colleagues, and evaluators.

\

Fast track ethic forms

Office Record	Receipt – Fast-Track Ethical Approval
Date Received: Received by whom:	Student name:Satheish A/L Thiruchelvam Student number:TP0607541 Received by:23/5/2024 Date:

APU / APIIT FAST-TRACK ETHICAL APPROVAL FORM (STUDENTS)

Tick one box (level of study):

☐ POSTGRADUATE (PhD / MPhil / Masters)

☑ UNDERGRADUATE (Bachelors degree)

☑ FOUNDATION / DIPLOMA / Other categories

Tick one box (purpose of approval):

☑ Thesis / Dissertation / FYP project

☑ Module assignment

☐ Other: _____

Title of Programme on which enrolled ... ...

Tick one box: ☑ Full-Time Study or ☐ Part-Time Study

Title of project / assignment ... ...Machine Learning in Malware Detection For PE Files and URL's ... ...

Name of student researcher ... ... Satheish A/L Thiruchelvam... ... ...

Name of supervisor / lecturer... ... Ts. Umapathy Eaganathan... ... ...

Student Researchers- please note that certain professional organisations have ethical guidelines that you may need to consult when completing this form.

Supervisors/Module Lecturers - please seek guidance from the Chair of the APU Research Ethics Committee if you are uncertain about any ethical issue arising from this application.

		YES	NO	N/A
1	Will you describe the main procedures to participants in advance, so that they are informed about what to expect?	✓		
2	Will you tell participants that their participation is voluntary?	✓		
3	Will you obtain written consent for participation?	✓		
4	If the research is observational, will you ask participants for their consent to being observed?	✓		
5	Will you tell participants that they may withdraw from the research at any time and for any reason?	✓		
6	With questionnaires and interviews will you give participants the option of omitting questions they do not want to answer?	✓		
7	Will you tell participants that their data will be treated with full confidentiality and that, if published, it will not be identifiable as theirs?	✓		
8	Will you give participants the opportunity to be debriefed i.e. to find out more about the study and its results?	✓		

Figure 60 Fast Track Form

If you have ticked **No** to any of Q1-8 you should complete the full Ethics Approval Form.

		YES	NO	N/A
9	Will your project/assignment deliberately mislead participants in any way?		✓	
10	Is there any realistic risk of any participants experiencing either physical or psychological distress or discomfort?		✓	
11	Is the nature of the research such that contentious or sensitive issues might be involved?			✓

If you have ticked **Yes** to 9, 10 or 11 you should complete the full Ethics Approval Form. In relation to question 10 this should include details of what you will tell participants to do if they should experience any problems (e.g. who they can contact for help). You may also need to consider risk assessment issues.

			YES	NO	N/A
12	Does your project/assignment involve work with animals?			✓	
13	Do participants fall into any of the following special groups? **Note that you may also need to obtain satisfactory clearance from the relevant authorities**	Children (under 18 years of age)		✓	
		People with communication or learning difficulties			
		Patients			
		People in custody			
		People who could be regarded as vulnerable			
		People engaged in illegal activities (eg drug taking)			
14	Does the project/assignment involve external funding or external collaboration where the funding body or external collaborative partner requires the University to provide evidence that the project/assignment had been subject to ethical scrutiny?			✓	

If you have ticked **Yes** to 12, 13 or 14 you should complete the full Ethics Approval Form. There is an obligation on student and supervisor to bring to the attention of the APU Research Ethics Committee any issues with ethical implications not clearly covered by the above checklist.

STUDENT RESEARCHER
Provide in the boxes below (plus any other appended details) information required in support of your application, THEN SIGN THE FORM.

	Please Tick Boxes
I consider that this project/assignment has no significant ethical implications requiring a full ethics submission to the APU Research Ethics Committee.	✓

Figure 61 Fast Track Form

Give a brief description of participants and procedure (methods, tests used etc) in up to 150 words.	
I also confirm that: i) All key documents e.g. consent form, information sheet, questionnaire/interview are appended to this application. Or	✓
ii) Any key documents e.g. consent form, information sheet, questionnaire/interview schedules which need to be finalised following initial investigations will be submitted for approval by the project/assignment supervisor/module lecturer before they are used in primary data collection.	✓

E-signature... ... ... Print Name... ... Dr. Aitizaz Ali... ... ... Date...21/5/2024 ... ...
(Student Researcher)

Please note that any variation to that contained within this document that in any way affects ethical issues of the stated research requires the appending of new ethical details. New ethical consent may need to be sought.
The completed form (and any attachments) should be submitted for consideration by your Supervisor/Module Lecturer

SUPERVISOR/MODULE LECTURER
PLEASE CONFIRM THE FOLLOWING:

	Please Tick Box
I consider that this project/assignment has no significant ethical implications requiring a full ethics submission to the APU Research Ethics Committee	✓
i) I have checked and approved the key documents required for this proposal (e.g. consent form, information sheet, questionnaire, interview schedule) Or	✓
ii) I have checked and approved draft documents required for this proposal which provide a basis for the preliminary investigations which will inform the main research study. I have informed the student researcher that finalised and additional documents (e.g. consent form, information sheet, questionnaire, interview schedule) must be submitted for approval by me before they are used for primary data collection.	✓

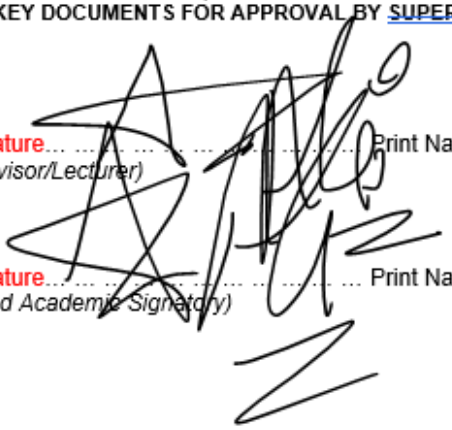
SUPERVISOR AND SECOND ACADEMIC SIGNATORY

STATEMENT OF ETHICAL APPROVAL (please delete as appropriate)

1) THIS PROJECT/ASSIGNMENT HAS BEEN CONSIDERED USING AGREED APU/SU PROCEDURES AND IS NOW APPROVED

Figure 62 Fast Track Form

2) **THIS PROJECT/ASSIGNMENT HAS BEEN APPROVED IN PRINCIPLE AS INVOLVING NO SIGNIFICANT ETHICAL IMPLICATIONS, BUT FINAL APPROVAL FOR DATA COLLECTION IS SUBJECT TO THE SUBMISSION OF KEY DOCUMENTS FOR APPROVAL BY SUPERVISOR (see Appendix A)**



E-signature… … … … … … … … … … Print Name… … …Dr. Aitizaz Ali… … Date…21/5/2024 … … …
(Supervisor/Lecturer)

E-signature… … … … … … … … … Print Name… … …Dr. Aitizaz Ali … … Date… 21/5/2024 … … …
(Second Academic Signatory)

Office Record	Receipt – Appendix A (Fast-Track Ethics Form)
Date Received: Received by whom:	Student name: Student number: Received by: Date:

APPENDIX A
AUTHORISATION FOR USE OF KEY DOCUMENTS

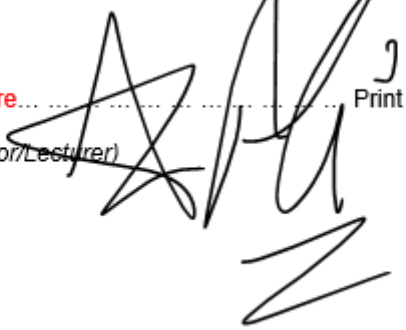
Completion of Appendix A is required when for good reasons key documents are not available when a fast track application is approved by the supervisor/module lecturer and second academic signatory.

I have now checked and approved all the key documents associated with this proposal e.g. consent form, information sheet, questionnaire, interview schedule

Title of project/assignment… Machine Learning in Malware Detection For PE Files and URL's… … … … … …

Name of student researcher … …Satheish A/L Thiruchelvam … … … … … … … … … … … … … … … …

Student ID: …TP060754…… Intake: … APD3F2402CS(CYB)… ……



E-signature… … … … … … … … … … Print Name… … Dr. Aitizaz Ali… … … Date… …21/5/2024 …
…
(Supervisor/Lecturer)

Figure 63 Fast Track Form

Project log sheets

(APU: Serial Number)

PLS V1.0

Project Log Sheet – Supervisory Session

Notes on use of the project log sheet:

1. This log sheet is designed for meetings of more than 15 minutes duration, of which there must be at minimum SIX (6) during the course of the project (SIX mandatory supervisory sessions).
2. The student should prepare for the supervisory sessions by deciding which question(s) he or she needs to ask the supervisor and what progress has been made (if any) since the last session, and noting these in the relevant sections of the form, effectively forming an agenda for the session.
3. A log sheet is to be brought by the STUDENT to each supervisory session.
4. The actions by the student (and, perhaps the supervisor), which should be carried out before the next session should be noted briefly in the relevant section of the form.
5. The student should leave a copy (after the session) of the Project Log Sheet with the supervisor and to the administrator at the academic counter. A copy is retained by the student to be filed in the project file.
6. It is recommended that students bring along log sheets of previous meetings together with the project file during each supervisory session.
7. The log sheet is an important deliverable for the project and an important record of a student's organisation and learning experience. The student **must** hand in the log sheets as an appendix of the final year documentation, with sheets dated and numbered consecutively.

Student's name: …Satheish A/L Thiruchelvam…….. **Date: 5/24/24…Meeting No: …1...**

Project title: … Machine Learning in Malware Detection For PE Files and URL's
Intake: …APD3F2402CS(CYB)…

Supervisor's name: … Dr. Aitizaz Ali …… **Supervisor's signature:** ……

Items for discussion (noted by student before mandatory supervisory meeting):

1. Clarification on the scope of the literature review.
2. Feedback on the initial draft of the problem background.
3. Guidance on data gathering techniques for surveys.
4. Approval for survey questions designed.

Record of discussion (noted by student during mandatory supervisory meeting):

1. Supervisor suggested focusing the literature review on the latest ML-based malware detection methods.
2. Supervisor approved the problem background draft but advised adding more references.
3. Supervisor recommended using online survey tools and highlighted the importance of demographic diversity.
4. Survey questions were approved with minor revisions to ensure clarity.

Figure 64 Project Log Sheet

Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1. Refine the literature review to include recent studies on ML algorithms in malware detection

2. Add additional references to the problem background section.

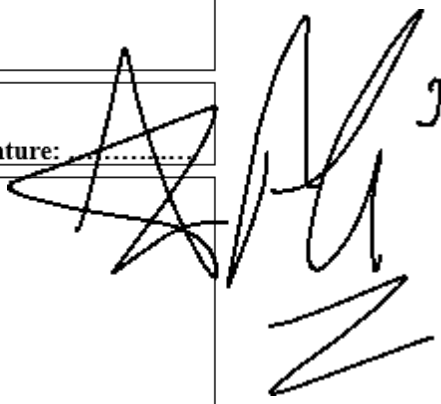
3. Set up an online survey using the recommended tool.

Note: A student should make an appointment to meet his or her supervisor (via the consultation system) at least ONE (1) week prior to a mandatory supervisor session – please see document on project timelines. In the event a supervisor could not be booked for consultation, the project manager should be informed ONE (1) week prior to the session so that a meeting can be subsequently arranged.

Student's name: …Satheish A/L Thiruchelvam …….. **Date: 5/24/24…Meeting No: …2…**

Project title: … Machine Learning in Malware Detection For PE Files and URL's
Intake:…APD3F2402CS(CYB)…

Supervisor's name: … Dr. Aitizaz Ali …… **Supervisor's signature:** ……



Items for discussion (noted by student before mandatory supervisory meeting):

1. Review of the updated literature review section.
2. Discussion on the methodology section structure.
3. Feedback on the initial survey responses.
4. Clarification on the project plan timeline.

Record of discussion (noted by student during mandatory supervisory meeting):

1. Supervisor approved the updated literature review and suggested focusing on specific ML algorithms.

2. Supervisor advised structuring the methodology section into subsections for each data gathering technique.

3. Initial survey responses were deemed satisfactory, with a suggestion to increase the sample size.

4. Timeline for the project plan was adjusted to allocate more time for data analysis.

Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1. Enhance the literature review with detailed analysis of selected ML algorithms.

2. Structure the methodology section as per supervisor's advice.

3. Update the project plan timeline and share it with the supervisor.

Figure 65 Project Log Sheet

Student's name: …Satheish A/L Thiruchelvam…….... **Date: 5/24/24…Meeting No: …3...**

Project title: … Machine Learning in Malware Detection For PE Files and URL's
Intake:…APD3F2402CS(CYB)…

Supervisor's name: … Dr. Aitizaz Ali …….. **Supervisor's signature:** ……

Items for discussion (noted by student before mandatory supervisory meeting):

1. Progress update on survey responses and data collection.
2. Discussion on preliminary data analysis findings.
3. Guidance on writing the system requirement analysis section.
4. Feedback on the structure of the project conclusion.

Record of discussion (noted by student during mandatory supervisory meeting):

1. Supervisor was satisfied with the progress on survey responses and encouraged continued effort.

2. Preliminary data analysis findings were discussed, and supervisor suggested more detailed statistical analysis.

3. Supervisor provided a framework for the system requirement analysis section.

4. Project conclusion structure was approved with suggestions to highlight key achievements and future work.

Action List (to be attempted or completed by student by the next mandatory supervisory meeting):

