End-to-End Implementation of Malware Detection Using Machine Learning

Subbulakshmi T[1], Josiga Subramanian[2], Suganya R[3],
Girish Subramanian[4], Yerramalli Sai Sreekar[5], Vasu Bansal[6]

{ research.subbulakshmi@gmail.com[1], sjosiga@gmail.com[2], suganya.ramamoorthy@gmail.com[3]}

School of Computer Science and Engineering, Vellore Institute of Technology, Chennai

Abstract. The trend in rising frequency of malware attacks provides the need for effective malware detection techniques. Machine learning has emerged as a promising approach for identifying and classifying malicious software, where the detection of new attacks are performed with the single or combination of machine learning algorithms. In this research work, a web application that utilizes machine learning techniques to classify the uploaded files as malware or not has been implemented. The system is based on a comprehensive dataset containing known malware and benign files, which are used for training and testing of the machine learning algorithms. Experimental results demonstrate that the implemented system is highly effective at detecting malware, achieving high accuracy rates even with previously unseen samples. The importance of machine learning for addressing the growing threat of malware attacks and provides a practical solution for identifying potentially harmful files has been highlighted.

Keywords: Machine Learning, Malware Detection, Cyber Security, Malware, Feature Extraction, Random Forest, Principal Component Analysis, Adaboost, Gaussian Naïve Bayes, PE File.

1 Introduction

The execution of malware attacks in recent years poses significant challenges in the global level. Malware attacks compromise sensitive data and lead to significant financial losses. As a result, the cyber security field prioritizes the development of effective malware detection methods. This research work introduces novel approach to malware detection using machine learning algorithms specifically focusing on the analysis of Portable Executable (PE) files, which are commonly used in Windows operating systems. A comprehensive dataset of known malware and non-malware PE files has been utilised to train and test several machine learning classifiers, including Random Forest, Adaboost, and Gaussian Naïve Bayes. These algorithms assess file attributes like size, type, and content. A publicly available dataset of malware and non-malware files is used for training and testing. Results indicate high malware detection accuracy. This

IACIDS 2023, November 23-25, Lavasa, India
Copyright © 2024 EAI
DOI 10.4108/eai.23-11-2023.2343241

research enhances cybersecurity by employing machine learning and advanced analysis to improve malware detection accuracy, mitigating risks associated with these evolving threats.

Malware attacks, growing in sophistication and frequency, challenge cybersecurity. In 2023, malware instances are surging at an alarming rate, with 300,000 new cases emerging daily, primarily through email channels. Detection of malware takes an average of 49 days. Websites are heavily impacted, with 4.1 million infected by malware, and 18% posing critical cybersecurity threats. Furthermore, a striking 97% of website breaches target WordPress plugins. SonicWall's 2023 Cyber Threat Report indicates a 2% year-over-year increase in malware attacks, reaching 5.5 billion attacks. This rise is driven by a 43% surge in cryptojacking and an 87% increase in IoT malware in 2022, counterbalancing the decline in ransomware volume and marking a significant shift in malware trends since 2018. Malware infections jeopardize computers, sensitive data, and financial standing. Traditional signature-based detection is increasingly ineffective. Machine learning offers a promising solution by analyzing complex data for malicious patterns.

2 Literature Review

Machine learning-based approach is used to detect emerging malware using network traffic features [1] involving a framework employs semi-supervised learning to enhance accuracy and strengthen network security by gathering malware traces for signature-based detection. A survey to explore the advanced machine learning techniques [2], for more effective malware detection and analysis was done to overcome the difficulties faced by the traditional methods. The challenges of detecting advanced malware are discussed along with existing techniques comprehensively [3].

Machine learning and traffic analysis are used to detect Android SMS malware effectively, achieving 99.96% accuracy for binary classification and 91.12% for multiclass classification [4] and the most challenging malware types, such as FakeNotify and Beanbot are identified. The research [5] presents a study focused on using machine learning with n-gram features from Android smali files to automate malware detection, enhancing security against threats like data theft and identity fraud. Machine Learning is used to detect and classify polymorphic malware [6], which continually changes its characteristics to evade signature-based detection methods, emphasizing behavioural pattern analysis for effective malware recognition.

