MALWARE DETECTION USING XAI

MINI PROJECT REPORT

Submitted by

VARSSHA BALASUNDARAM 210701325

KARISHMA KANNADASAN 210701106

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

ANNA UNIVERSITY:: CHENNAI 600 025

APRIL 2024

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Report titled "**Malware detection using XAI**" is the bonafide work of **"Varssha Balasundaram (210701325), Karishma Kannadasan (210701106)"** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Rahul Chiranjeevi. V

Assistant Professor,
Department of Computer Science and Engineering,

Rajalakshmi Engineering College,
Chennai – 602015

Submitted to Mini Project Viva-Voce Examination held on _________________

Internal Examiner **External Examiner**

ABSTRACT

A problem that has started since the last one to two decades, and has risen to a concerning level in recent times is malware. Malware can steal or manipulate our data. Especially confidential or sensitive data. Malware can also take control of our systems and cut off services. Malware has become more and more advanced. Traditional signature checking has proven to be ineffective against recent malwares. More technologically advanced methods need to be implemented. In latest times, machine learning has become popular in the field of malware detection. Machine learning and deep learning have proven to be very effective and less time consuming. This paper aims to delve into the application of machine learning models in malware detection. Classification is the most efficient method for this purpose. Under classification, the features and attributes of a set of files are analyzed. Then the files are flagged as malware or not. Classification techniques like decision trees and random forests are used. These techniques will help flag them. But many of these techniques might still give some false positives. False positives is when a file is safe but is still flagged as malicious software. So, now we need to understand that ML in malware detection can work efficiently up till detecting the malware, but it all takes place under wraps. The layers of algorithms all run in a black box and will not show up for the developer or the user. This makes understanding and debugging very difficult. Hence , to make the whole detection process more transparent, it will be very effective to combine eXplainable Ai with Ml. XAI method- LIME or local interpretable model agnostic explanation. This will help security systems, reduce false positives and help tremendously with debugging. It will give information of the most influential feature and also provide explanations on why a file was flagged as malware. By applying ml and xai techniques, we can create an efficient malware detection system.

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman **Mr. S.MEGANATHAN, B.E, F.I.E.**, our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN**, **Ph.D.,** for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P. KUMAR, Ph.D.**, Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide, **Mr. Rahul Chiranjeev.V** Professor, Department of Computer Science and Engineering. Rajalakshmi Engineering College for his valuable guidance throughout the course of the project.

Varssha Balasundaram: 210701325
Karishma Kannadasan:210701106

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ABBREVIATIONS

XAI	Explainable Ai
LIME	Local Interpretable model agnostic explanations
CART	Classification and Regression Tree
API	Application program interface
APK	Application Package
URL	Uniform resource locator

CHAPTER 1
INTRODUCTION

1.1 GENERAL

The widespread adoption of smartphones has revolutionized our lives, but it has also created vulnerabilities for malicious actors. They deploy malware disguised as legitimate apps to steal data, disrupt phone functionality, or launch attacks. To combat this threat, robust and explainable malware detection systems are crucial.This project tackles this challenge by building a system specifically for the Android platform. It leverages decision trees, known for their interpretability, allowing us to understand the logic behind the model's classifications of apps as malicious or benign. This is essential for security applications. Additionally, Explainable AI (XAI) techniques can provide even deeper insights into the decision-making process. By understanding "why" an app is flagged as malicious, we can build user trust and take appropriate actions. The project utilizes the "Android Malware Detection" dataset from Kaggle to train the decision tree model and analyze the features that contribute most to its classifications. XAI techniques, like LIME, might be explored to gain localized explanations for individual app classifications. Ultimately, this project aims to deliver a robust and interpretable malware detection system that safeguards the Android ecosystem and fosters user trust through clear explanations.

1.2 OBJECTIVE

This project tackles the growing threat of mobile malware on the Android platform by building a robust and explainable detection system. The system will leverage decision trees for classifying apps as malicious or benign, while also prioritizing interpretability. By analyzing the key features that influence these classifications, the project aims to provide clear explanations for why apps are flagged as malware. Additionally, Explainable AI (XAI) techniques might be explored to offer even deeper, localized insights into individual app classifications. This focus on interpretability fosters user trust and empowers informed decision-making when encountering potential threats. The project will utilize the "Android Malware Detection" dataset from Kaggle for training and evaluate the model's performance using established metrics. Ultimately, this project aspires to deliver a valuable tool that safeguards the Android ecosystem by offering a robust, transparent, and user-centric approach to malware detection.

1.3 EXISTING SYSTEM

This project doesn't directly address a single existing malware detection system. There are various commercial and open-source solutions available for Android, but their inner workings are often shrouded in secrecy. This project aims to build a new system from the ground up, prioritizing interpretability and explainability.

While existing systems might utilize signature-based detection or machine learning models, their decision-making logic can be opaque. This project tackles this by leveraging decision trees. These offer inherent interpretability, allowing us to understand the features (permissions, API calls) that most influence malware classifications. Additionally, we might explore Explainable AI (XAI) techniques to provide even deeper, localized explanations for why specific apps are flagged as malicious. This focus on transparency fosters user trust and empowers informed decisions when encountering potential threats. By offering a clear understanding of the system's reasoning, this project contributes a valuable tool to the overall landscape of Android malware detection.

1.4 PROPOSED SYSTEM

The project proposes a novel malware detection system for Android that prioritizes interpretability. It leverages decision trees, a machine learning model known for its clear decision-making logic. These trees classify apps as malicious or benign based on a sequence of features extracted from the app, such as requested permissions or code patterns. The project utilizes the "Android Malware Detection" dataset from Kaggle, which provides pre-processed features, eliminating the need for complex feature engineering.Data pre-processing might involve handling missing values or encoding categorical features. The pre-processed data is then split for training and testing the decision tree model. Training allows the model to learn the decision rules based on feature values. The testing set evaluates the model's performance using metrics like accuracy and precision.Beyond basic performance, the project emphasizes interpretability. By analyzing the decision tree, we can understand which features (permissions, code patterns) are most indicative of malware. Additionally, XAI techniques like LIME might be explored to provide localized explanations for why specific apps are flagged as malicious. This focus on interpretability fosters user trust and empowers informed decision-making when encountering potential threats. The proposed system offers a valuable tool for the Android ecosystem by providing a robust and explainable approach to malware detection.

CHAPTER 2
LITERATURE SURVEY

In [1], it is said that most of the current existing malwares are polymorphic and metamorphic malware. They are able to evolve from their older forms and can change their structure and function. Many malware detection models use static features of the malware files. But these cannot be consistently trusted all the time. But one other disadvantage of malware is that it cannot hide its malicious behaviour during run-time. Hence, a malware detection model built on the behaviour patterns of the model can be more resourceful. This will also help reduce the false-negative rate. For the process of decision making, a gradient boosting algorithm is used. All the features are fed into it. Hence, the detection rate is improved. This makes the model dependable.

In [2], we come to know that cyber dangers are more common as an effect of our growing dependence on technology and services provided by the cloud. Since it may be hard to identify attacks due to advanced persistent threats, Endpoint Detection and Response (EDR) was implemented in 2013. Each endpoint computer has a scanning application installed by EDR to track and record events and logs. However, a lightweight malware detector is required because EDR is prone to malware attacks. Since prior research has not included EDR, image-based malware classification uses a mechanism to group malware based on its representative image. The aim of this work is to combine EDR with a malware classifier that uses images. Deep Ocean Protection System (DOPS), a rudimentary EDR solution, has been created using two models that have been trained (Mobilenet V2 and Inception V3) that have been adjusted using BODMAS and MalImg.

[3] tells us that, recent research shows that the amount of malicious programs, or malware, is rising rapidly. Before it infects a lot of systems, the malware must be found in order to protect computer networks and the Internet against it. Many studies on methods for detecting malware have been done lately. Both heuristic- and signature-based detection techniques are quick and effective in finding known malware, but signature-based techniques especially have not been able to find unknown malware. This shows how difficult it is to develop a good malware detection tool and how numerous opportunities there are for new research and methods.
This study provides a thorough analysis of malware detection techniques and modern methods that make use of these methods.

The authors of [4] say that malware detection for Android is vital. Permission pair-based detection approaches show the most potential for actual detection among other kinds. Conventional mechanisms, however, are not capable of simultaneously meeting the efficiency demands for practical use. Although the most recent method depends on differences in frequently seen pairs between malware and secure software, it is not stable enough. This is due to the fact that most current malware requires additional rights in order to imitate real programs, leaving the use of the frequencies pointless. In this research, they suggest the detection of Android malware based on a Composition Ratio (CR) of permission pairs, which satisfies all the requirements.Additionally, the features can actually give precise information that helps in understanding of detection results by users. Our approach can be put to use effectively because it fulfills all the requirements.

Unwanted software is known as malicious software (malware), and cybercriminals commonly use it to launch cyberattacks. [5] suggests that it is necessary to use fresh methods that differ greatly from traditional methods in order to combat new virus types. This research proposes an innovative deep learning-based architecture that is capable of classifying malware types with a hybrid model. The study's main contribution is the introduction of a new hybrid design that effectively combines two different pre-trained network models. Data collection, deep neural network architecture design, deep neural network architecture training, and evaluation are the four main stages of this architecture. The results of the experiment show that the suggested method is capable of accurately effectively classifying malware.