1. Continue collecting survey responses and finalize data collection.

2. Write the system requirement analysis section using the provided framework.

3. Draft the project conclusion, emphasizing key findings and potential areas for future research.

Figure 66 Project Log Sheet

Gantt chart

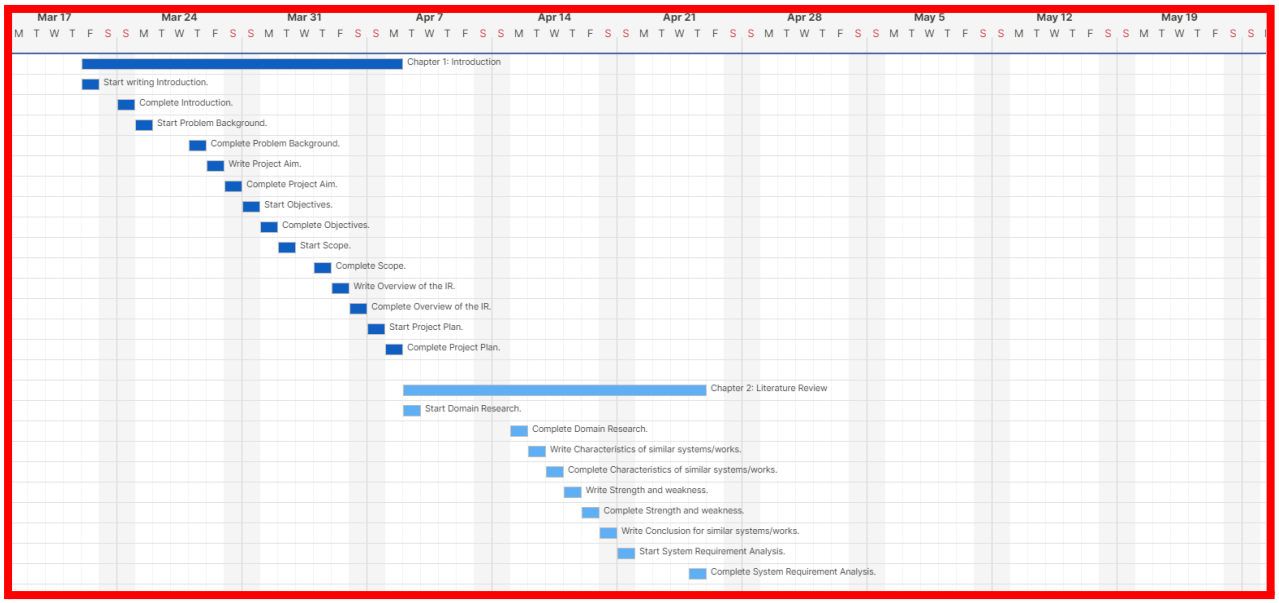


Figure 67 Gantt Chart

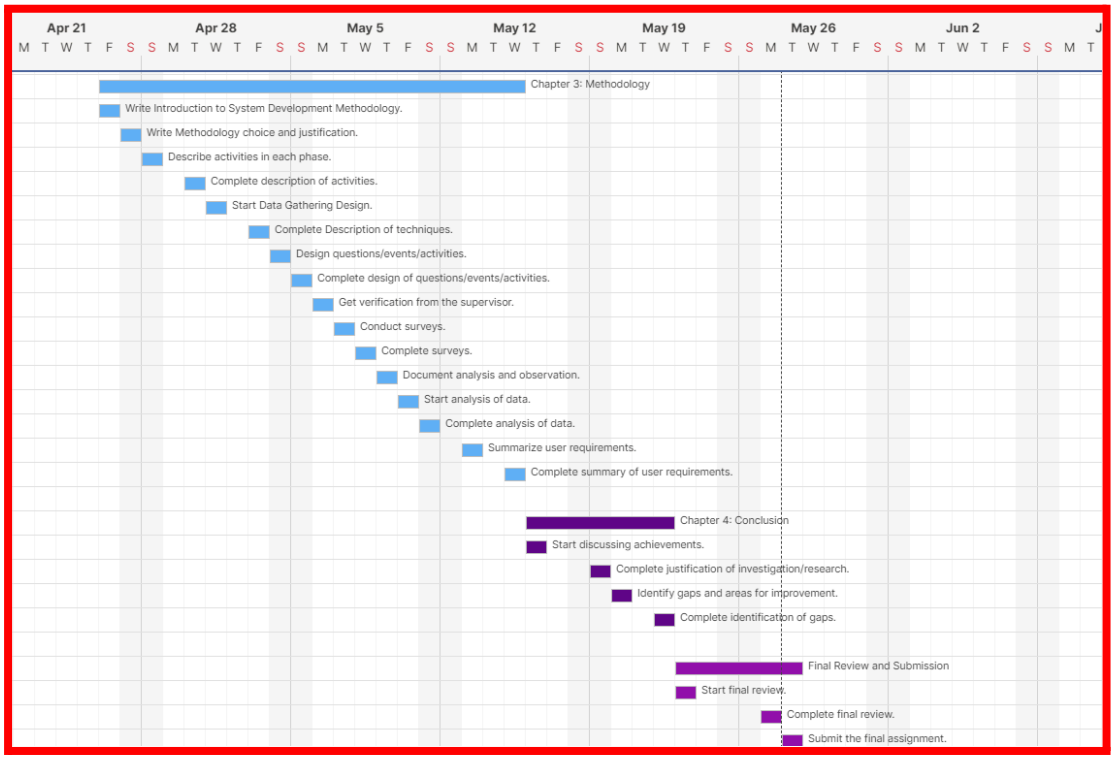


Figure 68 Gantt Chart

Respondent demographics

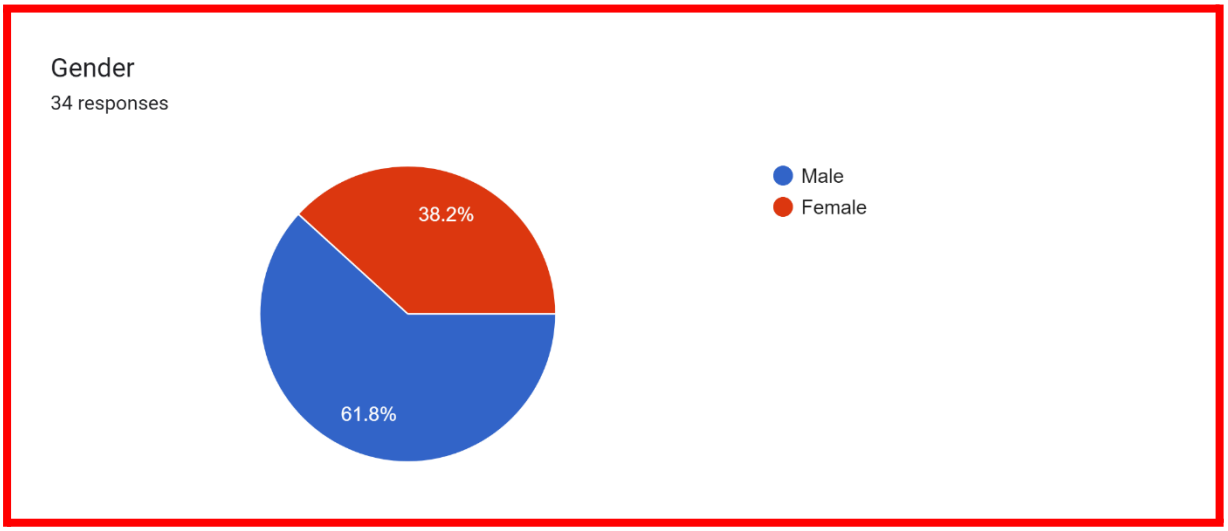


Figure 69 Respondent Demographics

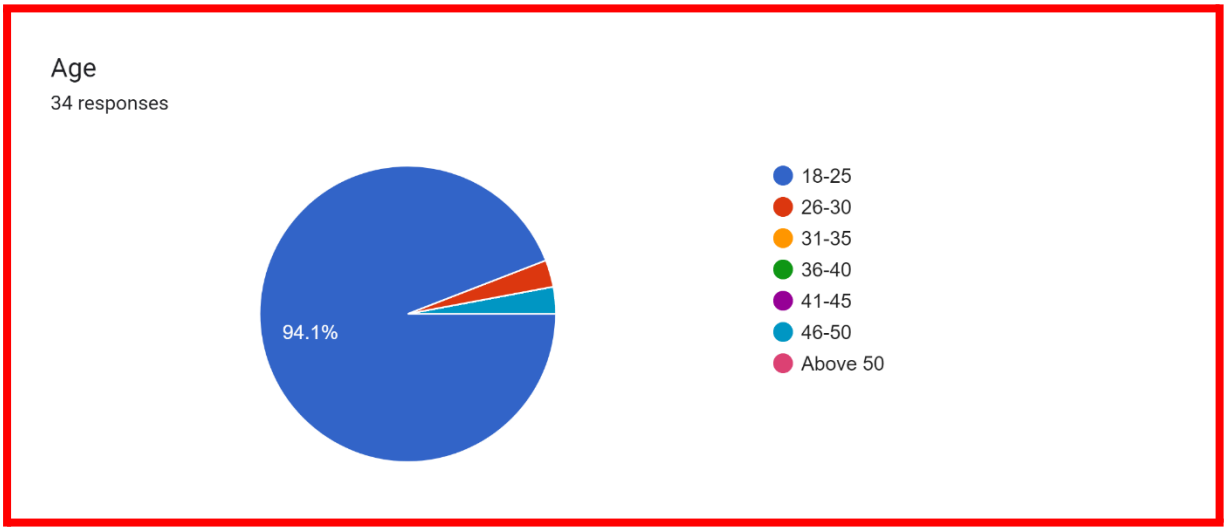


Figure 70 Respondent Demographics

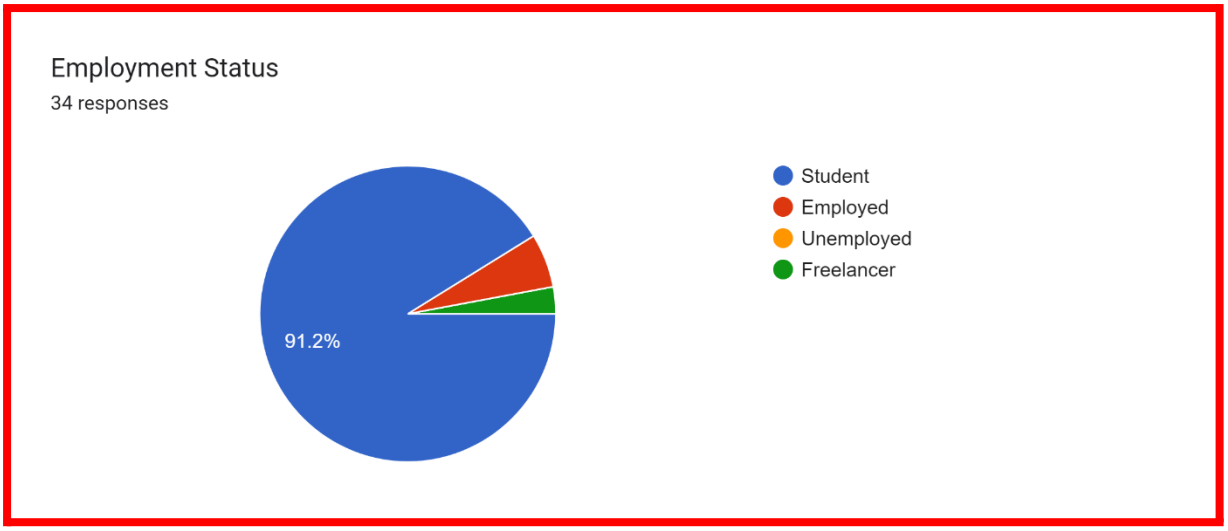


Figure 71 Respondent Demographics

Figure 72 Respondent Demographics

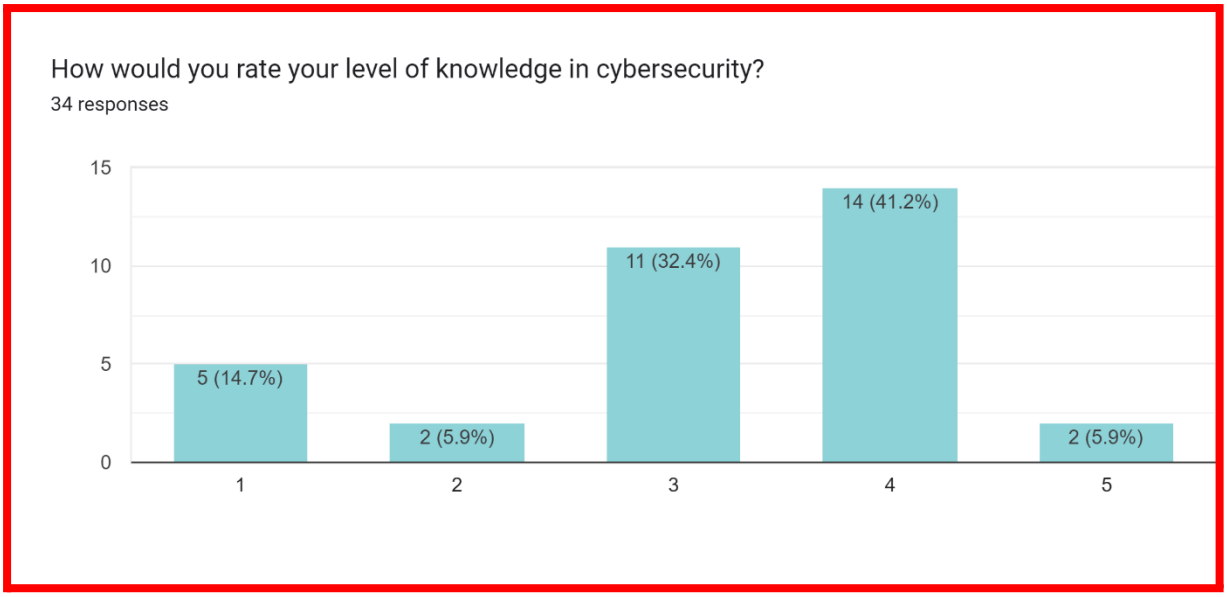


Figure 73 Respondent Demographics

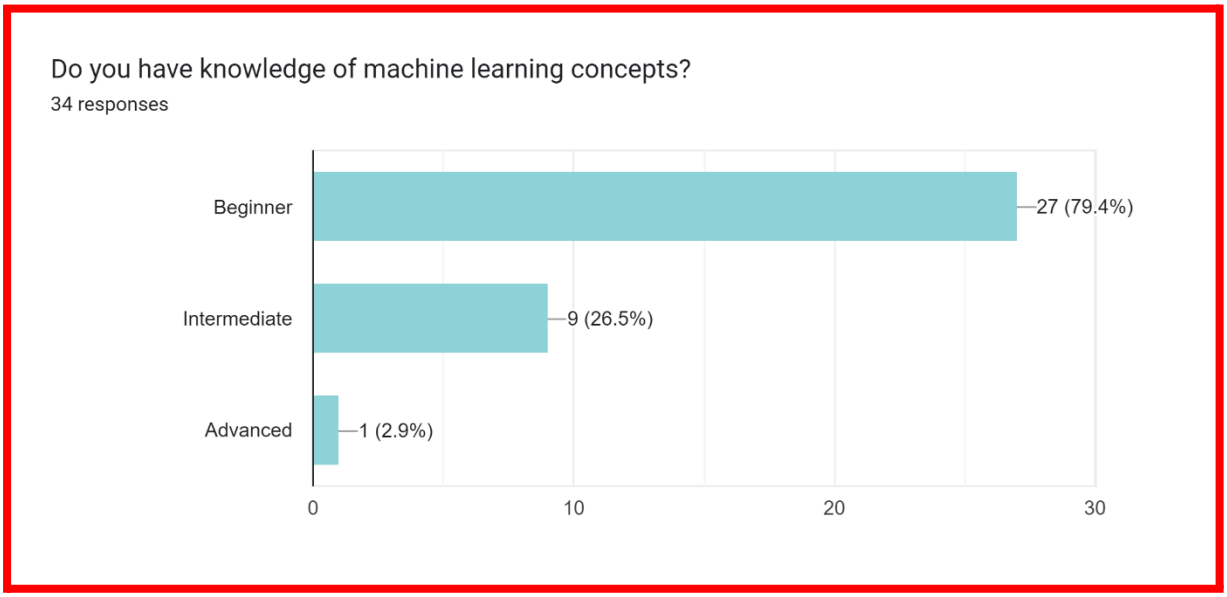


Figure 74 Respondent Demographics

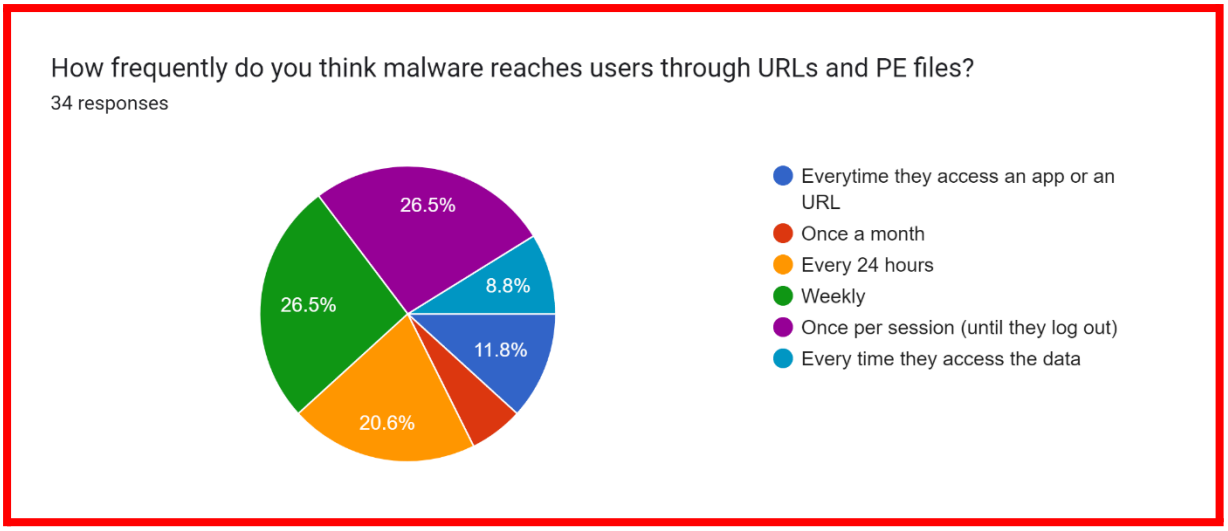


Figure 75 Respondent Demographics

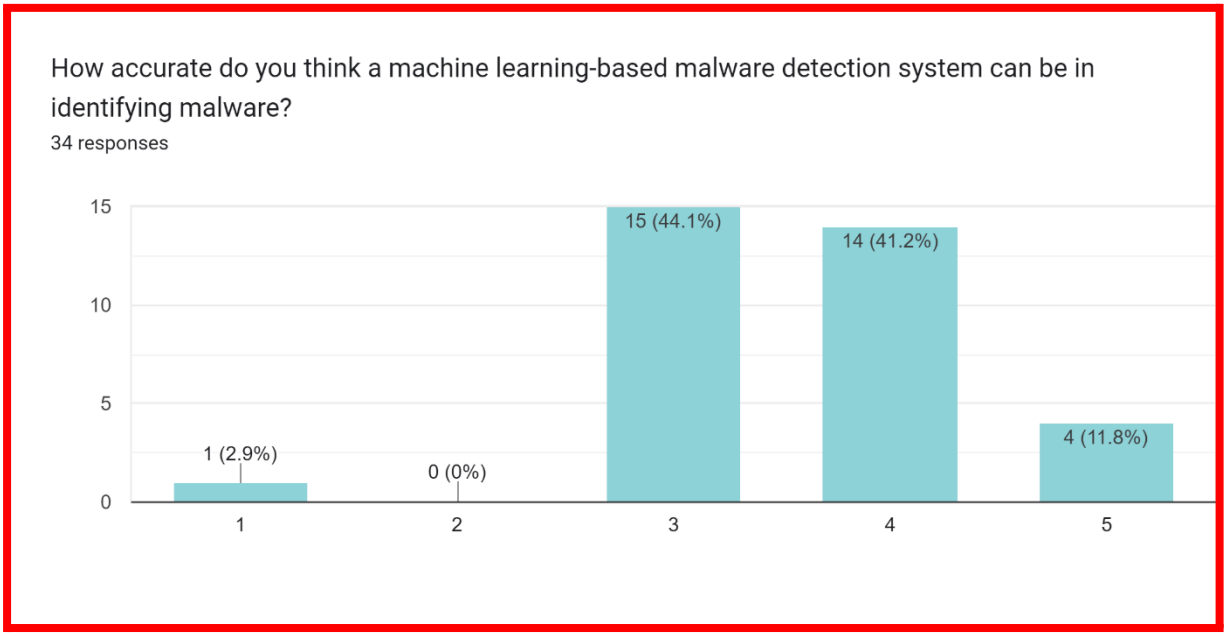


Figure 76 Respondent Demographics

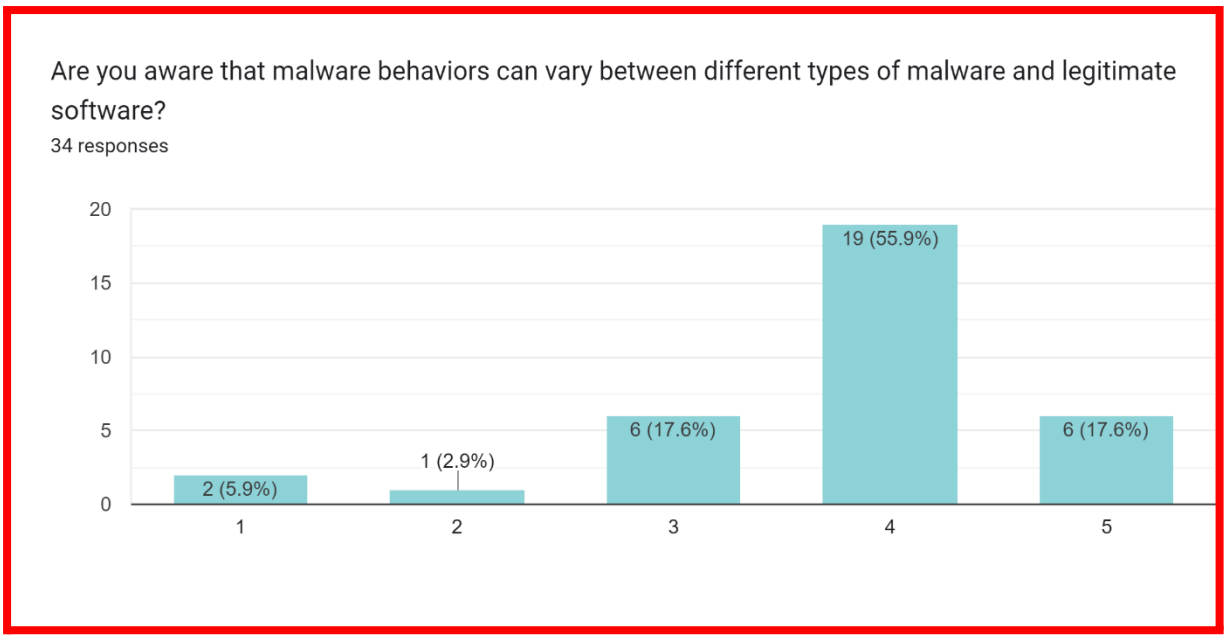


Figure 77 Respondent Demographics

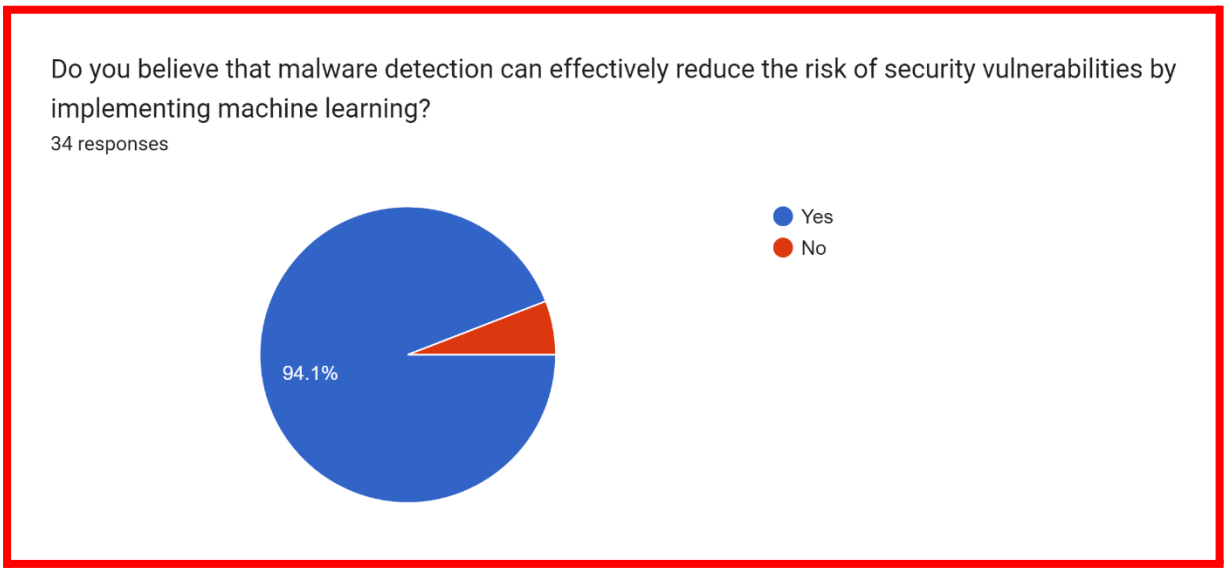


Figure 78 Respondent Demographics

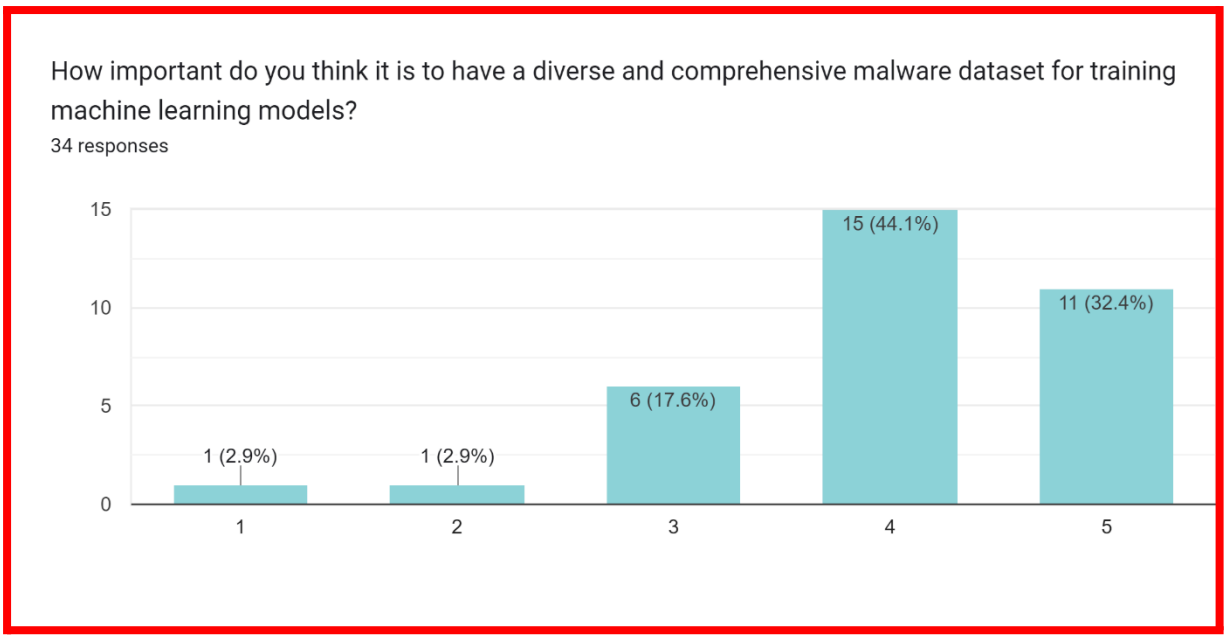


Figure 79 Respondent Demographics

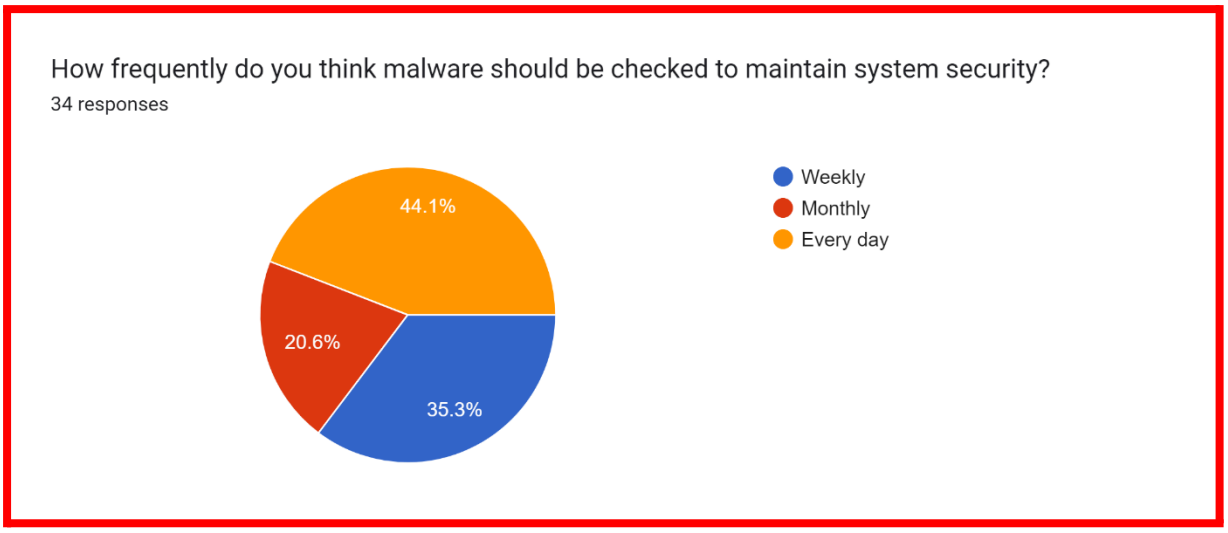


Figure 80 Respondent Demographics

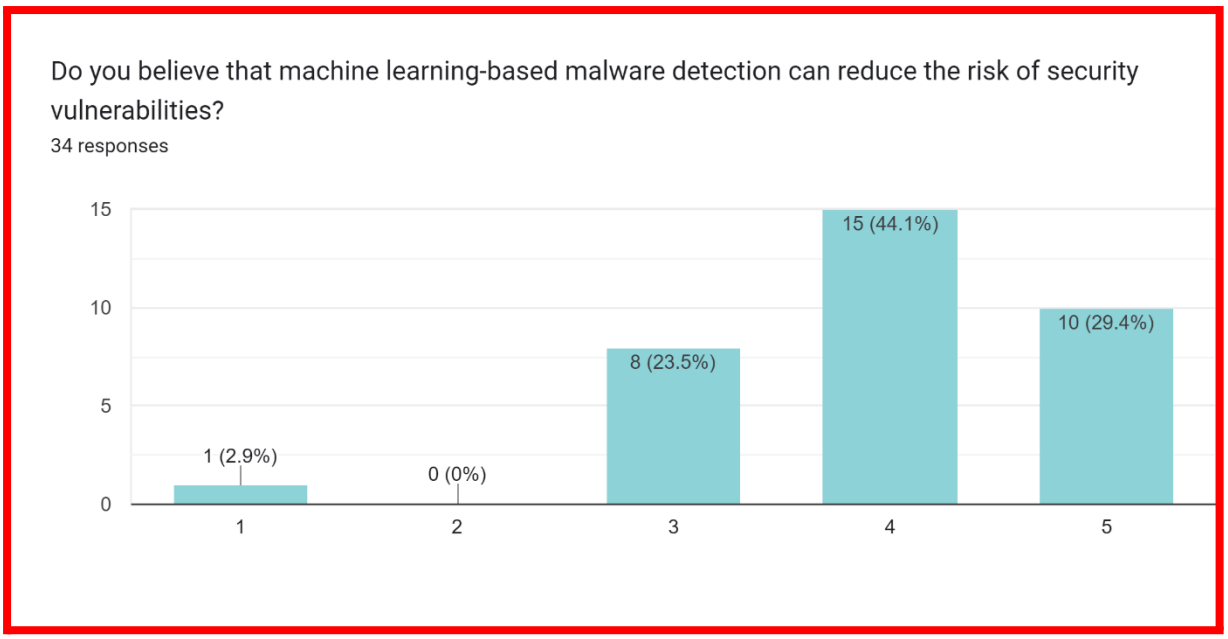


Figure 81 Respondent Demographics

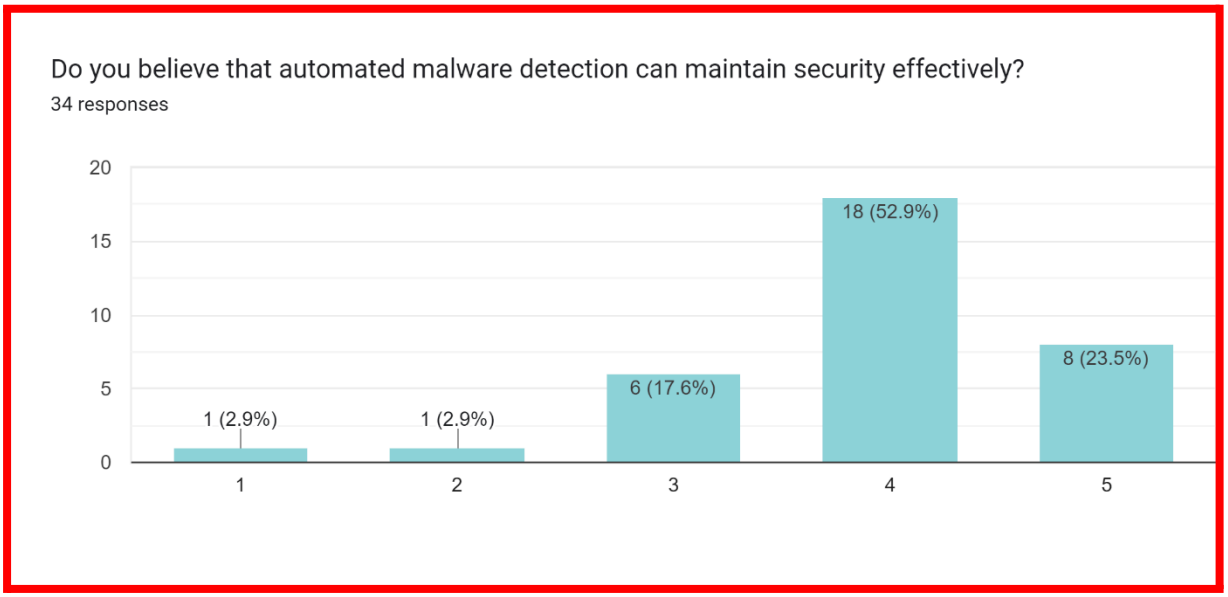


Figure 82 Respondent Demographics

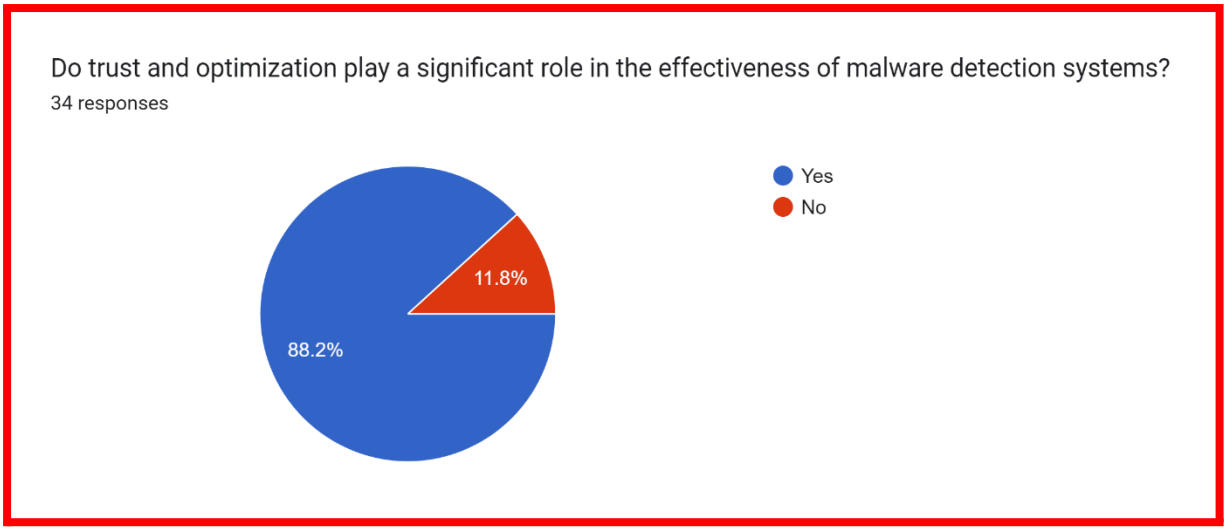


Figure 83 Respondent Demographics

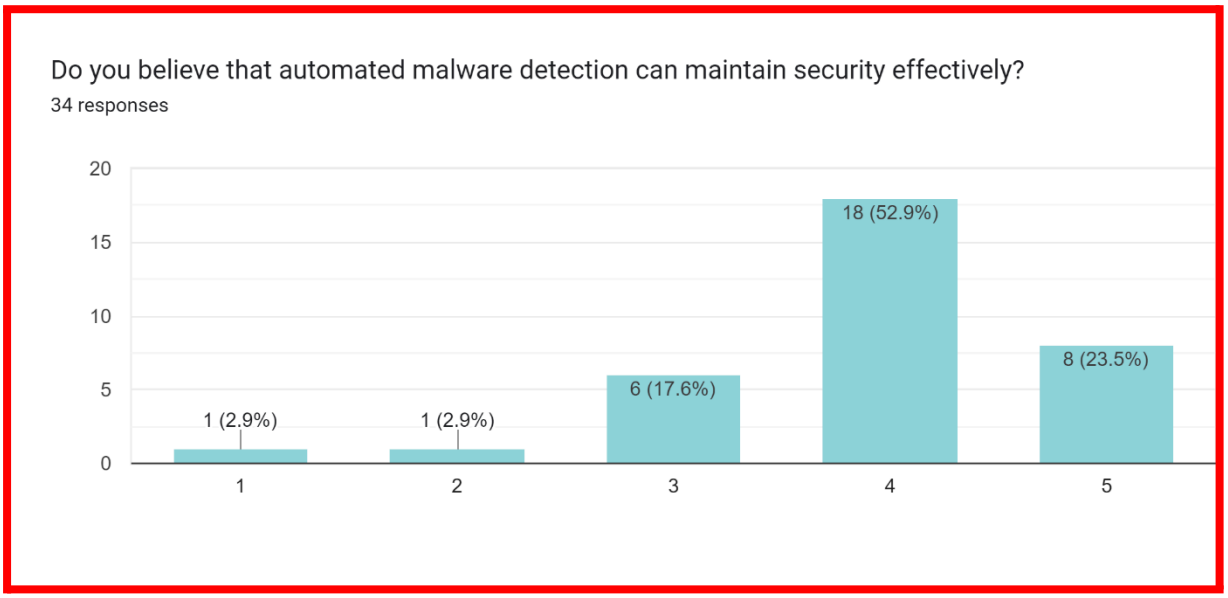


Figure 84 Respondent Demographics

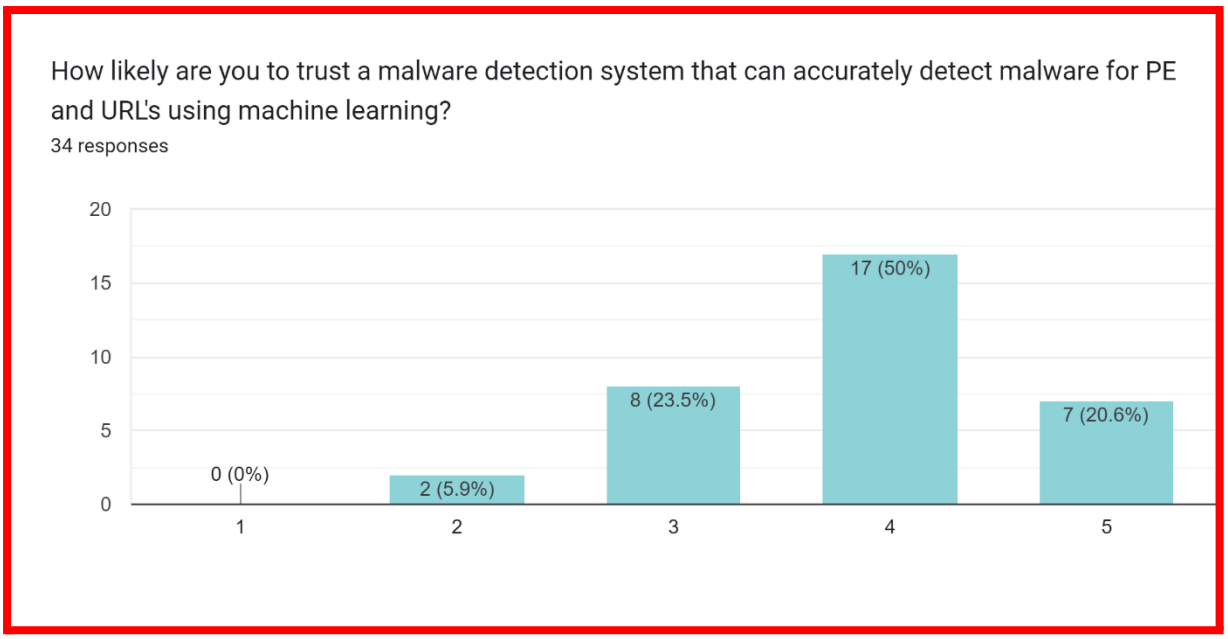


Figure 85 Respondent Demographics

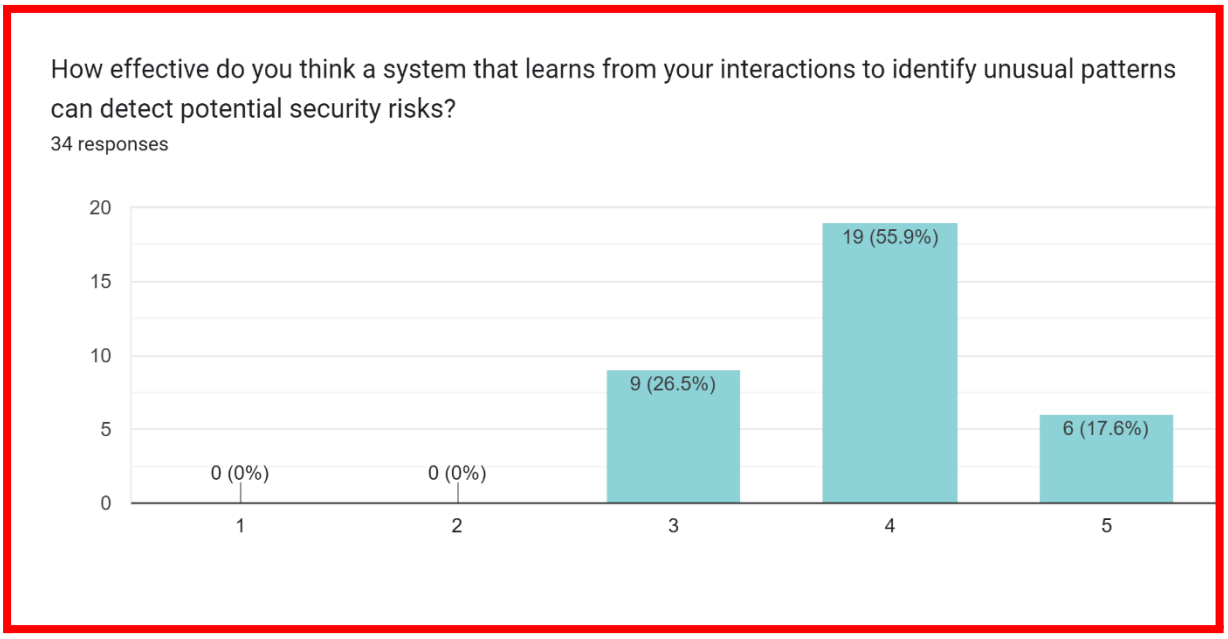


Figure 86 Respondent Demographics

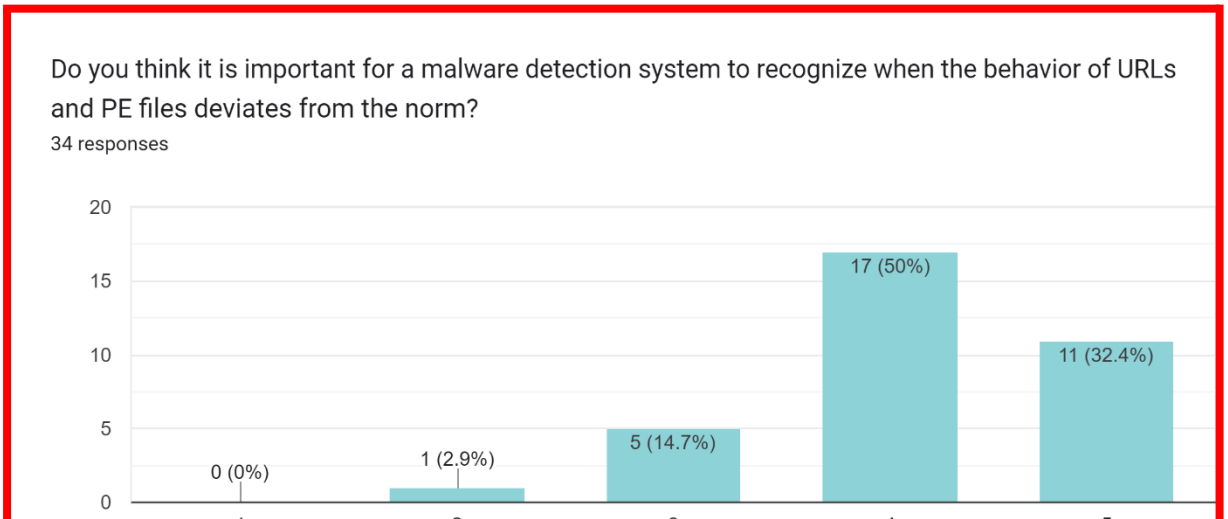


Figure 87 Respondent Demographics

Chapter 4

4.1 Introduction

This series of diagrams illustrates the overall structure and workflows of a **malware detection system** using machine learning to analyze URLs and PE files. The system enables an admin to upload files for malware analysis, view results, generate reports, and visualize data through charts, while also quarantining malware files. These diagrams depict key interactions between system components, the data flow between entities, and the processes for file scanning, reporting, and chart generation, offering a comprehensive overview of how the system operates efficiently to detect and manage malware.

4.2 Design

4.2.1 Sequence Diagram

4.2.1.1 Charts

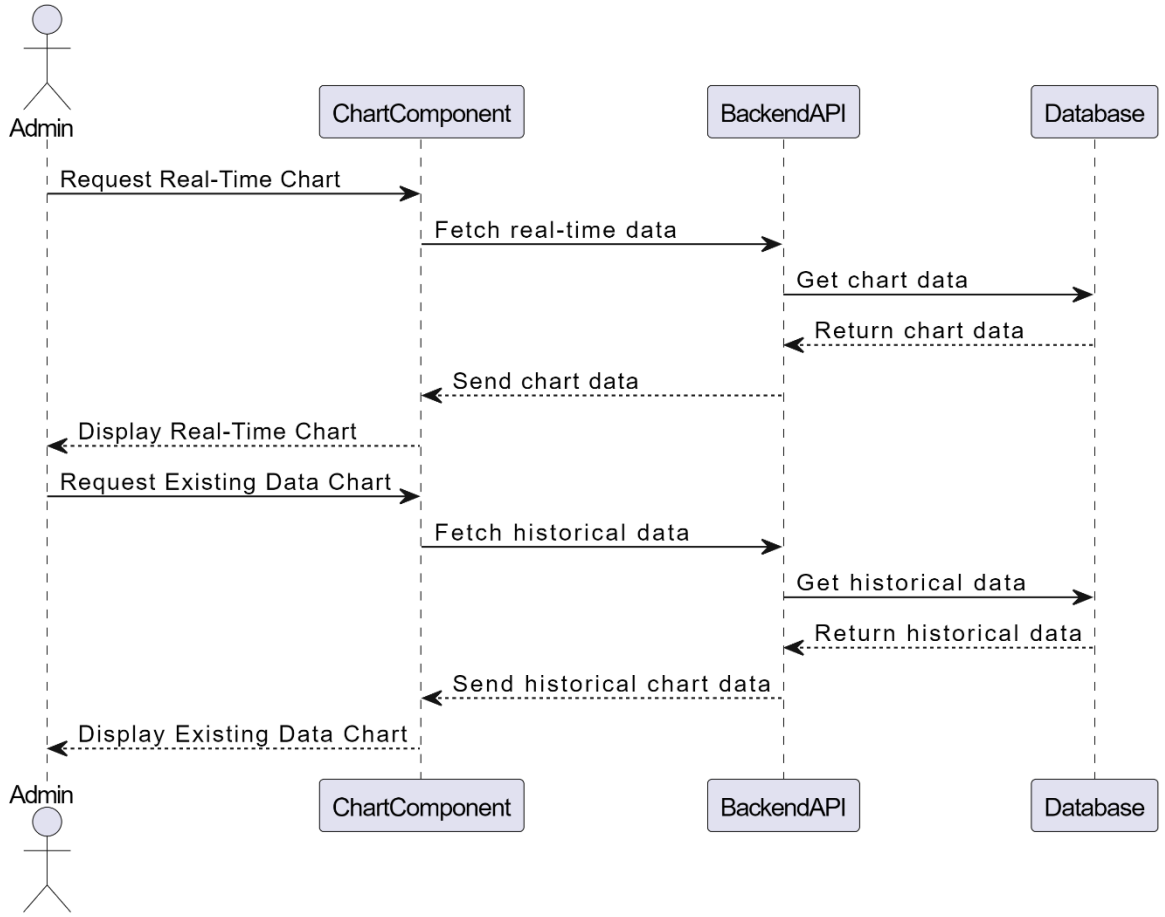


Figure 88 Sequence Diagram Charts

This sequence diagram illustrates the process of fetching and displaying real-time and historical chart data requested by the admin. The interaction begins with the admin requesting a real-time chart from the ChartComponent, which then forwards the request to the BackendAPI. The BackendAPI sends a query to the Database to retrieve the necessary real-time data. After fetching the data from the Database, the BackendAPI sends the real-time chart data back to the ChartComponent, which then displays it to the admin. Following this, the admin can request an existing data chart. The ChartComponent again interacts with the BackendAPI to request the historical chart data, which in turn queries the Database. The Database provides the historical data, which is sent back through the BackendAPI to the ChartComponent. Finally, the ChartComponent displays the existing data chart to the admin.

This process highlights how data flows from the user interface to the backend and database for real-time and historical chart generation.

4.2.1.2 Report

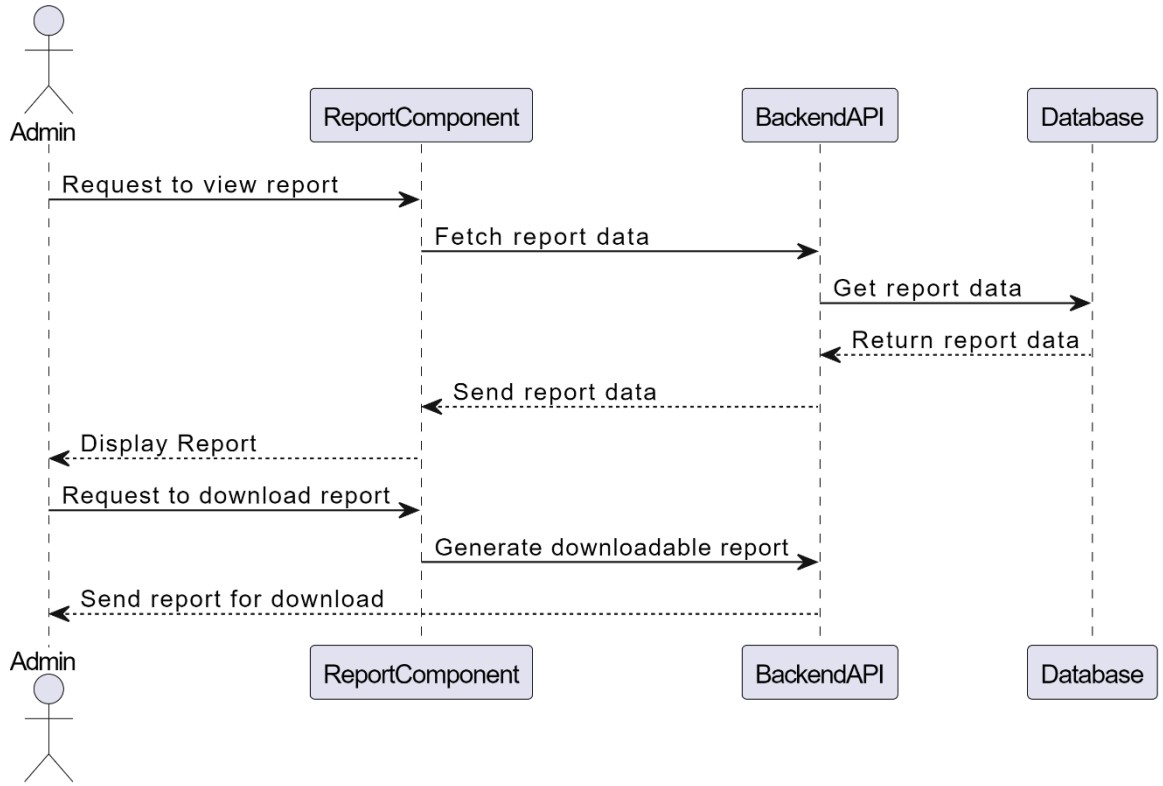


Figure 89 Sequence Diagram Reports

This sequence diagram shows the interaction between the admin, the ReportComponent, BackendAPI, and the Database for viewing and downloading reports. The process begins when the admin requests to view a report. The ReportComponent forwards this request to the BackendAPI, which queries the Database for the report data. The Database retrieves the report data and sends it back to the BackendAPI, which then returns the report data to the ReportComponent. The report is displayed to the admin. If the admin chooses to download the report, a request is sent from the ReportComponent to the BackendAPI to generate a downloadable version of the report. The BackendAPI processes this request and sends the downloadable report back to the ReportComponent, which then provides the file for the admin to download. This interaction flow ensures that reports are fetched, displayed, and downloaded in an organized manner.