Random Forest model of machine learning to detect malware in Windows PE files with a remarkable 99.7% accuracy in [7] and [8] by efficiently distinguishing between benign and malicious files and uses opcode frequency as a feature, showcasing its effectiveness in combating malware. Malware detection [9] and [10] utilizes Machine Learning to identify malware based on opcode sequences, offering an effective approach to combat the evolving nature of malware threats and a method to detect malware in Android smartphones using machine learning techniques and API classes with an average classification precision rate of 91.9% was presented.

This research introduces an innovative and expedient approach to malware detection, offering a distinctive contribution to the field. By leveraging a stateless classification method, the system outpaces conventional software in malware checks. Its capacity to swiftly identify potential threats sets it apart and addresses the escalating menace of malware attacks, providing individuals and organizations with enhanced cybersecurity defences.

3 Proposed Methodology

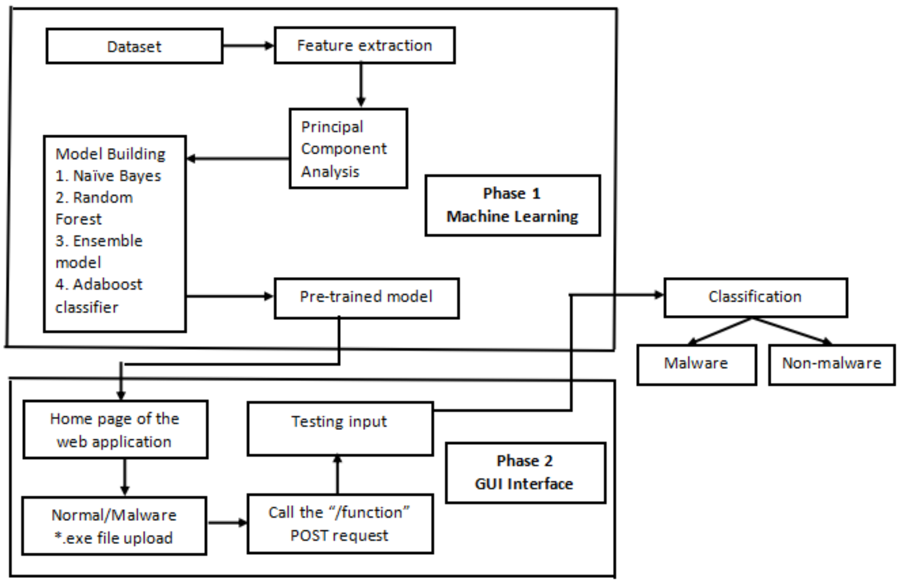


Fig. 1 Proposed architecture for the malware detection application using machine learning

A. Dataset

The dataset consists of 78 columns (out of which only 69 contain accessible or useful information) representing different features extracted from benign or harmful viruses and 19611 rows, each of which contains data of a particular virus abstracted from VirusShare. One of these columns contains information on whether the given sample is a virus or not in binary format. Unnecessary columns like name of the file, e_magic, e_cblp and some other error values are dropped in order to maintain high accuracy of the expected model.

B. PE File

PEFile, which stands for Portable Executable File, is a file format used in Microsoft Windows operating systems to represent executables, DLLs (Dynamic Link Libraries), and other binary files. The PEFile structure follows a well-defined format that includes a header and various sections. The header contains important information such as the file's signature, architecture, entry point, and size of the different sections. The sections in a PEFile store different types of data, including code, data, resources, and imports/exports.

C. Feature Extraction

Feature importance in machine learning is a critical aspect that helps determine the relative importance of various features in predicting target variable. By understanding feature importance, practitioners can make informed decisions regarding feature selection, model interpretation, and optimization. This information helps in building more accurate and interpretable machine learning models, leading to better insights and predictions. The required PEFile header features need to be extracted and passed to the trained model for prediction. The implemented code is able to extract 69 out of the 78 features available in the dataset.