[6] says that Artificial intelligence (AI) studies for developing methods for intrusion detection systems (IDS) are motivated by the rapid increase of intrusions on networked systems. Especially the use of explainable AI (XAI) methods in real-world intrusion detection systems is driven by the need to fully understand and interpret these AI models to security analysts (the person who is in charge of these IDS to safeguard their networks). We offer an end-to-end framework in this work to assess black-box XAI techniques for network intrusion detection systems. For network intrusion detection, we evaluate the global and local scopes of various black-box XAI techniques. We examine six unique evaluation standards for SHAP and LIME, two renowned black-box XAI techniques. Analytical accuracy, sparsity, stability, durability, and completeness are these measures. They include key KPIs from the AI and network security fields.

In [7], we see that ensuring that dangerous URLs are properly detected is crucial for maintaining Internet Web security. It is important yet challenging to develop an efficient detection system that can increase the accuracy of classifying harmful webpages since the detection outcomes of current techniques are low and their efficiency is low. In order to deal with this issue, this study proposes a Markov detection tree approach that uses the link connections of unified resource locators to automatically find and classify hazardous webpages. Two methods to process the null attribute values of webpages are provided in order to improve the detection accuracy for malicious webpages. We assess how well our algorithms work under different application circumstances.

According to [8], the anti-malware community is becoming more worried due to the growing trend of malware threats, even though there are many security solutions available. Malware still poses a serious risk to users of the web. We do a deep analysis of the malware field from 1970 to the present, focusing on attack paths, malware kinds, operational processes, and vulnerabilities. We also explore various defenses created in response to these dynamic threats. Our results demonstrate the increasing complexity of malware attack trends, such as an increase in ransomware, supply chain attacks, cloud-based attacks, ransomware attacks, attacks on mobile phones, attacks on Internet of Things devices, cryptojacking, fileless malware, advanced persistent threats, and attacks on edge networks. Along with these evolving dangers, defense techniques have also changed, giving higher importance for multilayered security measures.

In [9], the author states that corresponding users can't seem to optimize the decisions made by DL models, nor understand and trust their decisions. In order to get over these limitations, a fresh approach called Explainable Artificial Intelligence (XAI) offers a number of methods for understanding and deciphering the predictions made by DL models. To detect IoT-related intrusions, our framework depends on a unique intrusion detection system (IDS) for IoT networks that we also create using deep neural networks. Also, three main XAI approaches are used by our system, such as Interpretable Model-Agnostic Explanations (LIME). Our system can optimize how decisions based on DL are viewed. While broad reasons focus on finding the key factors that have led to every decision made, local explanations try to explain DL output. Therefore, our suggested model increases trust and transparency in the decision-making process.

According to [10], for a while, malware has been a problem on the internet and in computer systems. The industry and researchers have continually improved detection and prevention techniques to tackle malware that has grown more and more evasive. We provide an accurate and detailed review of the state of distribution of malware as it stands today in this article. We gathered a dataset over the course of 287 days that includes 99,312 malicious binary samples from 38,659 malware sites of distribution for the analysis. We conduct an extensive evaluation of the gathered malware binaries and URLs using our dataset to deliver current data and insight into the enemy's actions. We study malware distribution websites as well as malware binaries that are downloaded from them.

CHAPTER 3
SYSTEM DESIGN

3.1 GENERAL

System design involves the formulation and creation of systems that meet the specific needs of users. Fundamentally, the essence of studying system design lies in comprehending the individual elements and how they interact with each other.

3.2 DEVELOPMENT ENVIRONMENT

3.2.1 HARDWARE SPECIFICATIONS

This document offers a comprehensive overview of the hardware and its implementation, detailing the key components, their interactions, and the necessary requirements for seamless connectivity to utilities and installation.

Table 3.2.1 Hardware Specifications

PROCESSOR	Intel Core i5
RAM	4GB or above (DDR4 RAM)
GPU	Intel Integrated Graphics
HARD DISK	6GB
PROCESSOR FREQUENCY	1.5 GHz or above

3.2.2 SOFTWARE SPECIFICATIONS

The below table constitutes a thorough evaluation of requirements that precedes the more detailed phases of system design, aiming to minimize the need for subsequent revisions. Furthermore, it should offer a practical foundation for estimating product expenses, potential risks, and project timelines.

Table 3.2.2 Software Specifications

FRONT END	HTML, CSS, Bootstrap, JavaScript
FRAMEWORK	Python, Django
CODE EDITOR	Visual Studio Code

3.3 SYSTEM DESIGN

3.3.1 ARCHITECTURE DIAGRAM

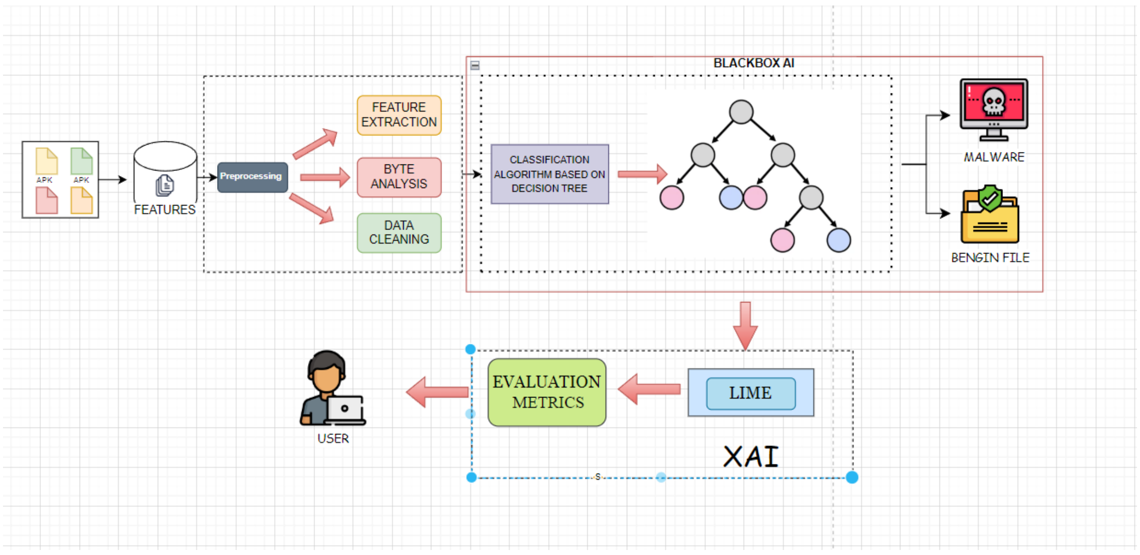


Fig 3.3.1 Malware detection

The architecture of the project can be seen as a sequence of related phases. Initially, information is obtained from "Android Malware Detection" dataset. After that, the data—likely in a CSV format—goes through pre-processing to deal with values that are missing, encode category characteristics, and maybe even scale or select features for the best possible model performance. Next, training and testing sets are created from the pre-processed data; a typical split is between 70–80% for training and 20–30% for testing. Should there be an imbalance of classes in the dataset, tiered splitting may be used.

Using algorithms like CART or C4.5, a decision tree model is trained during the core stage. Based on the features offered, our algorithm trains to categorize apps as either benign or malware. To maximize the decision tree's efficiency, hyperparameter tuning is an optional procedure. Once trained, metrics like accuracy, precision, recall, and F1 score are used to assess how useful the model is. An additional tool for visualizing the model's performance on both malicious and safe apps is a confusion matrix.

Ultimately, Explainable AI (XAI) methods are applied to obtain more profound understanding of the model's decision-making procedure. Decision trees' innate interpretability makes it possible to determine which characteristics are most important for classifying malware. Furthermore, local explanations for particular app classifications can be produced using methods such as LIME, which emphasize the characteristics that most affected the model's conclusion for each unique app. At the end of these phases is a malware detection system that has been trained and assessed, using decision trees and XAI.

CHAPTER 4
PROJECT DESCRIPTION

4.1 MODULE DESCRIPTION

APK-

Acts like a container for Android apps. It holds everything an app needs to run, from its code and resources to icons and images. Think of it as a box with all the ingredients for a mobile program! While handy for sideloading apps, be cautious about downloading APKs from untrusted sources, as they might harbor malware.

FEATURE EXTRACTION-

feature extraction is like summarizing a news article. You take a big chunk of text (raw data) and identify the key points (features) - who, what, when, where - to create a more concise and informative representation for training models.

DECISION TREE-

A decision tree is like that, but for machine learning. It asks a series of "yes" or "no" questions based on features (data points) to classify things. Easy to visualize, it shows the steps the model takes to reach a decision.

LIME-

LIME (Local Interpretable Model-Agnostic Explanations) is a technique used in machine learning to understand why a model makes a specific prediction for a single data point.

XAI-

XAI removes the smoke and mirrors, helping us understand how models arrive at their predictions. By analyzing features and their importance, XAI techniques like decision tree visualization or LIME explanations give us insights into the model's decision-making process.

CHAPTER 5
IMPLEMENTATION AND RESULTS

5.1 IMPLEMENTATION

The acquisition of pertinent data is the fundamental component of our powerful malware detection technology. Getting a pre-processed dataset with features taken from Android applications (APK files) will be our first step.The data set of the application that contain a APK file is necessary . Our model will be trained on this type of dataset, which will provide it with the information it needs to differentiate between safe and dangerous apps and also helps in determining why we have classified it into a malware and benign . Using a pre-processed dataset simplifies the data for our model and lets it concentrate on its primary function of learning by removing the need for laborious processes like parsing raw APK files.