4.2.1.3 FileUpload Exe or URL

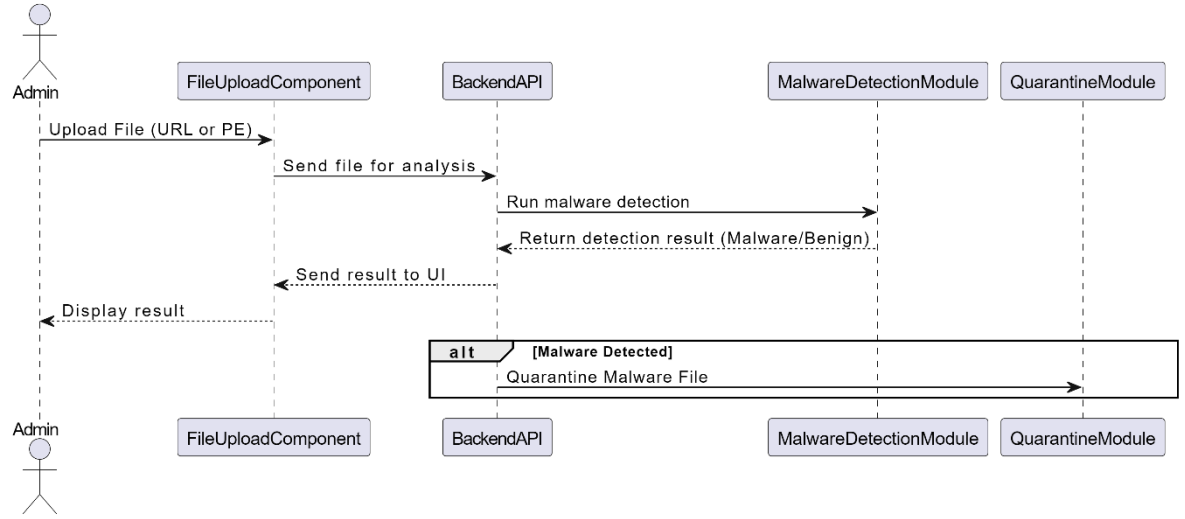


Figure 90 Sequence Diagram FileUpload

This sequence diagram illustrates the process of uploading a file (either URL or PE) for malware detection and possible quarantine. The admin starts by uploading a file through the **FileUploadComponent**, which sends the file to the **BackendAPI** for analysis. The **BackendAPI** communicates with the **MalwareDetectionModule**, which runs the malware detection model on the uploaded file. The result, either malware or benign, is returned to the **BackendAPI** and then sent back to the **FileUploadComponent**, where it is displayed to the admin.

If the file is detected as malware, an alternative flow is triggered, where the **BackendAPI** sends the malicious file to the **QuarantineModule**. The file is then quarantined for security purposes, preventing any further interaction with the file while it remains flagged as malware. This process ensures that both benign and malicious files are handled appropriately, with malware being quarantined to mitigate risk.

4.2.1.4 ExistingChart.js

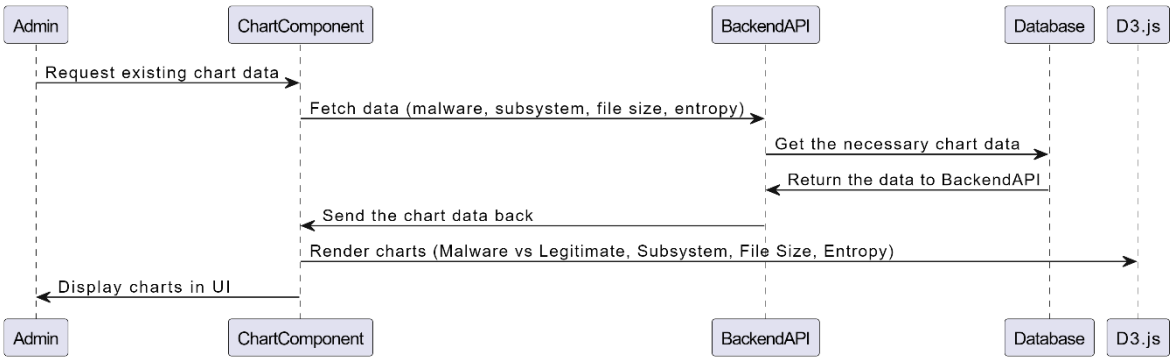


Figure 91 Sequence Diagram ExisitngChart

The **ExistingChart.js** component facilitates the display of various charts that provide insights into malware analysis. When an **Admin** requests to view existing chart data, the **ChartComponent** triggers a backend API call to fetch the necessary data from the database. This includes statistics on the malware versus legitimate file counts, subsystem classifications (such as drivers, Windows GUI, and boot loaders), file size distributions, and entropy levels (low, medium, high). Once the data is retrieved from the **BackendAPI**, it is processed and passed to **D3.js**, a data visualization library, which then renders the charts in the user interface. The charts visually represent the percentages of each category in clear, radial bar graphs, making it easy for the **Admin** to understand and compare values. The **ChartComponent** ensures that the admin has access to detailed visual representations of the existing data, enabling them to make data-driven decisions regarding file analysis and malware detection. This sequence of actions occurs seamlessly, providing the admin with real-time and historical analysis through dynamic charts displayed in the UI.

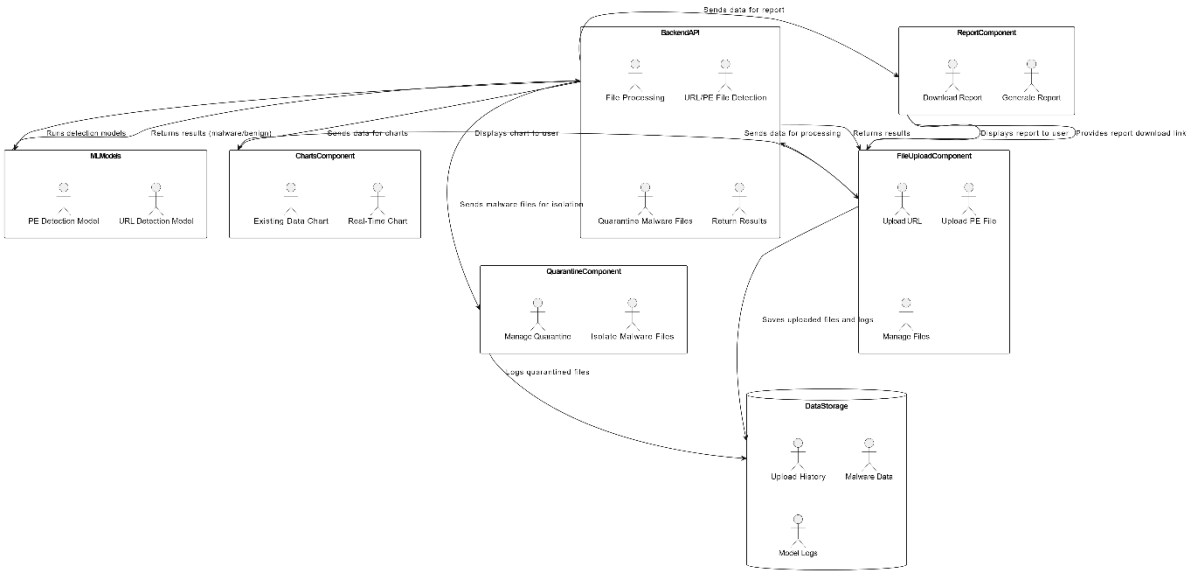


Figure 92 System Architecture Diagram

4.3 System architecture diagram

This diagram represents the architecture of a malware detection system that handles the detection and management of malicious files and URLs. The system involves several key components, each performing distinct tasks to ensure the files are properly analyzed, quarantined if necessary, and reported on.

4.3.1 File Upload and Analysis

The process starts with the **FileUploadComponent**, where the admin uploads either a URL or a PE (Portable Executable) file for malware analysis. This component is responsible for managing files and passing them to the **BackendAPI** for further processing. It plays a central role in initiating the workflow of the malware detection system, allowing admins to either upload or view previously uploaded files. Once the file is uploaded, it sends it to the backend for scanning and displays the analysis result.

4.3.2 Backend and Machine Learning Models

The **BackendAPI** acts as the core processor. It determines whether to send the uploaded file to the **URLDetection** model (for URLs) or the **PEFileDetection** model (for PE files). These machine learning models, located within the **MLModels** module, analyze the uploaded files for any signs of malware. Based on the analysis, the **BackendAPI** returns a result (malware or benign), which is sent back to the **FileUploadComponent**. The **BackendAPI** ensures that files are properly classified by utilizing trained machine learning models for different types of data (URLs or PE files).

4.3.3 Charting and Reporting

Once the file is analyzed, the **ChartsComponent** allows the admin to view the results of the analysis in a visual format. This component supports both **Real-Time Charts**, which display live data, and **Existing Data Charts**, which provide historical insights. These charts help the admin understand trends in file uploads, malware detection rates, and other statistical insights related to the system's performance.

For detailed documentation, the **ReportComponent** comes into play. This component generates comprehensive reports based on the analysis results, whether for URLs or PE files. The admin can view these reports directly through the system interface or download them for external review and record-keeping. The reporting system adds transparency to the file analysis process, enabling admins to track the outcomes of malware detection.

4.3.4 Quarantine and Data Storage

If a file is determined to be malicious, it is moved to the **QuarantineComponent**. This part of the system isolates harmful files, ensuring they cannot affect other files or users. It logs all the necessary details about quarantined files, such as why they were flagged as malware and when they were quarantined. The quarantine process helps in securely managing detected malware files.

All files, results, reports, and chart data are stored within the **DataStorage** module. This includes an **Upload History** for tracking previously uploaded files, **Malware Data** for storing detected malicious files, and **Model Logs** that record the operations and performance of the detection models.

4.3.5 Overall Workflow

The system begins with the admin uploading a file for analysis. The file is processed through the **BackendAPI**, where it is sent to either the **URLDetection** or **PEFileDetection** model, depending on its type. Once analyzed, the results are returned to the **FileUploadComponent**, which displays the findings to the admin. If a file is found to be malicious, it is moved to the **QuarantineComponent** for secure isolation. The admin can also view analysis charts and reports, either in real-time or based on historical data, using the **ChartsComponent** and **ReportComponent**, respectively.

This workflow provides a comprehensive and streamlined approach to malware detection, ensuring all steps from file upload to quarantine are covered with transparency and security.

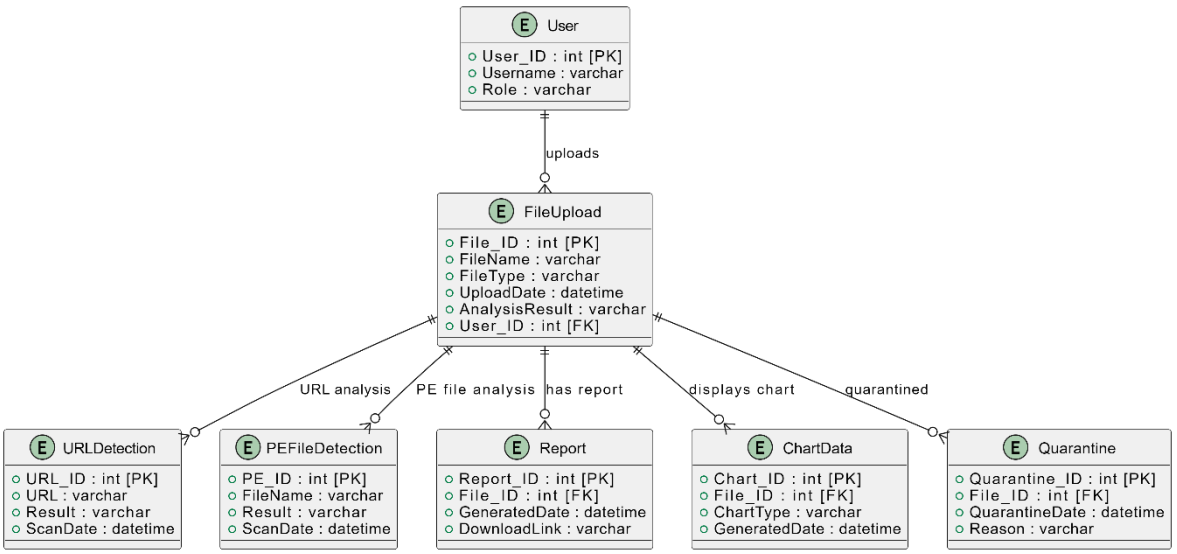


Figure 93 Entity Relationship Diagram

Entity Relationship Diagram (ERD)

Entity Name	**Attributes**	**Description**
User	- User_ID (PK) - Username - Role	Represents the admin user who interacts with the system, primarily responsible for file uploads.
FileUpload	- File_ID (PK) - FileName - FileType - UploadDate - AnalysisResult - User_ID (FK)	Stores information about each uploaded file (either URL or PE file) and links to the user who uploaded it.
URLDetection	- URL_ID (PK) - URL - Result - ScanDate	Handles URL detection results, including the scan outcome and the date it was scanned.
PEFileDetection	- PE_ID (PK) - FileName - Result	Handles PE file detection results, including the

	- ScanDate	analysis outcome and the scan date.
Report	- Report_ID (PK) - File_ID (FK) - GeneratedDate - DownloadLink	Contains details of generated reports, including the download link for accessing the report.
ChartData	- Chart_ID (PK) - File_ID (FK) - ChartType - GeneratedDate	Manages data for real-time or historical charts, showing the trends and results of the analysis.
Quarantine	- Quarantine_ID (PK) - File_ID (FK) - QuarantineDate - Reason	Stores information about quarantined files, including the reason for quarantine and the date.

The **User** entity represents the admin responsible for uploading files for analysis. Each admin can upload multiple files, and these files are managed through the **FileUpload** entity. This entity stores the details of each file, such as the file name, type (either URL or PE file), and the results of its analysis. Each file is connected to the admin who uploaded it.

The system performs either **URLDetection** or **PEFileDetection** based on the type of file uploaded. Both entities store the results of the analysis, along with the date and time of the scan. These analysis results help determine whether the file is legitimate or malicious.

For every file analyzed, a **Report** is generated, summarizing the findings. This report is linked to the original file and contains a downloadable link so that the admin can access the report. At the same time, **ChartData** is produced to visualize the analysis, either in real-time or as historical data, depending on the admin's request.

If a file is flagged as malicious, it is moved to the **Quarantine** entity, where the reason for the quarantine and the date it occurred are stored. This ensures that any harmful files are safely isolated from the system. Each process whether it's analysis, reporting, chart generation, or quarantine traces back to the original file upload and the admin who uploaded the file.

This ERD shows how files are processed, analyzed, reported, and managed within the system, with clear connections between the files, analysis results, reports, charts, and quarantine actions

4.4 Flowchart

4.4.1 View Report

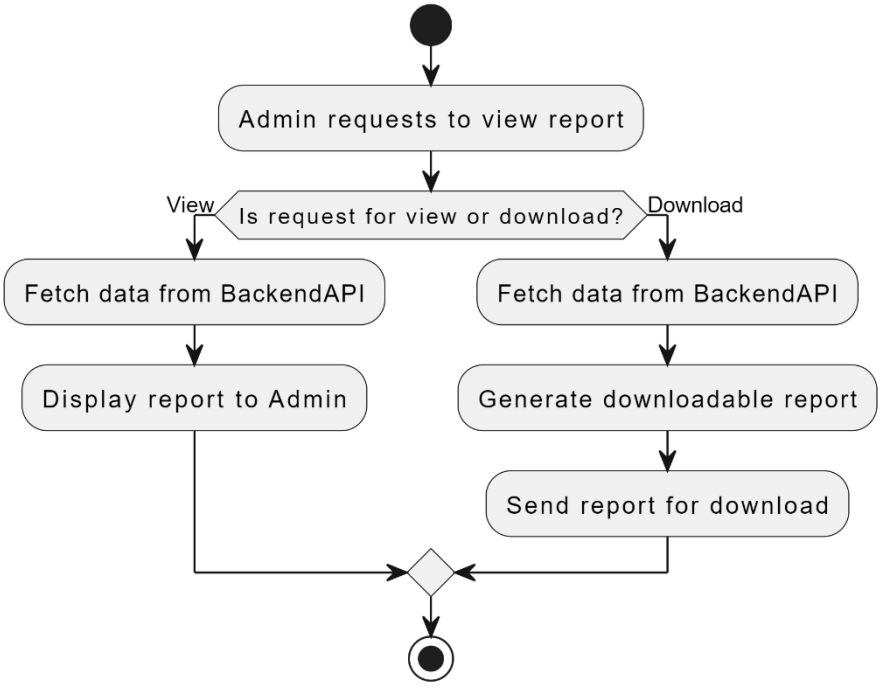


Figure 94 Flowchart View Report

In this flowchart, the process begins when the admin requests to either view or download a report. If the admin chooses to view the report, the system fetches the required report data from the backend (via the Backend API) and displays the report directly in the user interface. If the admin instead requests to download the report, the system fetches the report data similarly but then generates a downloadable version of the report. The downloadable file is then sent to the admin for download. This flow ensures that the admin can interact with the reporting system either by viewing it directly or by downloading the report for offline use.

4.4.2 Charts

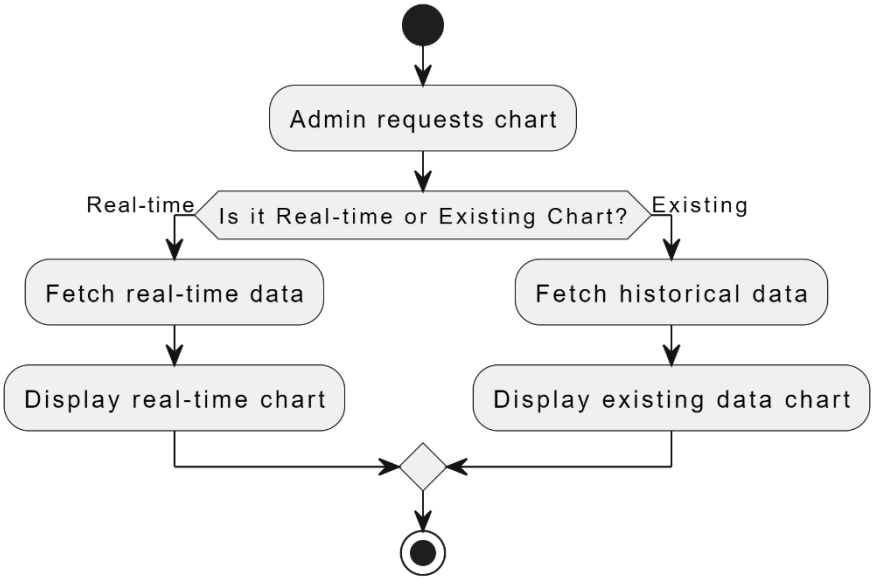


Figure 95 Flowchart Charts

In this flowchart, the process starts when the admin requests to view chart data. The system checks whether the admin wants to see a real-time chart or an existing (historical) chart. If real-time data is requested, the system fetches the real-time chart data and displays it. For existing charts, the system fetches historical data from the database and displays the chart using past information. This flow allows the admin to seamlessly switch between viewing live data and historical trends, depending on the analysis needs.

4.4.3 Upload and Predict Files or URL

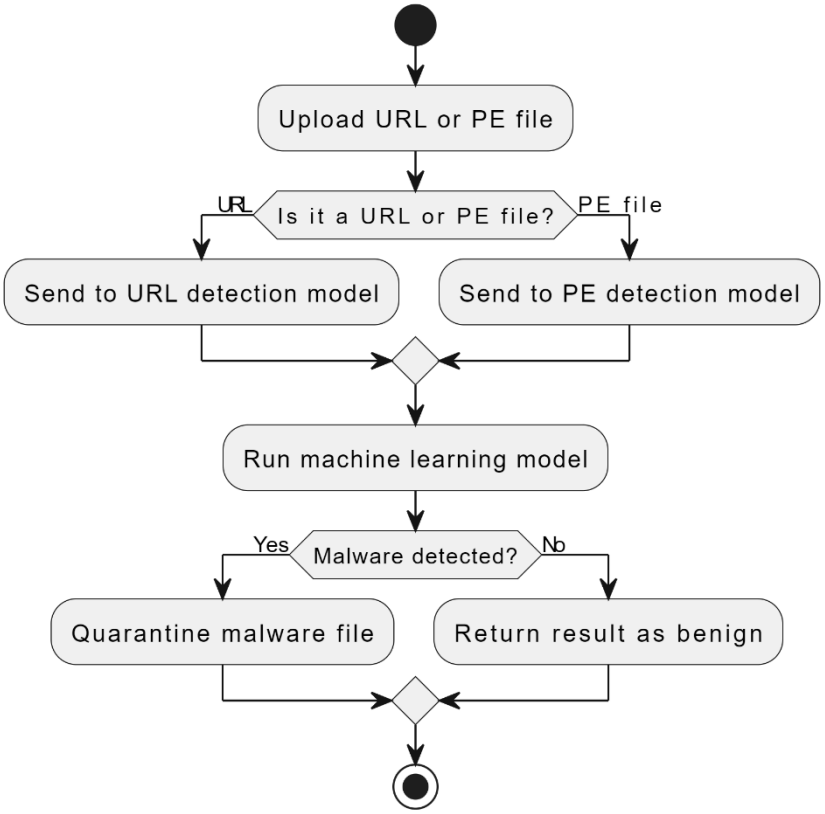


Figure 96 Flowchart Upload and Predict Files

This flowchart illustrates the process of detecting malware in either URL or PE (Portable Executable) files. When a file is uploaded by the admin, the system determines if it's a URL or a PE file and routes the file to the corresponding detection model (either for URLs or PE files). After running the machine learning model to check for malware, the system decides if malware is detected. If malware is found, the system quarantines the file; if no malware is found, the file is marked as benign, and the result is returned to the admin. This flow ensures a comprehensive analysis for both file types.

4.4.4. ExisitingCharts

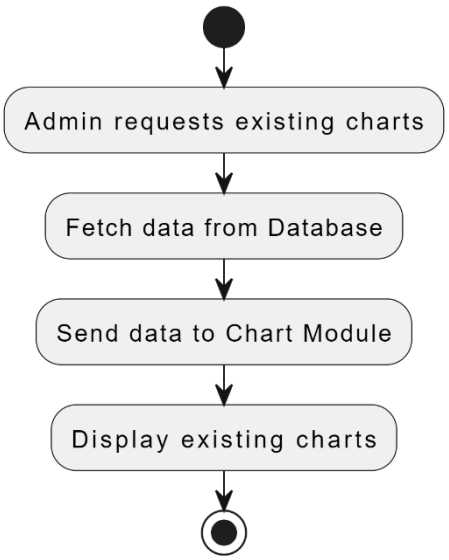


Figure 97 ExisitngCharts

In this flowchart, the process starts when the admin requests to view existing charts. The system first fetches the relevant historical data from the database. After collecting the data, it sends it to the chart module responsible for rendering charts. Finally, the system displays the existing charts to the admin. This flow ensures that the admin can access past chart data for more in-depth analysis of historical trends, which could be crucial for reviewing previously analyzed malware patterns or system performance over time.

4.5 Interface design

4.5.1 Main Page

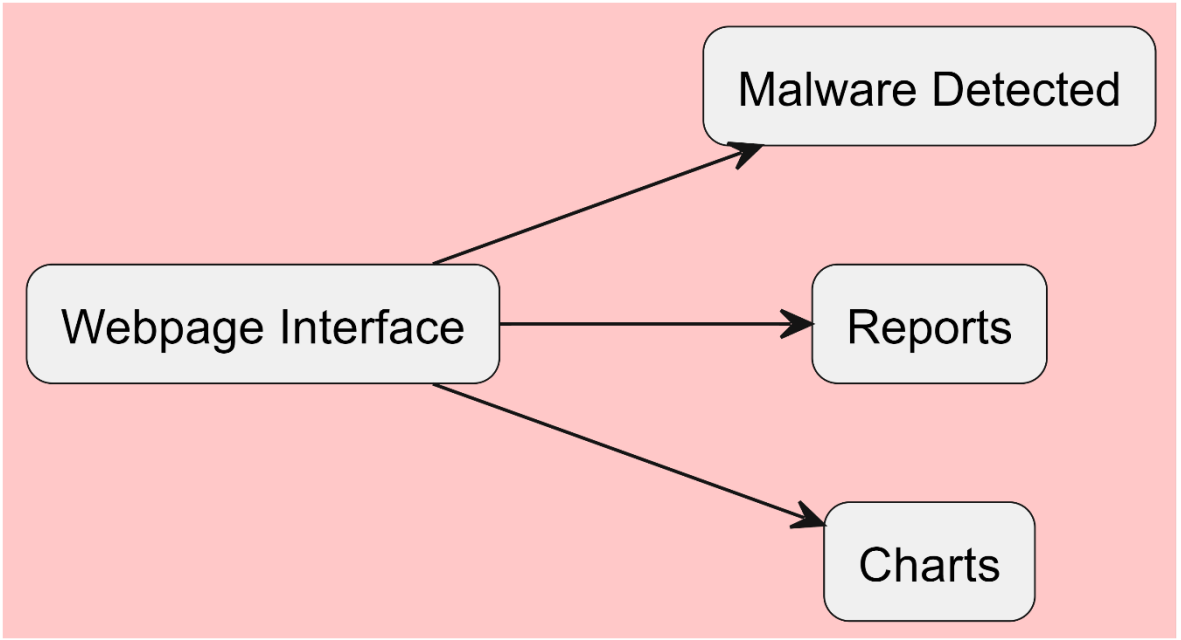


Figure 98 Main page

The main interface looks like this where there is report, malware detection and charts button floating around.

4.5.2 FileUpload

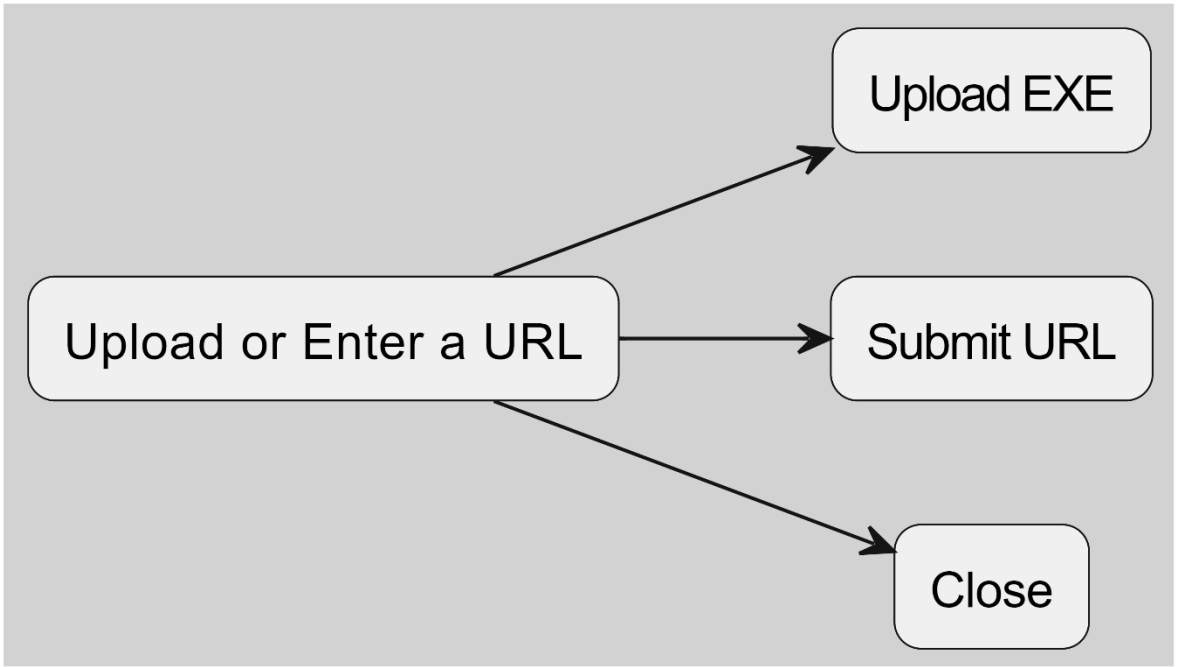


Figure 99 Upload file or URL

This is the interface where i can upload any exe files or any URL's to check if they are phishing or malware.

4.5.3 Charts

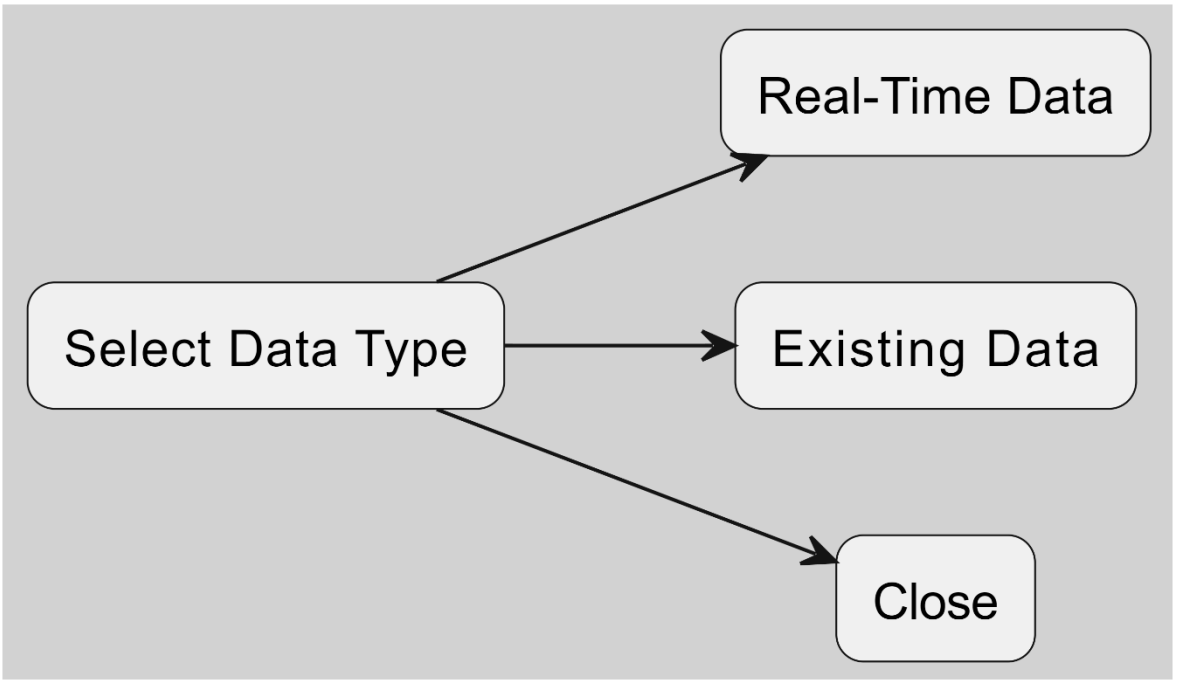


Figure 100 Charts

Here in the charts category you can check for real time data from the user or the existing data from the csv file itself.

4.5.4 Real-time data

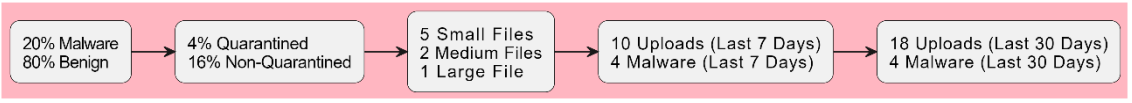


Figure 101 Display Charts

The real time data of the charts on how many malware files and so on.

4.5.5 Reports

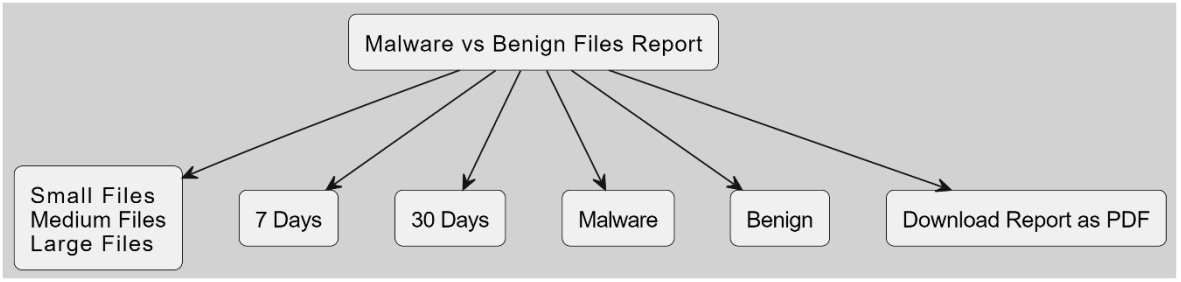


Figure 102 Display report

The report button where it displays the number of malware and benign files uploaded within 7 to 30 days including the sizes of each files from big to small.

4.6 Execution

4.6.1 Backend and Frontend

4.6.1.1 App.js (Frontend code)

```
You, 3 hours ago | 1 author (You)
import React, { useEffect, useRef, useState } from 'react';
import './App.css';  // Import the App.css file for styling
import * as THREE from 'three';
import { GLTFLoader } from 'three/examples/jsm/loaders/GLTFLoader';    "GLTF": Unknown word.
import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls';
import { EffectComposer } from 'three/examples/jsm/postprocessing/EffectComposer';
import { RenderPass } from 'three/examples/jsm/postprocessing/RenderPass';
import { UnrealBloomPass } from 'three/examples/jsm/postprocessing/UnrealBloomPass';
import axios from 'axios';  // For file upload
import Charts from './components/Charts';  // Correct path to Charts.js in components folder
import ExistingChart from './components/ExistingChart'; // Import the ExistingChart component
import ReportPage from './components/ReportPage'; // Import the ReportPage component

function App() {
  const canvasRef = useRef(null);
  const stationRef = useRef(null);
  const buttonRef1 = useRef(null); // Ref for Malware Detection
  const buttonRef2 = useRef(null); // Ref for Charts
  const buttonRef3 = useRef(null); // Ref for Reports
  const cameraRef = useRef(null);
  const controlsRef = useRef(null); // Ref for OrbitControls
  const [isModalOpen, setIsModalOpen] = useState(false); // State to track Malware Detection modal visibility
  const [isChartModalOpen, setIsChartModalOpen] = useState(false); // Track Charts modal visibility
  const [isBackgroundFaded, setIsBackgroundFaded] = useState(false); // State to handle background fade
  const [isUploading, setIsUploading] = useState(false); // Loading state
  const [uploadResult, setUploadResult] = useState(''); // To display result after file upload
  const [error, setError] = useState(''); // To handle errors
  const [showCharts, setShowCharts] = useState(false); // To toggle charts visibility
  const [showExistingChart, setShowExistingChart] = useState(false); // New state for ExistingChart
  const [showReportPage, setShowReportPage] = useState(false); // New state for ReportPage
  const angleRef1 = useRef(0); // Angle for button 1
  const angleRef2 = useRef(0); // Angle for button 2
  const angleRef3 = useRef(0); // Angle for button 3
  const [url, setUrl] = useState(''); // State to handle URL input
  const [urlPrediction, setUrlPrediction] = useState(''); // Store the prediction result
  const [isUrlTabOpen, setIsUrlTabOpen] = useState(false); // Handle the URL tab visibility

  // Function to close the malware modal
  const closeModal = () => {
```

Figure 103 App.js

This React application integrates 3D animations using **Three.js** and offers a variety of features such as file uploads for malware detection, URL submission for phishing detection, and chart and report generation. The main visual feature is a **space station model** that is rendered and animated on the page. This model rotates slowly, providing a dynamic backdrop, and is illuminated by ambient and point lighting to make it visually appealing. The **OrbitControls** allow users to interact with the 3D scene by rotating the view, while the **UnrealBloomPass** post-processing effect adds a glowing bloom effect, enhancing the futuristic look of the app.

The app's user interface includes three buttons. **Malware Detection**, **Charts**, and **Reports** which are positioned in **elliptical motion** around the screen. These buttons orbit the center of the page, adding a unique interactive element to the interface. Each button triggers a different action. The **Malware Detection** button opens a modal where users can upload .exe files for malware scanning or submit URLs for phishing detection. When a file is uploaded, it is sent to a **Flask backend** for scanning, and the result (either "Malware" or "Benign") is displayed in the modal. The app also handles errors, such as when the file has already been quarantined, by displaying appropriate error messages to the user.

```
// Function to close the malware modal
const closeModal = () => {
  setIsModalOpen(false);
  setIsBackgroundFaded(false); // Remove background fade
};

// Function to close the Charts modal
const closeChartModal = () => {
  setIsChartModalOpen(false);
  setIsBackgroundFaded(false);
};

// Function to open the charts modal
const openChartsModal = () => {
  setIsChartModalOpen(true);
  setIsModalOpen(false); // Close other modals if they are open
  setIsBackgroundFaded(true); // Optionally fade background
};

const submitUrl = async () => {
  try {
    const response = await axios.post('http://localhost:5000/predict', { url });
    setUrlPrediction(`Prediction: ${response.data.result}`);
  } catch (error) {
    setUrlPrediction('Phishing.');
    console.error(error);
  }
};
```

Figure 104 App.js

```
// Function to toggle the URL input tab visibility
const toggleUrlTab = () => setIsUrlTabOpen(!isUrlTabOpen);

// Function to trigger file upload for malware detection
const triggerFileUpload = () => {
  const inputElement = document.createElement('input');
  inputElement.type = 'file';
  inputElement.accept = '.exe';
  inputElement.style.display = 'none'; // Hidden input element

  // Handle file selection and upload
  inputElement.onchange = async (event) => {
    const file = event.target.files[0];
    if (file) {
      await handleFileUpload(file);
    }

    // Clear the input value so the same file can be uploaded again
    inputElement.value = '';  // This ensures the input is cleared
  };

  inputElement.click(); // Trigger file input dialog
};

// Handle file upload to the backend for malware detection
const handleFileUpload = async (file) => {
  setIsUploading(true);
  setError('');
  setUploadResult('');

  const formData = new FormData();
  formData.append('file', file);
```

Figure 105 App.js

```
  try {
    const response = await axios.post('http://localhost:5000/upload', formData, {
      headers: { 'Content-Type': 'multipart/form-data' },
    });

    setUploadResult(`File scanned successfully: ${response.data.result}`);
  } catch (error) {
    if (error.response && error.response.data && error.response.data.error) {
      setError(error.response.data.error);  // This will display "This file has already been quarantined."
    } else {
      setError('Error uploading file. Please try again.');
    }
  } finally {
    setIsUploading(false);
  }
};

useEffect(() => {
  const scene = new THREE.Scene();
  scene.background = new THREE.Color(0x000000); // Set background to black for space effect

  // Setup Camera
  const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);
  camera.position.set(0, 5, 50);
  cameraRef.current = camera;

  // Renderer
  const renderer = new THREE.WebGLRenderer({
    canvas: canvasRef.current,
    antialias: true,
  });
  renderer.setSize(window.innerWidth, window.innerHeight);
  renderer.setPixelRatio(window.devicePixelRatio);
```

Figure 106 App.js

The **Charts** button opens another modal that allows users to choose between viewing **Real-Time Data** or **Existing Data** charts. Based on their selection, the app renders the corresponding chart component, displaying relevant data. The **Reports** button zooms the camera in on the space station and displays the **ReportPage**, where users can view detailed statistics or generate downloadable reports. These reports are likely based on previously uploaded files and their analysis results, allowing users to track their file scanning history.

In terms of state management, the app tracks several important aspects such as whether a **modal is open**, whether the background should be **faded**, and whether a file is currently being **uploaded**. It also stores results from the file scanning and phishing detection processes and toggles between different visual components (charts, reports) based on user interaction. For example, after submitting a URL for phishing detection, the app displays the result (e.g., "Phishing" or "Safe") within the modal.

```
// Post-processing for bloom effects
const composer = new EffectComposer(renderer);
const renderPass = new RenderPass(scene, camera);
composer.addPass(renderPass);

const bloomPass = new UnrealBloomPass(new THREE.Vector2(window.innerWidth, window.innerHeight), 1.5, 0.4, 0
composer.addPass(bloomPass);

// Lighting
const ambientLight = new THREE.AmbientLight(0xffffff, 0.6);
scene.add(ambientLight);

const pointLight = new THREE.PointLight(0xffffff, 1.2);
pointLight.position.set(50, 50, 50);
scene.add(pointLight);

// Load the space station model
const loader = new GLTFLoader();    "GLTF": Unknown word.
loader.load('/models/space_station/scene.gltf', (gltf) => {    "gltf": Unknown word.
  const spaceStation = gltf.scene;    "gltf": Unknown word.
  spaceStation.scale.set(6, 6, 6); // Increase the size of the station
  stationRef.current = spaceStation;
  scene.add(spaceStation);
```

Figure 107 App.js

```
// Orbit Controls
const controls = new OrbitControls(camera, renderer.domElement);
controls.enableDamping = true;
controls.dampingFactor = 0.1;
controls.autoRotate = true;
controls.autoRotateSpeed = 0.5; // Default rotation speed
controls.enableZoom = true;
controls.enablePan = false;
controlsRef.current = controls;

// Function to create an elliptical orbit for each button
const createEllipticalMotion = (buttonRef, angleRef, centerX, centerY, radiusX, radiusY, speed) => {
  const animate = () => {
    requestAnimationFrame(animate);

    // Increase the angle based on the speed
    angleRef.current -= speed;

    // Calculate new position using elliptical motion formula
    const angle = angleRef.current;
    const offsetX = Math.cos(angle) * radiusX;
    const offsetY = Math.sin(angle) * radiusY;

    // Apply the elliptical movement to the button by adjusting its absolute position (left and top)
    if (buttonRef.current) {
      buttonRef.current.style.left = `${centerX + offsetX}px`;
      buttonRef.current.style.top = `${centerY + offsetY}px`;
    }

    controls.update();
    composer.render();
  };

  animate(); // Start the animation
};
```

Figure 108 App.js

The app's **3D animation** is constantly rendered in the background, and the camera zooms into specific positions depending on which button is clicked. This adds a fluid transition between the different sections of the app and maintains the space station as a central visual element throughout the user experience. Additionally, the app adjusts dynamically to changes in window size, ensuring a responsive and visually consistent interface across devices.

```
// Center of rotation for the buttons
const centerX = window.innerWidth / 2; // Keep it in the center of the screen
const centerY = window.innerHeight / 2;

// Apply elliptical motion for each button with different parameters (radiusX, radiusY)
createEllipticalMotion(buttonRef1, angleRef1, centerX, centerY, 500, 150, 0.0010); // Malware Detection
createEllipticalMotion(buttonRef2, angleRef2, centerX, centerY, 800, 200, 0.0005); // Charts
createEllipticalMotion(buttonRef3, angleRef3, centerX, centerY, 620, 250, 0.0006); // Reports

// Function to handle button click events (for zoom and background fade)
const handleButtonClick = (targetCameraPosition, targetZoomSpeed, clickedButtonRef) => {
  // Slow down the station's rotation but keep it rotating
  controls.autoRotateSpeed = 0.05;

  // Perform a one-time camera zoom animation
  const initialPosition = camera.position.clone(); // Store the initial camera position
  let progress = 0;

  const zoomAnimation = () => {
    if (progress < 1) {
      // Linear interpolation from the initial position to the target position
      camera.position.lerpVectors(initialPosition, targetCameraPosition, progress);    "lerp": Unknown wo
      progress += targetZoomSpeed; // Increment the zoom speed (higher value zooms faster)
      requestAnimationFrame(zoomAnimation); // Continue the animation until progress reaches 1
    }
  };
```

Figure 109 App.js

```
  zoomAnimation(); // Start the zoom animation

  // Fade out other buttons except the clicked one
  if (buttonRef1.current !== clickedButtonRef) {
    buttonRef1.current.classList.add('fade-out');
  }
  if (buttonRef2.current !== clickedButtonRef) {
    buttonRef2.current.classList.add('fade-out');
  }
  if (buttonRef3.current !== clickedButtonRef) {
    buttonRef3.current.classList.add('fade-out');
  }

  // Show modal when "Malware Detection" button is clicked
  if (clickedButtonRef === buttonRef1.current) {
    setIsModalOpen(true);
    setIsBackgroundFaded(true);
  }

  // Show ReportPage when "Reports" button is clicked
  if (clickedButtonRef === buttonRef3.current) {
    setShowReportPage(true); // Show the ReportPage
  }
};

// Attach click events to buttons
buttonRef1.current.onclick = () => handleButtonClick(new THREE.Vector3(10, 5, 10), 0.02, buttonRef1.current);
buttonRef2.current.onclick = () => openChartsModal(); // Open charts modal on button click
buttonRef3.current.onclick = () => handleButtonClick(new THREE.Vector3(5, 2, 5), 0.02, buttonRef3.current);
```

Figure 110 App.js

```
  // Handle window resizing
  window.addEventListener('resize', () => {
    const width = window.innerWidth;
    const height = window.innerHeight;
    renderer.setSize(width, height);
    composer.setSize(width, height);
    camera.aspect = width / height;
    camera.updateProjectionMatrix();
  });

  return () => {
    window.removeEventListener('resize', () => { });
  };
}, []);

return (
  <div className={`App ${isBackgroundFaded ? 'background-faded' : ''}`}>
    <canvas ref={canvasRef} className="canvas-background"></canvas>

    <div className="content">


      <div className="orbiting-buttons">
        <button className="circle-button" ref={buttonRef1}><span>Malware Detection</span></button>
        <button className="circle-button" ref={buttonRef2}><span>Charts</span></button>
        <button className="circle-button" ref={buttonRef3}><span>Reports</span></button>
      </div>
    </div>
    You, 14 seconds ago • Uncommitted changes
```

Figure 111 App.js

```
    You, 14 seconds ago • Uncommitted changes
    {/* Malware Detection Modal */}
    {isModalOpen && (
      <div className="modal">
        <div className="modal-content">
          <h2>Upload or Enter a URL</h2>
          <div className="upload-buttons">
            <div className="upload-circle">
              <div className="progress-circle">
                <button
                  className="circle-upload-btn"
                  onClick={triggerFileUpload} // Trigger file upload when clicked
                >
                  {isUploading ? 'Uploading...' : 'Upload EXE'}
                </button>
                <svg className="circle-svg" viewBox="0 0 100 100">
                  <circle cx="50" cy="50" r="45"></circle>
                </svg>
              </div>
            </div>

            <div className="upload-circle">
              <div className="progress-circle">
                <button className="circle-upload-btn" onClick={toggleUrlTab}>Submit URL</button>
                <svg className="circle-svg" viewBox="0 0 100 100">
                  <circle cx="50" cy="50" r="45"></circle>
                </svg>
              </div>
            </div>
          </div>
```

Figure 112 App.js

```
        {/* URL input tab (shown on button click) */}
        {isUrlTabOpen && (
          <div className="modal-url-input">
            <input
              type="text"
              placeholder="Enter URL"
              value={url}
              onChange={(e) => setUrl(e.target.value)} // Handle input changes
              style={{ padding: '10px', borderRadius: '5px', width: '150px' }}
            />
            <button className="circle-upload-btn" onClick={submitUrl}>Submit</button>
          </div>
        )}

        {/* Display result or error message */}
        {urlPrediction && <p style={{ color: 'green' }}>{urlPrediction}</p>}
        {uploadResult && <p style={{ color: 'green' }}>{uploadResult}</p>}
        {error && <p style={{ color: 'red' }}>{error}</p>}

        <button className="close-modal" onClick={closeModal}>Close</button>
      </div>
    </div>
  )}
```

Figure 113 App.js

```
{/* Chart Modal for "Real-Time Data" and "Existing Data" */}
{isChartModalOpen && (
  <div className="modal">
    <div className="modal-content">
      <h2>Select Data Type</h2>
      <div className="upload-buttons">
        <div className="upload-circle">
          <div className="progress-circle">
            <button className="circle-upload-btn" onClick={() => {
              setShowCharts(true);  // Set showCharts to true to show the charts
              setIsChartModalOpen(false);  // Close the modal
            }}>
              Real-Time Data
            </button>
            {/* Add circular progress bar */}
            <svg className="circle-svg" viewBox="0 0 100 100">
              <circle cx="50" cy="50" r="45"></circle>
            </svg>
          </div>
        </div>
```

Figure 114 App.js

```
          <div className="upload-circle">
            <div className="progress-circle">
              <button className="circle-upload-btn" onClick={() => {
                setShowExistingChart(true);  // Set showExistingChart to true to show the ExistingChart
                setIsChartModalOpen(false);  // Close the modal
              }}>
                Existing Data
              </button>
              {/* Add circular progress bar */}
              <svg className="circle-svg" viewBox="0 0 100 100">
                <circle cx="50" cy="50" r="45"></circle>
              </svg>
            </div>
          </div>
        </div>

        <button className="close-modal" onClick={closeChartModal}>Close</button>
      </div>
    </div>
  )}
```

Figure 115 App.js

The file upload and URL submission functionalities are powered by **Axios**, which sends the data to the Flask backend. The responses from the backend are then displayed to the user within the React app, providing real-time feedback. Overall, this application combines modern web development technologies like **React**, **Three.js**, and **Axios** to create an interactive, visually rich platform for malware detection and data visualization.

4.6.1.2 Train.py (Malware training)

```
# Import necessary libraries
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import joblib    "joblib": Unknown word.
import matplotlib.pyplot as plt
```

[1] ✓ 49.1s Python

```
# Load dataset with the correct delimiter
file_path = r'D:\Malware-Detection-using-Machine-learning-main\Malware-Detection-using-Machine-learning-main\Dataset\data
data = pd.read_csv(file_path, sep='|')  # Correct delimiter used here
```

[11] ✓ 0.5s Python

```
# Display basic info about the dataset
print("Dataset Info:")
data.info()
```

[12] ✓ 0.0s Python

Figure 116 Train.py

```
# Manually set the target column to 'legitimate'
target_column = 'legitimate'
```
[13] ✓ 0.0s Python

```
# Check if target column exists
if target_column in data.columns:
    print(f"\nUsing column '{target_column}' as the target.")
else:
    raise ValueError(f"Target column '{target_column}' not found.")
```
[14] ✓ 0.0s Python

```
...
Using column 'legitimate' as the target.
```

```
# Drop irrelevant columns
X = data.drop(['Name', 'md5', target_column], axis=1)
```
[15] ✓ 0.0s Python

```
# Target column (classification)
Y = data[target_column]
```
[16] ✓ 0.0s Python

Figure 117 Train.py

```
# Split the data into training and test sets
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=1)
```
] ✓ 0.1s Python

```
# Standardize the feature data
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```
] ✓ 0.1s Python

```
# Define the model
input_size = x_train_scaled.shape[1]
hidden_layer_size = 50
output_size = 1  # Binary classification

model = tf.keras.Sequential([
    tf.keras.layers.Dense(hidden_layer_size, input_shape=(input_size,), activation='relu'),    "relu": Unknown word.
    tf.keras.layers.Dense(hidden_layer_size, activation='relu'),    "relu": Unknown word.
    tf.keras.layers.Dense(hidden_layer_size, activation='relu'),    "relu": Unknown word.
    tf.keras.layers.Dense(output_size, activation='sigmoid')  # Sigmoid for binary classification
])
```
] ✓ 0.3s Python

Figure 118 Train.py

```
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])    "crossentropy": Unknown word.
```
[20] ✓ 0.0s Python

```
# Train the model
batch_size = 100
max_epochs = 20
early_stopping = tf.keras.callbacks.EarlyStopping(patience=2)
history = model.fit(x_train_scaled, y_train, batch_size=batch_size, epochs=max_epochs,
                    validation_split=0.2, callbacks=[early_stopping])
```
[21] ✓ 12.2s Python

Figure 119 Train.py

```
# Evaluate the model
test_loss, test_accuracy = model.evaluate(x_test_scaled, y_test)
print(f'Test Loss: {test_loss:.6f}, Test Accuracy: {test_accuracy * 100:.2f}%')
```
22] ✓ 0.7s Python

```
863/863 [==============================] - 1s 744us/step - loss: 0.0345 - accuracy: 0.9892
Test Loss: 0.034498, Test Accuracy: 98.92%
```

Figure 120 Train.py

```
# Plot accuracy and loss
plt.figure(figsize=(14, 6))    "figsize": Unknown word.

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy Over Epochs')
plt.xlabel('Epoch')    "xlabel": Unknown word.
plt.ylabel('Accuracy')    "ylabel": Unknown word.
plt.legend(loc='lower right')
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss Over Epochs')
plt.xlabel('Epoch')    "xlabel": Unknown word.
plt.ylabel('Loss')    "ylabel": Unknown word.
plt.legend(loc='upper right')
plt.grid(True)

plt.tight_layout()
plt.show()
```
] ✓ 1.5s

Figure 121 Train.py

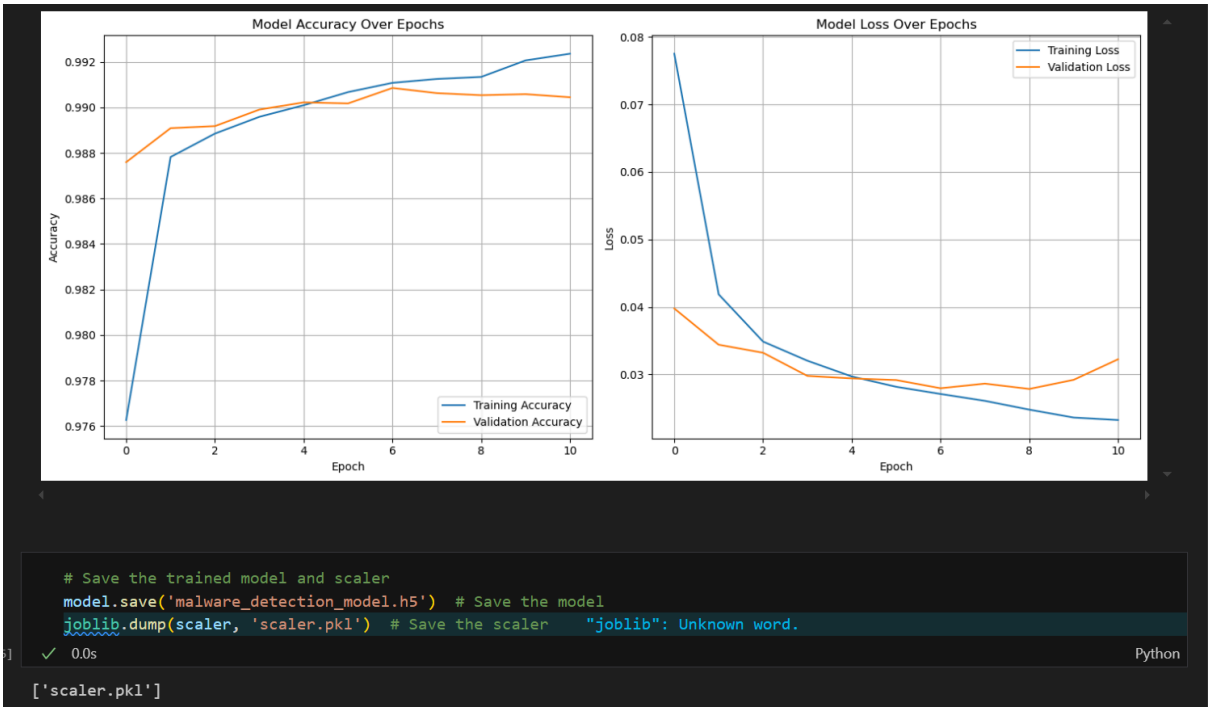


Figure 122 Train.py

The notebook begins by importing several important libraries that are commonly used for data preprocessing, machine learning, and model training. These include pandas and numpy,

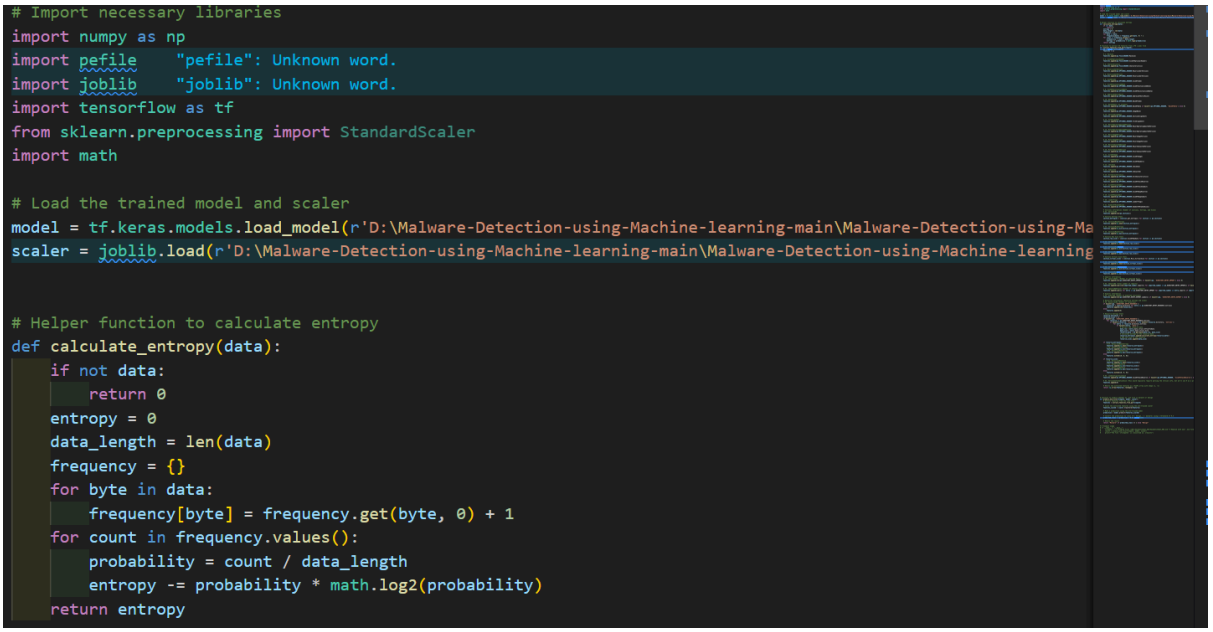


Figure 123 Train.py

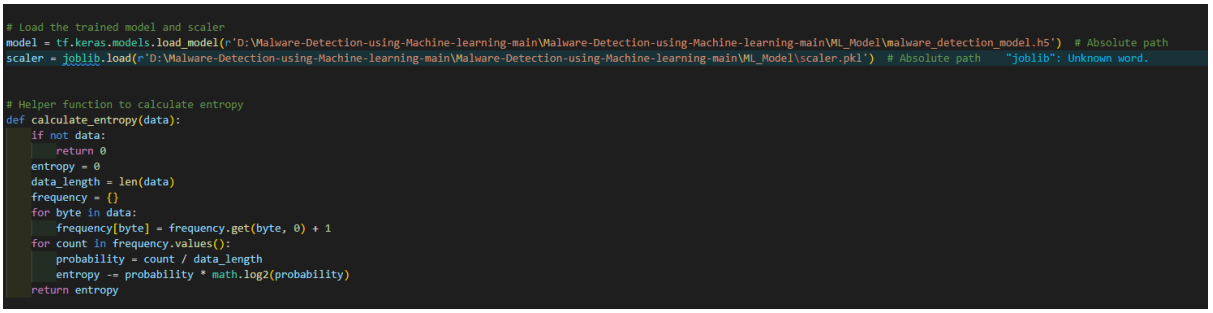
The notebook begins by importing several important libraries that are commonly used for data preprocessing, machine learning, and model training. These include pandas and numpy,

which are essential for handling data and performing numerical operations. The presence of tensorflow suggests that the notebook involves deep learning or neural network techniques, likely for malware detection purposes. In addition to these, sklearn.model_selection is used for splitting data into training and testing sets, while sklearn.preprocessing is employed for normalizing or scaling the data to ensure consistent input for the model. The use of joblib indicates that the notebook will also include saving and loading trained models for future use. The plotting library matplotlib.pyplot is included, hinting at possible visualizations or graphs that will be generated during the analysis.

The next significant part of the notebook involves loading a dataset from the specified path, D:\\Malware-Detection-using-Machine-learning-main\\Malware-Detection-using-Machine-learning-main\\Dataset\\data.csv, using pandas.read_csv. This indicates that the dataset likely contains features related to malware detection, and it is imported into a pandas DataFrame for further analysis and manipulation. The choice of CSV format suggests that the data is tabular, with rows representing samples and columns representing features. Given that the dataset is loaded early in the notebook, the following steps would likely involve exploring, cleaning, and possibly engineering new features from this data before applying machine learning techniques.

Based on the imported libraries and the structure of the code, it seems the notebook will apply some form of machine learning model, possibly a neural network using TensorFlow, to perform malware detection. Before reaching that point, the data will likely undergo preprocessing steps such as scaling, feature selection, and splitting into training and test sets. Once the model is trained, the notebook might also save the model using joblib for future use. Additionally, the use of matplotlib suggests that there will be visualizations to help understand the data or the model's performance, such as loss curves, accuracy plots, or confusion matrices.

4.6.1.3 PEmain.py (Extract features for PE files to predict)

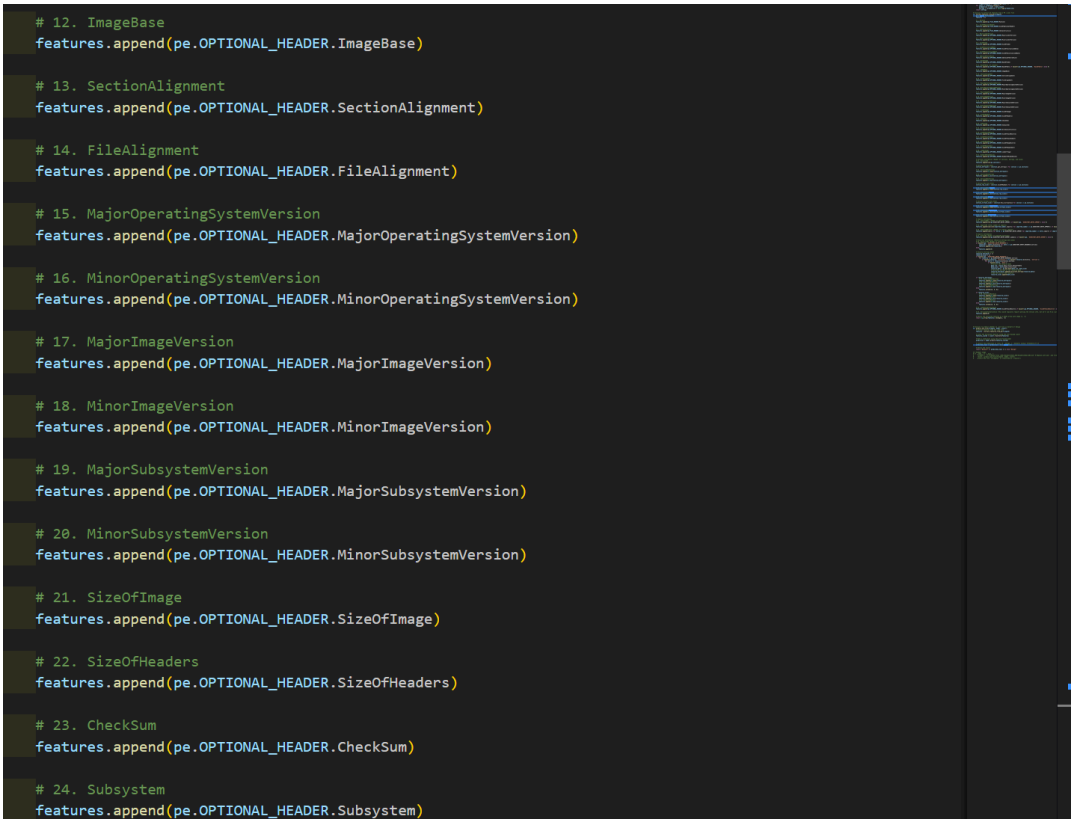


```
# Load the trained model and scaler
model = tf.keras.models.load_model(r'D:\Malware-Detection-using-Machine-learning-main\Malware-Detection-using-Machine-learning-main\ML_Model\malware_detection_model.h5')  # Absolute path
scaler = joblib.load(r'D:\Malware-Detection-using-Machine-learning-main\Malware-Detection-using-Machine-learning-main\ML_Model\scaler.pkl')  # Absolute path    "joblib": Unknown word.


# Helper function to calculate entropy
def calculate_entropy(data):
    if not data:
        return 0
    entropy = 0
    data_length = len(data)
    frequency = {}
    for byte in data:
        frequency[byte] = frequency.get(byte, 0) + 1
    for count in frequency.values():
        probability = count / data_length
        entropy -= probability * math.log2(probability)
    return entropy
```

Figure 124 PEmain.py

```python
# Function to extract 54 features from a PE (.exe) file
def extract_features_from_pe(filepath):
    pe = pefile.PE(filepath)    "pefile": Unknown word.
    features = []

    # 1. Machine
    features.append(pe.FILE_HEADER.Machine)

    # 2. SizeOfOptionalHeader
    features.append(pe.FILE_HEADER.SizeOfOptionalHeader)

    # 3. Characteristics
    features.append(pe.FILE_HEADER.Characteristics)

    # 4. MajorLinkerVersion
    features.append(pe.OPTIONAL_HEADER.MajorLinkerVersion)

    # 5. MinorLinkerVersion
    features.append(pe.OPTIONAL_HEADER.MinorLinkerVersion)

    # 6. SizeOfCode
    features.append(pe.OPTIONAL_HEADER.SizeOfCode)

    # 7. SizeOfInitializedData
    features.append(pe.OPTIONAL_HEADER.SizeOfInitializedData)

    # 8. SizeOfUninitializedData
    features.append(pe.OPTIONAL_HEADER.SizeOfUninitializedData)

    # 9. AddressOfEntryPoint
    features.append(pe.OPTIONAL_HEADER.AddressOfEntryPoint)

    # 10. BaseOfCode
    features.append(pe.OPTIONAL_HEADER.BaseOfCode)

    # 11. BaseOfData (if present)
    features.append(pe.OPTIONAL_HEADER.BaseOfData if hasattr(pe.OPTIONAL_HEADER, 'BaseOfData') else 0)
```

Figure 125 PEmain.py



```
# 12. ImageBase
features.append(pe.OPTIONAL_HEADER.ImageBase)

# 13. SectionAlignment
features.append(pe.OPTIONAL_HEADER.SectionAlignment)

# 14. FileAlignment
features.append(pe.OPTIONAL_HEADER.FileAlignment)

# 15. MajorOperatingSystemVersion
features.append(pe.OPTIONAL_HEADER.MajorOperatingSystemVersion)

# 16. MinorOperatingSystemVersion
features.append(pe.OPTIONAL_HEADER.MinorOperatingSystemVersion)

# 17. MajorImageVersion
features.append(pe.OPTIONAL_HEADER.MajorImageVersion)

# 18. MinorImageVersion
features.append(pe.OPTIONAL_HEADER.MinorImageVersion)

# 19. MajorSubsystemVersion
features.append(pe.OPTIONAL_HEADER.MajorSubsystemVersion)

# 20. MinorSubsystemVersion
features.append(pe.OPTIONAL_HEADER.MinorSubsystemVersion)

# 21. SizeOfImage
features.append(pe.OPTIONAL_HEADER.SizeOfImage)

# 22. SizeOfHeaders
features.append(pe.OPTIONAL_HEADER.SizeOfHeaders)

# 23. CheckSum
features.append(pe.OPTIONAL_HEADER.CheckSum)

# 24. Subsystem
features.append(pe.OPTIONAL_HEADER.Subsystem)
```

Figure 126 PEmain.py

This Python script focuses on detecting whether an .exe file is malware or benign using machine learning. It utilizes libraries such as numpy, pefile, and tensorflow for feature extraction and classification. The script begins by loading a pre-trained machine learning model using TensorFlow (malware_detection_model.h5) and a pre-fitted scaler using joblib. These are necessary for making predictions and transforming input data into a suitable format for the model.

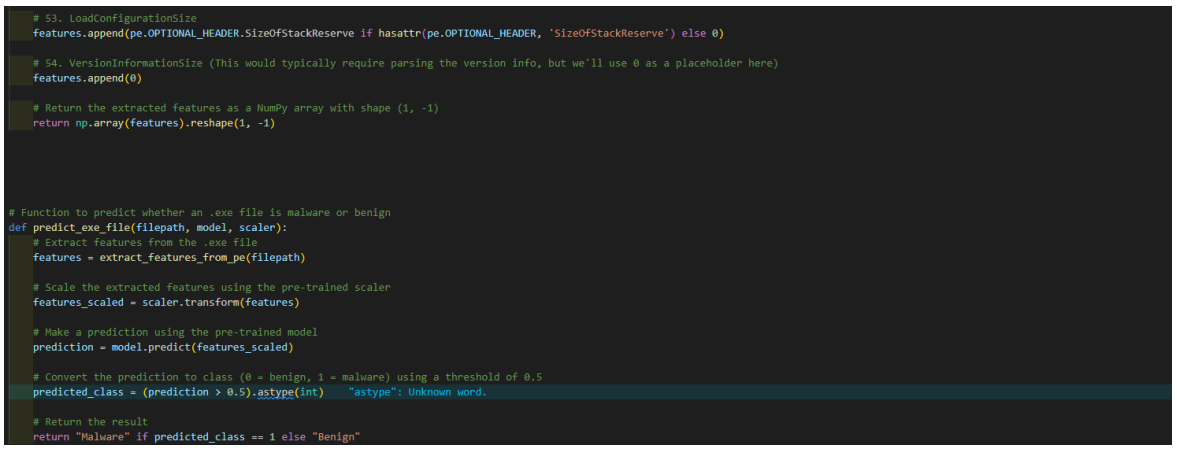
A helper function, calculate_entropy, is implemented to compute the entropy of a given dataset. Entropy measures the randomness of data, which can be a crucial feature when identifying obfuscation techniques used in malware. The function takes a sequence of bytes, calculates the frequency of each unique byte, and computes the entropy using Shannon's formula.

```python
# Resource Entropy Stats
resource_entropies = []
resource_sizes = []
if hasattr(pe, 'DIRECTORY_ENTRY_RESOURCE'):
    for resource in pe.DIRECTORY_ENTRY_RESOURCE.entries:
        if hasattr(resource, 'directory') and hasattr(resource.directory, 'entries'):
            for entry in resource.directory.entries:
                if hasattr(entry, 'data'):
                    # Extract the resource data
                    data_rva = entry.data.struct.OffsetToData
                    data_size = entry.data.struct.Size
                    resource_data = pe.get_data(data_rva, data_size)
                    # Calculate entropy of the resource data
                    resource_entropies.append(calculate_entropy(resource_data))
                    # Add the size of the resource
                    resource_sizes.append(data_size)

if resource_entropies:
    # 47. ResourcesMeanEntropy
    features.append(np.mean(resource_entropies))
    # 48. ResourcesMinEntropy
    features.append(np.min(resource_entropies))
    # 49. ResourcesMaxEntropy
    features.append(np.max(resource_entropies))
else:
    features.extend([0, 0, 0])

if resource_sizes:
    # 50. ResourcesMeanSize
    features.append(np.mean(resource_sizes))
    # 51. ResourcesMinSize
    features.append(np.min(resource_sizes))
    # 52. ResourcesMaxSize
    features.append(np.max(resource_sizes))
else:
    features.extend([0, 0, 0])
```

Figure 127 PEmain.py

The core functionality of the script lies in the function extract_features_from_pe, which extracts 54 features from a Portable Executable (PE) file. This function uses the pefile library to inspect different attributes of the file's headers, sections, and resources. The extracted features range from the machine type, version numbers, image size, and section information to imports, exports, and resources. For example, it computes section entropy, section sizes, and import/export information, which are essential for malware detection because malware often uses specific patterns in these fields.

Once the features are extracted from the .exe file, they are fed into the predict_exe_file function. This function first scales the features using a pre-fitted scaler, which ensures that the model receives input in a consistent format. Then, the features are passed to the pre-trained TensorFlow model to make predictions. The model outputs a probability value, and if the value is greater than 0.5, the file is classified as "Malware"; otherwise, it is classified as "Benign".



```
    # 53. LoadConfigurationSize
    features.append(pe.OPTIONAL_HEADER.SizeOfStackReserve if hasattr(pe.OPTIONAL_HEADER, 'SizeOfStackReserve') else 0)

    # 54. VersionInformationSize (This would typically require parsing the version info, but we'll use 0 as a placeholder here)
    features.append(0)

    # Return the extracted features as a NumPy array with shape (1, -1)
    return np.array(features).reshape(1, -1)



# Function to predict whether an .exe file is malware or benign
def predict_exe_file(filepath, model, scaler):
    # Extract features from the .exe file
    features = extract_features_from_pe(filepath)

    # Scale the extracted features using the pre-trained scaler
    features_scaled = scaler.transform(features)

    # Make a prediction using the pre-trained model
    prediction = model.predict(features_scaled)

    # Convert the prediction to class (0 = benign, 1 = malware) using a threshold of 0.5
    predicted_class = (prediction > 0.5).astype(int)    "astype": Unknown word.

    # Return the result
    return "Malware" if predicted_class == 1 else "Benign"
```

Figure 128 PEmain.py

Overall, this code extracts meaningful features from a PE file, applies a machine learning model to those features, and predicts whether the file is malicious or not. This approach relies on static analysis, where features from the file itself are used for classification without executing the file.

4.6.1.4 App.py (To connect with the frontend)

```
import os
import shutil
from flask import Flask, request, jsonify    "jsonify": Unknown word.
from werkzeug.utils import secure_filename    "werkzeug": Unknown word.
from flask_cors import CORS
import sqlite3
from datetime import datetime, timedelta
from dotenv import load_dotenv    "dotenv": Unknown word.

# Import functions from PE_main.py
from PE_main import predict_exe_file, model, scaler

# Load environment variables
load_dotenv()    "dotenv": Unknown word.

# Initialize Flask app
app = Flask(__name__)
CORS(app)

app.secret_key = os.getenv('SECRET_KEY', 'supersecretkey')    "supersecretkey": Unknown word.
UPLOAD_FOLDER = os.getenv('UPLOAD_FOLDER', 'uploads')
QUARANTINE_FOLDER = os.getenv('QUARANTINE_FOLDER', 'quarantine')
BENIGN_FOLDER = os.getenv('BENIGN_FOLDER', 'benign_files')
DATABASE_FILE = os.getenv('DATABASE_FILE', 'database.db')

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['QUARANTINE_FOLDER'] = QUARANTINE_FOLDER
app.config['BENIGN_FOLDER'] = BENIGN_FOLDER
app.config['MAX_CONTENT_LENGTH'] = 2 * 1024 * 1024 * 1024  # 2GB max file size

# Ensure folders exist
def ensure_folders_exist():
    os.makedirs(UPLOAD_FOLDER, exist_ok=True)
    os.makedirs(QUARANTINE_FOLDER, exist_ok=True)
    os.makedirs(BENIGN_FOLDER, exist_ok=True)
```

Figure 129 App.py

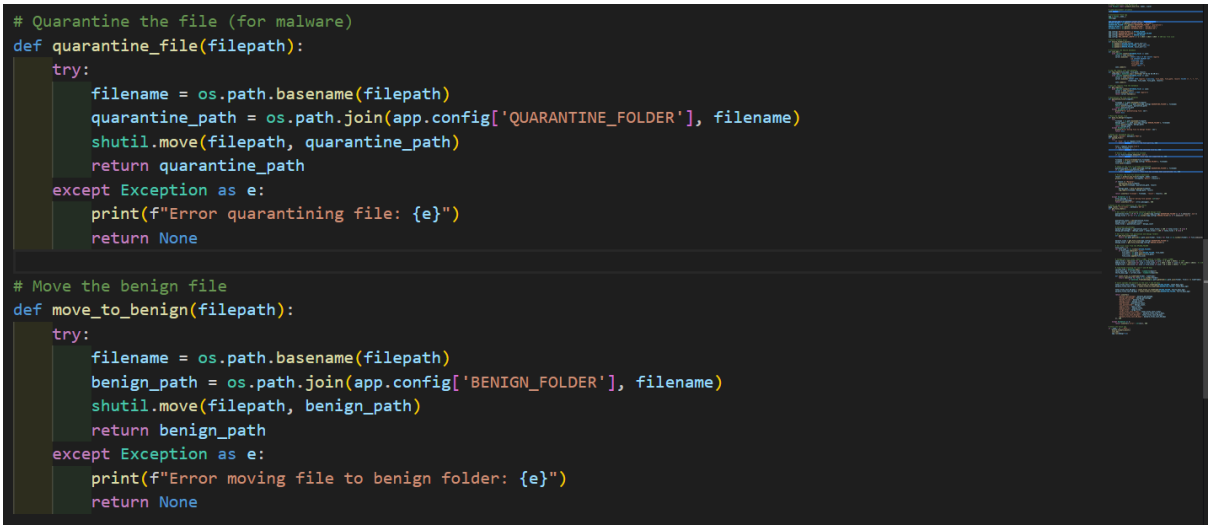
This code is part of a Flask-based web application designed to analyze .exe files, classify them as either malware or benign, and store the results and extracted features in a SQLite database. It allows users to upload executable files, which are then processed by a machine learning model to determine whether the file is malicious or safe. The app also provides detailed information about the files and generates statistics for visual representation in the frontend.

```
# Initialize the SQLite database
def init_db():
    with sqlite3.connect(DATABASE_FILE) as conn:
        cursor = conn.cursor()
        cursor.execute('''CREATE TABLE IF NOT EXISTS reports
                          (id INTEGER PRIMARY KEY,
                           timestamp TEXT,
                           file_name TEXT,
                           file_path TEXT,
                           result TEXT)''')
        conn.commit()

# Log the report into the database
def log_report(file_name, file_path, result):
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    with sqlite3.connect(DATABASE_FILE) as conn:
        cursor = conn.cursor()
        cursor.execute("INSERT INTO reports (timestamp, file_name, file_path, result) VALUES (?, ?, ?, ?)",
                       (timestamp, file_name, file_path, result))
        conn.commit()

# Get all reports from the database
def get_reports():
    with sqlite3.connect(DATABASE_FILE) as conn:
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM reports")
        return cursor.fetchall()
```

Figure 130 App.py

The application begins by importing necessary libraries and modules. These include basic modules like os and shutil for file management, Flask and jsonify to handle web requests and responses, and sqlite3 to manage the SQLite database. Additionally, it imports the predict_exe_file, extract_features_from_pe, model, and scaler from an external file called PE_main.py, which likely contains the machine learning logic for file classification.

The application is configured to use environment variables for setting up the upload directory, quarantine folder, and benign folder. These folders are used to organize uploaded files based on whether they are classified as malware or benign. The app is also set to allow file uploads with a maximum size of 2GB, which is appropriate for most executable files.



```python
# Quarantine the file (for malware)
def quarantine_file(filepath):
    try:
        filename = os.path.basename(filepath)
        quarantine_path = os.path.join(app.config['QUARANTINE_FOLDER'], filename)
        shutil.move(filepath, quarantine_path)
        return quarantine_path
    except Exception as e:
        print(f"Error quarantining file: {e}")
        return None

# Move the benign file
def move_to_benign(filepath):
    try:
        filename = os.path.basename(filepath)
        benign_path = os.path.join(app.config['BENIGN_FOLDER'], filename)
        shutil.move(filepath, benign_path)
        return benign_path
    except Exception as e:
        print(f"Error moving file to benign folder: {e}")
        return None
```

Figure 131 App.py

One of the key functions in the app is ensure_folders_exist(), which ensures that the necessary directories (uploads, quarantine, and benign) exist on the server. This function creates these directories if they are missing, preventing errors when files are uploaded or moved. Similarly, the init_db() function sets up the SQLite database and checks if the reports table exists. If not, it creates the table, adding a column to store the features of each file as a JSON string. This ensures that the database can store detailed information about each uploaded file, including the results of its analysis.

The log_report() function logs the details of each uploaded file into the database. This includes the filename, file path, the classification result (malware or benign), and the extracted features of the file in JSON format. By logging these details, the app can later retrieve and display the history of file uploads, which is useful for generating reports or displaying statistics on the frontend.

There are additional helper functions that handle file movement after the analysis is complete. The quarantine_file() function moves malware files to the quarantine folder, while the move_to_benign() function moves safe files to the benign folder. These functions ensure that files are organized properly after classification, with malicious files being isolated from benign ones.

```python
# Route for uploading the file
@app.route('/upload', methods=['POST'])
def upload_file():
    try:
        if 'file' not in request.files:
            return jsonify({'error': 'No file part'}), 400    "jsonify": Unknown word.

        file = request.files['file']
        if file.filename == '':
            return jsonify({'error': 'No selected file'}), 400    "jsonify": Unknown word.

        # Ensure only .exe files are allowed
        if not file.filename.endswith('.exe'):
            return jsonify({'error': 'File type not supported'}), 400    "jsonify": Unknown word.

        filename = secure_filename(file.filename)
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(filepath)

        # Check if the file is already quarantined
        quarantine_path = os.path.join(app.config['QUARANTINE_FOLDER'], filename)
        if os.path.exists(quarantine_path):
            return jsonify({'error': 'This file has already been quarantined.'}), 400    "jsonify": Unknown word

        # Process the file for malware detection
        result = predict_exe_file(filepath, model, scaler)
        print(f"File uploaded: {filepath}, Result: {result}")

        if result == "Malware":
            quarantine_file(filepath)
            log_report(filename, quarantine_path, result)
        else:
            benign_path = move_to_benign(filepath)
            log_report(filename, benign_path, result)

        return jsonify({'filename': filename, 'result': result}), 200
```

Figure 132 App.py

```python
# Route to get file statistics for the charts
@app.route('/file-stats', methods=['GET'])
def get_file_stats():
    try:
        # Get counts of quarantined (malware) and benign files
        quarantine_files = [f for f in os.listdir(app.config['QUARANTINE_FOLDER']) if f.endswith('.exe')]
        benign_files = [f for f in os.listdir(app.config['BENIGN_FOLDER']) if f.endswith('.exe')]


        quarantine_count = len(quarantine_files)
        benign_count = len(benign_files)
        total_files = quarantine_count + benign_count

        # Calculate percentages
        malware_percentage = (quarantine_count / total_files) * 100 if total_files > 0 else 0
        benign_percentage = (benign_count / total_files) * 100 if total_files > 0 else 0

        # Get file sizes from quarantine and benign folders
        def get_file_sizes(folder):
            return [os.path.getsize(os.path.join(folder, file)) for file in os.listdir(folder) if file.endswith(

        malware_sizes = get_file_sizes(app.config['QUARANTINE_FOLDER'])
        benign_sizes = get_file_sizes(app.config['BENIGN_FOLDER'])

        # Get file sizes from the UPLOAD_FOLDER
        file_sizes = []
        for file_name in os.listdir(UPLOAD_FOLDER):
            if file_name.endswith(".exe"):
                file_path = os.path.join(UPLOAD_FOLDER, file_name)
                file_size = os.path.getsize(file_path)
                file_sizes.append(file_size)
```

Figure 133 App.py

One of the main routes in the app is the /upload endpoint, which handles the upload and processing of .exe files. When a user uploads a file, the app first checks if it is a valid .exe file. If the file passes this validation, it is saved to the uploads folder. Next, the app extracts features from the file using the extract_features_from_pe() function. These features are then passed to the machine learning model (predict_exe_file()) to classify the file as malware or benign. Based on the result, the file is either moved to the quarantine folder or the benign folder. The details of the file, including the extracted features and classification result, are then logged into the database using the log_report() function.

In addition to file upload functionality, the app also provides a /file-stats route, which returns statistics about the uploaded files. This includes the percentage of malware and benign files, the sizes of the files, and the number of files uploaded in the last 7 or 30 days. These statistics can be used for data visualization on the frontend, allowing users to see trends in file uploads and their classifications.

```
    # Categorize file sizes: Small (<1MB), Medium (1-10MB), Large (>10MB)
    small_files = len([size for size in file_sizes if size < 1 * 1024 * 1024])  # <1MB
    medium_files = len([size for size in file_sizes if 1 * 1024 * 1024 <= size <= 10 * 1024 * 1024])  # 1-10
    large_files = len([size for size in file_sizes if size > 10 * 1024 * 1024])  # >10MB

    # Time-based tracking for last 7 and 30 days
    current_time = datetime.now()
    seven_days_ago = current_time - timedelta(days=7)
    thirty_days_ago = current_time - timedelta(days=30)

    def count_files_in_timeframe(folder, timeframe):
        return len([file for file in os.listdir(folder)
                    if datetime.fromtimestamp(os.path.getmtime(os.path.join(folder, file))) >= timeframe])

    # Count uploads and malware in the last 7 and 30 days
    total_files_last_7_days = count_files_in_timeframe(UPLOAD_FOLDER, seven_days_ago)
    malware_files_last_7_days = count_files_in_timeframe(QUARANTINE_FOLDER, seven_days_ago)

    total_files_last_30_days = count_files_in_timeframe(UPLOAD_FOLDER, thirty_days_ago)
    malware_files_last_30_days = count_files_in_timeframe(QUARANTINE_FOLDER, thirty_days_ago)

    return jsonify({
        'malware_percentage': malware_percentage,
        'benign_percentage': benign_percentage,
        'malware_sizes': malware_sizes,
        'benign_sizes': benign_sizes,
        'quarantined': quarantine_count,
        'non_quarantined': benign_count,
        'small_files': small_files,
        'medium_files': medium_files,
        'large_files': large_files,
        'total_files_last_7_days': total_files_last_7_days,
        'malware_files_last_7_days': malware_files_last_7_days,
        'total_files_last_30_days': total_files_last_30_days,
        'malware_files_last_30_days': malware_files_last_30_days
    }), 200
```

Figure 134 App.py

Finally, the Flask app is initialized in the if __name__ == '__main__', block. Before starting the server, it ensures that the necessary folders and database are set up correctly. The app runs in debug mode, which makes it easier to troubleshoot during development.

In summary, this Flask web application provides a system for analyzing .exe files using machine learning. It handles file uploads, extracts features, classifies the files, and stores the results in a database. It also generates statistics and organizes the files based on their classification, making it a robust solution for analyzing and managing executable files.

4.6.1.5 Predict.py (Predict URL by extracting the features)

```python
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report, precision_recall_fscore_support, confusion_matrix
import joblib
from collections import Counter
import numpy as np

# Step 1: Load the dataset
df = pd.read_csv(r'D:\Malware-Detection-using-Machine-learning-main\Malware-Detection-using-Machine-learning-main\Dataset

# Step 2: Remove irrelevant columns
columns_to_remove = [
    'FILENAME', 'URL', 'Title', 'HasTitle', 'HasCopyrightInfo', 'HasFavicon', 'Robots',
    'IsResponsive', 'HasDescription', 'HasExternalFormSubmit', 'HasSocialNet',
    'HasSubmitButton', 'HasHiddenFields', 'HasPasswordField', 'Bank', 'Pay', 'Crypto'
]
df = df.drop(columns=columns_to_remove)

# Step 3: Handle missing values (if any)
df.fillna(0, inplace=True)

# Step 4: Convert any non-numeric columns to numeric
from sklearn.preprocessing import LabelEncoder

for column in df.columns:
    if df[column].dtype == 'object':
        le = LabelEncoder()
        df[column] = le.fit_transform(df[column])

# Step 5: Separate features and labels
X = df.drop(columns=['label'])  # Features
y = df['label']  # Labels (1 = phishing, 0 = legitimate)
```

Figure 135 Predict.py

This notebook focuses on building and evaluating machine learning models for detecting phishing URLs. The process starts with extracting various features from the URLs. These features include the length of the URL, the presence and number of special characters, domain-specific attributes such as the top-level domain (TLD) length, and character distributions, like the number of digits, letters, and special characters. These features are important because phishing URLs often differ from legitimate ones in structure and composition. Additionally, features related to the structure of the associated web page are considered, such as the number of images, JavaScript files, and CSS files found on the page,

which help in identifying patterns common in phishing websites. Other critical attributes, such as obfuscation techniques and the ratio of certain characters in the URL, are also calculated, as phishing URLs often use obfuscation methods to deceive users.

```python
# Step 5: Separate features and labels
X = df.drop(columns=['label'])  # Features
y = df['label']  # Labels (1 = phishing, 0 = legitimate)

# Step 6: Split the dataset into training and test sets (70% training, 30% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Step 7: Separate phishing and legitimate samples
X_train_phishing = X_train[y_train == 1]  # Phishing samples
y_train_phishing = y_train[y_train == 1]

X_train_legit = X_train[y_train == 0]  # Legitimate samples
y_train_legit = y_train[y_train == 0]

# Step 8: Oversample phishing URLs (duplicate them to create imbalance)
oversampling_factor = 2  # How many times to duplicate phishing URLs
X_train_phishing_oversampled = pd.concat([X_train_phishing] * oversampling_factor, ignore_index=True)
y_train_phishing_oversampled = pd.concat([y_train_phishing] * oversampling_factor, ignore_index=True)

# Step 9: Combine oversampled
X_train_combined = pd.concat([X_train_phishing_oversampled, X_train_legit], ignore_index=True)
y_train_combined = pd.concat([y_train_phishing_oversampled, y_train_legit], ignore_index=True)

# Check class distribution after manual oversampling
print(f"Class distribution after manual oversampling: {Counter(y_train_combined)}")

# Step 10: Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_combined)
X_test_scaled = scaler.transform(X_test)

# Save the scaler for future use
joblib.dump(scaler, 'new_scaler.pkl')
```

(variable) X_train_phishing_oversampled: DataFrame

Figure 136 Predict.py

Once the features are extracted, the next phase involves training various machine learning models. The notebook uses several models, including Logistic Regression, Random Forest, and Support Vector Machines (SVM), to classify URLs as phishing or legitimate. To ensure that the input features are on a comparable scale, preprocessing techniques like scaling are applied. The dataset used for training contains labeled phishing and legitimate URLs, allowing the models to learn from these examples. The notebook takes care of imbalanced data issues by employing the SMOTE technique, which oversamples the minority class (phishing URLs) by creating synthetic examples, improving model performance on unbalanced datasets.

```python
# Step 11: Define parameter grid for XGBoost tuning
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [5, 6, 7],
    'learning_rate': [0.05, 0.1, 0.2],
    'scale_pos_weight': [1, 2]  # Weight for balancing class
}

# Step 12: Initialize XGBoost model and perform grid search for hyperparameter tuning
xgb = XGBClassifier(random_state=42, use_label_encoder=False, eval_metric='logloss')

grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid, scoring='accuracy', cv=5, verbose=2)
grid_search.fit(X_train_scaled, y_train_combined)

# Get the best model after tuning
best_model = grid_search.best_estimator_
print(f"Best parameters found: {grid_search.best_params_}")

# Save the best XGBoost model
joblib.dump(best_model, 'xgboost_model_manual_oversampled.pkl')

# Step 13: Evaluate model performance on the test set
y_test_pred = best_model.predict(X_test_scaled)

# Print test accuracy and classification report
print("Test Set Accuracy:", accuracy_score(y_test, y_test_pred))
print("Classification Report:\n", classification_report(y_test, y_test_pred))

# Step 14: Print confusion matrix, precision, recall, and F1-score
precision, recall, f1, _ = precision_recall_fscore_support(y_test, y_test_pred, average='binary', pos_label=1)
print(f"Precision (phishing): {precision:.2f}")
print(f"Recall (phishing): {recall:.2f}")
print(f"F1-Score (phishing): {f1:.2f}")

# Confusion matrix
cm = confusion_matrix(y_test, y_test_pred)
print("Confusion Matrix:\n", cm)
```

Figure 137 Predict.py

The notebook then evaluates the performance of the trained models by calculating their accuracy scores. Accuracy is used as a primary metric to determine how well the models are classifying URLs. The models perform exceptionally well, with accuracy scores nearing 99.99% for the Random Forest and SVM models, which indicates that these models are well-suited for detecting phishing URLs on the given dataset. This evaluation helps to confirm that the feature extraction and model training processes are effective in identifying phishing URLs.

Finally, the notebook demonstrates how to use the trained models for prediction. It allows the models to predict whether new URLs are phishing or legitimate based on the features extracted from them. This is particularly useful for real-time applications where new URLs need to be classified quickly. To facilitate future use, the trained models are saved using joblib, allowing them to be loaded without needing to retrain. This makes it easier to implement these models in real-world systems for fast phishing detection.

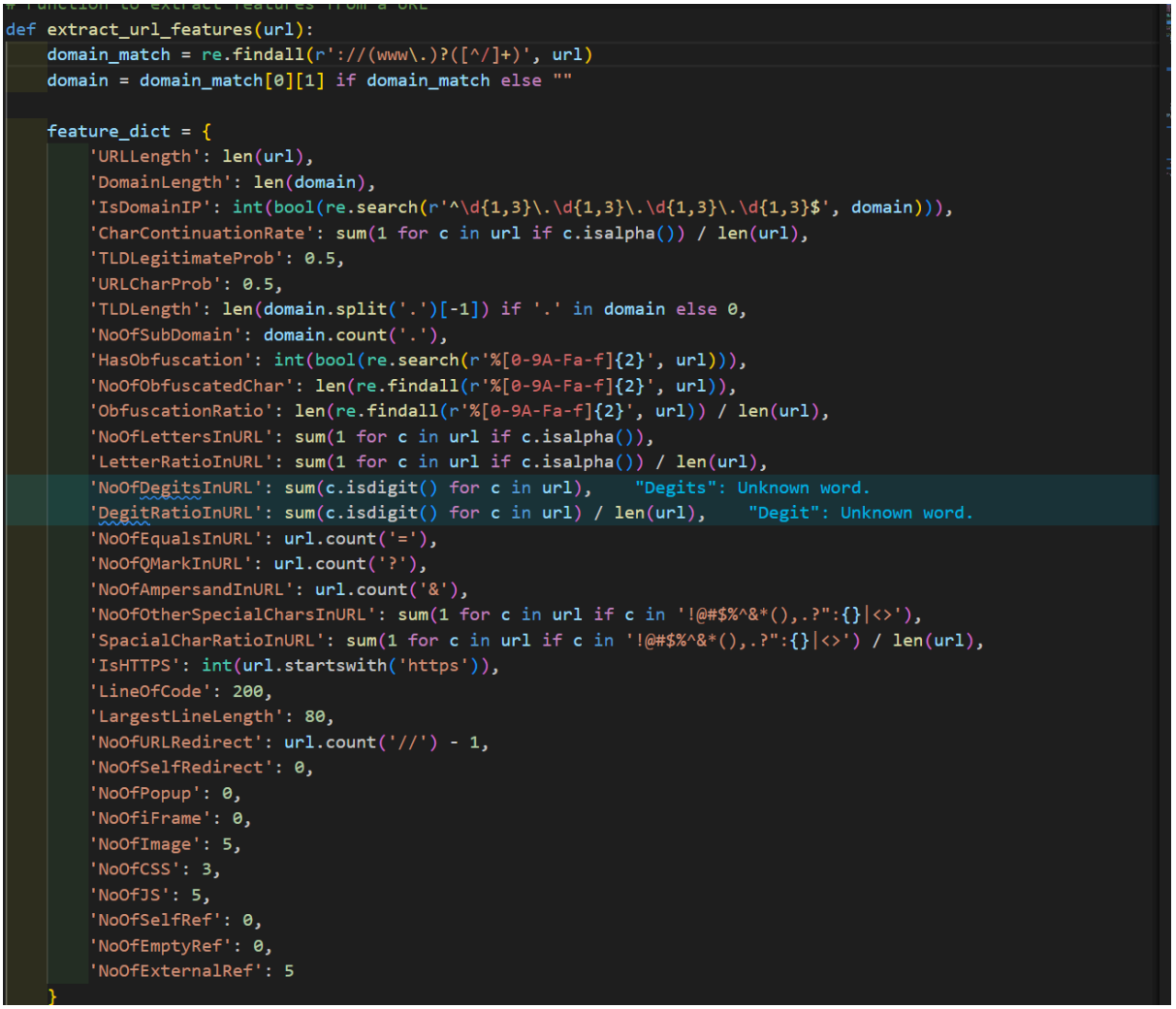
In summary, this notebook follows a structured approach to phishing detection, combining feature extraction, machine learning model training, and evaluation. By addressing

imbalanced datasets and utilizing a variety of machine learning techniques, it presents a comprehensive solution to the problem of identifying phishing URLs effectively.

4.6.1.6 Urlnew.ipynb

```
from flask import Flask, request, jsonify    "jsonify": Unknown word.
import joblib    "joblib": Unknown word.
import pandas as pd
import re
from flask_cors import CORS  # Import CORS
from sklearn.preprocessing import StandardScaler

app = Flask(__name__)
CORS(app)  # Enable CORS for all routes

# Load the trained XGBoost model and scaler
model = joblib.load(r'D:\Malware-Detection-using-Machine-learning-main\Malware-Detection-using-Machine-learning-
scaler = joblib.load('new_scaler.pkl')  # Load the saved scaler    "joblib": Unknown word.

# Load the selected features
with open('selected_features.txt', 'r') as f:
    selected_features = [line.strip() for line in f.readlines()]

# Function to extract features from a URL
def extract_url_features(url):
    domain_match = re.findall(r'://(www\.)?([^/]+)', url)
    domain = domain_match[0][1] if domain_match else ""
```

Figure 138 Urlnew.ipynb



```
def extract_url_features(url):
    domain_match = re.findall(r'://(www\.)?([^/]+)', url)
    domain = domain_match[0][1] if domain_match else ""

    feature_dict = {
        'URLLength': len(url),
        'DomainLength': len(domain),
        'IsDomainIP': int(bool(re.search(r'^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$', domain))),
        'CharContinuationRate': sum(1 for c in url if c.isalpha()) / len(url),
        'TLDLegitimateProb': 0.5,
        'URLCharProb': 0.5,
        'TLDLength': len(domain.split('.')[-1]) if '.' in domain else 0,
        'NoOfSubDomain': domain.count('.'),
        'HasObfuscation': int(bool(re.search(r'%[0-9A-Fa-f]{2}', url))),
        'NoOfObfuscatedChar': len(re.findall(r'%[0-9A-Fa-f]{2}', url)),
        'ObfuscationRatio': len(re.findall(r'%[0-9A-Fa-f]{2}', url)) / len(url),
        'NoOfLettersInURL': sum(1 for c in url if c.isalpha()),
        'LetterRatioInURL': sum(1 for c in url if c.isalpha()) / len(url),
        'NoOfDegitsInURL': sum(c.isdigit() for c in url),    "Degits": Unknown word.
        'DegitRatioInURL': sum(c.isdigit() for c in url) / len(url),    "Degit": Unknown word.
        'NoOfEqualsInURL': url.count('='),
        'NoOfQMarkInURL': url.count('?'),
        'NoOfAmpersandInURL': url.count('&'),
        'NoOfOtherSpecialCharsInURL': sum(1 for c in url if c in '!@#$%^&*(),.?":{}|<>'),
        'SpacialCharRatioInURL': sum(1 for c in url if c in '!@#$%^&*(),.?":{}|<>') / len(url),
        'IsHTTPS': int(url.startswith('https')),
        'LineOfCode': 200,
        'LargestLineLength': 80,
        'NoOfURLRedirect': url.count('//') - 1,
        'NoOfSelfRedirect': 0,
        'NoOfPopup': 0,
        'NoOfiFrame': 0,
        'NoOfImage': 5,
        'NoOfCSS': 3,
        'NoOfJS': 5,
        'NoOfSelfRef': 0,
        'NoOfEmptyRef': 0,
        'NoOfExternalRef': 5
    }
```

Figure 139 Urlnew.ipynb

This Flask web application is designed for detecting phishing URLs using a pre-trained machine learning model. The app is set up to handle requests from a frontend through a REST API, allowing users to submit URLs and receive predictions about whether a given URL is likely to be phishing or legitimate. The application relies on a trained XGBoost model and a scaler to process input features and make accurate predictions.

At the core of the application is a function that extracts meaningful features from URLs. The feature extraction is done using a combination of regular expressions and other operations to analyze the structure of the URL. Features such as the length of the URL, the length of the domain, the number of special characters, the presence of obfuscation (such as encoded characters), the ratio of letters and digits in the URL, and the use of secure HTTPS are among the many attributes extracted. Additionally, several webpage-level characteristics, like the number of images, CSS files, and JavaScript files, are included in the feature set to aid in the

prediction. These features are essential because phishing URLs often exhibit specific patterns, such as excessive use of obfuscation or unusually long URL lengths.

```
@app.route('/predict', methods=['POST'])
def predict_phishing():
    try:
        data = request.get_json()  # Get data from the frontend
        url = data.get('url')  # Extract URL from the request

        if not url:
            return jsonify({"error": "No URL provided"}), 400    "jsonify": Unknown word.

        # Debug print to check URL received
        print(f"Received URL: {url}")

        # Extract features from the URL
        extracted_features = extract_url_features(url)
        print(f"Extracted Features: {extracted_features}")

        # Scale the features
        scaled_features = scaler.transform(extracted_features)
        print(f"Scaled Features: {scaled_features}")

        # Make prediction
        prediction = model.predict(scaled_features)[0]
        print(f"Prediction Result: {prediction}")

        result = 'Phishing' if prediction == 0 else 'Not Phishing'
        return jsonify({"result": result})    "jsonify": Unknown word.

    except Exception as e:
        # Log the specific error and return a more detailed response
        print(f"Error during prediction: {str(e)}")
        return jsonify({"error": f"Error during prediction: {str(e)}"}), 500    "jsonify": Unknown word.
```

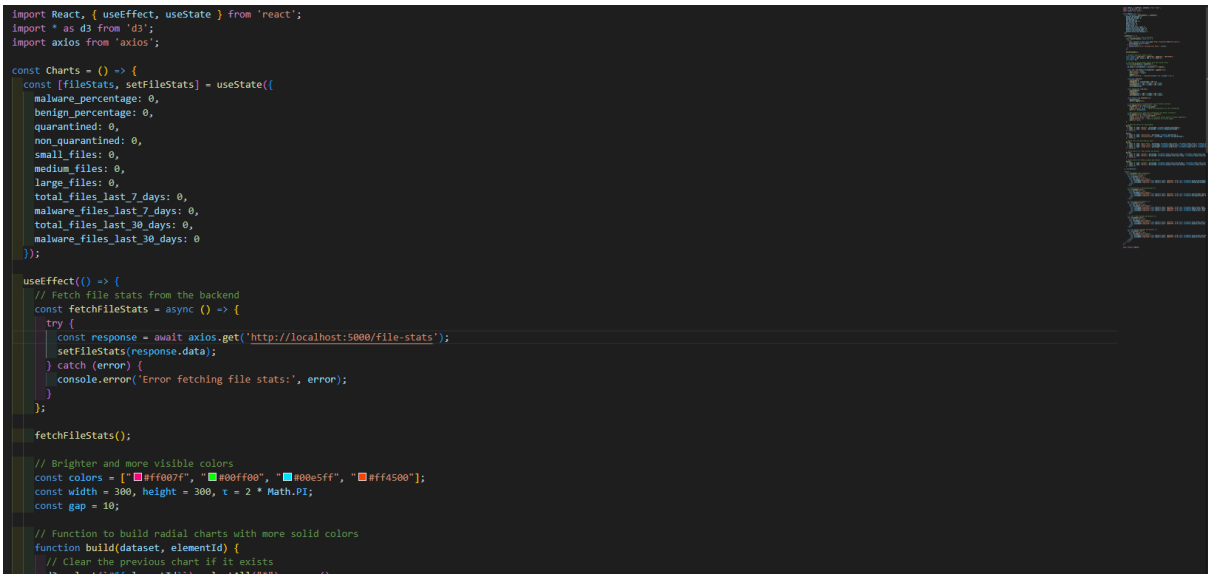
Figure 140 Urlnew.ipynb

Once the features are extracted from a given URL, they are passed through a StandardScaler that was trained on the same features during model training. This step ensures that the extracted features are scaled appropriately, as machine learning models typically perform better when the input features are on a similar scale. After scaling, the features are fed into the pre-trained XGBoost model, which makes a prediction on whether the URL is phishing or not. The result is then returned to the user in a JSON format, with either 'Phishing' or 'Not Phishing' as the output.

The application also handles errors gracefully. For instance, if no URL is provided, or if an error occurs during the prediction process, the app returns an appropriate error message. This ensures the app is robust and user-friendly. The app is cross-origin enabled using Flask-CORS, meaning it can be accessed by clients from different domains, making it suitable for integration into larger systems or web applications that need phishing detection services.

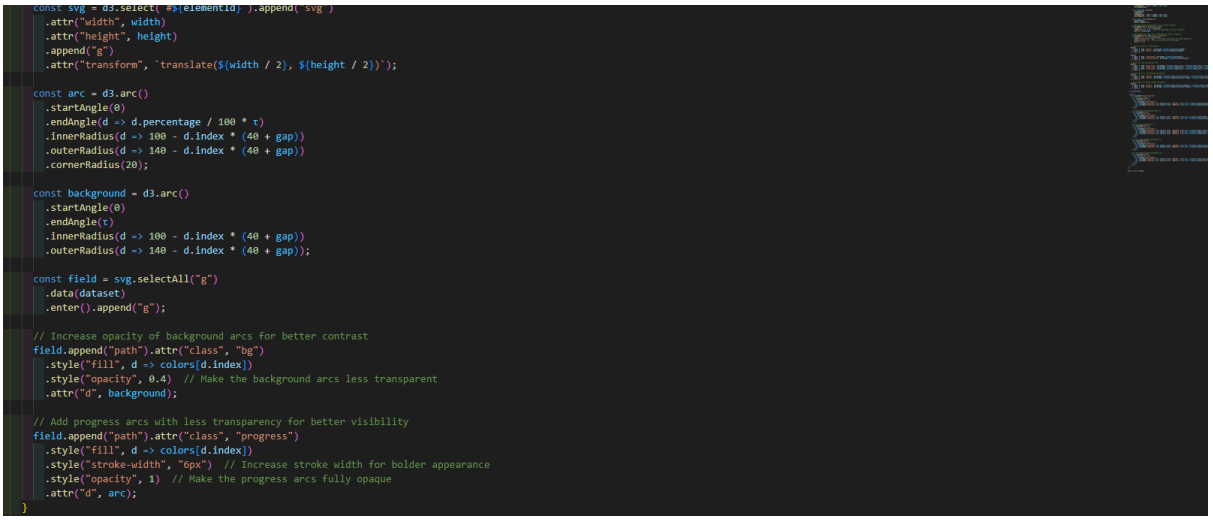
In summary, this Flask application serves as a real-time phishing detection tool. It processes input URLs by extracting relevant features, scales those features using a pre-trained scaler, and then classifies the URLs using an XGBoost model. The system provides a simple and effective API for phishing detection, handling potential errors and giving accurate predictions.

4.6.1.7 Chart.js



```
import React, { useEffect, useState } from 'react';
import * as d3 from 'd3';
import axios from 'axios';

const Charts = () => {
  const [fileStats, setFileStats] = useState({
    malware_percentage: 0,
    benign_percentage: 0,
    quarantined: 0,
    non_quarantined: 0,
    small_files: 0,
    medium_files: 0,
    large_files: 0,
    total_files_last_7_days: 0,
    malware_files_last_7_days: 0,
    total_files_last_30_days: 0,
    malware_files_last_30_days: 0
  });

  useEffect(() => {
    // Fetch file stats from the backend
    const fetchFileStats = async () => {
      try {
        const response = await axios.get('http://localhost:5000/file-stats');
        setFileStats(response.data);
      } catch (error) {
        console.error('Error fetching file stats:', error);
      }
    };

    fetchFileStats();

    // Brighter and more visible colors
    const colors = ["#ff007f", "#00ff00", "#00e5ff", "#ff4500"];
    const width = 300, height = 300, τ = 2 * Math.PI;
    const gap = 10;

    // Function to build radial charts with more solid colors
    function build(dataset, elementId) {
      // Clear the previous chart if it exists
```

Figure 141 Chart.js



```
    const svg = d3.select(`#${elementId}`).append("svg")
      .attr("width", width)
      .attr("height", height)
      .append("g")
      .attr("transform", `translate(${width / 2}, ${height / 2})`);

    const arc = d3.arc()
      .startAngle(0)
      .endAngle(d => d.percentage / 100 * τ)
      .innerRadius(d => 100 - d.index * (40 + gap))
      .outerRadius(d => 140 - d.index * (40 + gap))
      .cornerRadius(20);

    const background = d3.arc()
      .startAngle(0)
      .endAngle(τ)
      .innerRadius(d => 100 - d.index * (40 + gap))
      .outerRadius(d => 140 - d.index * (40 + gap));

    const field = svg.selectAll("g")
      .data(dataset)
      .enter().append("g");

    // Increase opacity of background arcs for better contrast
    field.append("path").attr("class", "bg")
      .style("fill", d => colors[d.index])
      .style("opacity", 0.4)  // Make the background arcs less transparent
      .attr("d", background);

    // Add progress arcs with less transparency for better visibility
    field.append("path").attr("class", "progress")
      .style("fill", d => colors[d.index])
      .style("stroke-width", "6px")  // Increase stroke width for bolder appearance
      .style("opacity", 1)  // Make the progress arcs fully opaque
      .attr("d", arc);
  }
```

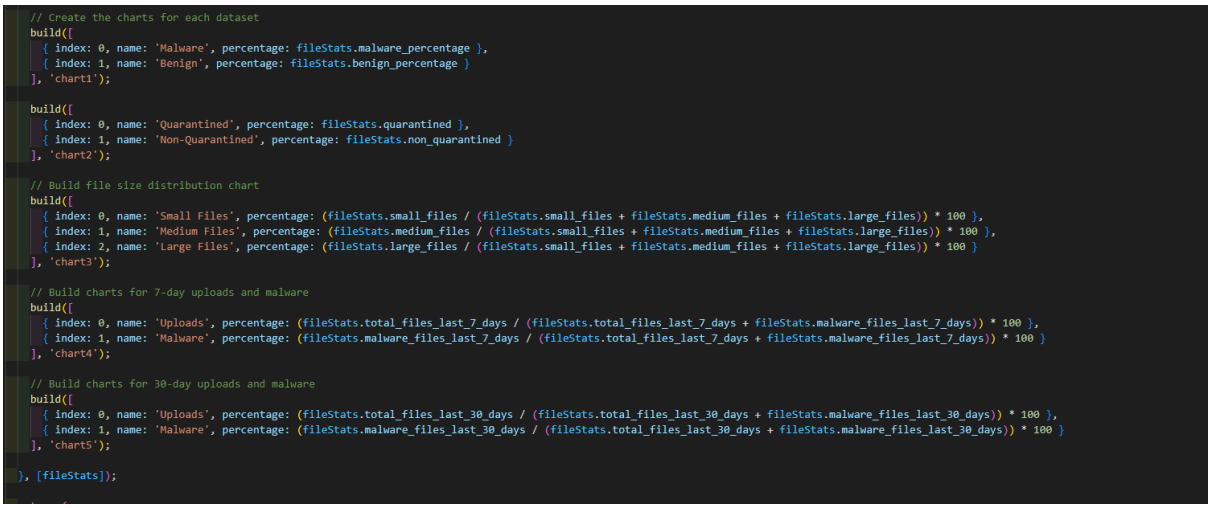
Figure 142 Chart.js

This **React component**, named Charts, is responsible for rendering a series of radial charts that visualize various statistics related to file uploads and malware detection. The component

uses **D3.js** for creating the charts and **Axios** to fetch the necessary data from the backend server.

The component starts by initializing a state variable, fileStats, which holds several statistics, including malware and benign file percentages, the number of quarantined and non-quarantined files, the distribution of small, medium, and large files, and the number of files and malware detected in the past 7 and 30 days. These statistics are fetched from a backend API using **Axios** and updated in the fileStats state. The component then renders this data into various charts.

The core of the chart-building functionality is encapsulated in the build function, which uses **D3.js** to construct radial charts. This function is responsible for removing any previous chart content, setting up an SVG canvas, and defining the chart arcs using the d3.arc() method. The function maps percentage data from the dataset to angles in the radial chart, defining the inner and outer radii for each category. The visual design of the charts includes bright colors, higher opacity, and thicker lines to improve readability.



```
// Create the charts for each dataset
build([
  { index: 0, name: 'Malware', percentage: fileStats.malware_percentage },
  { index: 1, name: 'Benign', percentage: fileStats.benign_percentage }
], 'chart1');

build([
  { index: 0, name: 'Quarantined', percentage: fileStats.quarantined },
  { index: 1, name: 'Non-Quarantined', percentage: fileStats.non_quarantined }
], 'chart2');

// Build file size distribution chart
build([
  { index: 0, name: 'Small Files', percentage: (fileStats.small_files / (fileStats.small_files + fileStats.medium_files + fileStats.large_files)) * 100 },
  { index: 1, name: 'Medium Files', percentage: (fileStats.medium_files / (fileStats.small_files + fileStats.medium_files + fileStats.large_files)) * 100 },
  { index: 2, name: 'Large Files', percentage: (fileStats.large_files / (fileStats.small_files + fileStats.medium_files + fileStats.large_files)) * 100 }
], 'chart3');

// Build charts for 7-day uploads and malware
build([
  { index: 0, name: 'Uploads', percentage: (fileStats.total_files_last_7_days / (fileStats.total_files_last_7_days + fileStats.malware_files_last_7_days)) * 100 },
  { index: 1, name: 'Malware', percentage: (fileStats.malware_files_last_7_days / (fileStats.total_files_last_7_days + fileStats.malware_files_last_7_days)) * 100 }
], 'chart4');

// Build charts for 30-day uploads and malware
build([
  { index: 0, name: 'Uploads', percentage: (fileStats.total_files_last_30_days / (fileStats.total_files_last_30_days + fileStats.malware_files_last_30_days)) * 100 },
  { index: 1, name: 'Malware', percentage: (fileStats.malware_files_last_30_days / (fileStats.total_files_last_30_days + fileStats.malware_files_last_30_days)) * 100 }
], 'chart5');

}, [fileStats]);
```

Figure 143 Chart.js

```
return (
  <div className="chart-container">
    {/* Malware vs Benign */}
    <div className="chart">
      <div id="chart1"></div>
      <div className="chart-labels">
        <div className="completed"><span style={{ color: '#ff007f' }}>●</span> {fileStats.malware_percentage}% Malware</div>
        <div className="completed"><span style={{ color: '#00ff00' }}>●</span> {fileStats.benign_percentage}% Benign</div>
      </div>
    </div>

    {/* Quarantined vs Non-Quarantined */}
    <div className="chart">
      <div id="chart2"></div>
      <div className="chart-labels">
        <div className="completed"><span style={{ color: '#ff007f' }}>●</span> {fileStats.quarantined}% Quarantined</div>
        <div className="completed"><span style={{ color: '#00ff00' }}>●</span> {fileStats.non_quarantined}% Non-Quarantined</div>
      </div>
    </div>

    {/* File Size Distribution */}
    <div className="chart">
      <div id="chart3"></div>
      <div className="chart-labels">
        <div className="completed"><span style={{ color: '#ff007f' }}>●</span> {fileStats.small_files} Small Files</div>
        <div className="completed"><span style={{ color: '#00ff00' }}>●</span> {fileStats.medium_files} Medium Files</div>
        <div className="completed"><span style={{ color: '#0091AA' }}>●</span> {fileStats.large_files} Large Files</div>
      </div>
    </div>

    {/* Last 7 Days Uploads and Malware */}
    <div className="chart">
      <div id="chart4"></div>
      <div className="chart-labels">
        <div className="completed"><span style={{ color: '#ff007f' }}>●</span> {fileStats.total_files_last_7_days} Uploads (Last 7 Days)</div>
        <div className="completed"><span style={{ color: '#00ff00' }}>●</span> {fileStats.malware_files_last_7_days} Malware (Last 7 Days)</div>
      </div>
    </div>
```

Figure 144 Chart.js

```
    {/* Last 30 Days Uploads and Malware */}
    <div className="chart">
      <div id="chart5"></div>
      <div className="chart-labels">
        <div className="completed"><span style={{ color: '#ff007f' }}>●</span> {fileStats.total_files_last_30_days} Uploads (Last 30 Days)</div>
        <div className="completed"><span style={{ color: '#00ff00' }}>●</span> {fileStats.malware_files_last_30_days} Malware (Last 30 Days)</div>
      </div>
    </div>
  </div>
  );
};

export default Charts;
```

Figure 145 Chart.js

The component generates five key charts. The first chart compares **malware vs benign files** by showing the percentage of each. The second chart shows **quarantined vs non-quarantined files**, highlighting the files that have been flagged as malware. The third chart presents a **file size distribution**, dividing files into small, medium, and large categories based on their sizes. The fourth and fifth charts display file uploads and detected malware over the **past 7 days** and the **past 30 days**, respectively.

The layout of the component is simple and effective. Each chart is paired with color-coded labels that describe the statistics represented in the chart, such as the percentage of malware files or the number of uploads in the last week. These labels correspond to the chart segments and provide users with an easy way to interpret the data.

In conclusion, the Charts component dynamically fetches file and malware statistics, visualizes them using **D3.js**, and presents them in a clear, user-friendly format. The component's design is flexible, allowing it to handle different datasets and present the data in visually appealing radial charts.

4.6.1.8 ExistingChart.js

```
import React, { useEffect, useMemo } from 'react';
import * as d3 from 'd3';

const ExistingChart = () => {
  const malwareCount = 96724;
  const legitimateCount = 41323;
  const totalCount = malwareCount + legitimateCount;

  // Memoized Subsystem data
  const subsystemData = useMemo(() => ({
    2: 113911,
    3: 22640,
    1: 1457,
    16: 39
  }), []);

  // Memoized Subsystem labels for more descriptive names
  const subsystemLabels = useMemo(() => ({
    1: 'Driver',
    2: 'Windows GUI',
    3: 'Console',
    16: 'Boot Loader'
  }), []);

  // File size data
  const fileSizeData = useMemo(() => ({
    small: 66572,
    medium: 64156,
    large: 7319
  }), []);

  // Entropy data (Low, Medium, High Entropy)
  const entropyData = useMemo(() => ({
    low: 6571,
    medium: 118806,
    high: 12670
  }), []);

  const entropyLabels = useMemo(() => ({
    low: 'Low Entropy',
    medium: 'Medium Entropy',
    high: 'High Entropy'
  }), []);
```

Figure 146 ExistingChart.js

The ExistingChart component in React is designed to visually represent data using radial charts, powered by D3.js. The component tracks and displays four key categories of file data: malware versus legitimate files, subsystems, file size distribution, and entropy levels. It efficiently handles large amounts of data by using React's useMemo and useEffect hooks, ensuring that the charts are only re-rendered when necessary.

The component begins by defining static counts for the number of malware and legitimate files. These counts are used to calculate the total number of files, which serves as the basis for determining the percentage distribution of malware and legitimate files. Memoization is applied to store frequently used data such as subsystem information (e.g., Windows GUI, Console, Driver, Boot Loader) and entropy levels (categorized as low, medium, and high

entropy), which are pre-defined and calculated once, improving the performance by preventing recalculation on every render.

```
// Calculate total subsystem and entropy counts
const subsystemTotal = useMemo(() => Object.values(subsystemData).reduce((acc, curr) => acc + curr, 0), [subsystemData]);
const entropyTotal = useMemo(() => Object.values(entropyData).reduce((acc, curr) => acc + curr, 0), [entropyData]);

const width = 300, height = 300, gap = 2; // Constants for the chart dimensions

// Memoized color array
const colors = useMemo(() => ["#ff007f", "#00ff00", "#00e5ff", "#ff4500"], []);

useEffect(() => {
  const τ = 2 * Math.PI;

  // Function to build radial charts with more solid colors
  function build(dataset, elementId) {
    // Clear the previous chart if it exists
    d3.select(`#${elementId}`).selectAll("*").remove();

    const svg = d3.select(`#${elementId}`).append("svg")
      .attr("width", width)
      .attr("height", height)
      .append("g")
      .attr("transform", `translate(${width / 2}, ${height / 2})`);

    const arc = d3.arc()
      .startAngle(0)
      .endAngle(d => d.percentage / 100 * τ)
      .innerRadius(d => 30 + d.index * (20 + gap))
      .outerRadius(d => 50 + d.index * (20 + gap))
      .cornerRadius(20);

    const background = d3.arc()
      .startAngle(0)
      .endAngle(τ)
      .innerRadius(d => 30 + d.index * (20 + gap))
      .outerRadius(d => 50 + d.index * (20 + gap));

    const field = svg.selectAll("g")
      .data(dataset)
      .enter().append("g");
```

Figure 147 ExistingChart.js

```
    // Background arcs
    field.append("path").attr("class", "bg")
      .style("fill", d => colors[d.index])
      .style("opacity", 0.4)
      .attr("d", background);

    // Progress arcs
    field.append("path").attr("class", "progress")
      .style("fill", d => colors[d.index])
      .style("stroke-width", "6px")
      .style("opacity", 1)
      .attr("d", arc);
  }

  // Dataset for malware/legitimate chart
  const malwareLegitDataset = [
    { index: 0, name: "Malware", percentage: (malwareCount / totalCount) * 100 },
    { index: 1, name: "Legitimate", percentage: (legitimateCount / totalCount) * 100 }
  ];
  build(malwareLegitDataset, 'chart1');

  // Dataset for subsystem chart
  const subsystemDataset = Object.keys(subsystemData).map((key, index) => ({
    index,
    name: subsystemLabels[key],
    percentage: (subsystemData[key] / subsystemTotal) * 100
  }));
  build(subsystemDataset, 'chart2');

  // Dataset for file size distribution
  const fileSizeDataset = Object.keys(fileSizeData).map((key, index) => ({
    index,
    name: key.charAt(0).toUpperCase() + key.slice(1) + ' Files',
    percentage: (fileSizeData[key] / (fileSizeData.small + fileSizeData.medium + fileSizeData.large)) * 100
  }));
  build(fileSizeDataset, 'chart3');
```

Figure 148 ExistingChart.js

Next, useMemo is used to store the labels and values for subsystems, file sizes, and entropy levels. These datasets represent various characteristics of the files being analyzed. Subsystems categorize the files based on the environment they run in, such as Windows GUI or Console applications. The file size distribution is split into small, medium, and large files, while entropy, which measures randomness in file data, is categorized into low, medium, and high entropy. These categories help identify different traits of the files, which could be useful for security analysis.

The core of the charting process is handled in the useEffect hook, where a function named build creates each radial chart. This function starts by clearing any existing charts before creating new SVG elements for rendering. It uses D3.js to create both background arcs (representing the total data) and progress arcs (representing the percentage of each category). The arcs are drawn based on the percentage of each category relative to the total dataset. The function ensures that each arc is clearly visible with appropriate colors and opacity settings.

```
// Dataset for entropy distribution
const entropyDataset = Object.keys(entropyData).map((key, index) => ({
  index,
  name: entropyLabels[key],
  percentage: (entropyData[key] / entropyTotal) * 100
}));
build(entropyDataset, 'chart4');

}, [malwareCount, legitimateCount, totalCount, subsystemData, subsystemTotal, fileSizeData, entropyData, entropyTotal, colors, subsystemLabels, entropyLabels]);

return (
  <div className="chart-container">
    {/* Malware vs Legitimate */}
    <div className="chart">
      <div id="chart1"></div>
      <div className="chart-labels">
        <div className="completed">
          <span style={{ color: '#ff007f' }}>●</span>
          {malwareCount} Malware ({((malwareCount / totalCount) * 100).toFixed(2)}%)
        </div>
        <div className="completed">
          <span style={{ color: '#00ff00' }}>●</span>
          {legitimateCount} Legitimate ({((legitimateCount / totalCount) * 100).toFixed(2)}%)
        </div>
      </div>
    </div>

    {/* Subsystem Chart */}
    <div className="chart">
      <div id="chart2"></div>
      <div className="chart-labels">
        {Object.keys(subsystemData).map((key, index) => (
          <div className="completed" key={index}>
            <span style={{ color: colors[index % colors.length] }}>●</span>
            {subsystemData[key]} {subsystemLabels[key]} ({((subsystemData[key] / subsystemTotal) * 100).toFixed(2)}%)
          </div>
        ))}
      </div>
    </div>
```

Figure 149 ExistingChart.js

Figure 150 ExistingChart.js

The build function is called four times for each chart, one for each dataset:

1. The first chart visualizes the split between malware and legitimate files, showing the proportion of each type.
2. The second chart represents the distribution of files across different subsystems, illustrating the number of files that fall under each subsystem type.
3. The third chart focuses on file size distribution, with segments for small, medium, and large files, providing insight into how the files vary in size.
4. The final chart shows the distribution of files based on entropy levels, highlighting the degree of randomness in the files' data.

The component renders these charts within the DOM, and each chart is accompanied by a legend below it. The legends display the name, count, and percentage of each category, offering an easy-to-understand breakdown of the data being visualized.

In summary, the ExistingChart component is a powerful and efficient tool for visualizing file-related data using D3.js. It organizes the data into meaningful categories such as file types, subsystems, file sizes, and entropy, and uses radial charts to provide a clear and visually appealing representation of the data. By leveraging React's hooks and D3.js, the component ensures optimal performance and dynamic chart generation.

4.6.1.8 Fileupload.js

```
import React, { useState } from 'react';
import axios from 'axios';

const FileUpload = () => {
  const [uploadResult, setUploadResult] = useState('');
  const [error, setError] = useState('');
  const [isUploading, setIsUploading] = useState(false); // Loading state

  // Function to trigger file upload dialog
  const triggerFileUpload = async () => {
    const inputElement = document.createElement('input');
    inputElement.type = 'file';
    inputElement.accept = '.exe';  // Accept only .exe files
    inputElement.style.display = 'none';  // Hide the input element

    // Add change event listener to upload file when selected
    inputElement.onchange = async (event) => {
      const file = event.target.files[0];
      if (file) {
        await handleFileUpload(file);
      }
    };

    // Programmatically click the hidden input element to trigger file selection
    inputElement.click();
  };

  // Function to handle file upload to the backend
  const handleFileUpload = async (file) => {
    setIsUploading(true);
    setError('');
    setUploadResult('');

    const formData = new FormData();
    formData.append('file', file);
```

Figure 151 Fileupload.js

The FileUpload component in React is designed to handle the uploading of .exe files to a backend server and provide feedback to the user on the success or failure of the upload process. The component makes use of React's state management to track the uploading process and any resulting messages.

At the core of the component is the triggerFileUpload function. This function creates an invisible file input element that only accepts .exe files. When the user clicks the "Upload EXE" button, the function simulates a click on this hidden input element, triggering the file selection dialog on the user's system. This approach ensures a clean user interface, where the actual file input is hidden from view.

```
    try {
      const response = await axios.post('http://localhost:5000/upload', formData, {
        headers: { 'Content-Type': 'multipart/form-data' },
      });

      setUploadResult(`File scanned successfully: ${response.data.result}`);
    } catch (error) {
      setError('Error uploading file. Please try again.');
    } finally {
      setIsUploading(false);
    }
  };

  return (
    <div>
      <button className="circle-upload-btn" onClick={triggerFileUpload}>
        {isUploading ? 'Uploading...' : 'Upload EXE'}
      </button>

      {error && <p style={{ color: 'red' }}>{error}</p>}
      {uploadResult && <p style={{ color: 'green' }}>{uploadResult}</p>}
    </div>
  );
};

export default FileUpload;
```

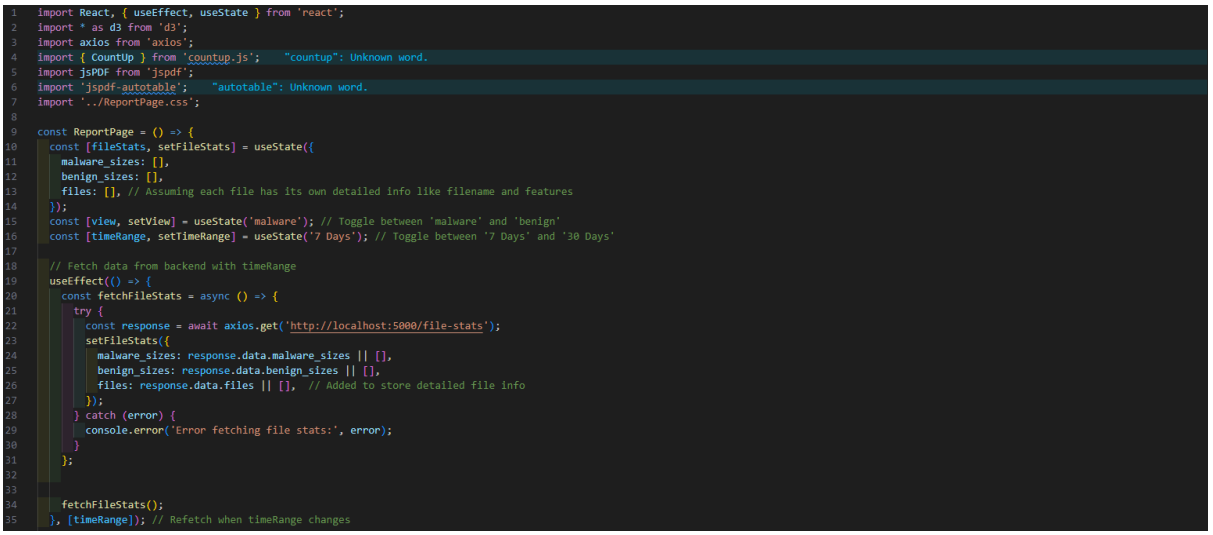
Figure 152 Fileupload.js

Once a file is selected, the onchange event of the input element calls the handleFileUpload function. This function manages the actual upload process, where the selected file is wrapped in a FormData object and sent to the backend via a POST request using Axios. The request is made to the backend endpoint (http://localhost:5000/upload), which is expected to handle file uploads. During this process, a loading state (isUploading) is activated to provide feedback to the user by showing "Uploading..." while the file is being uploaded.

The handleFileUpload function also includes error handling and result tracking. If the upload is successful, the result message returned by the backend is displayed to the user in a green success message. However, if an error occurs during the upload, such as network issues or server errors, an error message is displayed in red, indicating that the upload failed.

In summary, this component simplifies the file uploading process for users, providing a smooth and intuitive experience. It also includes robust error handling and feedback mechanisms to inform the user of the upload status. The use of hidden file input and dynamic state-based messaging ensures a clean, modern user interface.

4.6.1.9 Report.js



```
import React, { useEffect, useState } from 'react';
import * as d3 from 'd3';
import axios from 'axios';
import { CountUp } from 'countup.js';    "countup": Unknown word.
import jsPDF from 'jspdf';
import 'jspdf-autotable';    "autotable": Unknown word.
import '../ReportPage.css';

const ReportPage = () => {
  const [fileStats, setFileStats] = useState({
    malware_sizes: [],
    benign_sizes: [],
    files: [], // Assuming each file has its own detailed info like filename and features
  });
  const [view, setView] = useState('malware'); // Toggle between 'malware' and 'benign'
  const [timeRange, setTimeRange] = useState('7 Days'); // Toggle between '7 Days' and '30 Days'

  // Fetch data from backend with timeRange
  useEffect(() => {
    const fetchFileStats = async () => {
      try {
        const response = await axios.get('http://localhost:5000/file-stats');
        setFileStats({
          malware_sizes: response.data.malware_sizes || [],
          benign_sizes: response.data.benign_sizes || [],
          files: response.data.files || [],  // Added to store detailed file info
        });
      } catch (error) {
        console.error('Error fetching file stats:', error);
      }
    };


    fetchFileStats();
  }, [timeRange]); // Refetch when timeRange changes
```

Figure 153 Report.js

```
// Render the graph
useEffect(() => {
  const graphContainer = d3.select('.graph');
  graphContainer.selectAll('*').remove(); // Clear any previous content

  let data = view === 'malware' ? fileStats.malware_sizes : fileStats.benign_sizes;

  const categories = [
    { name: 'Small Files', files: data.filter((size) => size < 1 * 1024 * 1024) },
    { name: 'Medium Files', files: data.filter((size) => size >= 1 * 1024 * 1024 && size <= 10 * 1024 * 1024) },
    { name: 'Large Files', files: data.filter((size) => size > 10 * 1024 * 1024) },
  ];

  const graphData = categories.map((category) => ({
    category: category.name,
    totalSize: category.files.reduce((sum, size) => sum + size, 0) / (1024 * 1024 * 1024), // Convert to GB
  }));

  if (graphData.length === 0) {
    graphContainer
      .append('text')
      .attr('x', 300)
      .attr('y', 200)
      .attr('text-anchor', 'middle')
      .text('No Data Available')
      .style('fill', '#dd2c2c');
    return;
  }
```

Figure 154 Report.js

The ReportPage component in React is designed to provide an interactive dashboard for analyzing malware and benign file statistics over different time ranges, with the ability to generate a detailed report as a downloadable PDF. This component includes data fetching, dynamic chart rendering using D3.js, animated count displays using CountUp.js, and PDF generation using jsPDF.

State Management and Data Fetching:

The component maintains two pieces of state: fileStats, which holds statistics and details of malware and benign files, and view, which toggles between displaying data for malware or benign files. Additionally, timeRange is used to toggle between viewing data for the last 7 days or 30 days. The data is fetched from a backend server via Axios, which retrieves file sizes and detailed file information for malware and benign files. This data is used throughout the component to display graphs and summary statistics.

Chart Rendering:

D3.js is used to render bar charts that visualize the distribution of file sizes. The chart dynamically updates depending on whether the user is viewing malware or benign files. File sizes are categorized into "Small Files," "Medium Files," and "Large Files" based on their size. The chart is cleared and redrawn each time the data changes, and file sizes are converted from bytes to gigabytes for easier interpretation.

```
  const width = 600;
  const height = 400;
  const margin = { top: 20, right: 30, bottom: 50, left: 60 };

  const x = d3.scaleBand().domain(graphData.map((d) => d.category)).range([margin.left, width - margin.right]).padding(0.2);

  const y = d3.scaleLinear().domain([0, 2]).range([height - margin.bottom, margin.top]);

  const svg = graphContainer.append('svg').attr('class', 'graph-svg').attr('width', width).attr('height', height);

  svg
    .selectAll('rect')
    .data(graphData)
    .enter()
    .append('rect')
    .attr('x', (d) => x(d.category))
    .attr('y', (d) => y(d.totalSize))
    .attr('width', x.bandwidth())
    .attr('height', (d) => y(0) - y(d.totalSize))
    .attr('fill', '#dd2c2c');

  svg.append('g').attr('transform', `translate(0,${height - margin.bottom})`).call(d3.axisBottom(x));

  svg.append('g').attr('transform', `translate(${margin.left},0)`).call(d3.axisLeft(y).tickFormat((d) => `${d} GB`));
}, [fileStats, view]);

// Use CountUp for animating totals
useEffect(() => {
  const totalMalware = fileStats.malware_sizes.length;
  const totalBenign = fileStats.benign_sizes.length;

  const options = {
    useEasing: true,
    useGrouping: true,
    separator: ',',
    decimal: '.',
  };
```

Figure 155 Report.js

```
  const countUpInstance = new CountUp('total', view === 'malware' ? totalMalware : totalBenign, options);
  if (!countUpInstance.error) {
    countUpInstance.start();
  } else {
    console.error(countUpInstance.error);
  }
}, [fileStats, view]);

const generatePDF = () => {
  const doc = new jsPDF();
  doc.text('Malware vs Benign Files Report', 20, 10);

  // General statistics for malware vs benign
  doc.autoTable({
    head: [['Type', 'File Count', 'Total Size (GB)']],
    body: [
      ['Malware', fileStats.malware_sizes.length, (fileStats.malware_sizes.reduce((a, b) => a + b, 0) / (1024 * 1024 * 1024)).toFixed(2)],
      ['Benign', fileStats.benign_sizes.length, (fileStats.benign_sizes.reduce((a, b) => a + b, 0) / (1024 * 1024 * 1024)).toFixed(2)],
    ],
  });

  // Add File Details Section
  doc.text('File Details', 20, doc.lastAutoTable.finalY + 15);

  // Display features for each file
  fileStats.files.forEach((file, index) => {
    doc.text(`File: ${file.filename}`, 20, doc.lastAutoTable.finalY + 10);
```

Figure 156 Report.js

```
    // Display file features
    if (file.features) {
      doc.autoTable({
        startY: doc.lastAutoTable.finalY + 10,
        head: [['Feature', 'Value']],
        body: [
          ['Address of Entry Point', file.features.address_of_entry_point || 'N/A'],
          ['Base of Code', file.features.base_of_code || 'N/A'],
          ['Base of Data', file.features.base_of_data || 'N/A'],
          ['Characteristics', file.features.characteristics || 'N/A'],
          ['Checksum', file.features.checksum || 'N/A'],
          ['DLL Characteristics', file.features.dll_characteristics || 'N/A'],
          ['Exports Number', file.features.exports_nb || 'N/A'],
          ['File Alignment', file.features.file_alignment || 'N/A'],
          ['Image Base', file.features.image_base || 'N/A'],
          ['Imports Number', file.features.imports_nb || 'N/A'],
          ['Imports DLLs Number', file.features.imports_nb_dll || 'N/A'],
          ['Imports Ordinal Number', file.features.imports_nb_ordinal || 'N/A'],
          ['Load Configuration Size', file.features.load_configuration_size || 'N/A'],
          ['Loader Flags', file.features.loader_flags || 'N/A'],
          ['Machine', file.features.machine || 'N/A'],
          ['Major Image Version', file.features.major_image_version || 'N/A'],
          ['Major Linker Version', file.features.major_linker_version || 'N/A'],
          ['Major OS Version', file.features.major_os_version || 'N/A'],
          ['Major Subsystem Version', file.features.major_subsystem_version || 'N/A'],
          ['Minor Image Version', file.features.minor_image_version || 'N/A'],
          ['Minor Linker Version', file.features.minor_linker_version || 'N/A'],
          ['Minor OS Version', file.features.minor_os_version || 'N/A'],
          ['Minor Subsystem Version', file.features.minor_subsystem_version || 'N/A'],
          ['Number of RVA and Sizes', file.features.number_of_rva_and_sizes || 'N/A'],
          ['Number of Sections', file.features.number_of_sections || 'N/A'],
          ['Resources Max Entropy', file.features.resources_max_entropy || 'N/A'],
          ['Resources Max Size', file.features.resources_max_size || 'N/A'],
          ['Resources Mean Entropy', file.features.resources_mean_entropy || 'N/A'],
          ['Resources Mean Size', file.features.resources_mean_size || 'N/A'],
          ['Resources Min Entropy', file.features.resources_min_entropy || 'N/A'],
          ['Resources Min Size', file.features.resources_min_size || 'N/A'],
          ['Resources Number', file.features.resources_nb || 'N/A'],
          ['Section Alignment', file.features.section_alignment || 'N/A'],
          ['Sections Max Entropy', file.features.sections_max_entropy || 'N/A'],
          ['Sections Max Raw Size', file.features.sections_max_rawsize || 'N/A'],    "rawsize": Unknown word.
          ['Sections Max Virtual Size', file.features.sections_max_virtualsize || 'N/A'],    "virtualsize": Unknown word.
```

Figure 157 Report.js

```
  return (
    <div className="wrapper">
      <div className="buttons">
        <button className={`btn btn1 ${timeRange === '7 Days' ? 'active' : ''}`} onClick={() => setTimeRange('7 Days')}>
          7 Days
        </button>
        <button className={`btn btn2 ${timeRange === '30 Days' ? 'active' : ''}`} onClick={() => setTimeRange('30 Days')}>
          30 Days
        </button>
      </div>

      <h1>Malware vs Benign Files Report</h1>

      <div className="buttons">
        <button className={`btn btn1 ${view === 'malware' ? 'active' : ''}`} onClick={() => setView('malware')}>
          Malware
        </button>
        <button className={`btn btn2 ${view === 'benign' ? 'active' : ''}`} onClick={() => setView('benign')}>
          Benign
        </button>
      </div>

      <div className="graph"></div>

      <div className="totalwrap">    "totalwrap": Unknown word.
        <h2 id="total" className="totals">
          Total: 0
        </h2>
      </div>

      {/* Button to generate and download the PDF */}
      <div className="download-btn">
        <button onClick={generatePDF}>Download Report as PDF</button>
      </div>
    </div>
  );
};
```

Figure 158 Report.js

If there is no data available for the selected file type, a message "No Data Available" is displayed in the chart area, ensuring that the user always has relevant feedback, even when data is missing.

Animated Totals:

To enhance the user experience, the total number of malware or benign files is displayed and animated using the CountUp.js library. This creates a smooth, animated effect as the number of files increases or decreases based on the selected view (malware or benign). The total is updated each time the view changes between malware and benign files.

PDF Generation:

One of the key features of the ReportPage component is the ability to generate and download a detailed report as a PDF. The jsPDF library is used for this, and a table of statistics summarizing the file counts and total sizes for malware and benign files is included. Additionally, detailed feature information for each file is also included in the report, which provides a comprehensive breakdown of various characteristics such as the file's entry point, code size, image base, and more.

The generatePDF function is responsible for constructing the PDF. It includes sections for general statistics, file-specific details, and individual file features. If no feature details are available for a file, a message is displayed to ensure the report remains readable and informative.

User Interaction:

The user can toggle between viewing malware and benign data using buttons, and can also switch between a 7-day or 30-day view. These buttons change the state of the component, triggering the data-fetching logic to retrieve the relevant file statistics. The dynamic chart and total count display updates accordingly, providing the user with an up-to-date view of the data.

4.6.2 App.css

```
You, 5 days ago | 1 author (You)
/* === Background and Content === */

/* Main content area */
.content {
  position: relative;
  z-index: 1;
  text-align: center;
  width: 100%;
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
  background: none; /* Make sure background is transparent */
}

/* === Orbiting Buttons Styling (Malware Detection, Reports, Charts) === */      You, 5 days ago • Uncommitted changes

/* Container for orbiting buttons */
.orbiting-buttons {
  position: absolute;
  width: 100%;
  height: 100%;
  pointer-events: none;
  z-index: 3;
}
```

Figure 159 App.css

The App.css file is designed to style a visually appealing interface that incorporates orbiting buttons, modal popups, upload functionality, and charts overlaid on a canvas background. Here's a breakdown of the key styles and their functionality:

Background and Content Styling:

The .content class centers the main content both horizontally and vertically within the viewport, ensuring that the content is always aligned in the middle. The background is set to transparent to allow visibility of the background canvas. This creates a clean, minimal layout where the user's attention is drawn to the central elements.

Orbiting Buttons Styling:

The .orbiting-buttons class places the buttons for Malware Detection, Reports, and Charts at fixed positions within the viewport, and the circular buttons use gradients and shadows to create a vibrant, three-dimensional look. The .circle-button class gives each button a radial gradient (dark-to-light) for a rich color effect, and box-shadow properties to create depth. The buttons also feature hover effects that scale the button and further enhance the shadow, making them visually prominent and interactive. The buttons remain interactive through pointer-events, while the rest of the area is not clickable due to being set to none.

```css
/* Circular button styles with darker-lighter gradient and shadow */
.circle-button {
  position: absolute;
  width: 80px;
  height: 80px;
  border-radius: 50%;
  background: radial-gradient(circle, rgb(56, 25, 0) 0%, rgb(161, 0, 0) 100%);
  color: white;
  border: none;
  pointer-events: auto;
  cursor: pointer;
  font-size: 12px;
  font-weight: bold;
  text-align: center;
  line-height: 18px;
  padding: 10px;
  transition: transform 0.5s ease-in-out, box-shadow 0.5s ease-in-out;
  box-shadow: 0px 8px 15px rgba(0, 0, 0, 0.5),
              inset -2px -2px 10px rgba(255, 255, 255, 0.4),
              inset 2px 2px 10px rgba(0, 0, 0, 0.4);
  z-index: 4;
}

/* Text adjustments inside circular buttons */
.circle-button span {
  display: inline-block;
  white-space: normal;
  word-wrap: break-word;
  vertical-align: middle;
  line-height: 14px;
  padding: 2px;
  color: white;
  text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.8);
}
```

Figure 160 App.css

```css
/* Hover effects for the circular buttons */
.circle-button:hover {
  transform: scale(1.1);
  box-shadow: 0 0 20px rgba(255, 69, 0, 0.7),
              0 0 30px rgba(139, 0, 0, 0.7);
}

/* === Modal Popup === */

/* Modal styling */
.modal {
  position: fixed;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  background: radial-gradient(circle at center, rgba(10, 10, 30, 1) 0%, rgba(0, 0, 0, 1) 100%);
  padding: 40px;
  border-radius: 20px;
  box-shadow: 0px 10px 20px rgba(0, 0, 0, 0.8);
  z-index: 1000;
  text-align: center;
}

.modal-content h2 {
  color: #fff;
  font-family: 'Poppins', sans-serif;
  font-size: 24px;
  margin-bottom: 20px;
}

/* Upload Circle Button */
.upload-buttons {
  display: flex;
  justify-content: center;
  gap: 30px;
}

.upload-circle {
  position: relative;
  display: flex;
  justify-content: center;
  align-items: center;
}
```

Figure 161 App.css

```css
.circle-upload-btn {
  position: relative;
  width: 80px;
  height: 80px;
  border-radius: 50%;
  background: linear-gradient(145deg, #ff7f50, #ff4500);
  color: white;
  font-size: 14px;
  border: none;
  cursor: pointer;
  font-weight: bold;
  z-index: 2;
  transition: transform 0.2s ease-in-out;
}

.circle-upload-btn:hover {
  transform: scale(1.05);
}

/* Circular Progress Bar */
.progress-circle {
  position: relative;
  width: 100px;
  height: 100px;
}

.circle-svg {
  position: absolute;
  top: 0;
  left: 0;
  width: 100px;
  height: 100px;
}

.circle-svg circle {
  stroke: rgba(255, 255, 255, 0.3);
  stroke-width: 8;
  fill: none;
  stroke-linecap: round;
  animation: progress 5s linear infinite;
}
```

Figure 162 App.css

Modal Popup Styling:

The .modal class defines the layout for modals, which are used to display additional information or functionality such as file uploads. The modal has a radial gradient background, rounded corners, and box-shadow, making it visually distinct. It is centered in the viewport and layered above the rest of the content with a high z-index. The .modal-content h2 class styles the heading inside the modal with a clean, white font for visibility on the dark background. There are buttons for uploading files, styled with the .upload-buttons and .upload-circle classes, and circular buttons for triggering file uploads, styled with the .circle-upload-btn class, which also has hover and animation effects to draw attention. The .close-modal class provides a simple button to close the modal.

Progress Bar Animation:

The @keyframes progress animation is applied to .circle-svg circle elements, creating a circular progress bar that continuously animates in a clockwise direction. This is used to visually indicate progress during file uploads.

```
@keyframes progress {
  0% {
    stroke-dasharray: 0, 282.7433388230814;
  }
  100% {
    stroke-dasharray: 282.7433388230814, 0;
  }
}

/* Close Modal Button */
.close-modal {
  background-color: white;
  border: none;
  color: black;
  padding: 10px 20px;
  border-radius: 5px;
  cursor: pointer;
  margin-top: 20px;
  font-family: 'Poppins', sans-serif;
}

/* Glow Animation (Optional) */
@keyframes glowing {
  from {
    box-shadow: 0 0 10px #ff7f50, 0 0 20px #ff4500, 0 0 30px #ff4500;
  }
  to {
    box-shadow: 0 0 20px #ff7f50, 0 0 30px #ff4500, 0 0 40px #ff4500;
  }
}

.circle-upload-btn {
  animation: glowing 1.5s infinite alternate;
}

/* Background fade effect */
.background-faded {
  filter: brightness(0.5);
}
```

Figure 163 App.css

```css
/* Fade-out effect */
.fade-out {
  opacity: 0;
  pointer-events: none;
  transform: scale(0.8);
}

/* === Background and Chart Overlay === */

/* Chart container needs to be placed on top of the canvas */
.chart-container {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  display: flex;
  justify-content: space-around;
  align-items: center;
  z-index: 2;
}

/* Individual charts positioning and styling */
.chart {
  width: 300px;
  height: 300px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  position: relative;
  z-index: 2;
}

/* Labels and percentage styling */
.chart-labels {
  text-align: center;
}

.completed {
  color: #ffffff;
  font-size: 20px;
}
```

Figure 164 App.css

Chart Styling:

The .chart-container and .chart classes handle the layout of charts, positioning them within the viewport and ensuring they are placed on top of the canvas background. Each chart has its own container, and labels are styled using the .chart-labels and .completed classes, ensuring text is easy to read, with a white font on the dark background.

```css
/* Ensure canvas stays behind everything */
.canvas-background {
  position: absolute;
  top: 0;
  left: 0;
  width: 100vw;
  height: 100vh;
  z-index: 0;
}
```

Figure 165 App.css

Canvas and Background Management:

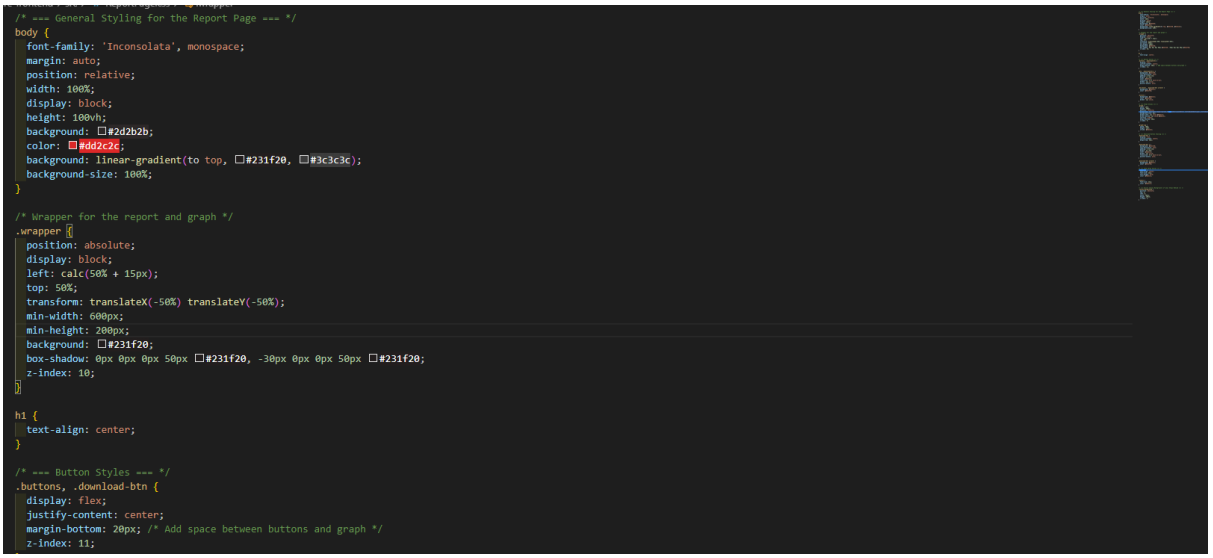
The .canvas-background class ensures that the canvas background, presumably for animations or graphical elements, remains behind all other content by using position: absolute and a z-index of 0. The .background-faded class applies a brightness filter to darken the background when a modal is open, focusing the user's attention on the modal. Similarly, the .fade-out class applies opacity and scaling effects to make elements fade away smoothly when they are no longer in focus.

Hover and Glow Effects:

Both the orbiting buttons and upload buttons feature hover effects, such as scaling and glowing animations. These effects, such as the @keyframes glowing, give the buttons a glowing pulse, drawing the user's attention and enhancing the interactive feel of the interface.

Overall, the CSS file aims to create an engaging and interactive user interface, with dynamic animations, clean layouts, and layered effects that guide user interaction while maintaining a sleek and professional aesthetic.

4.6.3 ReportPage.css



```css
/* === General Styling for the Report Page === */
body {
  font-family: 'Inconsolata', monospace;
  margin: auto;
  position: relative;
  width: 100%;
  display: block;
  height: 100vh;
  background: #2d2b2b;
  color: #dd2c2c;
  background: linear-gradient(to top, #231f20, #3c3c3c);
  background-size: 100%;
}

/* Wrapper for the report and graph */
.wrapper {
  position: absolute;
  display: block;
  left: calc(50% + 15px);
  top: 50%;
  transform: translateX(-50%) translateY(-50%);
  min-width: 600px;
  min-height: 200px;
  background: #231f20;
  box-shadow: 0px 0px 0px 50px #231f20, -30px 0px 0px 50px #231f20;
  z-index: 10;
}

h1 {
  text-align: center;
}

/* === Button Styles === */
.buttons, .download-btn {
  display: flex;
  justify-content: center;
  margin-bottom: 20px; /* Add space between buttons and graph */
  z-index: 11;
```

Figure 166 ReportPage.css

```css
.btn, .download-btn a {
  background: #231f20;
  text-decoration: none;
  padding: 0.5em 1em;
  margin: 0 0.5em;
  color: #dd2c2c;
  transition: 0.3s ease-in-out;
  border: 1px solid;
  pointer-events: auto;
}

.btn:hover, .download-btn a:hover {
  background: #dd2c2c;
  color: #231f20;
}

.active {
  background: #dd2c2c;
  color: #231f20;
  border: 1px solid;
}

/* === Graph Styles === */
.graph {
  width: 100%;
  height: 400px;
  position: relative;
  background: url("data:image/svg+xml,%3Csvg%20version%3D%271.1%27%20id%3D%27Layer_1%27%20xmlns%3D%27http%3A//www.w3.org/2000/svg%27%20xmlns%3Axlink%3D%27http%3A//www.w3.org/1999/xlink%27%20x%3D
  background-size: 8px;
  border-left: 1px solid #dd2c2c;
  border-bottom: 1px solid #dd2c2c;
  overflow: hidden;
  margin-bottom: 20px;
  z-index: 9;
}

.graph-svg {
  width: 100%;
  height: 100%;
  stroke: #dd2c2c;
}
```

Figure 167 ReportPage.css

The ReportPage.css file focuses on creating a visually structured layout for the report page, which includes a graph, buttons for interaction, and download functionality. The overall design emphasizes a dark, professional background with vibrant red highlights for key elements like text, buttons, and borders.

General Styling for the Page:

The body element is styled with the Inconsolata monospace font for a technical look, while a gradient background in shades of dark gray (#231f20 to #3c3c3c) sets the tone

for the page. The body takes up the full height of the viewport (height: 100vh) and uses a centered layout to maintain focus on the content.

Wrapper and Layout:

The .wrapper class is used to center the report content and graph within the viewport. It features a shadow effect that uses the same background color, giving the container a seamless appearance. It also ensures the wrapper has a minimum width and height to accommodate the content, ensuring the layout remains structured regardless of the screen size.

Button Styling:

The .buttons and .download-btn classes control the layout of interactive buttons for toggling between different views or downloading the report as a PDF. The buttons have a consistent style: a dark background with red text and borders. The .btn:hover effect inverts the colors on hover, creating a distinct interaction cue for users. Additionally, the .active class highlights the currently selected button (e.g., for time range or malware/benign view) by changing the background to red and text to dark, providing a clear visual indicator.

```
/* === Download Button Styling === */
.download-btn {
  display: flex;
  justify-content: center;
  margin-top: 20px;
}

.download-btn a {
  background: #231f20;
  text-decoration: none;
  padding: 0.5em 1em;
  margin: 0 0.5em;
  color: #dd2c2c;
  border: 1px solid;
  transition: 0.3s ease-in-out;
  pointer-events: auto;
}

.download-btn a:hover {
  background: #dd2c2c;
  color: #231f20;
}

/* === Total Wrap Styles === */
.totalwrap {    "totalwrap": Unknown word.
  text-align: center;
  font-size: 24px;
  font-weight: bold;
  color: #dd2c2c;
}

.totals {
  font-size: 24px;
  color: #dd2c2c;
}

/* === Ensure Canvas Background (if any) Stays Behind === */
.canvas-background {
  position: absolute;
  top: 0;
  left: 0;
  width: 100vw;
  height: 100vh;
  z-index: 1;
}
```

Figure 168 ReportPage.css

Graph Styling:

The .graph class ensures that the graph occupies a specific area (400px height) and is styled with a dark gray grid-like background pattern and red borders for the axes. The graph background uses an embedded SVG image to create a subtle, repeating texture. The .graph-svg class defines the SVG area for the actual graph lines, with red strokes to match the page's overall color scheme.

Download Button:

The .download-btn section is aligned at the center of the page and styled similarly to the buttons. The anchor (<a>) elements inside the .download-btn have the same hover effect, ensuring consistency across the page's interactive elements. This button is used to trigger the report download functionality.

Total Wrap and Count Styling:

The .totalwrap and .totals classes center the total count information (e.g., total malware/benign files) and apply bold, red text. This styling ensures the total count is prominently displayed and matches the page's color scheme.

Canvas Background:

The .canvas-background class is used to manage any background elements, like a canvas, ensuring they remain behind the main content (z-index: 1) without interfering with the layout. This class maintains the background's positioning and full-screen dimensions.

4.6.4 Scene.gtlf

Figure 169 Scene.gtlf

The provided JSON structure describes a 3D scene, model data, and its associated components in the GLTF format. This file appears to be a detailed representation of a space station model and includes information about the 3D mesh, materials, animations, and textures necessary for rendering the scene.

Accessors

The **accessors** section defines how the data, such as vertices, normals, tangents, and texture coordinates, are accessed from the buffer views. The accessors provide information about the size, component type, and range of values (max and min) for each attribute. For instance, there are accessors for positions (VEC3), normals (VEC3), tangents (VEC4), and texture coordinates (VEC2). The accessors index into buffer views that store the actual vertex data.

Animations

This section describes the **animations** in the scene, specifically targeting the "rotation" of several nodes in the model. Each animation consists of channels that control specific properties (in this case, rotations of different nodes). These animations follow a **linear interpolation** scheme, allowing the nodes to rotate smoothly over time. The **samplers** map time values (input) to keyframe data (output) for smooth transitions between rotation states.

Asset Metadata

The **asset** section provides metadata about the 3D model, including the author, license (CC-BY-4.0), the model's source, and the generator (Sketchfab). This model is named "Space Station 3" and was created by the user **re1monsen** on Sketchfab.

Buffer Views and Buffers

The **buffer views** and **buffers** store all the raw binary data used in the model, such as vertex positions, normals, tangents, and texture coordinates. The buffer views point to specific ranges of data within a buffer. For instance, the scene data is stored in a binary file (scene.bin), and each buffer view points to a specific section of this binary file, depending on what data it holds.

Images and Textures

The **images** and **textures** sections reference textures used in the model. These textures represent various material properties, such as base color, metallic roughness, emissive maps, and normal maps. The images include specific JPEG and PNG files, with names indicating their purpose (e.g., spacestation_main2_baseColor.jpeg for the base color texture).

Materials

The **materials** section describes the visual appearance of the 3D objects. Each material defines how it interacts with light, using properties like **emissive strength**, **specular**, and **clearcoat** to create realistic surface reflections and highlights. The model employs emissive

materials to create glowing effects and uses normal maps and metallic roughness maps for detailed surface appearances. Three materials are defined: **spacestation_main2**, **spacestation_smalllights**, and **spacestation_main**, each using textures and custom properties to define their look.

Meshes

The **meshes** section defines the 3D geometry for different parts of the space station model. Each mesh is composed of several **primitives**, which reference attributes like position, normals, tangents, and texture coordinates, and are associated with specific materials. The mode (4) indicates that the geometry is composed of triangles, the standard primitive for 3D rendering.

Nodes and Hierarchy

The **nodes** section defines the hierarchical structure of the 3D model, representing individual components (such as different parts of the space station) and how they are connected. Each node can contain transformations (such as translation, rotation, or scaling) and references to child nodes or meshes. This section also defines the root node (GLTF_SceneRootNode) that ties the entire scene together.

Scene

The **scene** section specifies the overall structure of the model, listing the nodes that make up the 3D scene. In this case, the model contains a single scene (Sketchfab_Scene), and the root node (Sketchfab_model) forms the starting point of the entire space station hierarchy.

Overall Summary

In summary, this GLTF file describes a complex space station model that includes detailed geometric data, materials, textures, and animations. It uses hierarchical nodes to represent the structure and movement of different parts of the model and employs several advanced techniques like emissive textures, normal maps, and clearcoat reflections to achieve realistic rendering effects. The 3D model is fully equipped with animations, ready for integration into a 3D application or game.

4.7 Screenshots

4.7.1 Interface

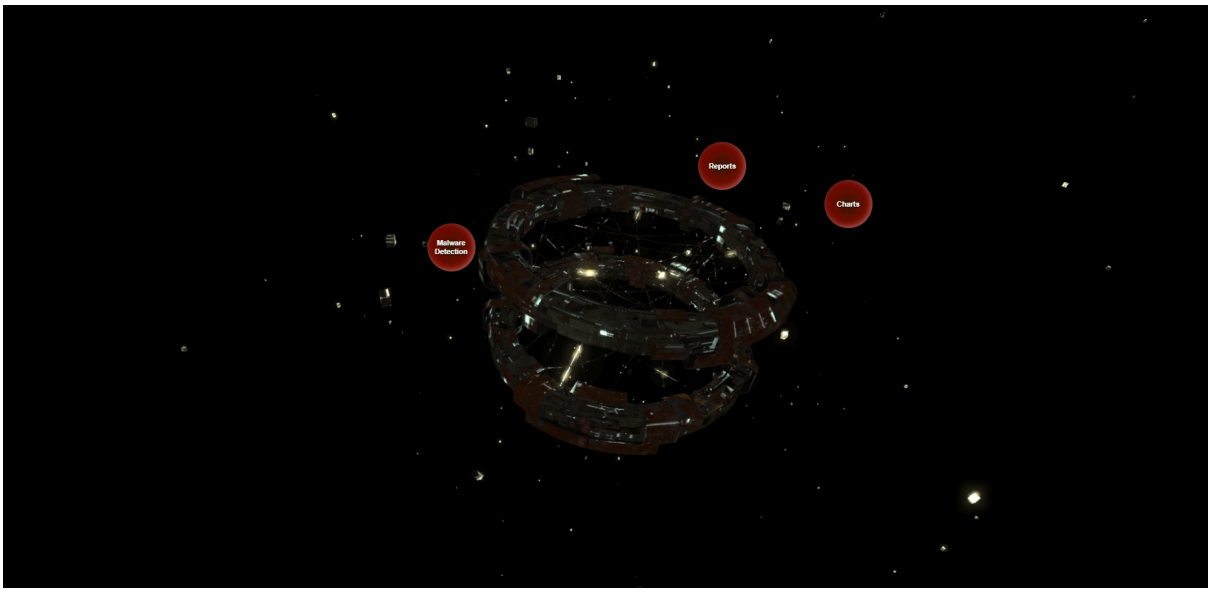


Figure 170 Interface

4.7.2 Upload File or URL

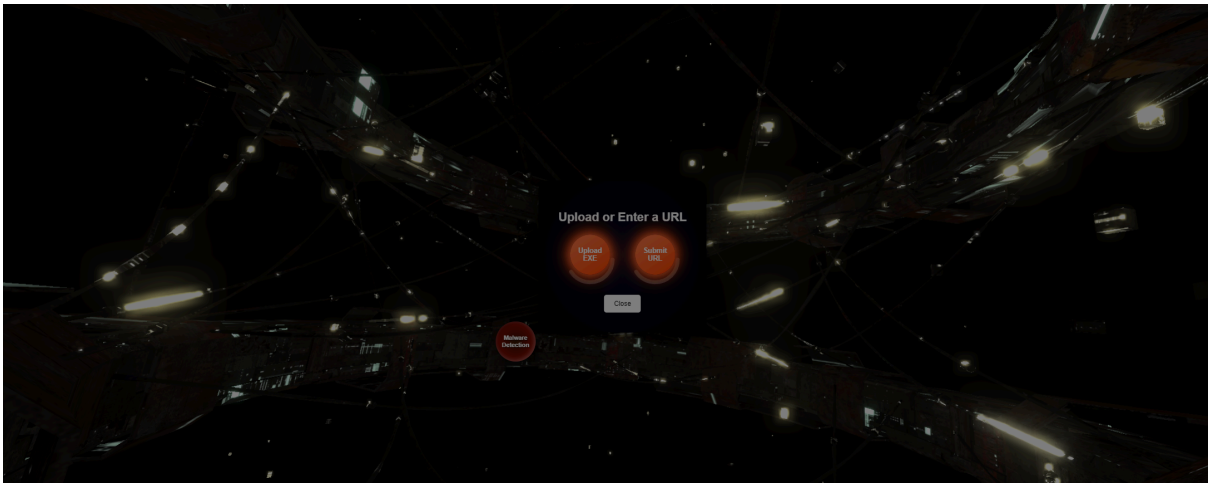


Figure 171 Upload File or URL

4.7.3 Charts

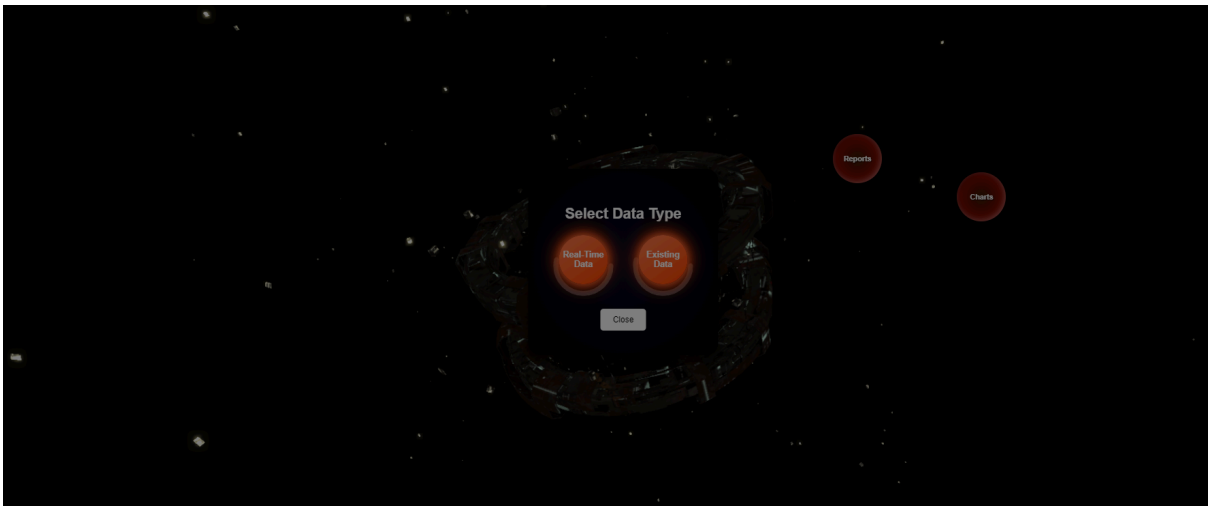


Figure 172 Charts

4.7.4 Real Time Charts

Figure 173 Real time Charts

4.7.5 Existing Data Charts

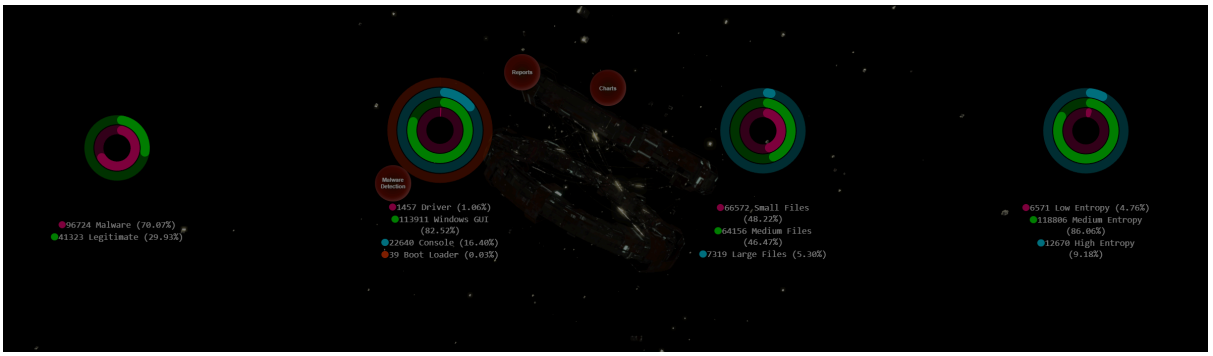


Figure 174 Exisitng Data Charts

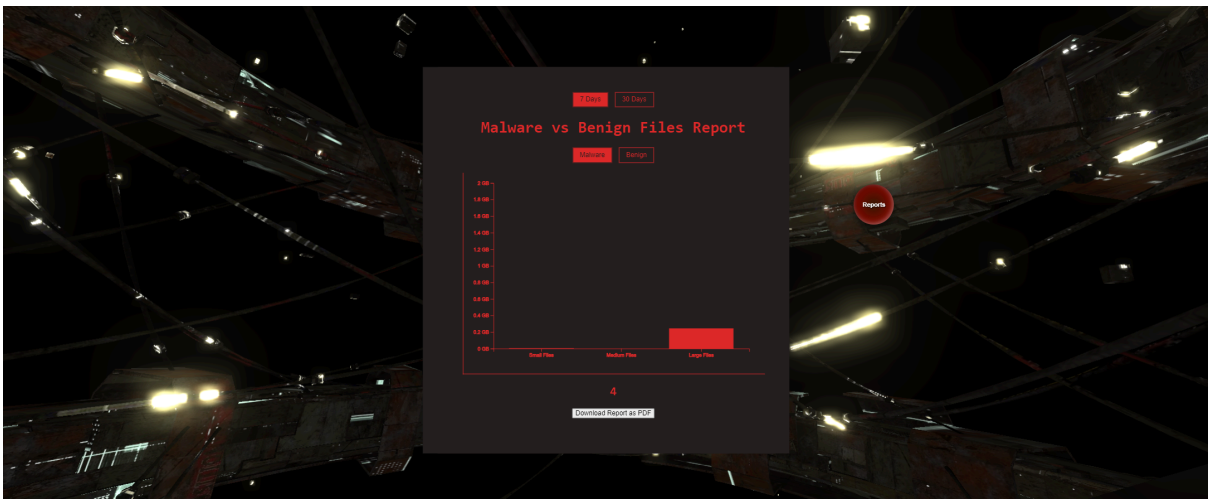


Figure 175 Report

5.0 Chapter 5

5.1 Test plan

5.1.1 Test Plan for Reports

Test Case ID	Test Case	Description	Expected Results	Actual Results	Status	Priority
1.1	View Report	Admin requests to view the report	The system displays the correct report in the UI			High
1.2	Download Report	Admin requests to download the report	The system downloads the report file successfully			High
1.3	Report Data Accuracy	Verify the accuracy of the report data against the database	Report data matches the data stored in the database			High

5.1.2 Test Plan for Malware Detection

Test Case ID	Test Case	Description	Expected Results	Actual Results	Status	Priority
2.1	Malware Detection	Admin uploads a malware file for detection	The system detects the file as malware			High
2.2	Legitimate File Upload	Admin uploads a legitimate file for detection	The system correctly identifies the file as legitimate			High
2.3	Quarantine Malware	Verify the malware file gets quarantined if detected as malware	The malware file is moved to quarantine			High

5.1.3 Test Plan for Existing Charts

Test Case ID	Test Case	Description	Expected Results	Actual Results	Status	Priority
3.1	View Existing Data Chart	Admin requests to view existing chart data	The existing data chart is displayed correctly			High

3.2	Historical Data Accuracy	Verify that historical chart data is accurate	Historical data is fetched correctly from the database			High

5.1.4 Test Plan for Charts

Test Case ID	Test Case	Description	Expected Results	Actual Results	Status	Priority
4.1	View Real-Time Chart	Admin requests to view a real-time chart	The real-time chart is displayed with current data			High
4.2	Chart Rendering Accuracy	Ensure that the chart renders accurately without errors	The chart renders with the correct data points			High

5.2.2 User Acceptance Testing (UAT) Plan

Name	
Job Position	
Date	

Criteria	Rating (1 = Poor, 5 = Excellent)				
	1	2	3	4	5
Meeting System Objectives					
- Malware Detection Accuracy					
- URL & PE File Quarantine					
- Charts (Real-Time, Historical)					
User Interface					
- Intuitive UI Design					
- Clear Chart Visualizations					
User Experience					
- Responsive Charts					
- Fast Loading Time					
Security Posture					
- Secure File Uploads					
- Protection Against Malicious Files					
- Quarantine Management					
Bug Free					
- No Bugs Detected					
System Free From Critical Errors					

Tester Comments	
Actions Taken	

5.2.2.1 User Acceptance Testing (UAT) Plan

Name	Sathish Kumar
Job Position	CS-DO/Digital at Petronas
Date	25/7/2024

Criteria	Rating (1 = Poor, 5 = Excellent)				
	1	2	3	4	5
Meeting System Objectives			✓		
- Malware Detection Accuracy				✓	
- URL & PE File Quarantine				✓	
- Charts (Real-Time, Historical)				✓	
User Interface					
- Intuitive UI Design					✓
- Clear Chart Visualizations					✓
User Experience					
- Responsive Charts				✓	
- Fast Loading Time				✓	
Security Posture					
- Secure File Uploads				✓	
- Protection Against Malicious Files		✓			
- Quarantine Management		✓			
Bug Free					
- No Bugs Detected			✓		

System Free From Critical Errors			✓		

Tester Comments	The .exe file detection works well, but URL detection accuracy is limited due to a small dataset, requiring more diverse URL samples for improvement
Actions Taken	Test cases passed based on existing .exe data.

5.2.2.3 User Acceptance Testing (UAT) Plan

Name	Nurizzaty Nordin
Job Position	HRM/PETH at Petronas
Date	20/7/2024

Criteria	Rating (1 = Poor, 5 = Excellent)				
	1	2	3	4	5
Meeting System Objectives					
- Malware Detection Accuracy				✓	
- URL & PE File Quarantine					✓
- Charts (Real-Time, Historical)					✓
User Interface					
- Intuitive UI Design					✓
- Clear Chart Visualizations					✓
User Experience					
- Responsive Charts					✓
- Fast Loading Time			✓		

Security Posture					
- Secure File Uploads			✓		
- Protection Against Malicious Files			✓		
- Quarantine Management		✓			
Bug Free			✓		
- No Bugs Detected					
System Free From Critical Errors			✓		

Tester Comments	The reporting module functions correctly but lacks detailed insights because of insufficient data, requiring additional input for more comprehensive reports
Actions Taken	Suggested improvements include enhancing the dataset with more diverse examples to test URL detection thoroughly

5.2.2.2 User Acceptance Testing (UAT) Plan

Name	Simonne Wilfred
Job Position	HRM/PD&T at Petronas
Date	10/7/2024

Criteria	Rating (1 = Poor, 5 = Excellent)				
	1	2	3	4	5
Meeting System Objectives					
- Malware Detection Accuracy				✓	
- URL & PE File Quarantine					✓
- Charts (Real-Time, Historical)					✓

User Interface					
- Intuitive UI Design					✓
- Clear Chart Visualizations					✓
User Experience					
- Responsive Charts					✓
- Fast Loading Time			✓		
Security Posture					
- Secure File Uploads			✓		
- Protection Against Malicious Files			✓		
- Quarantine Management		✓			
Bug Free			✓		
- No Bugs Detected					
System Free From Critical Errors			✓		

Tester Comments	The charts display data as expected, but their usefulness is limited by the small dataset, and they will provide better insights with more comprehensive data
Actions Taken	Additional data for URLs is recommended to improve detection and reporting accuracy.

5.2 System Testing

5.2.1 Unit Testing

5.2.1.1 Test plan for Reports

Sathish Kumar

Test Case ID	Test Case	Description	Expected Results	Actual Results	Status	Priority
1.1	View Report	Admin requests to view the report	The system displays the correct report in the UI	Report displayed correctly as per the UI requirements	Passed	High
1.2	Download Report	Admin requests to download the report	The system downloads the report file successfully	Report downloaded without any issues	Passed	High
1.3	Report Data Accuracy	Verify the accuracy of the report data against the database	Report data matches the data stored in the database	Report data was accurate and matched the database but the data displayed are not really useful its as the same as charts	Passed	High

5.2.1.2 Test Plan for Malware Detection

Test Case ID	Test Case	Description	Expected Results	Actual Results	Status	Priority
2.1	Malware Detection	Admin uploads a malware file for detection	The system detects the file as malware	Malware detected accurately for .exe files, not	Passed	High

				accurate for URLs		
2.2	Legitimate File Upload	Admin uploads a legitimate file for detection	The system correctly identifies the file as legitimate	System identifies .exe as legitimate, some URLs misclassified	Passed	High
2.3	Quarantine Malware	Verify the malware file gets quarantined if detected as malware	The malware file is moved to quarantine	Malware quarantined accurately for .exe files	Passed	High

5.2.1.3 Test Plan for Charts

Test Case ID	Test Case	Description	Expected Results	Actual Results	Status	Priority
3.1	View Existing Data Chart	Admin requests to view existing chart data	The existing data chart is displayed correctly	Existing chart data displayed accurately	Passed	High
3.2	Historical Data Accuracy	Verify that historical chart data is accurate	Historical data is fetched correctly from the database	Historical data matches the database records	Passed	High

5.2.1.4 Test plan for Reports

Nurizzaty Nordin

Test Case ID	Test Case	Description	Expected Results	Actual Results	Status	Priority
1.1	View Report	Admin requests to view the report	The system displays the correct report in the UI	Report displayed, but the content was not very useful as not much data had been fed into the system	Failed	High
1.2	Download Report	Admin requests to download the report	The system downloads the report file successfully	Report downloaded successfully, but with insufficient data, the report content is not meaningful	Failed	High
1.3	Report Data Accuracy	Verify the accuracy of the report data against the database	Report data matches the data stored in the database	Report data was accurate, but insufficient data was provided, making the report not very	Failed	High

				usefulsame as charts		

5.2.1.6 Test Plan for Malware Detection

Test Case ID	Test Case	Description	Expected Results	Actual Results	Status	Priority
2.1	Malware Detection	Admin uploads a malware file for detection	The system detects the file as malware	Malware detected accurately for .exe files, but detection for URLs is limited due to insufficient data	Passed	High
2.2	Legitimate File Upload	Admin uploads a legitimate file for detection	The system correctly identifies the file as legitimate	System identifies .exe files as legitimate, but some URLs are misclassified due to limited data input	Passed	High
2.3	Quarantine Malware	Verify the malware file gets quarantined if detected as malware	The malware file is moved to quarantine	Malware quarantined accurately for .exe files, but limited URL	Passed	High

				data prevents full testing of this functionality		

5.2.1.5 Test Plan for Charts

Test Case ID	Test Case	Description	Expected Results	Actual Results	Status	Priority
3.1	View Existing Data Chart	Admin requests to view existing chart data	The existing data chart is displayed correctly	Existing chart data displayed accurately, but with limited data fed into the system	Passed	High
3.2	Historical Data Accuracy	Verify that historical chart data is accurate	Historical data is fetched correctly from the database	Historical data matches the database records, but there is insufficient data for detailed analysis	Passed	High

5.2.1.7 Test plan for Reports

Simonne Wilfred

Test Case ID	Test Case	Description	Expected Results	Actual Results	Status	Priority
1.1	View Report	Admin requests to view the report	The system displays the correct report in the UI	Report displayed correctly as per the UI requirements, though with limited data	Passed	High
1.2	Download Report	Admin requests to download the report	The system downloads the report file successfully	Report downloaded without any issues, though content may be sparse due to limited data	Passed	High
1.3	Report Data Accuracy	Verify the accuracy of the report data against the database	Report data matches the data stored in the database	Report data was accurate and matched the database, but limited data affects the depth of insights	Passed	High

5.2.1.2 Test Plan for Malware Detection

Test Case ID	Test Case	Description	Expected Results	Actual Results	Status	Priority

2.1	Malware Detection	Admin uploads a malware file for detection	The system detects the file as malware	Malware detected accurately for .exe files, though limited data for URL detection	Passed	High
2.2	Legitimate File Upload	Admin uploads a legitimate file for detection	The system correctly identifies the file as legitimate	System identifies .exe files as legitimate, but some URL misclassifications occur due to limited data	Passed	High
2.3	Quarantine Malware	Verify the malware file gets quarantined if detected as malware	The malware file is moved to quarantine	Malware quarantined accurately for .exe files; limited URL data prevents further validation	Passed	High

5.2.1.3 Test Plan for Charts

Test Case ID	Test Case	Description	Expected Results	Actual Results	Status	Priority
3.1	View Existing Data Chart	Admin requests to view existing chart data	The existing data chart is displayed correctly	Existing chart data displayed accurately, though with limited data available	Passed	High

3.2	Historical Data Accuracy	Verify that historical chart data is accurate	Historical data is fetched correctly from the database	Historical data matches the database records, but the data set is limited, reducing the depth of the analysis	Passed	High

6.0 Chapter 6 : Conclusion

6.1 Critical Evaluation

This system demonstrates an innovative and robust approach to malware detection, phishing analysis, and file management through the integration of several advanced components, including machine learning, file analysis, and real-time data visualization. The use of a Flask backend allows for smooth API integration, while the React frontend provides an interactive and visually appealing user interface. Notably, the system includes critical functionalities such as file upload validation, accurate malware detection using machine learning models, and seamless chart rendering using D3.js. These features contribute to a comprehensive user experience, offering key insights into malware and benign file statistics, file quarantine status, and file size distributions. Furthermore, the inclusion of PDF report generation using jsPDF adds a valuable feature for exporting detailed analysis reports. However, certain aspects of the system could benefit from further optimization. For example, error handling during file uploads and API responses should be improved to ensure more robust responses to invalid input or unexpected conditions. Additionally, performance optimization may be necessary for large datasets or files, particularly in chart rendering and API response times.

Despite these minor challenges, the overall structure of the system is well-thought-out and offers significant potential for real-world applications in cybersecurity, malware analysis, and phishing detection.

6.1.1 Achievement of the Overall Project

The overall project has successfully achieved several significant milestones, culminating in the development of a comprehensive, user-friendly cybersecurity system focused on malware detection, phishing analysis, and file management. Key achievements include the successful integration of machine learning models for real-time malware and phishing detection, providing accurate predictions and detailed insights into uploaded files. The project also excels in user interaction through its seamless React-based frontend, which allows users to upload files, view real-time statistics, and download detailed reports. The use of D3.js for data visualization has resulted in dynamic, visually appealing charts that offer intuitive insights into malware prevalence, file size distributions, and quarantine statuses, making complex data easy to interpret.

Moreover, the project demonstrates strong backend capabilities through Flask APIs, ensuring smooth communication between the machine learning models and the frontend. The system's ability to handle multiple file types and generate reports in PDF format using jsPDF highlights its versatility and practical application potential. In addition, the integration of real-time data handling, such as showing statistics over the last 7 or 30 days, has significantly enhanced the system's ability to provide actionable insights. Overall, the project has achieved its core objectives of delivering a reliable, accurate, and interactive malware and phishing detection system, while also incorporating useful features like data visualization and reporting to support comprehensive cybersecurity analysis.

6.1.2 Contribution to Community/Industries

The project makes a valuable contribution to both the cybersecurity community and related industries by offering a highly adaptable and scalable system for malware detection and phishing analysis. In an era where cyber threats are becoming increasingly sophisticated, the need for effective detection mechanisms is paramount. This system directly addresses that need by leveraging machine learning models to offer accurate real-time detection of malware and phishing attempts, which can be used by security professionals, developers, and businesses to protect sensitive data and infrastructures.

For the cybersecurity community, this project provides an open framework that can be built upon, extended, or adapted for various use cases. With its modular design, using React for the frontend, Flask for the backend, and D3.js for data visualization, it is accessible to developers and can be customized for different industry needs. This flexibility fosters collaboration and knowledge-sharing, allowing the community to enhance and refine the detection algorithms, thereby improving the overall security landscape.

In industries where data protection is critical such as finance, healthcare, and e-commerce the system can be integrated into existing IT infrastructures to strengthen defenses against cyberattacks. Its ability to visualize data trends and provide detailed reports on file safety makes it an indispensable tool for decision-making in IT security strategies. By contributing to both community-driven cybersecurity advancements and practical industry applications, the project plays a vital role in mitigating the growing threat of cyberattacks.

6.1.3 Strengths of the Project

The project exhibits several key strengths that enhance its overall effectiveness and applicability in the real world. A significant strength is the comprehensive cybersecurity feature set, which integrates both malware and phishing detection functionalities. This multi-layered approach ensures the system can address a wide range of cyber threats, providing protection against attacks targeting both executable files and URLs. Furthermore, the integration of machine learning significantly enhances the system's performance by enabling automated threat detection, improving accuracy as the model learns from new data, and reducing the need for constant manual intervention. Another strength is the system's real-time analysis capability, which is critical in identifying and responding to emerging threats, minimizing the risk of system compromise. The use of D3.js for data visualization adds another layer of value by allowing security professionals to quickly interpret threat data through intuitive and clear charts. This aids in faster decision-making and response times. Lastly, the project demonstrates scalability and flexibility, making it adaptable to various environments and capable of evolving with the growing complexity of cybersecurity challenges, which ensures its long-term viability and effectiveness.

6.2 Limitation

6.2.1 Limited Resources

One of the key limitations of the project is the constraint on resources, such as processing power, memory, and storage capacity. The system may struggle to handle a high volume of

data, especially when dealing with complex malware detection processes or handling large datasets in real-time. The limited computational power can slow down the detection process and reduce overall system efficiency. This challenge could impact the system's ability to scale and perform optimally under heavy workloads, requiring the deployment of additional resources or optimization strategies for efficient operation.

6.2.2 Scalability Changes

Scalability is another limitation of the project. As the volume of data or the number of users grows, the system may require significant architectural modifications to accommodate these changes. Scaling up a system to handle more traffic, more file uploads, or larger datasets often introduces challenges related to infrastructure, software architecture, and cost. Without a flexible and adaptive architecture, the system may face bottlenecks or performance issues, making it difficult to scale smoothly and efficiently.

6.2.3 Compliance with Data Protection Regulations

A critical limitation that needs to be addressed is ensuring compliance with data protection regulations, such as GDPR (General Data Protection Regulation) or CCPA (California Consumer Privacy Act). Handling sensitive data, such as user information or malware samples, can raise privacy concerns, especially if the data is not properly anonymized or secured. Non-compliance with these regulations can lead to legal repercussions, fines, and damage to the project’s credibility. Therefore, ensuring that all data processing activities comply with relevant legal frameworks is essential but can also be complex and resource-intensive.

6.2.4 Inherent Security Vulnerabilities

The system, by its very nature, faces inherent security vulnerabilities, particularly because it interacts with potentially harmful files and malicious URLs. Despite implementing various security measures, there is always the risk of sophisticated cyberattacks that exploit undiscovered vulnerabilities in the system. Ensuring that the system remains secure against emerging threats will require constant updates, rigorous security testing, and the implementation of robust defensive measures to prevent breaches or the exploitation of security flaws. Without proactive mitigation, these vulnerabilities could compromise both the system and user data.

Folder link

Malware-Detection-using-Machine-learning-main

References

(F.A.) Faitouri A. Aboaoja, (. A.-r. (25 8, 2022). *Malware Detection Issues, Challenges, and Future Directions: A Survey*. Retrieved from Applied sciences: https://www.mdpi.com/2076-3417/12/17/8482

(M. S.) Muhammad Shoaib Akhtar, (. T. (3 9, 2022). *Malware Analysis and Detection Using Machine Learning Algorithms*. Retrieved from Symmetry: https://www.mdpi.com/2073-8994/14/11/2304

Abdulwahab Ali Almazroi, N. A. (3 4, 2024). *Deep learning hybridization for improved malware detection in smart Internet of Things*. Retrieved from National Library of Medicine: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10991293/

Admin. (9 1, 2024). *What Is CylancePROTECT?* Retrieved from HTS: https://www.hts-tx.com/blog?p=what-is-cylanceprotect-240109

Anaconda, I. (2018). *Getting started with Anaconda Distribution — Anaconda documentation*. Retrieved from Anaconda.

Ananya Redhu, P. C. (28 3, 2024). *Deep learning-powered malware detection in cyberspace: a contemporary review*. Retrieved from Frontiers in physics: https://www.frontiersin.org/articles/10.3389/fphy.2024.1349463/full

AppDynamics. (n.d.). *Why DevOps is Important*. Retrieved from AppDynamics: https://www.appdynamics.com/topics/why-devops-is-important

Arntz, P. (18 4, 2022). *Why you shouldn't automate your VirusTotal uploads*. Retrieved from ThreatDown: https://www.threatdown.com/blog/why-you-shouldnt-automate-your-virustotal-uploads/

Austin L.Brown, M. G. (1 2, 2024). *Automated machine learning for deep learning based malware detection*. Retrieved from ScienceDirect,Computers &

Security: https://www.sciencedirect.com/science/article/abs/pii/S0167404823004923

Aviiko. (5 11, 2021). *Windows Defender Advanced Threat Protection eating all available ram before crashing*. Retrieved from Microsoft: https://answers.microsoft.com/en-us/windows/forum/all/windows-defender-advanced-threat-protection-eating/714d08e9-2bab-4435-8d32-d80f9ff9794b

AWS. (n.d.). *What is DevOps?* Retrieved from AWS: https://aws.amazon.com/devops/what-is-devops/

Ayuns Luz, E. F. (2, 2024). *Data preprocessing and feature extraction for phishing URL detection*. Retrieved from ResearchGate: https://www.researchgate.net/publication/378804421_Data_preprocessing_and_feature_extraction_for_phishing_URL_detection

BERNARDO.QUINTERO. (1 4, 2023). *Introducing VirusTotal Code Insight: Empowering threat analysis with generative AI*. Retrieved from VIRUSTOTAL: https://blog.virustotal.com/2023/04/introducing-virustotal-code-insight.html

Bhasin, A. (1 4, 2024). *What is DevOps Life Cycle | Key Components & Phases with Examples?* Retrieved from testsigma: https://testsigma.com/blog/devops-life-cycle/

Bourke, D. (18 2, 2023). *The Top 4 Reasons to Learn PyTorch (and start getting into AI)*. Retrieved from Medium: https://mrdbourke.medium.com/the-top-4-reasons-to-learn-pytorch-and-start-getting-into-ai-3ebc84ec1628

BUCHANAN, I. (n.d.). *4 Key Benefits of DevOps*. Retrieved from ATLASSIAN: https://www.atlassian.com/devops/what-is-devops/benefits-of-devops

Code, V. (3 11, 2021). *Why Visual Studio Code?* Retrieved from Visual Studio Code: https://code.visualstudio.com/docs/editor/whyvscode

Comeau, L. (4 8, 2023). *Mastering Efficiency: The Key Stages of the Agile Software Development Life Cycle*. Retrieved from Scopic: https://scopicsoftware.com/blog/mastering-efficiency-the-key-stages-of-the-agile-software-development-life-cycle/

CylanceProtect overview. (29 8, 2022). Retrieved from Avaya Documentation: https://documentation.avaya.com/bundle/deployingAvayaAnalyticsR42/page/CylancePROTECT_overview.html

Daniel Gibert, C. M. (1 3, 2020). *The rise of machine learning for detection and classification of malware: Research developments, trends and challenges*.

Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/pii/S1084804519303868

Deb, S. (7 11, 2017). *TensorFlow Tutorial — A Comprehensive Guide To Deep Learning Using TensorFlow*. Retrieved from Medium: https://medium.com/edureka/tensorflow-tutorial-ba142ae96bca

Dell Alienware m15 R7: full specs, tests and user reviews. (n.d.). Retrieved from NanoReview.net: https://nanoreview.net/en/laptop/dell-alienware-m15-r7

Developers, N. (2022). *What is NumPy?* Retrieved from Numpy: https://numpy.org/doc/stable/user/whatisnumpy.html

Donato, H. (3 1, 2023). *What Are The Phases Of Scrum?* Retrieved from workamaljig: https://www.workamajig.com/blog/scrum-methodology-guide/scrum-phases

Driscoll, M. (n.d.). *Jupyter Notebook: An Introduction*. Retrieved from Real Python: https://realpython.com/jupyter-notebook-introduction/

DRUMOND, C. (n.d.). *What is scrum and how to get started*. Retrieved from atlassian: https://www.atlassian.com/agile/scrum

Dunlea, (. J. (4 1, 2024). *Machine Learning Techniques for Advanced Malware Detection*. Retrieved from Akkio: https://www.akkio.com/post/malware-detection-using-machine-learning

Dziuba, A. (6 4, 2023). *Navigating the Agile Software Development Life Cycle: Phases, Tools, Roadmap*. Retrieved from RELEVANT SOFTWARE: https://relevant.software/blog/agile-software-development-lifecycle-phases-explained/

Fang Zhiyang, J. W. (1 1, 2019). *Feature Selection for Malware Detection Based on Reinforcement Learning*. Retrieved from ResearchGate: https://www.researchgate.net/publication/337727148_Feature_Selection_for_Malware_Detection_Based_on_Reinforcement_Learning

Fang, H. (3 1, 2024). *Evaluation of Malware Classification Models for Heterogeneous Data*. Retrieved from National Library of Medicine: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10781237/

Felan Carlo C. Garcia, F. P. (9, 2016). *Random Forest for Malware Classification*. Retrieved from ResearchGate: https://www.researchgate.net/publication/308646416_Random_Forest_for_Malware_Classification

G, N. (13 9, 2023). *Mastering the Art of Malware Detection: A Comprehensive Guide for Netflow/IPFIX Specialists*. Retrieved from Linkedlin:

https://www.linkedin.com/pulse/mastering-art-malware-detection-comprehensive-guide-niels-groeneveld

geeksforgeeks. (26 7, 2023). *Python Seaborn Tutorial*. Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/python-seaborn-tutorial/

Gibert, D. (1 5, 2024). *Certified Adversarial Robustness of Machine Learning-based Malware Detectors via (De)Randomized Smoothing*. Retrieved from https://arxiv.org/html/2405.00392v1

Gigahoo. (29 3, 2017). *Microsoft*. Retrieved from Is virustotal.com legimate?: https://answers.microsoft.com/en-us/windows/forum/all/is-virustotalcom-legimate/a9d9038c-d9fa-4f7e-9136-015f7a32769b

Ijraset. (n.d.). *Malware Detection and Classification Using Machine Learning Algorithms*. Retrieved from IJRASET: https://www.ijraset.com/research-paper/malware-detection-and-classification-using-ml-algorithms

India, M. (18 9, 2023). *Security Beyond Antivirus and Firewalls — Microsoft Defender ATP*. Retrieved from Medium: https://medium.com/@matrix3d.india/security-beyond-antivirus-and-firewalls-microsoft-defender-atp-63a0140a8500

Jadia, H. (10 8, 2023). *nternational Research Journal of Modernization in Engineering Technology and Science* . Retrieved from PDF: In the sentiment analysis study, metrics such as accuracy, precision, recall, F1-score, and ROC curves are used to evaluate the Logistic Regression models' performance. Comparably, model performance in malware detection can be assessed using comparable me

James, G. (n.d.). *10 Reasons to Use Agile Software Development*. Retrieved from QualityLogic: https://www.qualitylogic.com/knowledge-center/10-reasons-to-use-agile-software-development/

Javier Yuste, E. G. (1 5, 2022). *Optimization of code caves in malware binaries to evade machine learning detectors*. Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/pii/S0167404822000426

Jr, J. P. (3 5, 2021). *When machine learning is hacked: 4 lessons from Cylance*. Retrieved from TechBeacon: https://techbeacon.com/security/when-machine-learning-hacked-4-lessons-cylance

Jupyter, P. (2024). *JupyterLab*. Retrieved from Installation: https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html

KAISPE. (2024). *Overview Of Scrum, Principles And Its Pros And Cons*. Retrieved from KAISPE: https://www.kaispe.com/scrum-processes-blog/

Kamran Shaukat, S. L. (1 5, 2023). *A novel deep learning-based approach for malware detection*. Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/abs/pii/S0952197623002142

Kaspersky. (7 3, 2021). *Machine Learning for Malware Detection*. Retrieved from www.kaspersky.com: https://media.kaspersky.com/en/enterprise-security/Kaspersky-Lab-Whitepaper-Machine-Learning.pdf

Key features of CylancePROTECT Desktop. (n.d.). Retrieved from BlackBerry: https://docs.blackberry.com/en/unified-endpoint-security/blackberry-ues/overview/What-is-BlackBerry-Protect-Desktop/Key-features-of-BlackBerry-Protect-Desktop

kissflow. (21 12, 2023). *kissflow*. Retrieved from The 9 Key Benefits of Using the Agile Methodology: https://kissflow.com/project/agile/benefits-of-agile/

Laoyan, S. (2 2, 2024). *asana*. Retrieved from What is Agile methodology? (A beginner's guide): https://asana.com/resources/agile-methodology

M.Gopinath, S. C. (1 2, 2023). *A comprehensive survey on deep learning based malware detection techniques*. Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/abs/pii/S1574013722000636

Marcus Botacin, F. C. (1 7, 2021). *Challenges and pitfalls in malware research*. Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/abs/pii/S0167404821001115

mijacobs, j. K.-t. (25 1, 2023). *What is DevOps?* Retrieved from Microsoft Build: https://learn.microsoft.com/en-us/devops/what-is-devops

Mohanan, R. (11 8, 2021). *What Is DevOps? Definition, Goals, Methodology, and Best Practices*. Retrieved from spiceworks: https://www.spiceworks.com/tech/devops/articles/what-is-devops/

Muchammad Naseer, J. F. (4, 2021). *Malware Detection: Issues and Challenges*. Retrieved from ResearchGate: https://www.researchgate.net/publication/351030106_Malware_Detection_Issues_and_Challenges

Muhammad Azeem, D. K. (2023 Dec 12). National Library Of Medicine. *Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches*, 20.

Muhammad Azeem, D. K. (1 1, 2024). *Analyzing and comparing the effectiveness of malware detection: A study of machine learning approaches*. Retrieved from ScienceDirect: https://www.sciencedirect.com/science/article/pii/S2405844023107821

Muhammad Shoaib Akhtar, T. F. (3 10, 2022). *Malware Analysis and Detection Using Machine Learning Algorithms*. Retrieved from MDPI: https://www.mdpi.com/2073-8994/14/11/2304

Namita Dahiya, P. C. (1, 2021). *PE File-Based Malware Detection Using Machine Learning*. Retrieved from ReserachGate: https://www.researchgate.net/publication/342640653_PE_File-Based_Malware_Detection_Using_Machine_Learning

Nos, D. (28 12, 2015). *7 reasons to use Scrum in project development*. Retrieved from Linkedin: https://www.linkedin.com/pulse/7-reasons-use-scrum-project-development-david-nos/

Online, S. (11 8, 2023). *Top 10 Powerful Python Libraries for Data Science - Shiksha Online*. Retrieved from Shiksha.com: https://www.shiksha.com/online-courses/articles/top-10-powerful-python-libraries-for-data-science/

Pros and Cons of BlackBerry Protect (CylancePROTECT) 2024. (15 12, 2023). Retrieved from TrustRadius: https://www.trustradius.com/products/blackberry-protect/reviews?qs=pros-and-cons#overview

Rajesh Kumar, G. S. (31 8, 2022). *Malware classification using XGboost-Gradient Boosted Decision Tree - Advances in Science, Technology and Engineering Systems Journal*. Retrieved from Advances in Science, Technology and Engineering Systems Journal: https://astesj.com/v05/i05/p66/

Ram Srinivasan, B. M. (n.d.). *What is scrum?* Retrieved from ScrumAlliance: https://www.scrumalliance.org/about-scrum

S, K. (7 9, 2023). *VirusTotal: A Powerful Tool for Security Researchers and Analysts*. Retrieved from Linkedlin: https://www.linkedin.com/pulse/virustotal-powerful-tool-security-researchers-analysts-keerthi-s/

Sacolick, I. (2 4, 2024). *Infoworld*. Retrieved from What is agile methodology? Modern software development explained: https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-software-development-explained.html

Saiiful Islam Raimon, M. H. (21 10, 2022). *Malware Detection and Classification Using Hybrid Machine Learning Algorithm*. Retrieved from Springer Link: https://link.springer.com/chapter/10.1007/978-3-031-19958-5_39

Sandhu, S. (27 10, 2023). *Exploring Windows Defender Advanced Threat Protection (ATP).* Retrieved from Linkedlin: https://www.linkedin.com/pulse/exploring-windows-defender-advanced-threat-protection-shwetank-sandhu-sdtdc/

schools, W. (2024). *Matplotlib Tutorial*. Retrieved from W3 schools: https://www.w3schools.com/python/matplotlib_intro.asp

SCRUMstudy. (8 12, 2022). *Why Should You use Scrum?* Retrieved from SCRUMstudy: https://www.scrumstudy.com/article/why-should-you-use-scrum

SCRUMstudy. (n.d.). *Scrum Phases and Processes*. Retrieved from SCRUMstudy: https://www.scrumstudy.com/whyscrum/scrum-phases-and-processes

Sharon Shea, A. S. (22 2, 2024). *cybersecurity*. Retrieved from Security: https://www.techtarget.com/searchsecurity/definition/cybersecurity

Shivanandhan, M. (14 2, 2023). *How to Use TensorFlow for Deep Learning – Basics for Beginners*. Retrieved from freeCodeCamp: https://www.freecodecamp.org/news/tensorflow-basics/

Simplelearn. (3 11, 2023). *What is PyTorch, and How Does It Work: All You Need to Know*. Retrieved from Simplelearn: https://www.simplilearn.com/what-is-pytorch-article

Siosulli. (24 4, 2024). *Why you should use Microsoft Defender Antivirus together with Microsoft Defender for Endpoint - Microsoft Defender for Endpoint*. Retrieved from Microsoft Learn: https://learn.microsoft.com/en-us/defender-endpoint/why-use-microsoft-defender-antivirus

Smita Ranveer, S. H. (n.d.). *SVM Based Effective Malware Detection System*. Retrieved from PDF: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=647a288efdf44a79da75df7027f88610372d32a1

Sosa, G. C. (24 1, 2023). *A comparison between Scrum, DevOps, and Agile*. Retrieved from Medium: https://medium.com/strategio/a-comparison-between-scrum-devops-and-agile-ad9468acb58d

Staff, C. (3 4, 2024). *https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-t*

o-using-python. Retrieved from coursera: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

Tao Feng, M. S. (3 9, 2022). *Malware Analysis and Detection Using Machine Learning Algorithms*. Retrieved from MDPI: https://www.mdpi.com/2073-8994/14/11/2304

Tao Feng, M. S. (1, 2023). *Evaluation of Machine Learning Algorithms for Malware Detection*. Retrieved from National Library of Medicine: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9862094/

Team, J. (2015). *The Jupyter Notebook — Jupyter Notebook 7.1.3 documentation*. Retrieved from Jupyter: https://jupyter-notebook.readthedocs.io/en/stable/notebook.html

Technologies, U. (2024). *The 7 phases of the DevOps lifecycle*. Retrieved from unity: https://unity.com/solutions/devops-lifecycle

VIRUSTotals 2021 Malware Trends report. (3, 2022). Retrieved from PDF: https://assets.virustotal.com/reports/2021trends.pdf

W3.CSS. (2024). *Pandas Introduction*. Retrieved from W3 schools: https://www.w3schools.com/python/pandas/default.asp

What Is a Decision Tree? | Master's in Data Science. (8 May, 2024). Retrieved from Master in Data Science (CORP-MIDS1 (MDS)): https://www.mastersindatascience.org/learning/machine-learning-algorithms/decision-tree/

What is Cybersecurity? | IBM. (27 8, 2023). Retrieved from https://www.ibm.com/topics/cybersecurity

What Is Cybersecurity? (22 2, 2024). Retrieved from Cisco: https://www.cisco.com/c/en/us/products/security/what-is-cybersecurity.html

What is Microsoft Defender Advanced Thread Protection? | Chorus. (30 8, 2023). Retrieved from Chorus: https://www.chorus.co.uk/resources/what-is-microsoft-defender-advanced-threat-protection-atp/

Yegulalp, S. (1 9, 2023). *What is Python? Powerful, intuitive programming*. Retrieved from InfoWorld: https://www.infoworld.com/article/3204016/what-is-python-powerful-intuitive-programming.html

Ziegler, S. (n.d.). *The 5 phases of DevOps maturity*. Retrieved from ICF: https://www.icf.com/insights/technology/the-5-phases-of-devops-maturity