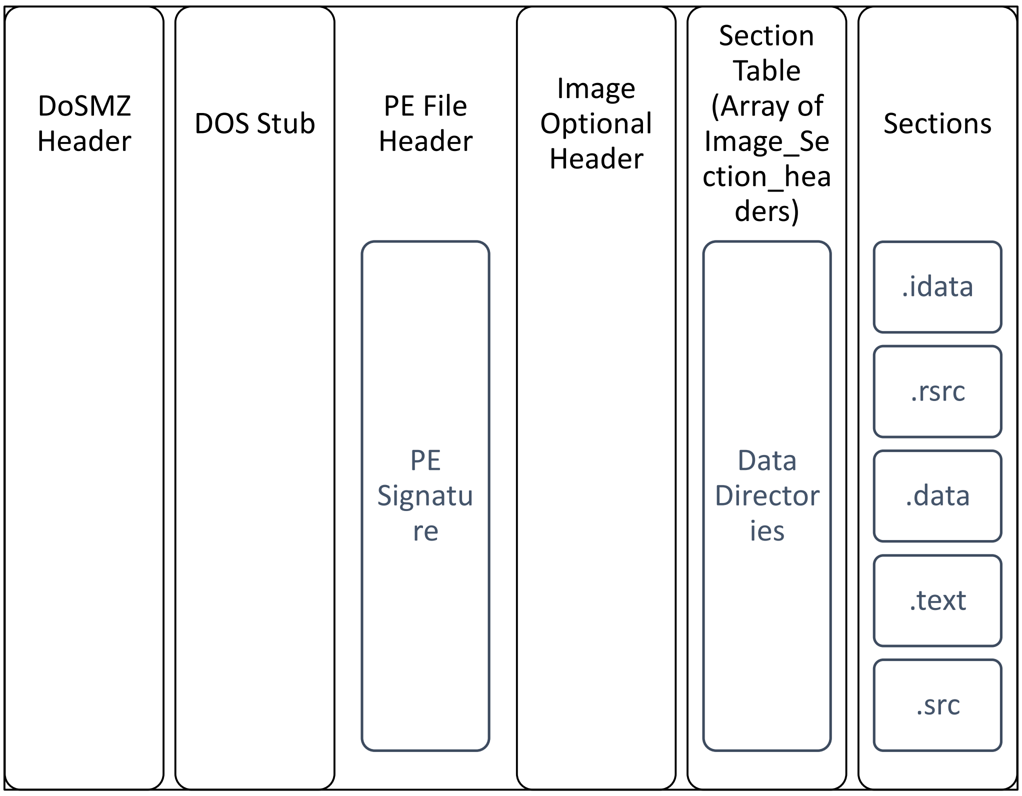


Fig 2: PEFile Structure

D. Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique used to transform a dataset with multiple correlated features into a lower-dimensional representation while preserving the most important information. PCA is widely used for feature extraction, visualization, and data compression. It can help reduce the complexity of a dataset, improve model performance by removing redundant information, and provide insights into the underlying structure of the data.

E. Model Building

80% of the dataset is utilised for training. In order to find the best accuracy, various models like Gaussian Naïve Bayes, Random Forest, Adaboost Classifier and an Ensemble model were used. The remaining 20% of the data was used for testing. The accuracy, precision, recall and F1-score of each of the model was calculated and the model with the best score was chosen as shown in Table 1.

F. FrontEnd

The backend implements a Python program which functions as shown in Figure 3. Once a suspected file is uploaded, it is sent to the backend where required features are extracted using PEFile module of Python. A web application that uses Python in the backend, FastAPI as the module for connection to the frontend and HTML, CSS and JS to make a user-friendly UI in order for any person to use as can be seen in Fig. 3. The application will check if the a suspected malware file that is uploaded is malware or benign. The flow of the code will go something like this - the python code is kept running live with 'uvicorn', an ASGI server implementation for Python. The user accesses the home page and uploads the file they feel requires some checking.

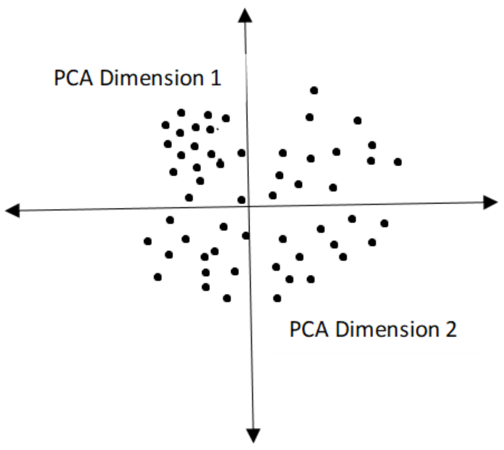


Fig 3: Working of Principal Component Analysis

G. Prediction

Once the features are extracted, the chosen model is used to predict whether uploaded file is a malware or benign. Based on application's prediction, the user is redirected to different pages to further show the required output and provide some guidance to user considering a malware does exist.

4 Implementation and Results

Once a suspected file is uploaded, it is sent to the backend where required features are extracted using PEFile module of Python. The structures defined header files can be accessed as attributes in a PE instance. The following features were found to be the most important in the analysis that has been performed as shown in Figure 2.

1. e_ss: “Stack Segment” specifies the size of the stack segment in the executable file. It can be relevant in analyzing malware to understand the memory layout and usage of the program, as abnormal or suspicious stack sizes may indicate potential buffer overflow or stack-based attacks.

2. Characteristics: “Characteristics” field contains flags that describe various characteristics of the executable, such as whether it is a DLL or an executable, if it is a console or GUI application, whether it is a 32-bit or 64-bit binary, etc. The characteristics can provide insights into the intended behavior of the program and identify potential anomalies or deviations from expected behaviors, which could be indicative of malware.

3. SizeOfInitializedData: “SizeofInitializedData” field specifies the size of the initialized data section. It contains data that is initialized by the operating system when the executable file is loaded into memory. Malware applications may attempt to manipulate this section to store malicious code or data, or to hide its presence by modifying the expected size of this section.

4. e_lfanew: “e_lfanew” field is an offset to the PE header, that contains data on structure and layout of a Portable Executable file. The value of this field is used by the operating system to locate the PE header. Malware applications may attempt to tamper with this field to avoid being detected or alter the expected execution flow of the program in the backend.

5. SizeOfHeapCommit: “SizeOfHeapCommit” field specifies the size of the committed heap memory that is allocated when the executable file is loaded. An unusually large value in the field indicates potential heap-based attacks or abnormal memory allocation behavior, which could detect the presence of malware.

6. SizeOfUninitializedData: “SizeOfUninitializedData” field specifies the size of the uninitialized data section as opposed to the “SizeOfInitializedData” field. It contains data that is not initialized by the operating system when the executable file is loaded into the memory. Malware applications may attempt to manipulate this section to store destructive code or data.

7. Magic: “Magic” field is a signature that identifies the file as a valid Portable Executable file. Different values of this field indicate the type of the PE files, such as executable, DLL, etc. Analyzing this value can help predict if the uploaded file is a legitimate PE file or a suspicious disguised malware.

8. PointerToSymbolTable: “PointerToSymbolTable” field points to the symbol table, which contains information about the symbols (e.g., functions, variables) in the portable executable file. This field is used to analyze and identify disguised malware using methods such as missing or manipulated symbol table entries, that might indicate attempts to hide the suspicious file or program.

9. AddressOfEntryPoint: “AddressOfEntryPoint” field specifies the entry point of the executable, that is the first instruction that is executed when the program starts. Malware may attempt to modify this field to alter the expected execution flow of the program or to inject malicious code at the entry point, which can be used to detect abnormal activities and malware.

10. TimeDateStamp: “TimeDateStamp” field contains a timestamp indicating the time at which the executable file was compiled. Analyzing this field will provide information on the age of the executable and identify potential problems, such as executables with future timestamps or unusually old timestamps, which may indicate suspicious activity or attempts to evade detection of malware.

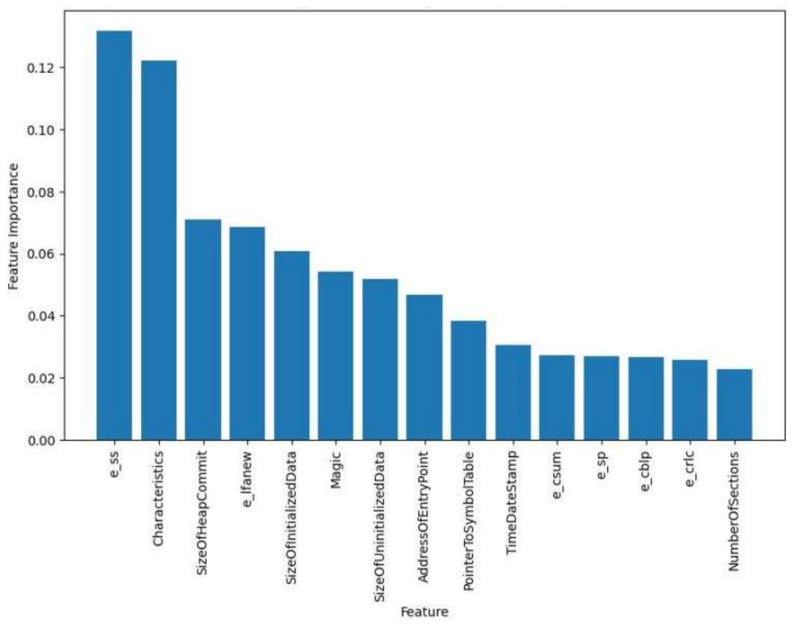


Fig. 4 Top Features Sorted on Basis of Importance.

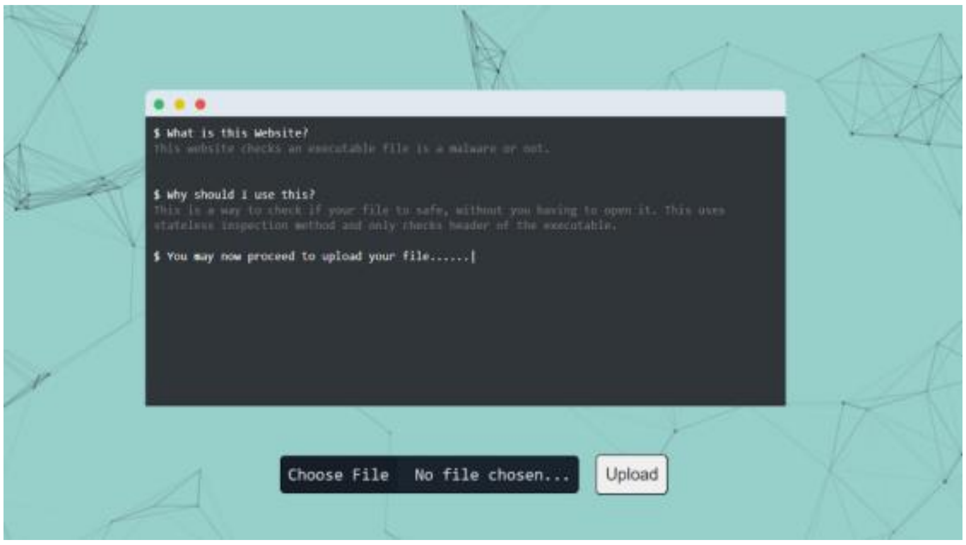
Comparison of the pre-trained model is tested and the precision, recall, F1 score and Support obtained in Gaussian Naïve Bayes, Random Forest Model, Ensemble Model, Adabooster algorithm with and without Principal Component Analysis are calculated using the program. Table 1 shows the comparison of all the algorithms used. Ensemble technique with Principal Component Analysis is used in the Web Application as it demonstrates the highest accuracy of 99.60%.

TABLE I. Summary of Reports of All Models Used

Algorithm /Model Used	**Accuracy (%)**		**Precision**		**Recall**		**F1 score**	
	Using PCA	Using PCA	Using PCA	Using PCA	Using PCA	Using PCA	Using PCA	Using PCA
Gaussian Naïve Bayes	28.80	32.24	0.94	0.93	0.29	0.32	0.39	0.41

Random Forest	98.72	98.72	0.99	0.99	0.99	0.99	0.99	0.99
Adaboost	94.51	99.29	0.95	0.99	0.98	0.99	0.99	0.99
Ensemble	99.60	98.39	0.99	0.99	0.99	0.99	0.99	0.99

The homepage of the web application contains a typing animation that guides the users about the features of the website as shown in Figure 3a. The suspected file is passed onto the backend using the Choose file option present in the homepage of the web application. Python functions in the backend predict whether the uploaded file is malware or benign. If the input file has a risk of being a malware, a red light is displayed in the homepage of the application and as shown in Figure 3b. Some methods are suggested on how to check if their PC has been damaged. If the file is benign, a green light is displayed on the home page of the web application.

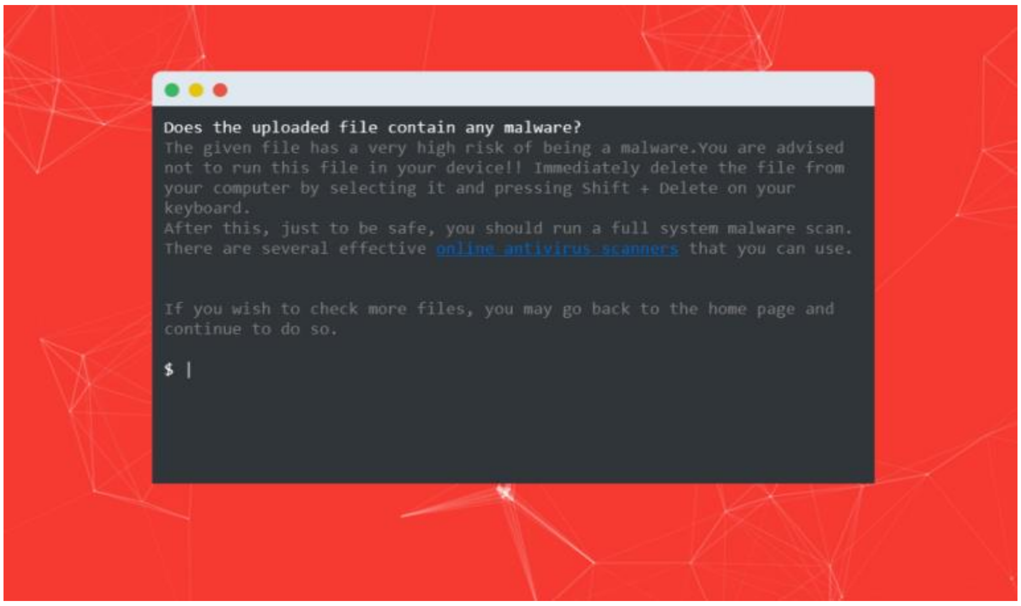


```
$ What is this Website?
This website checks an executable file is a malware or not.


$ why should I use this?
This is a way to check if your file to safe, without you having to open it. This uses
stateless inspection method and only checks header of the executable.

$ You may now proceed to upload your file......|
```

Fig. 5a Typing animation in the home page of the application



```
Does the uploaded file contain any malware?
The given file has a very high risk of being a malware.You are advised
not to run this file in your device!! Immediately delete the file from
your computer by selecting it and pressing Shift + Delete on your
keyboard.
After this, just to be safe, you should run a full system malware scan.
There are several effective online antivirus scanners that you can use.


If you wish to check more files, you may go back to the home page and
continue to do so.

$ |
```

Fig. 5b The page turns red as the application detects malware

5 Conclusion

In conclusion, this research work on end-to-end implementation of malware detection using multiple machine learning models had provided us with a deep understanding of the complexity involved in the detection of malware in the modern digital landscape. After experimenting with models mentioned in Table 1, we had selected Random Forest with Principal Component Analysis as the final model due to its ability to improve the accuracy of the detection process and reduce the risk of false positives. It is believed that if stateless inspection of the PE header file gives such a high accuracy (99.41%), it is worth using rather than stateful inspection which takes a lot of time. By automating the process of malware detection, individuals and organizations can utilize this research work to detect malware infections early, prevent data breaches, and reduce downtime. These benefits can significantly enhance the security of sensitive information and maintain business continuity. This research work serves as a foundation for future research in the domain of malware detection, and we hope it can inspire others to build upon our work and contribute to the development of more robust and accurate machine learning models. This research work aims to provide individuals and organizations a powerful tool to stay safe in an ever-evolving digital landscape.

References

[1] S. D. Mukesh, J. A. Raval, and H. Upadhyay, "Real-Time Framework for Malware Detection Using Machine Learning Technique", in Smart Innovation, Systems and Technologies, vol. 83, August 2017.
[2]S. Soja Rani and S. R. Reeja, "A Survey on Different Approaches for Malware Detection Using Machine Learning Techniques", in Lecture Notes on Data Engineering and Communications Technologies, vol. 39., June 2019.
[3]Namita and Prachi, "PE File-Based Malware Detection Using Machine Learning", in Advances in Intelligent Systems and Computing, vol. 1164, 02 July 2020.
[4]Kanis R. Mim, Md. Sakir Hossain, Sumona A. Tisha, Kallul R. Kalpo, Mehedi H. Bakul, and Md. Shakhawat Hossain", Traffic Analysis-Based Android SMS Malware Detection Using Machine Learning," in Lecture Notes in Networks and Systems, vol. 437, 04 October 2022.
[5]B. Tahtaci and B. Canbay, "Android Malware Detection Using Machine Learning", IEEE, 2018.
[6]S. Choudhary and A. Sharma, "Malware Detection & Classification using Machine Learning", in 2020 International Conference on Emerging Trends in Communication, Control and Computing (ICONC3), Lakshmangarh, India, 2020
[7]A. Kumar, K. Abhishek, K. Shah, D. Patel, Y. Jain, H. