iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 11 **Issue:** VI **Month of publication:** June 2023

DOI: https://doi.org/10.22214/ijraset.2023.54181

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VI Jun 2023- Available at www.ijraset.com

Malware Detection and Prediction System Using Advanced Machine Learning Algorithms

Asst. Prof. T. Rajesh[1], K. Divya[2], Shaista Firdouse[3], Sharia Areeb[4], Navya Thakur[5]
[1, 2, 3, 4, 5]*G Narayanamma Institute of Technology and Science (for Women) Shaikpet, Hyderabad, Telangana, India*

Abstract: ***Given how many websites are now disseminating malware, malicious software is one of the biggest hazards in today's digital environment. Computer systems connected to the Internet increasingly require malware analysis and protection techniques. Without the user's knowledge, malicious software takes advantage of system flaws to steal important data and covertly send it to distant servers under the control of attackers.***
Static or dynamic analysis methods are used to analyse malware. These methods categorise and forecast distinctive patterns to effectively detect malware. Software assaults have the potential to corrupt or harm devices, compromise entire systems, steal information, change data, and deny service.
The detection and prediction system is implemented as a user-friendly website with a variety of admin and user-facing modules. The file that requires detection and prediction can be uploaded by the admin user. If there are any API calls used in the file with malware purpose, the system will examine them. This API can provide security dangers to the computer because it was designed with malware in mind; as a result, it must be found and immediately removed to prevent security difficulties.
The system that was built can determine the sort of malware that is encoded in the file and can assist users in determining the threat that such malware may pose to their computer.

I. INTRODUCTION

With an annual increase in the total number of known mobile malware cases, the large-scale Internet is no longer the primary target of malware attacks (i.e., viruses, spam bots, worms, and other harmful software). The cause is online data exchange, which unintentionally encourages malware. Malware that once resided on the wired Internet can now spread through the use of devices and networks. The complexity and exponential expansion of malware pose a serious threat to computer and network security. The reliance on computer systems has grown significantly over the last ten years. Practically all tasks, from those involved in daily living to those involved in business, have been mechanized.

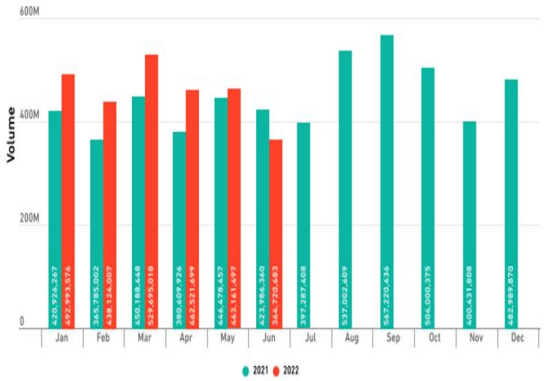


Figure 1 Global Malware Volume

The first half of 2022 saw 2.8 billion malware strikes worldwide, an 11% rise year to date over 2021, according to threat experts at SonicWall Capture Labs. This equals an average of 8,240 attempts to install malware per customer.
In 2020, there were 304 million reported ransomware attacks worldwide, according to Matthew and Woodward. This translates into roughly 37,700 ransomware attacks per hour, or about 578 per minute. The goals of two research communities are being pursued concurrently.

The first is creating harmful software, and the second is creating defense against malware attacks on the systems. Static and dynamic analysis are the two fundamental approaches to malware analysis.

Malware detection system is created based on features using malware analysis techniques. Both signature-based and behavior-based techniques are used to find malware. The dynamic analysis environment is set up for this technique's development, and malware samples are subjected to classification algorithms testing. The malware detection system then used a variety of behavioral artifacts, including PSI, API calls, registry changes, file operations, etc. The dataset of malware API calls is categorized and evaluated using K-Nearest Neighbors (KNN), Random Forest, AdaBoost, Naive Bayes, and Decision Tree algorithms.

Two major approaches in dynamic analysis are control flow analysis and API call analysis. Both approaches detect malware based on analysis of similarity between the behavior of the new and the known ones. However, malware authors try to circumvent those techniques through inserting meaningless codes or shuffling the sequence of program. Several of the API call analysis tools now in use are unable to find malware that uses such a circumvention.

Some methods concentrate on removing APIs that are frequently used in malware across all classes. They track the usage of APIs and determine how frequently and how many times an API function has been called overall. Despite the fact that they immediately highlight the traits of malware in the same class, they are easily circumvented by malware developers by injecting and running bogus and repeated API calls. Others take each class's API call sequence and build static signatures based on it. Because they keep track of the calls' order and program flow, they are superior from a semantic perspective.However,they are unable to recognize malware in polymorphic or unidentified forms by merely extracting call sequences that are frequently seen for each class of malware and creating signatures from those sequences. Malware developers' evasion techniques, including injecting duplicated API calls, can also be used to get around it. This necessitates the development of fresh methods for API call sequence analysis.

New malware attacks increases day by day. The different types of malware [12] with their features are listed below.

A. Ransomware

Ransomware refers to a type of software that encrypts a target's data, blocking access until a ransom is paid. In such instances, the victim organization becomes partially or entirely incapacitated until the payment is made. However, there is no guarantee that the decryption key will be provided upon payment, or even if provided, that it will effectively decrypt the data.