After obtaining the data, we will initiate a thorough investigation process to thoroughly examine its complexities. To determine the characteristics available, this entails carefully analyzing the structure of the dataset that is provided . These features could include the kinds of API calls the app makes to communicate with the Android system, the permissions it requests (such as location or camera access), and even code snippets that are contained within the app. It is essential to comprehend the various data kinds (numerical or categorical) and the descriptions that go along with them in order to properly grasp the model's conclusions. We may examine any connections between the traits and display their distribution to obtain even more profound understanding. This primary stage is crucial because it enables us to understand the data that the model will use to efficiently classify apps as benign or malicious.Thet model needs to understand what type of method is used to classify it into malware or not.

DATA PRE-PROCESSING:

Pre-processing procedures make sure the data is prepared for use in the decision tree model after it has been evaluated. Cleaning the data and preparing it for a model that uses machine learning are necessary steps in the data preprocessing process, which also improves the machine learning model's accuracy and efficiency. Removing data points with missing values is done if the missing values are rare and do not correspond to a particular app activity. use statistical techniques such as mean/median imputation to replace missing values in the data with typical values. This method is predicated on the random distribution of missing values. As an alternative, we can use different method to deal with missing values.

For example, if a particular permission is absent from an app, add a new binary feature that indicates that it is lacking. When there are several features available, we should take into account feature selection strategies to determine which are most useful for malware detection. Analyse the relationship between the target variable (malware or benign) and the attributes using correlation analysis. It is likely that features that have a strong association with the target variable are more informative.

Recursive Feature Elimination method retrains the model after repeatedly eliminating the least significant feature according to a predefined criteria. It assists in locating a more focused subset of highly predictive traits. By concentrating on the most relevant features, feature selection enhances interpretability, lowers model complexity, and may even result in improved performance. Look at scaling the dataset's features to a common range if they have significantly dissimilar scales (e.g., number of permissions vs. frequency of API calls). Using normalization, features are scaled to a range of 0 to 1.This sets a mean of 0 and a standard deviation of 1 for the features. By ensuring that every feature has the same effect on the decision-making process, scaling can enhance the decision tree's performance. Scaling might not be required, yet, if the features already have accessible scales built in.

SPLITTING DATA:

A machine learning model which is required in dividing the data into training and testing set for the malware detection .Always set of data is having two categories one is “malware “ and another is Benign “.The data splitting is required to know that our decision tree model is effectively used to determine .Their are two primary methods of splitting of data exist which mainly be known as random split and stratified split. The data is divided arbitrarily based on the selected ratio using a random split. Although simple, it might not ensure that both malicious and safe apps are fairly represented in both sets. This might be an issue for malware detection, since it's important to detect even a tiny percentage of harmful software.Stratified splitting is the recommended method for datasets that are imbalanced. This method guarantees that the proportionate representation of both classes (malicious and benign) exists in the testing and training datasets. This is necessary for a strong malware detection model to function successfully in real-world malware identification settings and to learn from a wide variety of data sets.And this type of splitting requires to maintain a clear splitting of data into malware and safe.

DECISION TREE:

While there are many decision tree algorithms available, CART (Classification And Regression Trees) and C4.5 are common options for classification. Using an iterative methodology, CART starts at the root node with all of the training data. The feature that produces the "purest" split—that is, the child nodes that result have a higher concentration of a single class—malware or benign—than the parent node—is chosen after all features have been analyzed. Until a stopping requirement is satisfied, this procedure iteratively splits each child node by determining the best feature for additional splitting. Though it has more features, such as managing missing values and "pruning" the tree, C4.5 is comparable to CART. Pruning ensures that the final tree generalizes by removing extraneous branches that may have arisen from overfitting on the training set.

The training data is divided into two parts according to the selected feature and its value. Data points having the selected feature value are represented by one branch, while those without are represented by the other. In essence, a certain feature is being used to categorize the data.

Recursive learning: The first split is not where the process ends. Every child node is where the algorithm continues recursively. Each child node's data is analyzed, and the feature that best divides the data into subsequent child nodes is chosen once more. Such recursive splitting keeps going until a stopping criterion is met, making sure the tree catches the most important differences in the data between malicious and benign programs.Overfitting occurs when a decision tree grows out of control, making the model function badly on unknown input and memorize the training set too well. Stopping criteria are put in place to stop this. Typical standards consist of:

The method of maximum depth is restricting the tree's maximum depth minimizes overfitting and limits its complexity. It could be difficult for a deeper tree to generalize to unknown facts.The another method of minimum samper per leaf is determining the least quantity of data points necessary at a leaf node guarantees that the model has sufficient information to confidently classify the data at that node.

MODEL EVALUATION:

To evaluate the model's performance in identifying apps as malicious or benign, we need use a variety of metrics. The percentage of apps in the testing set that were accurately classified as either benign or malicious. Although a high accuracy is ideal, if the dataset is uneven (more benign apps than malware), it could not be the most revealing metric. The percentage of apps that the model successfully detected as malware (true positives) out of all apps that were predicted to be malware. A high precision means the model detects malware accurately and doesn't produce a lot of false positives, which are harmless apps that are mistakenly flagged as malicious. The percentage of real malware apps in the testing set that are true positives, or apps that are successfully detected as malicious.

XAI and LIME:

Significant security concerns are introduced by the wide app ecosystem and the widespread use of Android smartphones. Apps that are malicious yet look safe might steal confidential information, tamper with how a device works, or even transmit other malware. Strong and understandable malware detection technologies are essential to counter this danger. This paper describes how to use Explainable AI (XAI) methods, particularly LIME (Local Interpretable Model-Agnostic Explanations), when combined with decision trees. A large number of pre-processed features taken from Android applications (APK files) are available in the "Android Malware Detection" dataset. These functions provide clarity on app behaviour without getting into the complex details of raw APK parsing. We explored a few important feature categories and how they may be used in a malware detection system that relies on decision trees.

Permissions: In order for Android apps to access certain features (such the camera and location), they must have permissions. Malicious intent may be indicated by specific permissions being present or absent. When a game software that looks innocent asks to access location data, for example, it may be cause for concern. We can create branches in the decision tree according to whether an application asks for rights (like location access for a photo-editing app) that are frequently abused by malware.

API Calls: Application Programming Interfaces (APIs) are the ways through which applications interact with the Android OS. An application's usage habits and frequency of API calls might provide information about how it operates. Suspicious API calls, like those for network connections to known malware domains or for gaining access to private device functions, may be made by malicious programs. We can include branches in the decision tree according to the frequency with which the application calls the network or the existence of particular API calls linked to malware.

Code N-grams: These are word or instruction sequences from the app's code. Despite the fact that the raw code is not included in the dataset, n-grams can reveal information about the features of the program. Malicious intent may be indicated by the existence of code n-grams, such as those used for data encryption or deception, that are known to be employed in malware. Depending on which code n-grams are present and linked to known malware families, the decision tree may have branches.

LIME focuses on providing context for each of a model's unique predictions. LIME can be used in the context of the malware detection system to explain the decision tree's classification of a particular testing set app as malicious. LIME first looks for applications that, based on their features, are similar to the app that is being stated in the training data. These comparable apps serve as a point of reference for understanding the categorization of the main app. Based on the individual app instance, LIME builds a more basic model (often a linear model) that roughly represents the decision tree's behaviour for that app and its surrounding apps. Using an analysis of the weights given to the attributes in this location, the prediction is explained.

In a situation LIME explains why a program is categorized as malicious. The explanation may indicate that the main elements that shifted the balance towards categorizing the app as malware were its request for a mix of permissions (such as access to location and storage) and the existence of code n-grams linked to data theft methods. LIME's detailed explanation is really helpful. It helps to recognize dangerous programs and comprehend the particular traits that led to their categorization. This is particularly helpful for more research and possible decision tree model improvement. Decision trees' interpretability and LIME's explanatory ability is combined to create a reliable malware detection solution for Android apps that will protect consumers.

5.2 OUTPUT SCREENSHOTS

Using decision trees and LIME for interpretability, we have established a structure for a transparent malware detection system. What has been done and the expected results are laid down as follows.We loaded the CSV file, which included a class label and malware traits from extracted features of android (binary coded as 0s and 1s). We investigated the data types and found that, while one or two columns may be categorical, most of the columns show binary attributes.Depending on the data and analytic demands, we eliminated rows containing missing values or imputed the missing values (e.g., using mean for numerical columns).

The data was divided into testing and training sets. The testing set is used to evaluate the model's performance once it has been constructed with the training set.We developed a decision tree model, a class of machine learning models characterized by their interpretability.Metrics like accuracy, true positive rate (TPR), and false positive rate (FPR) have been computed systematically. These metrics, however, are available through the model evaluation procedure. To provide meaning for the decision tree model's individual predictions, we employed LIME (Local Interpretable Model-Agnostic Explanations).LIME enables us to comprehend which features—such as certain malware traits—have the biggest impact on the model's ability to determine if a given data item is malicious or not.

Using the attributes in the information, the decision tree model will grow to differentiate between samples that are malicious and those that are benign.LIME will explain each prediction, highlighting the key elements the model took into account when classifying the data. This enables us to evaluate the transparency of the model and comprehend the reasoning behind its judgments.To improve the decision tree model's performance, tuning its hyperparameters appropriately is done and followed by testing the model

.The predicted labels and actual labels from the testing set can be used to create the confusion matrix once the decision tree model has been trained and tested.The accuracy, TPR, and FPR can be obtained by inserting the number of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) from the confusion matrix into the appropriate formulas.To evaluate the success rate of the model, computation of the evaluation metrics (accuracy, TPR, and FPR) on the testing set is done. To understand the behavior of the model, use LIME to explain predictions on samples that are misclassified as well as those that are correctly classified as malware or benign.Examined the LIME explanations to determine which characteristics are most important for the model's choices and to find possible areas for development.

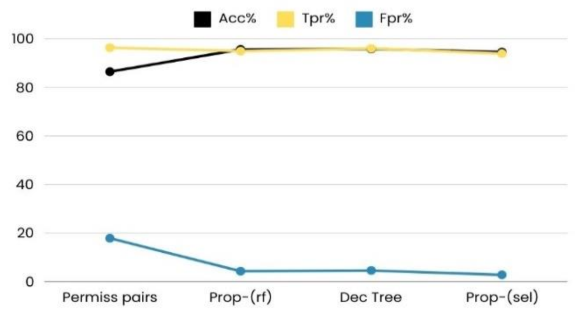


Fig :5.2 Analysis on DREBIN android dataset

SCHEMES	DREBIN android dataset		
	Acc %	Tpr %	Fpr %
Permiss pairs	86.5	96.4	17.9
Prop-(Rf)	95.7	96.3	4.3
Dec Tree	95.9	96.1	4.6
Prop-(sel)	94.6	94.2	2.8

Table 5.3.1 : Precision table

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENTS

6.1 CONCLUSION

A serious security risk brought about by the widespread use of Android smartphones and the extensive app store is the presence of illegal apps which appear as trustworthy ones. These applications could spread further viruses, steal confidential information, or disrupt device operations. We need effective and understandable malware detection solutions to counter this danger. The paper explores developing a transparent defense against mobile threats by utilizing Explainable AI (XAI) techniques in alongside decision trees. Decision trees, in contrast to complex black-box models, provide built-in interpretability. Their arrangement shows an arrangement of features used for classification, similar to an if-then-else flowchart. These attributes in our system cover permissions that the app requests and are taken from the "Android Malware Detection" dataset. API calls it makes, and code n-gram sequences. By analyzing the decision tree, we can understand which features are most indicative of malware.

For instance, a branch in the tree might check if an app requests location access for a seemingly unrelated purpose like a photo editing app. This interpretability fosters trust and transparency in the model's decision-making process, as users can understand the rationale behind malware classifications.It produces code n-gram sequences and API calls. We can identify the characteristics that are most suggestive of malware by examining the decision tree. For example, a branch in the tree might determine whether an application requests location access for a reason that doesn't seem connected. Due to users' capacity to comprehend the reasoning behind malware classifications, the interpretability of the model promotes transparency and trust in its decision-making process. It takes time to develop a trustworthy and understandable malware detection system. This project shows how decision trees and XAI work well together to achieve transparency. We can build a reliable system by adding explainable features, complex algorithms, and feature engineering to LIME. In addition, the inclusion of real-time detection, dynamic updates combining fresh malware samples, and user feedback helps guarantee that the system stays flexible and efficient within a continually evolving mobile threat environment. This effort opens the door for more XAI research for mobile security applications in addition to helping to construct a clear and understandable malware detection system.

6.2 FUTURE ENHANCEMENTS

For future enhancements, we can add several features. At the moment, feature engineering uses human judgment to find features that may be instructional. This procedure can be somewhat automated through the use of XAI algorithms. XAI could suggest additional features or feature combinations that might increase detection accuracy without compromising interpretability by examining the behavior of the model and feature interactions. Reports on incorrectly classified apps can be submitted by users under the existing system. This can be further enhanced by adding user-supplied labels (benevolent or malicious) for individual programs. Over time, the system could improve its decision-making process by including insights from these labels from users. But measures must be taken to stop bad actors from tampering with the feedback loop. Interpretable reasons for on-device classifications are essential as the system approaches real-time detection. Even in environments with limited resources, techniques such as LIME can be modified for real-time situations and provide consumers with basic justifications for why an app is reported as malicious.

REFERENCES

[1]A. A. Al-Hashmi et al., "Deep-Ensemble and Multifaceted Behavioral Malware Variant Detection Model," in IEEE Access, vol. 10, pp. 42762-42777, 2022, doi: 10.1109/ACCESS.2022.3168794.

[2]T. H. Hai, V. Van Thieu, T. T. Duong, H. H. Nguyen and E. -N. Huh, "A Proposed New Endpoint Detection and Response With Image-Based Malware Detection System," in IEEE Access, vol. 11, pp. 122859-122875, 2023, doi: 10.1109/ACCESS.2023.3329112.

[3]Ö. A. Aslan and R. Samet, "A Comprehensive Review on Malware Detection Approaches," in IEEE Access, vol. 8, pp. 6249-6271, 2020, doi: 10.1109/ACCESS.2019.2963724.

[4]H. Kato, T. Sasaki and I. Sasase, "Android Malware Detection Based on Composition Ratio of Permission Pairs," in IEEE Access, vol. 9, pp. 130006-130019, 2021, doi: 10.1109/ACCESS.2021.3113711.

[5]Ö. Aslan and A. A. Yilmaz, "A New Malware Classification Framework Based on Deep Learning Algorithms," in IEEE Access, vol. 9, pp. 87936-87951, 2021, doi: 10.1109/ACCESS.2021.3089586.

[6]O. Arreche, T. R. Guntur, J. W. Roberts and M. Abdallah, "E-XAI: Evaluating Black-Box Explainable AI Frameworks for Network Intrusion Detection," in IEEE Access, vol. 12, pp. 23954-23988, 2024, doi: 10.1109/ACCESS.2024.3365140.

[7]J. Liu, M. Xu, X. Wang, S. Shen and M. Li, "A Markov Detection Tree-Based Centralized Scheme to Automatically Identify Malicious Webpages on Cloud Platforms," in IEEE Access, vol. 6, pp. 74025-74038, 2018, doi: 10.1109/ACCESS.2018.2882742.

[8]J. Ferdous, R. Islam, A. Mahboubi and M. Z. Islam, "A Review of State-of-the-Art Malware Attack Trends and Defense Mechanisms," in IEEE Access, vol. 11, pp. 121118-121141, 2023, doi: 10.1109/ACCESS.2023.3328351.

[9]Z. A. E. Houda, B. Brik and L. Khoukhi, "“Why Should I Trust Your IDS?”: An Explainable Deep Learning Framework for Intrusion Detection Systems in Internet of Things Networks," in IEEE Open Journal of the Communications Society, vol. 3, pp. 1164-1176, 2022, doi: 10.1109/OJCOMS.2022.3188750.

[10]S. Huh, S. Cho, J. Choi, S. Shin and H. Lee, "A Comprehensive Analysis of Today’s Malware and Its Distribution Network: Common Adversary Strategies and Implications," in IEEE Access, vol. 10, pp. 49566-49584, 2022, doi: 10.1109/ACCESS.2022.3171226.

PLAGIARISM REPORT (PROJECT REPORT)

PAPER WITH PLAGIARISM REPORT

Malware Detection using XAI

Karishma Kannadasan
Computer Science and Engineering
Rajalakshmi Engineering College
Chennai, India
210701106@rajalakshmi.edu.in

Varssha Balasundaram
Computer Science and Engineering
Rajalakshmi Engineering
Chennai, India
210701325@rajalakshmi.edu.in

line 1: 3rd Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

Abstract*—An issue that has started since the last one to two decades, and has risen to a concerning level in recent times is malware. Malware can steal or manipulate our data. Especially confidential or sensitive data. Malware can also take control of our systems and cut off services. Malware has become more and more advanced. Traditional signature checking has proven to be ineffective against recent malwares. More technologically advanced methods need to be implemented. In latest times, machine learning has become popular in the area of malware detection. Machine learning and deep learning have proven to be very effective and less time consuming. This paper aims to look into the application of machine learning algorithms in malware detection. Classification is the most efficient method for this purpose. Under classification, the features and attributes of a set of files are analyzed. Then the files are flagged as malware or not. Classification techniques like decision trees and random forests are used. These techniques will help flag them. But many of these techniques might still give some false positives. False positives is when a file is safe but is still flagged as malicious software. So, now we need to understand that ML in malware detection can work efficiently up till detecting the malware, but it all takes place under wraps. The layers of algorithms all run in a black box and will not show up for the developer or the user. This makes understanding and debugging very difficult. Hence , to make the whole detection process more transparent, it will be very effective to combine eXplainable Ai with Ml. XAI method- LIME or local interpretable model agnostic explanation. This will help security systems, reduce false positives and help tremendously with debugging. It will give information of the most influential feature and also provide explanations on why a file was flagged as malware. By applying ml and xai techniques, we can create an efficient malware detection system.

I. Introduction

The coined term malware is from the foundation of malicious software-which is termed as a harmful program that enters the computer system without the owner's knowledge. Malware is of various types and of different forms , which contains its own method of dealing with the users system and the hackers motive of financial gain or stealing personal detail .As similar to the biological virus ,it requires a host for the action to take place in the users system and spread from one system to another system when the file or the program moves to the neighbouring system.Worms are the other form of malware which jump from one device to another device due to the network flaws .The other malware so called the Trojan horses which may mislead the user to install the programs in the system and steal the data from the user.Silently watching the activities of the user in the background is Spyware .Their is a another worst malware which would encrypt the malware user data and then ask for ransom which is called Ransomeware ,after giving the ransom the system would be let to unlock.The human error which would likely give rise to the opportunity and motivation for the malware proliferation.The rise of vulnerabilities arise due to the connections of the virtual environment.Malware can enter systems through unpatched software, unsecure networks, and human error (clicking on dubious links).

The thought of malware may arise due to many reasons ,one among the rest may be the motivation for money which is by ransomware ,demand for money and steak bank informations for improper uses, Espionoge and disruption are another factors where the hackers may target the sensitive informations and the infrastructure of the user's system. The ease with which malware can be created and distributed, along with its potential for gain or destructiveness, ultimately encourages malevolent actors to continue producing these always changing threats.
A worldwide ransomware attack which is called as WannaCry on 2017 which lead to a very high impact to the computer system all over the world and damaged them disruptly . Due to the use of the vulnerabilities on Microsoft Windows the malware -worm spread over thousands of computer system across different countries in more then 150 nations .The worldwide impact spread over many businesses, hospitals and governamental organisations ,some of users need to pay a millions of dollars in order to restore the lost data. After the impact of the worldwide attack the cybersecurity came into light where all the businesses where to do software updates and backups .After the impact everyone were caution about the malware and were ready to defend against these ever-evolving danger which could happen any time and watched them carefully.

Another wave of malware attack was "Heartbleed Bug" that had a serious attack on the security protocol OpenSSL. The vulnerabilities that allowed the hackers to access the control over the private data that is sent through internet, including the bank credentials that was floating through the internet. This attack left over numerous website and millions of users which had an impact on social media, businesses, governmental and non-governamental organisations. Strong encryption and ongoing security vulnerability monitoring and patching of vital software were made imperative by the Heartbleed Bug occurrence. It also brought attention to how seemingly small errors in the digital infrastructure we depend on can have far-reaching consequences. The NotPetya Wiper

Attack of 2017 which is often known as the wiper malware which was initiated to erase data. This is a ransomeware and demanded money. This attack happened all over the world and cost billions of dollars to damage which was directed on Ukrinian entities. NotPetya illustrated how malware may wreak havoc on a large scale in addition to generating profits.

Malware had been a problem since computing started booming. When ARPANET became famous and was used by many in 1971, the self-replicating virus creeper began attacking the systems through ARPANET. This is when the official war against malware began. The early detection methods were very intuitive and relied on gut feeling and spotting unusual behaviour. It was basically based on the human's knowledge and ability to spot it. It was not logical or uniform. It had to be done manually. So as malware evolved, the need for more uniform and systematic malware detection also rose. Then in the 1980s, a new method of malware detection came to be born. This new method was signature-based detection. This was a new effective way to fight against malware. The people who are well versed in security do the job of breaking down and analysing malware samples. They then identify unique patterns in their code or actions. These patterns become signatures and they are used to detect malware. These signatures are maintained in a vast database and are downloaded by anti-virus vendors to create detection software. This software scans suspicious files and secludes the file if a signature match is found. However, this method has a drawback. The malware scene keeps changing with codes and can escape from getting detecting. Signature based technique is useless against zero-day attacks. Zero-day attacks are ones never tackled before. So, they can easily harm a system before the attacks' signature is distributed. Hence, they go undetected.

The older method of detecting malware was through signatures. This was based on the property that specific patterns existed for malware. And then these repeating patterns were recognized. But this was no feasible because malware kept evolving and new strains or malware were constantly coming up. So now the area of machine learning can contribute great help in this matter. Machine learning is also an evolving field and is as fast or even faster than malware. Hence, it provides and adaptable solution. Machine learning algorithm have the feature of learning by themselves from given data and from past activities. Programming need not be done for every single detection rule and be update every time a new data is added. They have the ability to skim through huge datasets of malware and identify patterns. They can pull out repeating patterns and relationships. Hence, by analysing they are able to pick out malicious files from legitimate files. Features analysed are file size, code, system calls etc. Hence, using machine learning can help catch up to the evolving field of machine learning.

In machine learning, there are several different models. One of them is called classification. Classification is used to categorize data points into separate classes. When this is applied in malware detection, the two classes will be malware and benign. The machine learning model will segregate the input data points depending on the characteristics and then predict the class to which the data points will belong. For this project, we will be using the decision tree model. Decision tree also falls under the category of Classification. It functions by dividing the data into different categories based on the rules derived from the features. These rules are easy and not complicated. This is best for interpretable machine learning. The tree keeps dividing based on questions for each feature. Each step will answer a yes or no or true or false question. Based on that the node will be pushed to one direction. By repeatedly doing so for each feature, the node will finally reach a leaf node, and this leaf node represents the final class or malware or benign. Decision tree has several benefits like interpretability and simplicity. In complex models, we can easily understand the logic behind the model's prediction. Decision trees are also simple to understand and implement. They also make predictions efficiently.

II. LITERATURE SURVEY

In [1], it is said that most of the current existing malwares are polymorphic and metamorphic malware. They are able to evolve from their older forms and can change their structure and function. Many malware detection models use static features of the malware files. But these cannot be consistently trusted all the time. But one other disadvantage of malware is that it cannot hide its malicious behavior during run-time. Hence, a malware detection model made from the behavior patterns of the model can be more resourceful. This will also help reduce the false-negative rate. For the process of decision making, a gradient boosting algorithm is used. All the features are fed into it. Hence, the detection rate is improved. This makes the model dependable.

In [2], we come to know that cyber dangers are more common as an effect of our growing dependence on technology and services provided by the cloud. Since it may be hard to identify attacks due to advanced persistent threats, Endpoint Detection and Response (EDR) was implemented in 2013. Each endpoint computer has a scanning application installed by EDR to track and record events and logs. However, a lightweight malware detector is required because EDR is prone to malware attacks. Since prior research has not included EDR, image-based malware classification uses mechanism to group malware based on its visuals. The aim of this work is to combine EDR with a malware classifier that uses images. Deep Ocean Protection System (DOPS), a rudimentary EDR solution, has been created using two models that have been that have been adjusted using BODMAS and MalImg.

[3] tells us that, current studies shows that the amount of malicious programs, or malware, is rising rapidly. Before it attacks multiple of systems, the malware must be found in order to safeguard network of computer. Several studies on methods for identifying malware have been done lately. Both heuristic- and signature-based detection techniques are fast and efficient in finding known malware, but signature-based techniques especially have not been able to find unknown malware. This shows how difficult it is to develop a good malware detection tool and how numerous opportunities there are for new studies and methods.This study provides a proper analysis of malware detection techniques and modern methods that use of these methods.

The authors of [4] say that malware detection for Android is vital. Permission pair-based detection approaches show the most potential for actual detection among other kinds. Conventional mechanisms, however, are not capable of simultaneously meeting the efficiency demands for practical use. Although the most recent method depends on differences in frequently seen pairs between malware and secure software, it is not stable enough. This is due to the fact that most current malware requires additional rights in order to imitate real programs, leaving the use of the frequencies pointless. In this research they suggest the detection of Android malware depends on a Composition Ratio (CR) of permission pairs, which satisfies all the requirements. Additionally,the features can actually give precise information that helps in understanding of detection results by users. Our approach can be put to use effectively because it fulfills all the requirements.

Unwanted software is known as malicious software (malware), and cybercriminals commonly use it to launch cyberattacks. [5] suggests that it is necessary to use fresh methods that differ greatly from traditional methods in order to combat new virus types. This research proposes an innovative deep learning-based architecture that is capable of classifying malware types with a hybrid model. The study's main contribution is the introduction of a new hybrid design that effectively combines two different pre-trained network models. Data collection, deep neural network design, deep neural network architecture instruction, and evaluation are the four main stages of this architecture. The results of the experiment show that the suggested method is able to accurately and effectively segregating malware.

[6] says that Artificial intelligence (AI) studies for developing methods for intrusion detection systems (IDS) are motivated by rapid increase of intrusions on networked systems. Especially the use of explainable AI (XAI) methods in reality intrusion detection systems is driven by the need to fully comprehend and interpret these AI models to security analysts (the person who is in charge of these IDS). We offer an end-to-end framework in this work to assess black-box XAI techniques for network intrusion detection systems. For network intrusion detection, we evaluate the global and local areas of various black-box XAI techniques. We examine six unique evaluation standards for SHAP and LIME, two renowned black-box XAI techniques. Analytical accuracy, sparsity, stability, maintenance, and complete are these measures. They include key KPIs from the AI and network security fields.

In [7], we see that ensuring that dangerous URLs are properly detected is crucial for maintaining Internet Web security. It is important yet tough to develop an effective detection system that is able to increase the accuracy of classifying harmful webpages since the detection outcomes of present techniques are low and their effectiveness is low. In order to deal with this issue, this study proposes a Markov detection tree approach that uses the joint connections of unified resource locators to automatically find and classify hazardous webpages. Two methods to process the null attribute values of webpages are provided in order to better the detection accuracy for malicious webpages. We then assess how well our algorithms work under different application circumstances.

According to [8], the anti-malware community is becoming more worried due to the growing trend of malware threats, even though there are many security solutions available. Malware still poses a serious risk to users of the web. We do a deep analysis of the malware field from 1970 to the present, focusing on attack paths, malware kinds, operational processes, and vulnerabilities. We also explore various defenses created as response to these dynamic threats. Our results demonstrate the highering complexity of malware attack trends, such as an increase in ransomware, supply chain attacks, cloud-based attacks, ransomware attacks, attacks on mobile phones, attacks on Internet of Things devices, cryptojacking, fileless malware, advanced consistent threats, and attacks on edge networks. Along with these evolving dangers, defense techniques have also changed, giving higher importance for multilayered security measures.

In [9], the author states that corresponding users can't seem to optimize the decisions made by DL models, nor understand and trust their decisions. In order to get over these limitations, a fresh approach called Explainable Artificial Intelligence (XAI) offers a number of methods for understanding and deciphering the predictions made by DL models. To detect IoT-related attacks, our frame depends on a unique intrusion detection system (IDS) for IoT networks that we also create using deep neural networks. Also, three main XAI approaches are used by our system, such as Interpretable Model-Agnostic Explanations (LIME). Our system can optimize how decisions based on DL are viewed. While broad reasons concentrate on finding the key factors that have led us to every decision made, local understandings try to explain DL output. Therefore, our suggested model increases trust and transparency in the decision-making process.

According to [10], for a while, malware has been a problem on the internet and in computer systems. The industry and researchers have continually improved detection and prevention techniques to tackle malware that has grown more and more evasive. We provide an accurate and detailed review of the state of distribution of malware as it stands today in this article. We gathered a dataset over the course of 286 days that includes 98,321 malicious binary samples from 37,689 malware sites of distribution for the analysis. We conduct an extensive evaluation of gathered malware binaries and URLs using our dataset deliver current data and insight into the enemy's actions. We study malware distribution websites as well as malware binaries that are downloaded from them.

In [11] Malware is a threat to user's computer system, mobile phones and things that are connected to internet like (IoT) which gave a rise to the online services The methods like obfuscation, packaging, coding and encryption are done to then control the operations of malware attack. the novel malware version which include the method of evading to figure out the malware detection. The three areas which mainly concentrate on the malware detection are : feature selection (FS) approaches, machine learning (ML) techniques, and deep learning (DL) techniques. We have determined the shortcomings and research spaces as well as some future guidelines to create an effective malware detection framework based on the review of the literature.

The MeMalDet from [12] approach where the memory analysis based stacked ensemble learning and deep autoencoders. MeMalDet will help prevent human feature engineering by employing deep autoencoders to extract optimised attributes from memory bundles in an unsupervised way. Then the process is carried out by stacked ensemble of supervised classifier is done. The previously conducted malware detection is of 98.82% and 98.72% which lately had a improvement over a memory analysis based malware detection. MeMalDet uses memory analysis to combine the advantages of supervised machine learning ensemble segregation with representation learning for long-term, efficient malware detection.

The anti-detection capabilities of [13] which deals with the RL-based technique which elevates the window Pe malware in black-box scenarios. The method mainly deal with the reinforcement learning which is capable of modifying the environmental feedback. The agent investigation of uncharted territory and pick up practical evasion technique. The generative antagonistic arrange(GAN) helps to utilize and change engineered information, that replaces the antagonistic payload to decrease the hazard of hard-coded programs. Even after the utilization of these event's victory rate remains 48%-67%.

The past 10 years of research to examine the drawbacks of android malware detector. this work on EvadeDroid [14] ,a problem spaced adversarial approach which lead to the malware detection in android through black-boxed .The EvadeDroid works when the malware programmes has similar opcode-level in the problem space and is found out using the n-gram technique repeatedly iterating this process will reduce malicious instances into benign .The EvadeDroid which is run on 1000 malware app proved to show that this is one of the optimisation method that helps to transform malware into a query-efficient manner. The evasion rate after the conduction of many practices is sad to be 79%.

The Heterophilic antibodies which have the potential in the [15] laboratory test results. When a well-old man of 55 years came up with a Rituximab-treated Mantel cell which was falsely positive as HIV test result. On the go of further investigation a high level of human anti-mouse antibodies(HAMA) was found. The PEG treatment which lead to a negative result for HIV test, indicated that HAMA was the result of positive in HIV serologic test.

The IoT based health care system [16] that would likely provide a proper precise and more accurate data that can be diagnosed easily for decision making in healthcare.The decision tree technique that can handle real time medical parameter for the patients data analysis .The new technology of Mobile edge computing that is used in collaboration of BS generated data and then lately transform it to real time data.The Internet of Medical Things to Fog Interoperability of task scheduling that helps in optimisation of latency Fog Computing and Data segregation.

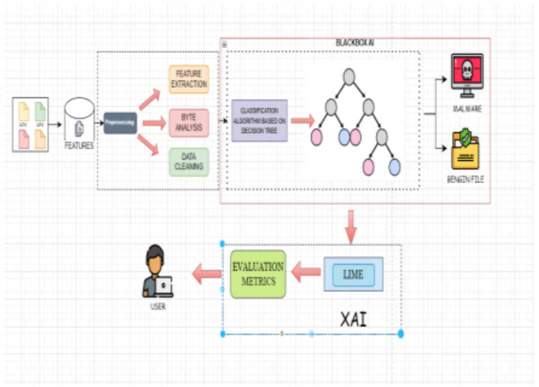
The malware detection of [17] using mobile computing to provide secure and privacy -preserving machine learning .Researchers use many techniques to prevent the hacking of the personal informations from the mobile devices to analyse and detect .They help to keep this informations protected by using machine and deep learning.Many methods are being used by the researches like differential privacy ,homomorphic encryption and federal leraning.The risk of malware id reduced by the federated learning in real time data-set that is used for mobile malware detection.

The technique that uses the Internet of thing (IoT) for the detection of malware by using the Android malware detection [18] framework.This kind of framework is used to characterize by the lightweight and the easy use of the algorithm.This helps to classify between the malicious and benign apps.As this method has executed the most accurate form and helps in the negligence of the false positive and decreases the rate of error.

A Deep learning research [19] with the real time data which provide the predictions for the model,which is a lightweight that is used in certain use-cases.The convolution neural network which is used Android malware detection which uses two method ,a novel method which helps to detect the location from the CNN in Android mobile detection,a second method which uses state-of-the-art by the method of LIME.This is a method which uses the user data to find out the malicious software to benefit the full of CNNs and LIME.

Due [20] to the rise of the various devices in the current generation the devices like IoT and gives rise to malware activities which control the automated devices and the open source platform.The hackers can access the personal data of the users and the network worldwide.The Deep squeezed -booosted and Ensemble learning which put froward the STm for the multi-path for both homogeneous and heterogeneous malicious patterns.Which shows the accuracy performance of 98.50% accuracy and helps in timely detection of the malware activities.

III. PROPOSED MODEL



The acquisition of pertinent data is the fundamental component of our powerful malware detection technology. Getting a pre-processed dataset with features taken from Android applications (APK files) will be our first step.The data set of the application that contain a APK file is necessary . Our model will be trained on this type of dataset, which will provide it with the information it needs to differentiate between safe and dangerous apps and also helps in determining why we have classified it into a malware and benign . Using a pre-processed dataset simplifies the data for our model and lets it concentrate on its primary function of learning by removing the need for laborious processes like parsing raw APK files.

After obtaining the data, we will initiate a thorough investigation process to thoroughly examine its complexities. To determine the characteristics available, this entails carefully analyzing the structure of the dataset that is provided . These features could include the kinds of API calls the app makes to communicate with the Android system, the permissions it requests (such as location or camera access), and even code snippets that are contained within the app. It is essential to comprehend the various data kinds (numerical or categorical) and the descriptions that go along with them in order to properly grasp the model's conclusions. We may examine any connections between the traits and display their distribution to obtain even more profound understanding. This primary stage is crucial because it enables us to understand the data that the model will use to efficiently classify apps as benign or malicious.The model need to understand what type of method is used to classify it into malware or not.

A. **Data pre-processing**:

Pre-processing procedures make sure the data is prepared for use in the decision tree model after it has been evaluated. Cleaning the data and readying it for a model that uses machine learning are important steps in the data preprocessing process, which also betters the machine learning model's precision and speed. Removing data points with missing values is done if the missing values are rare and do not correspond to a particular app activity. use statistical techniques such as mean/median imputation to replace missing values in the data with typical values. This method is predicated on the random distribution of missing values. As an alternative, we can use different method to deal with missing values. For example, if a particular permission is absent from an app, add a new binary feature that indicates that it is lacking. When there are several features available, we should take into account feature selection strategies to determine which are most useful for malware detection. Analyse the relationship between the target variable (malware or benign) and the attributes using correlation analysis. It is likely that features that have a strong association with the target variable are more informative.

Recursive Feature Elimination method retrains the model after repeatedly eliminating the least significant feature according to a predefined criteria. It assists in locating a more focused subset of highly predictive traits. By concentrating on the most relevant features, feature selection enhances interpretability, lowers model complexity, and may even result in improved performance. Look at scaling the dataset's features to a common range if they have significantly dissimilar scales (e.g., number of permissions vs. frequency of API calls). Using normalization, features are scaled to a range of 0 to 1.This sets a mean of zero and then a standard deviation of one for the features. By ensuring that every feature has the same effect on the decision-making process, scaling can enhance the decision tree's performance. Scaling might not be required, yet, if the features already have accessible scales built in.

B. **SPLITTING DATA:**

A machine learning algorithm which is required in dividing data into training, testing set for the malware detection .Always set of data is having two categories one is “malware “ and another is Benign “.The data splitting is required to know that our decision tree model is effectively used to determine .Their are two primary methods of splitting of data exist which mainly be known as random split and stratified split. The data is divided arbitrarily based on the selected ratio using a random split. Although simple, it might not ensure that both malicious and safe apps are fairly represented in both sets. This might be an issue for malware detection, since it's important to detect even a tiny percentage of harmful software.Stratified splitting is the recommended method for datasets that are imbalanced. This method guarantees that the proportionate representation of both classes (malicious and benign) exists in the testing and training datasets. This is necessary for a strong malware detection model to function successfully in real-world malware identification settings and to learn from a wide variety of data sets.And this type of splitting requires to maintain a clear splitting of data into malware and safe.

C. **DECISION TREE:**

While there are many decision tree models available, CART (Classification And Regression Trees) and C4.5 are common options for classification. Using an iterative methodology, CART starts at the start node with all of the training data. The attribute that produces "purest" split—that is, the child nodes that result have a higher concentration of a single class—malware or benign—than the parent node—is chosen after all features have been analyzed. Until a stopping requirement is satisfied, this procedure iteratively splits each child node by determining the best feature for additional splitting. Though it has more features, such as managing missing values and "pruning" the tree, C4.5 is comparable to CART. Pruning ensures that the final tree generalizes by removing extraneous branches that may have arisen from overfitting on the training set.

The training data is divided into two parts according to the selected feature and its value. Data points having the selected feature value are represented by one branch, while those without are represented by the other. In essence, a certain feature is being used to categorize the data.
Recursive learning: The first split is not where the process ends. Every child node is where the algorithm continues

recursively. Each child node's data is analyzed, and the feature that best divides the data into subsequent child nodes is chosen once more. Such recursive splitting keeps going until a stopping criterion is met, making sure the tree catches the most important differences in the data between malicious and benign programs.Overfitting occurs when a decision tree grows out of control, making the model function badly on unknown input and memorize the training set too well. Stopping criteria are put in place to stop this. Typical standards consist of:

The method of maximum depth is restricting the tree's maximum depth minimizes overfitting and limits its complexity. It could be difficult for a deeper tree to generalize to unknown facts.The another method of minimum samper per leaf is determining the least quantity of data points necessary at a leaf node guarantees that the model has sufficient information to confidently classify the data at that node.

D. **Model evaluation**:

To evaluate the model's performance in identifying apps as malicious or benign, we need use a variety of metrics. The percentage of apps in the testing set that were accurately classified as either benign or malicious. Although a high accuracy is ideal, if the dataset is uneven (more benign apps than malware), it could not be the most revealing metric. The percentage of apps that the model successfully detected as malware (true positives) out of all apps that were predicted to be malware. A high precision means the model detects malware accurately and doesn't produce a lot of false positives, which are harmless apps that are mistakenly flagged as malicious. The percentage of real malware apps in the testing set that are true positives, or apps that are successfully detected as malicious.

E. **XAI and LIME**:

Significant security concerns are introduced by the wide app ecosystem and the widespread use of Android smartphones. Apps that are malicious yet look safe might steal confidential information, tamper with how a device works, or even transmit other malware. Strong and understandable malware detection technologies are essential 3 counter this danger. This paper describes how to use Explainable AI (XAI) methods, particularly LIME (Local Interpretable Model-Agnostic Explanations), when combined with decision trees. A large number of pre-processed features taken from Android applications (APK files) are available in the "Android Malware Detection" dataset. These functions provide clarity on app behaviour without getting into the complex details of raw APK parsing. We explored a few important feature categories and how they may be used in a malware detection system that relies on decision trees.

Permissions: In order for Android apps to access certain features (such the camera and location), they must have permissions. Malicious intent may be indicated by specific permissions being present or absent. When a game software that looks innocent asks to access location data, for example, it may be cause for concern. We can create branches in the decision tree according to whether an application asks for rights (like location access for a photo-editing app) that are frequently abused by malware.

API Calls: Application Programming Interfaces (APIs) are the ways through which applications interact with the Android OS. An application's usage habits and frequency of API calls might provide information about how it operates. Suspicious API calls, like those for network connections to known malware domains or for gaining access to private device functions, may be made by malicious programs. We can include branches in the decision tree according to the frequency with which the application calls the network or the existence of particular API calls linked to malware.

Code N-grams: These are word or instruction sequences from the app's code. Despite the fact that the raw code is not included in the dataset, n-grams can reveal information about the features of the program. Malicious intent may be indicated by the existence of code n-grams, such as those used for data encryption or deception, that are known to be employed in malware. Depending on which code n-grams are present and linked to known malware families, the decision tree may have branches.

LIME focuses on providing context for each of a model's unique predictions. LIME can be used in the area of the malware detection model to explain every decision tree's classification of a particular testing set app as malicious. LIME first looks for applications that, based on their features, are similar to the app that is being stated in the training data. These comparable apps serve as a point of reference for understanding the categorization of the main app. Based on the individual app instance, LIME builds a more basic model (often a linear model) that roughly represents the decision tree's behaviour for that app and its surrounding apps. Using an analysis of the weights given to the attributes in this location, the prediction is explained.

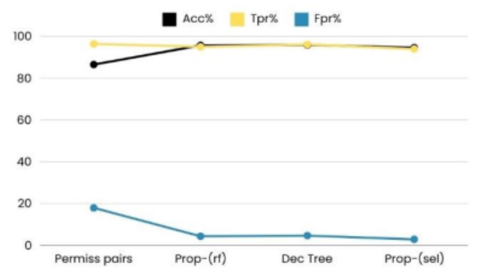
In a situation LIME explains why a program is categorized as malicious. The explanation may indicate that the main elements that shifted the balance towards categorizing the app as malware were its request for a mix of permissions (such as access to location and storage) and the existence of code n-grams linked to data theft methods. LIME's detailed explanation is really helpful. It helps to recognize dangerous programs and comprehend the particular traits that led to their categorization. This is particularly helpful for more research and possible decision tree model improvement. Decision trees' interpretability and LIME's explanatory ability is combined to create a reliable malware detection solution for Android apps that will protect consumers.

IV. RESULT

Using decision trees and LIME for interpretability, we have established a structure for a transparent malware detection system. What has been done and the expected results are laid down as follows.We loaded the CSV file, which included a class label and malware traits from extracted features of android (binary coded as 0s and 1s). We investigated the data types and found that, while one or two columns may be categorical, most of the columns show binary attributes.Depending on the data and analytic demands, we eliminated rows containing missing values or imputed the missing values (e.g., using mean for numerical columns). The data was divided into testing and training sets. The testing set is helpful to evaluate the model's performance once it has been constructed with the training set.We developed a decision tree model, a class of machine learning models characterized by their interpretability.Metrics like accuracy, true positive rate (TPR), and false positive rate (FPR) have been calculated systematically. The metrics, however, are available through the model evaluation procedure. To provide meaning for the decision tree model's individual predictions, we employed LIME (Local Interpretable Model-Agnostic Explanations).LIME enables us to comprehend which features—such as certain malware traits—have the biggest impact on the model's ability to determine if a given data item is malicious or not.

SCHEMES	DREBIN android dataset		
	Acc %	Tpr %	Fpr %
Permiss pairs	86.5	96.4	17.9
Prop-(Rf)	95.7	96.3	4.3
Dec Tree	95.9	96.1	4.6
Prop-(sel)	94.6	94.2	2.8

Using the attributes in the information, the decision tree model will grow to differentiate between samples that are malicious and those that are benign.LIME will explain each prediction, highlighting the key elements the model took into account when classifying the data. This enables us to evaluate the transparency of the model and comprehend the reasoning behind its judgments.To improve the decision tree model's performance, tuning its hyperparameters appropriately is done and followed by testing the model.The predicted labels and actual labels from the testing set can be used to create the confusion matrix once the decision tree model has been trained and tested.The accuracy, TPR, and FPR can be obtained by inserting the number of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) from the confusion matrix into the appropriate formulas.To evaluate the success rate of the model, computation of the evaluation metrics (accuracy, TPR, and FPR) on the testing set is done. To understand the behavior of the model, use LIME to explain predictions on samples that are misclassified as well as those that are correctly classified as malware or benign.Examined the LIME explanations to determine which characteristics are most important for the model's choices and to find possible areas for development.



V.CONCLUSION

A serious security risk brought about by the widespread use of Android smartphones and the extensive app store is the presence of illegal apps which appear as trustworthy ones. These applications could spread further viruses, steal confidential information, or disrupt device operations. We need effective and understandable malware detection solutions to counter this danger. The paper explores developing a transparent defense against mobile threats by utilizing Explainable AI (XAI) techniques in alongside decision trees. Decision trees, in contrast to complex black-box models, provide built-in interpretability. Their arrangement shows an arrangement of features used for classification, similar to an if-then-else flowchart. These attributes in our system cover permissions that the app requests and are taken from the "Android Malware Detection" dataset. API calls it makes, and code n-gram sequences. By analyzing the decision tree, we can understand which features are most indicative of malware.

For instance, a branch in the tree might check if an app requests location access for a seemingly unrelated purpose like a photo editing app. This interpretability fosters trust and transparency in the model's decision-making process, as users can understand the rationale behind malware classifications.It produces code n-gram sequences and API calls. We can identify the characteristics that are most suggestive of malware by examining the decision tree. For example, a branch in the tree might determine whether an application requests location access for a reason that doesn't seem connected. Due to users' capacity to comprehend the reasoning behind malware classifications, the interpretability of the model promotes transparency and trust in its decision-making process. It takes time to develop a trustworthy and understandable malware detection system. This project shows how decision trees and XAI work well together to achieve transparency. We can build a reliable system by adding explainable features, complex algorithms, and feature engineering to LIME. In addition, the inclusion of real-time detection, dynamic updates combining fresh malware samples, and user feedback helps guarantee that the system stays flexible and efficient within a continually evolving mobile threat environment. This effort opens the door for more XAI research for mobile security applications in addition to helping to construct a clear and understandable malware detection system.

For future enhancements, we can add several features. At the moment, feature engineering uses human judgment to find features that may be instructional. This procedure can be somewhat automated through the use of XAI algorithms. XAI could suggest additional features or feature combinations that might increase detection accuracy without compromising interpretability by examining the behavior of the model and feature interactions. Reports on incorrectly classified apps can be submitted by users under the existing system. This can be further enhanced by adding user-supplied labels (benevolent or malicious) for individual programs. Over time, the system could improve its decision-making process by including insights from these labels from users. But measures must be taken to stop bad actors from tampering with the feedback loop. Interpretable reasons for on-device classifications are essential as the system approaches real-time detection. Even in environments with limited resources, techniques such as LIME can be modified for real-time situations and provide consumers with basic justifications for why an app is reported as malicious.

REFERENCES

[1] A. A. Al-Hashmi et al., "Deep-Ensemble and Multifaceted Behavioral Malware Variant Detection Model," in IEEE Access, vol. 10, pp. 42762-42777, 2022, doi: 10.1109/ACCESS.2022.3168794.

[2] T. H. Hai, V. Van Thieu, T. T. Duong, H. H. Nguyen and E. -N. Huh, "A Proposed New Endpoint Detection and Response With Image-Based Malware Detection System," in IEEE Access, vol. 11, pp. 122859-122875, 2023, doi: 10.1109/ACCESS.2023.3329112.

[3] Ö. A. Aslan and R. Samet, "A Comprehensive Review on Malware Detection Approaches," in IEEE Access, vol. 8, pp. 6249-6271, 2020, doi: 10.1109/ACCESS.2019.2963724.

[4] H. Kato, T. Sasaki and I. Sasase, "Android Malware Detection Based on Composition Ratio of Permission Pairs," in IEEE Access, vol. 9, pp. 130006-130019, 2021, doi: 10.1109/ACCESS.2021.3113711.

[5] Ö. Aslan and A. A. Yilmaz, "A New Malware Classification Framework Based on Deep Learning Algorithms," in IEEE Access, vol. 9, pp. 87936-87951, 2021, doi: 10.1109/ACCESS.2021.3089586.

[6] O. Arreche, T. R. Guntur, J. W. Roberts and M. Abdallah, "E-XAI: Evaluating Black-Box Explainable AI Frameworks for Network Intrusion Detection," in IEEE Access, vol. 12, pp. 23954-23988, 2024, doi: 10.1109/ACCESS.2024.3365140.

[7] J. Liu, M. Xu, X. Wang, S. Shen and M. Li, "A Markov Detection Tree-Based Centralized Scheme to Automatically Identify Malicious Webpages on Cloud Platforms," in IEEE Access, vol. 6, pp. 74025-74038, 2018, doi: 10.1109/ACCESS.2018.2882742.

[8] J. Ferdous, R. Islam, A. Mahboubi and M. Z. Islam, "A Review of State-of-the-Art Malware Attack Trends and Defense Mechanisms," in IEEE Access, vol. 11, pp. 121118-121141, 2023, doi: 10.1109/ACCESS.2023.3328351.

[9] Z. A. E. Houda, B. Brik and L. Khoukhi, ""“Why Should I Trust Your IDS?”: An Explainable Deep Learning Framework for Intrusion Detection Systems in Internet of Things Networks," in IEEE Open Journal of the Communications Society, vol. 3, pp. 1164-1176, 2022, doi: 10.1109/OJCOMS.2022.318875,

[10] S. Huh, S. Cho, J. Choi, S. Shin and H. Lee, "A Comprehensive Analysis of Today’s Malware and Its Distribution Network: Common Adversary Strategies and Implications," in IEEE Access, vol. 10, pp. 49566-49584, 2022, doi: 10.1109/ACCESS.2022.3171226.

[12] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, “MeMalDet: A Memory analysis-based Malware Detection Framework using deep autoencoders and stacked ensemble under temporal evaluations,” *Computers & security*, Apr. 01, 2024. https://www.sciencedirect.com/science/article/pii/S0167404824001652

[13] G. Renjith, S. Laudanna, S. Aji, C. A. Visaggio, and P. Vinod, “GANG-MAM: GAN based enGine for Modifying Android Malware,” *SoftwareX*, Jun. 01, 2022. https://www.sciencedirect.com/science/article/pii/S2352711022000036

[14] H. Bostani and V. Moonsamy, “EvadeDroid: A Practical Evasion Attack on Machine Learning for Black-box Android Malware Detection,” *Computers & security*, Apr. 01, 2024. https://www.sciencedirect.com/science/article/pii/S0167404823005862

[15] D. R. J. Verboogen *et al*., “Heterophilic antibodies leading to falsely positive D-dimer concentration in an adolescent,” *Research and practice in thrombosis and haemostasis*, Jan. 01, 2023. https://www.sciencedirect.com/science/article/pii/S2475037922002267

[16] H. Bolhasani, M. Mohseni, and A. M. Rahmani, “Deep learning applications for IoT in health care: A systematic review,” *Informatics in medicine unlocked*, Jan. 01, 2021. https://www.sciencedirect.com/science/article/pii/S235291482100040X

[17] F. Nawshin, R. S. Gad, D. Ünal, A. Al-Ali, and P. N. Suganthan, “Malware detection for mobile computing using secure and privacy-preserving machine learning approaches: A comprehensive survey,” *Computers & electrical engineering*, Jul. 01, 2024. https://www.sciencedirect.com/science/article/pii/S0045790624001617

[18] A. Wajahat *et al*., “Securing Android IoT devices with GuardDroid transparent and lightweight malware detection,” *Ain Shams Engineering Journal/Ain Shams Engineering Journal*, Feb. 01, 2024. https://www.sciencedirect.com/science/article/pii/S2090447924000170

[19] M. Kinkead, S. Millar, N. McLaughlin, and P. O’Kane, “Towards Explainable CNNs for Android Malware Detection,” *Procedia computer science*, Jan. 01, 2021. https://www.sciencedirect.com/science/article/pii/S1877050921007663

[20] S. H. Khan *et al*., “A new deep boosted CNN and ensemble learning based IoT malware detection,” *Computers & security*, Oct. 01, 2023. https://www.sciencedirect.com/science/article/pii/S016740482300295X

varssha

ORIGINALITY REPORT

4%	3%	3%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	doaj.org Internet Source	1%
2	nrl.northumbria.ac.uk Internet Source	<1%
3	thesai.org Internet Source	<1%
4	Omer Aslan, Refik Samet. "A Comprehensive Review on Malware Detection Approaches", IEEE Access, 2020 Publication	<1%
5	ijcsmc.com Internet Source	<1%
6	kristiania.brage.unit.no Internet Source	<1%
7	Jannatul Ferdous, Rafiqul Islam, Arash Mahboubi, Md. Zahidul Islam. "A Review of State-of-the-Art Malware Attack Trends and Defense Mechanisms", IEEE Access, 2023 Publication	<1%

8	Submitted to University of Nottingham Student Paper	<1%
9	www.frontiersin.org Internet Source	<1%
10	ijeci.lgu.edu.pk Internet Source	<1%
11	www.researchsquare.com Internet Source	<1%
12	Ananya Redhu, Prince Choudhary, Kathiravan Srinivasan, Tapan Kumar Das. "Deep learning-powered malware detection in cyberspace: a contemporary review", Frontiers in Physics, 2024 Publication	<1%
13	Atul Kumar, Ishu Sharma. "CNN-based Approach for IoT Intrusion Attack Detection", 2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), 2023 Publication	<1%
14	www.ijrdet.com Internet Source	<1%
15	P Kanaga Priya, T Sivakumar. "Artificial Intelligence Techniques and Methodology Available for Lung Cancer Detection", 2023	<1%

International Conference on Sustainable Computing and Smart Systems (ICSCSS), 2023
Publication

16	dokumen.pub Internet Source	<1 %
17	www.researchgate.net Internet Source	<1 %
18	Jianhua Liu, Mengda Xu, Xin Wang, Shigen Shen, Minglu Li. "A Markov Detection Tree-Based Centralized Scheme to Automatically Identify Malicious Webpages on Cloud Platforms", IEEE Access, 2018 Publication	<1 %
19	Osvaldo Arreche, Tanish R. Guntur, Jack W. Roberts, Mustafa Abdallah. "E-XAI: Evaluating Black-Box Explainable AI Frameworks for Network Intrusion Detection", IEEE Access, 2024 Publication	<1 %
20	Pascal Maniriho, Abdun Naser Mahmood, Mohammad Jabed Morshed Chowdhury. "A systematic literature review on windows malware detection: Techniques, research issues, and future directions", Journal of Systems and Software, 2023 Publication	<1 %