B. Fileless

Fileless malware, on the other hand, operates differently. It doesn't require initial installation, instead making modifications to native operating system files like PowerShell or WMI. As these edited files are deemed legitimate by the operating system, antivirus software fails to detect fileless attacks. Moreover, due to their stealthy nature, fileless attacks have a success rate up to ten times higher than traditional malware attacks.

C. Spyware

Spyware is a form of malware that clandestinely gathers information about users' activities without their knowledge or consent. This can encompass sensitive data like passwords, PINs, payment information, and unstructured messages.

D. Adware

Adware although similar to spyware, serves a different purpose. It tracks a user's browsing behavior to determine which ads are most suitable to display. Unlike spyware, adware does not install software on the user's computer or capture keystrokes.

E. Trojan

A Trojan disguises itself as desirable code or software to deceive users. Once unsuspecting users download the Trojan, it gains control over their systems for malicious intent. Trojans often hide within games, apps, software patches, or attachments found in phishing emails.

F. Worms

Worms, on the other hand, exploit vulnerabilities in operating systems to infiltrate networks. They can gain access through backdoors in software, unintentional software vulnerabilities, or even through the use of flash drives. Once established, worms serve as tools for malicious actors to launch Distributed Denial of Service (DDoS) attacks, steal sensitive data, or carry out ransomware attacks.

G. Virus

A virus is a piece of code that inserts itself into an application and activates when the application is run. After infiltrating a network, viruses can be utilized to pilfer sensitive data, initiate DDoS attacks, or execute ransomware attacks.

H. Rootkits

Rootkits are software that bestows remote control of a victim's computer upon malicious actors, granting them full administrative privileges. They can be injected into applications, kernels, hypervisors, or firmware. Rootkits spread through phishing attempts, malicious attachments, downloads, or compromised shared drives. Furthermore, rootkits can be employed to conceal other forms of malware, such as keyloggers.

I. Bots/Botnets

Bots or botnets are software applications designed to perform automated tasks upon receiving commands. While bots have legitimate uses like search engine indexing, they can also be employed for malicious purposes. In such cases, they transform into self-propagating malware that establishes connections back to a central server.

II. LITERATURE SURVEY

Various approaches have been proposed for malware detection by N.Idika,[1]. Detection techniques proposed earlier were based on static analysis. Static analysis examines the binary code, analyzes all possible execution paths, and identifies malicious code without execution as proposed by P.Vinod,S.Cesare[2-3]. However, analyzing binary code turns out to be difficult nowadays. As obfuscation techniques become more sophisticated, static analysis can be bypassed by various obfuscation techniques, such as polymorphism, encryption, or packing . In addition, as static analysis relies on a prebuilt signature database, it cannot easily detect new unknown malware until the signature is updated as stated by P.Okane[4]. Besides, some execution paths can be only explored after execution . To overcome these limitations of static analysis and complement it, dynamic analysis has been proposed by A. Moser[5] and is widely used to achieve more effective malware detection.

A behavioral model based on conditional graph structure presented by M.Conti [6] is an overview of the findings on trigger-based malware behavior elicitation, classification, modeling, and behavioral signature generation.

PbMMD [7] proposed a model based on system call sequences to detect Multi-process malware with an attempt to inspect the whole processes running on the system and discover collaborative processes by finding processes running along a common execution policy. A. Bushby [8] proposed the use of deception technology as an effective approach in modern cybersecurity for active and intelligent post-breach defense. This emerging phenomenon emphasizes the need for an active defense strategy to lure, detect, and defend against malware and intruders that are moving laterally within the network.

I.K. Cho, T.G. Kim, Y.J. Shim, M. Ryu, and E.G. Im [9] proposed a novel approach for classifying unknown or new malware into known malware families. They developed a malware family classification framework that utilizes a sequence alignment method. This framework is capable of identifying common parts within the invoked API sequences of malware, which can then be used to uncover similar behavioral patterns among malware variants**.**

Francesco Mercaldo [10] conducted research on detecting malicious files using diverse datasets that included both real and synthetic malware samples. The study found that the Random Forest classifier outperformed other machine learning algorithms, achieving superior results. Additionally, a fine-tuned deep neural network achieved high F1-scores of 99.1% and 98.48%.

Omer Aslan [11] conducted a comprehensive review of state-of-the-art techniques and algorithms employed in malware detection. The review discusses the advantages and disadvantages of each malware detection approach, covering areas such as data mining, machine learning, and the utilization of emerging technologies like deep learning, cloud computing, mobile devices, and IoT-based detection schemes.

A. Challenges In The Existing Malware Detection System

The challenges in the existing system is as follows:

1) Static analysis is the primary method used by websites to identify harmful software. Static analysis has the drawback that it takes a long time to perform manually. Automatic tools may provide the impression that everything has been taken care of while also giving false positives and false negatives. Runtime environment-introduced vulnerabilities are not discovered via static analysis. Because it can be easily avoided via obfuscation techniques, static analysis isn't always correct (polymorphism and metamorphism).

2) In order to detect anomalies and identify malware threats, machine learning algorithms need to be taught to evaluate data patterns and form conclusions. With a high number of samples, the algorithm won't be able to discriminate between legitimate and malicious files if the database is corrupted or not labeled appropriately, hence the solution will produce inaccurate results. Time required for malware detection and classification was comparatively more. Since there are different types of malware with unique features. A huge data set is required and to training time is also more. Prediction of type of malware is not done. Previous models only detected if the software is malicious or benign, there are no approaches to finding the type of malware. Due to security and privacy reasons the malware dataset is not available easily and the size of dataset is also less.The existing system has less accuracy and failed to decrease the false positive rate.

III. PROPOSED MALWARE DETECTION SYSTEM

Malware analysis techniques might be static or dynamic. These techniques classify malware and foresee recognizable patterns to efficiently find it. Software attacks have the power to damage or corrupt hardware, compromise entire networks, steal data, alter data, and disable services. The System will develop an ensemble machine learning technique-based system for identifying malicious software using its dynamic analysis efforts. The psychological proof Application programming interface calls are retrieved for malware detection systems to build a model with high accuracy. Compared to models trained purely on API calls as the dynamic feature, application programming interface (API) calls offer a minor speed improvement. Furthermore, in order to determine the type of malware, predictions are produced using sophisticated learning algorithms.

The detection and prediction system is implemented as a user-friendly website with a variety of admin and user-facing modules. The file that requires detection and prediction can be uploaded by the admin user. If there are any API calls used in the file with malware purpose, the system will examine them. This API can provide security dangers to the computer because it was designed with malware in mind; as a result, it must be found and immediately removed to prevent security difficulties.

The system that was built can determine the sort of malware that is encoded in the file and can assist users in determining the threat that such malware may pose to their computer. The system can then be made completely safe by deleting this file. The suggested malware detection and prediction system, which employs cutting-edge learning algorithms for high accuracy, may effectively accomplish this.

A. *Objectives*

The system is built to develop a behavior based malicious software detection and prediction system using advanced machine learning algorithms.

1) To employ various simple algorithms like K-Nearest Neighbor algorithm, Naïve-Bayes algorithm, and decision tree algorithm. Advanced ensemble machine learning algorithms like random forest and AdaBoost are also implored in this study.
2) To analyze the performance of the five algorithms and select the one that give best performance.
3) edicts the type of malware found in a given file.
4) To make the system available as a user-friendly website that can be accessed using any search engine.
5) A system should be created to assist users, testers, and developers in analyzing their files and spotting any security threats.

B. *Methodology*

The datasets are the focus of the architecture's initial section. To extract the necessary characteristics from the datasets that will aid in the training and testing of the algorithm, the data set must first be collected. The data set is fed into five distinct classifiers after feature extraction has been successfully completed: K-Nearest Neighbors classifier, Decision Tree classifier, Nave-Bayes classifier, Adaboost classifier, and Random Forest classifier.

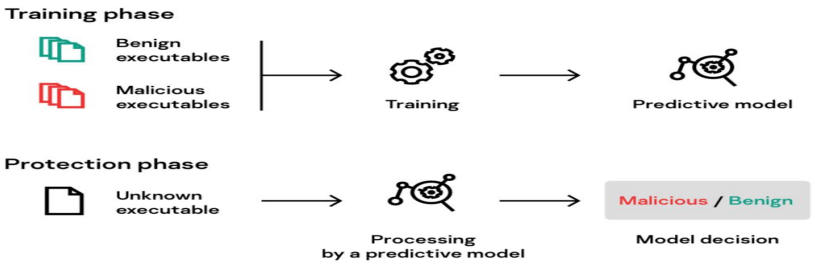


Figure 2 Machine Learning algorithm lifecycle

Each of the given classifiers must follow a life cycle which includes testing and training for each algorithm involved. The benign executables also known as goodware files,
and the malicious executables, also known as malicious files are given in training to the predictive model.

C. Hardware and Software Requirements

The system requirements to produce a malware detection and prediction system are as follows.

System Requirements	
Hardware Requirements	
RAM	4 GB Minimum
Processor	i3 Minimum
Hard disk	500 GB
Software Requirements	
Technology	Python 3.6
Operating System	Windows Family
IDE	VS Code
Technology	Python, Django
Database Server	Postgresql
Front Design Technology	HTML, CSS, JS

Figure 3 System Requirements for the system

IV. PROPOSEDMALWARE DETECTION SYSTEM

The design of the system is done by incorporating the various machine learning models.

A. Architecture Of Proposed System

The architecture of the proposed malware prediction and detection system is as follows:

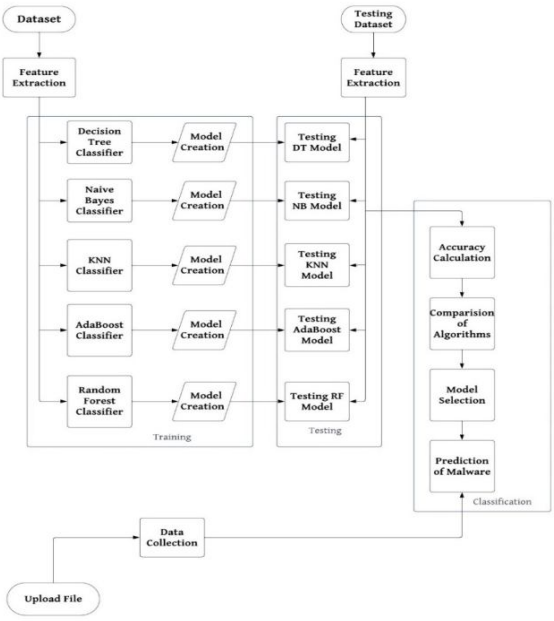


Figure 4 Architecture of the system

The first part of the architecture is concerned with the datasets. The data set is first collected, and feature extraction is done to extract the required features from the datasets which will help in training and testing of the algorithm. After feature extraction has been successfully completed, the data set is passed into five different classifiers namely K-Nearest Neighbors classifier, Decision Tree classifier, Naïve-Bayes classifier, AdaBoost classifier and Random Forest classifier.

These are all machine learning classifiers, and they have been shown to be quite effective at testing and training different kinds of data in datasets. For each of these classifiers, which are stored as.sav files, a model must be created as the next step. Training begins after the model has been created,and it is carried out independently for each classifier. Testing is the following step. The testset, a new data set from which features are extracted and individually fed into each model for testing, is needed for the system's testing component. Following thorough testing of each model, the accuracy is determined using a variety of metrics, and the accuracy scores of each classifier are compared to determine which model is the most effective at making predict malware in file.

On the user's end of the system, the file uploaded is collected from which API calls are determined to see if they match into any of the training data to check if they're goodware or malware.

B. Description of Algorithms used

The proposed system employs a variety of machine learning algorithms, including ensemble machine learning algorithms, to ensure an effective outcome. Below, each algorithm's description is provided.

C. Classifier for K-Nearest Neighbors

A supervised machine learning algorithm is K-Nearest Neighbor (KNN). A new data point is classified based on similarity and all previously stored data is stored.As opposed to an eager algorithm, KNN is a lazy algorithm.The KNN algorithm simply stores the dataset during the training phase, and when it receives new data, it categorizes it into a category that is very similar to the new data. The training data points are not used to make any generalizations.

Thus, there is either no explicit training phase present or very little of it. Additionally,the training period is fairly brief because of this. KNN retains all of the training data because there is no generalization.More precisely ,the testing phase requires all (or the majority) of the training data. The KNN algorithm is based on feature similarity: We classify a given data point based on how closely out-of-sample features resemble our training set.

D. Decision Tree Classifier

Classifiers using Decision Trees (DT) are an example of supervised machines.Learning is the process of creating a model, feeding it training data that is matched with the desired outputs, and then letting the model pick up on these patterns. In order to predict a value from a decision tree, one would start at the root node and ask questions that are specific to each node. Afterward, based on the node answer, the appropriate branch is selected, and we keep going until we reach a leaf node, arriving at a decision.

E. The Naive-Bayes classifier

The classification method known as Naive Bayes (NB) is based on the Bayes' Theorem and makes the assumption that all of the features that predict the target value are unrelated to one another. It determines each class's probability before selecting the one with the highest likelihood. It has been successfully applied to a variety of tasks, but natural language processing (NLP) issues are where it excels.

F. Classifier with AdaBoost.

The most effective application of AdaBoost is to improve decision tree performance in binary classification problems. Any machine learning algorithm's performance can be improved with AdaBoost. It works best with reluctant students. These models perform well enough on classification problems to achieve accuracy slightly above chance. As it is used for classification rather than regression, it has been referred to more as discrete AdaBoost.

G. Random Forest Classifier

The widely-used machine learning algorithm known as "random forest" was created by Leo Breiman and Adele Cutler. It combines the output of various decision trees to produce a single outcome. Given that it can solve classification and regression issues, its popularity has beenfueled by its simplicity andflexibility.

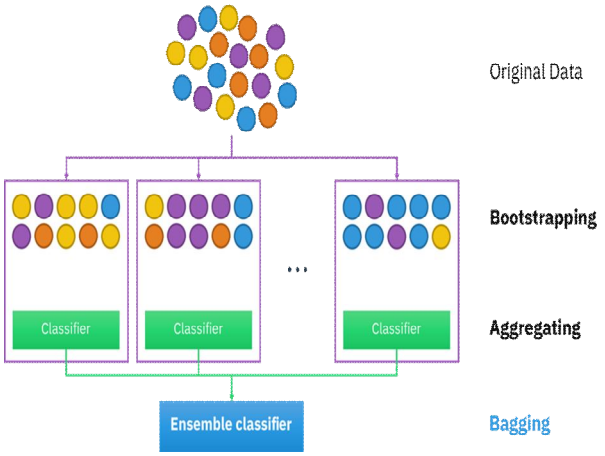


Figure 5 Working of Ensemble classifier

V. IMPLEMENTATION OF MODULES

The data set is fed into the K-Nearest Neighbors classifier, Decision Tree classifier, Nave-Bayes classifier, Adaboost classifier, and Random Forest classifier after feature extraction has been successfully completed.

Every provided classifier needs to follow a life cycle that includes training and testing for every associated algorithm. When the prediction model is being trained, it is given both the malicious executables, also known as malicious files, and the benign executables, also known as goodware files.

A. Dataset Description

Building a malware prediction and detection system necessitates a sizable collection of API calls, preferably in the form of a database to aid in the training and testing of the learning algorithms. The dataset gathered from the HCRL (Hacking and Countermeasure Research Lab) is one example of one such dataset. There are 1,200 Goodware calls and approximately 14,000 API sequence calls from malware..

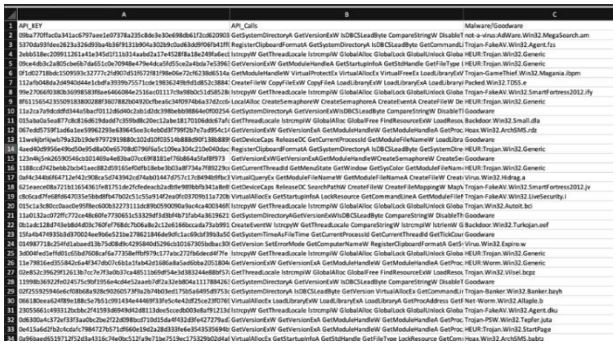


Figure 6 HCRL Dataset

The API_KEY,API_Calls,and the malware connected to each call are the three properties of each instance.

B. Description of Technologies/Tools used

1) Python

Python is a high-level, interpreted programming language that is widely used in various domains such as web development, data science, machine learning, and automation. It is known for its simplicity, readability, and ease of use, making it a popular choice among developers of all levels of experience. Python has a vast library of built-in modules and packages and a wide range of libraries such as Pandas, Numpy, and Scikit-learn, which are ideal for handling data analysis and visualization tasks, making it highly versatile and capable of handling a wide range of programming tasks. It also supports multiple programming paradigms, including object-oriented, procedural, and functional programming, giving developers the flexibility to choose the approach that best suits their needs.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 11 Issue VI Jun 2023- Available at www.ijraset.com

2) VS Code

Developers don't need to use a separate terminal program because .VS Code includes an integrated terminal that enables them to execute commands and scripts right inside the editor. A flexible and effective code editor,VSCode can be tailored to meet the requirements of developers working with a variety of platforms and programming languages.

3) GoogleColab

Colaboratory, also known as "Colab," is a product of Google Research. Colab, which makes it possible for anyone to write and run arbitrary Python code through a web browser, is particularly well-suited for machine learning, data analysis, and education. Technically speaking, Colab is a hosted Jupyter notebook service that can be accessed instantly and is free of charge, including access to GPUs.

4) Django

To manage data for a machine learning project,use.Django's robust Object-Relational Mapping (ORM) system. When working with large datasets, this can be especially helpful. Django can be used to incorporate machine learning models into web applications, enabling users to interact with the model via a web interface. It is the perfect option for machine learning projects due to its strengths in data management, model integration, user management, task scheduling, and API development.

5) HTML

A key language in web development for creating web applications is HTML, or Hypertext Markup Language. The structure and content of web pages are defined using it. HTML can be used in conjunction with other web development tools like JavaScript for interactivity and CSS for styling. Building robust and interactive web applications is a cinch thanks to its advantages in defining page structure, content markup, form creation, links and navigation, and integration with other technologies.

6) CSS

Cascading Style Sheets, also known as CSS, is a language used in web design to style and layout web pages. As it enables developers to design aesthetically pleasing and user-friendly interfaces, it is a crucial tool for developing web applications that integrate machine learning. It is the best option for enhancing the application's visual aspects and enhancing the user experience because of its strengths in responsive design, styling of data visualizations, user interface design, branding, animation.

7) JS

JavaScript can be used to dynamically update web page content, such as updating predictions made by machine learning models or instantly refreshing data visualizations. User interactions with the web application, such as clicking buttons, submitting forms, or scrolling, can be made possible with JavaScript. The application could become more interactive as a result, enhancing the user experience overall.

C. Module Description

1) Collecting Data

Machines begin by learning from the data that is provided to them. Collecting trustworthy data is crucial so that your machine learning model can identify the appropriate patterns. The accuracy of the model depends on the quality of the data you feed it. Inaccurate or out-of-date data will result in inaccurate results or predictions that are irrelevant. A large number of API calls must be made, preferably in the form of a database to help with training and testing the learning algorithms, in order to create a malware prediction and detection system. One example is the data gathered from the HCRL (Hacking and Countermeasure Research Lab). A total of 14,000 API sequence calls from malware and 1,200 calls from goodware are present.

2) Getting the Data Ready

The dataset must be vectorized in order to make it suitable for the training phase. Tf-idf vectorizer is utilized in this situation. Term Frequency Inverse Document Frequency is referred to as TF-IDF. This is a well-known algorithm that turns text into a meaningful representation of numbers that can be used to fit machine prediction algorithms. The formula listed below is used for this.The vectorizer formula shown in Figure 12 TFIDF.

For a term i in document j :

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tf_3 = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Figure 7 TFIDF vectorizer formula

3) Picking a Model.

The output you get after applying a machine learning algorithm to the gathered data is determined by a machine learning model.Selecting a model that is applicable to the current task is crucial. Many models have been developed over the years by scientists and engineers that are suitable for various tasks like speech recognition, image recognition, prediction, etc. In addition, it's critical to decide whether the model is best suited for categorical or numerical data.

The following models for this system have been researched.

•The K-Nearest Neighbors classifier.
• Decision Tree classifier.
•The Naive-Bayes classifier.
•Classifier AdaBoost.
•Classifier from the Random Forest.

4) Training the Models

The training phase of machine learning is the most crucial. The machine learning model receives the prepared data during training in order to look for patterns and make predictions. The model ends up learning from the data in order to complete the given task. The model improves over time at predicting thanks to training.

5) Evaluating the Models

Accuracy scores are used to rate the models. After training, each model is put to the test and evaluated based on how well it performs.

Comparing the models

The accuracy scores are compared of all five classifiers and the one which is producing the most efficient results is chose to be in the final system. The classifier which produces a higher accuracy is chosen.

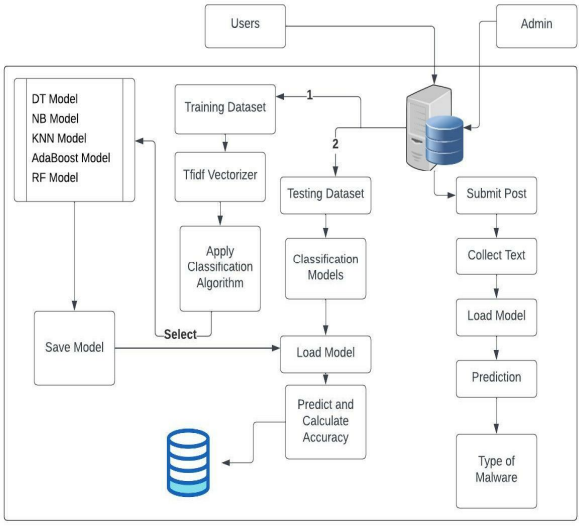


Figure 8 Modules of the system

The above figure represents the modules of the system.

VI. RESULTS AND DISCUSSIONS

Given that malware was created to harm computer systems and spread via Internet connections, it has evolved into a serious threat. Infiltrating a computer system without the owner's knowledge or consent is the main goal of malware, which is software. Even if malware is built in a benign manner and is not malicious in nature, its presence causes problems because it can be hostile, destructive, and intrusive. By constructing a model with the help of cutting-edge machine learning techniques, the malware detection and prediction system aims to accurately identify different types of malware and detect malicious software. We need to test the system using various machine learning and cutting-edge machine learning algorithms in order to create one that is highly accurate. The final system would use the one that had the highest accuracy.Below, all of the algorithms' testing accuracy is discussed below.

Algorithm	Accuracy
Random Forest Classifier	96.9%
Adaboost Classifier	90.45%
K Nearest Neighbor Classifier	49.95%
Naïve Bayes Classifier	47.55%
Decision Tree Classifier	13.9%

Figure 9 Accuracies of the classifiers

The Malware API dataset is used to train the supervised machine learning algorithm known as the Naive Bayes algorithm. The phrase "Naive Bayes is Successfully trained" notifies the algorithm when it has completed training. It's helpful to use NB Classifier to quickly predict outcomes. The testing dataset is then used to test the Naive Bayes code, which yields a 47.55 percent accuracy.
The other algorithms' accuracy is checked using the same method for each one. With a 49.95 percent accuracy, KNN classifier appeared to have slightly higher accuracy than Nave- Bayes. The decision tree classifier, a simple classifier but one that is widely used for supervised learning, is the next algorithm that is tested. The decision tree classifier, on the other hand, performs poorly with only a 13.9 percent accuracy for the dataset used.
The ensemble machine learning algorithms are then evaluated, both of which had accuracy levels greater than 90%. Random Forest performed better than Adaboost with an accuracy of 96% compared to 90.45% for the Adaboost classifier. With its highest accuracy, Random Forest was undoubtedly the algorithm that was most suitable for this particular dataset.

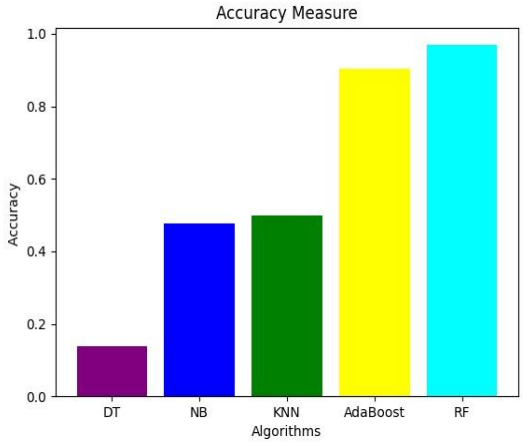


Figure 15 Comparison of accuracies of Algorithms

Plotting the accuracy, precision, recall, and F1 scores for all five algorithms revealed and improvement when going from ensemble machine learning models to basic machine learning models. The Random Forest model scored the highest on each of the five performance measures.

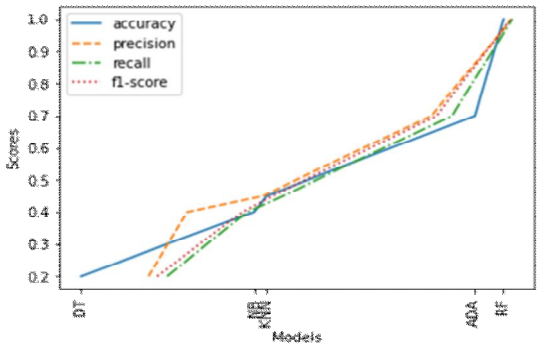


Figure 16 Different Performance measures of the models

The random Forest classifier was finalized as the one to be used for the final system. This model was embedded into the system and the web pages were developed as follows.

Figure 10 Graphical user interface when malware isn't detected

The graphical user interface displays an icon for detection which allows users to upload a file for malware detection. In case the system finds no malware after scanning the file, it displays as “Not a Malware file”.

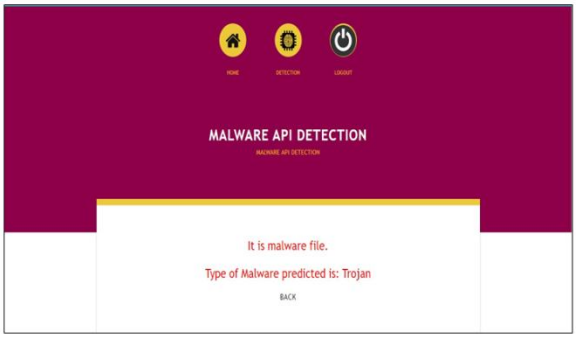


Figure 11 Graphical interface when the malware trojan has been detected

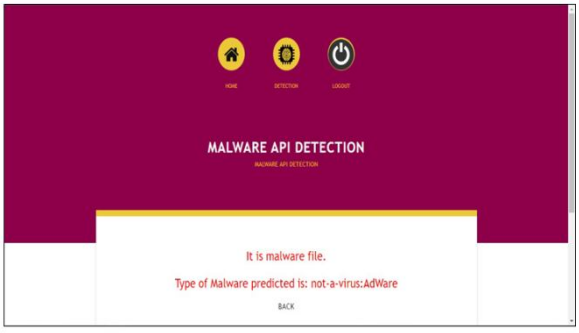


Figure 12 Graphical interface when the malware not-a-virus has been detected

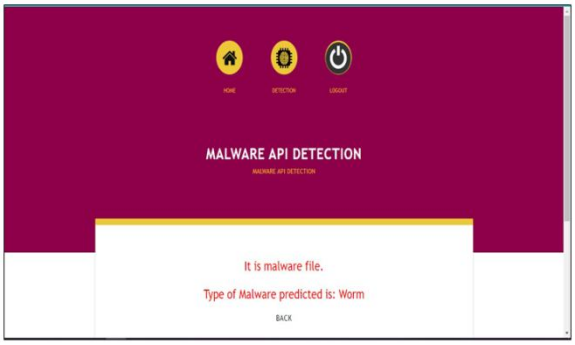


Figure 13 Graphical interface when the malware Worm has been detected

In case a malware has been detected, the system displays that it is a malware file along with the type of malware that has been detected. The above figures show the graphical user interface incase the malwares trojan, not-a-virus and Worm has been detected.

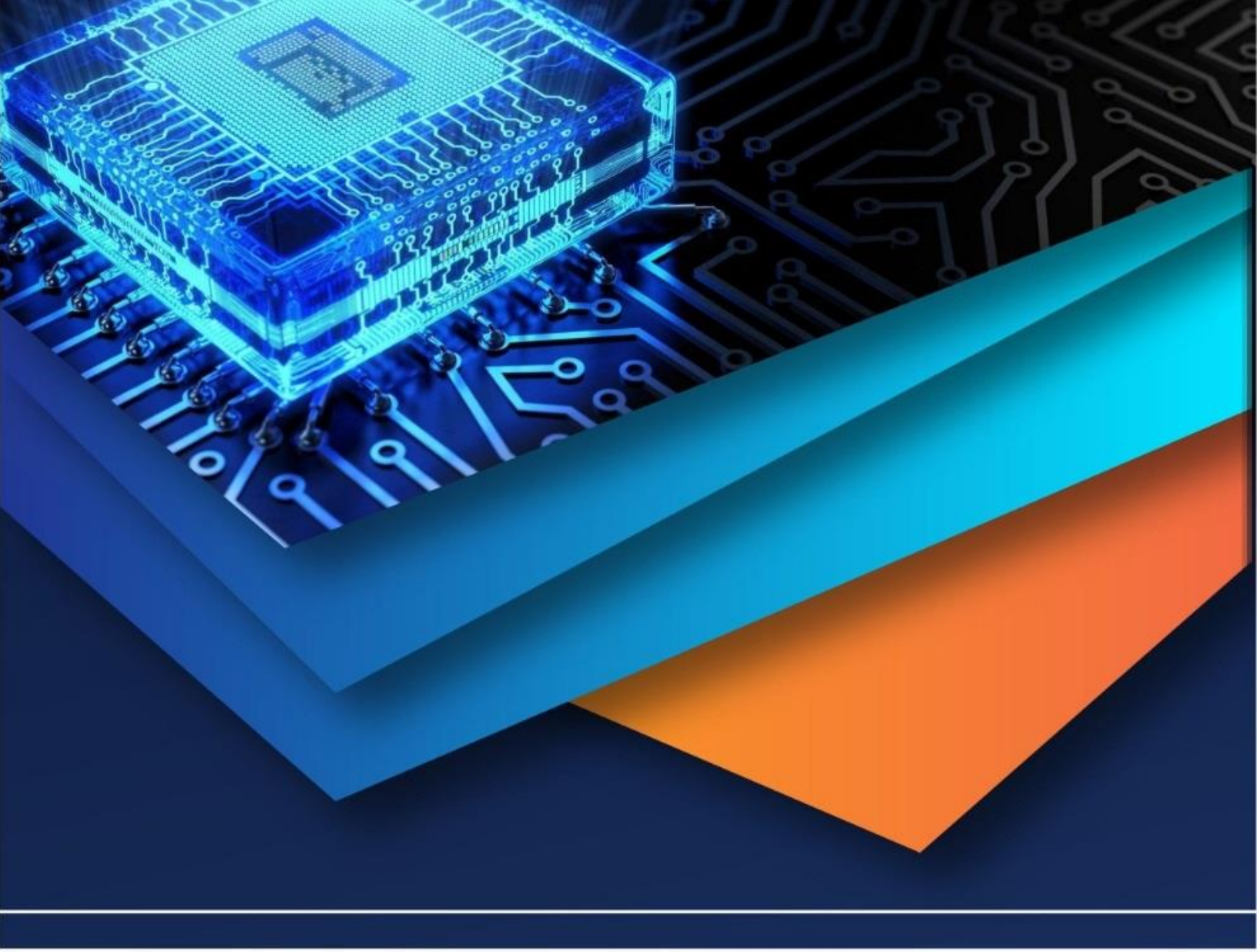
VII. CONCLUSION AND UPCOMING AMENDMENTS

A malware detection and prediction system is created using cutting-edge machine learning algorithms and integrated with a website. Whether a file is malicious or benign can be determined with great accuracy by the system. The website also offers the capability of malware type prediction. the malware dataset obtained from HCRL, which includes numerous examples of both benign and malicious samples. The accuracy of the dataset's classification using the Decision Tree Classifier and the Naive Bayes Classifier is noted. The project's goal is to develop a system using sophisticated machine learning algorithms and evaluate the algorithms' accuracy. The system warns the user of potentially harmful software that could damage the computer. Security and authentication tools are built into an interactive website to find malicious software.

Future work on the project will involve training a malware dataset using sophisticated machine learning algorithms and evaluating accuracy. A machine learning ensemble model will pick the most accurate classifier. The Malware Detection and Prediction System will soon have the ability to predict the type of malware.

REFERENCES

[1] N. Idika and A. P. Mathur, A survey of malware detection techniques[Predoctoral Fellowship, and Purdue Doctoral Fellowship], Purdue University, 2007.
[2] P. Vinod, R. Jaipur, V. Laxmi, and M. S. Gaur, “Survey on malware detection methods,” in Proceedings of the 3rd Hackers' Workshop on Computer and Internet Security (IITKHACK '09), 2009.
[3] S. Cesare and Y. Xiang, Software Similarity and Classification, Springer Science & Business Media, 2012.
[4] P. Okane, S. Sezer, and K. McLaughlin, “Obfuscation: the hidden malware,” IEEE Security & Privacy, vol. 9, no. 5, pp. 41–47, 2011. View at Publisher · View at Google Scholar · View at Scopus
[5] A.Moser, C. Kruegel, and E. Kirda, “Limits of static analysis for malware detection,” in Proceedings of the 23rd Annual Computer Security Applications Conference (ACSAC '07), pp. 421–430, December 2007.
[6] M.Alaeiyan, S.Parsa, M.Conti, Analysis and Classification of context based malware, Comp. Commun. 136(2019) 76-90 2019, doi:10.1016/jcomcom
[7] S.M. Bidoki, S. Jalili, A. Tajoddin, PbMMD: a novel policy based multi-process malware detection, Eng. Appl. Artif. Intell. 60 (August 2016) (2017) 57–70, doi:10.1016/j.engappai.2016.12.008.
[8] A.Bushby, F. Cybersecurity, How deception can change cyber security defences, Comp. Fraud Secur. Bull. 2019 (1) (2019) 12–14, doi:10.1016/S13613723(19)30008-9.
[9] I.K. Cho, T.G. Kim, Y.J. Shim, M. Ryu, E.G. Im, (2016). Malware analysis and classification using sequences 8587(June).10.1080/10798587.2015.1118916 .
[10] Neamat Al Saraha , Fahmida Yasmin Rifata , Md. Shohrab Hossainb , Husnu S. Narman,” An Efficient Android Malware Prediction Using Ensemble machine learning algorithms”, The 18th International Conference on Mobile Systems and Pervasive Computing (MobiSPC) August 9-12, 2021
[11] Ömer Aslan, Refik Samet,” A Comprehensive Review on Malware Detection Approaches”, 10.1109/ACCESS.2019.
[12] Subrahmanian, V. S., Ovelgönne, M., Dumitras, T., & Prakash, B. A. (2015). Types of Malware and Malware Distribution Strategies. Terrorism, Security, and Computation, 33–46. doi:10.1007/978-3-319-25760-0_2
[13] Bronshtein, A. (2019, May 6). A quick introduction to K-nearest neighbors algorithm. Medium. Retrieved January 10, 2023, from https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearestneighbors-algorithm-62214cea29c7 Borcan, M. (2020, March 21). Decision tree classifiers explained. Medium. Retrieved January 10, 2023, from



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY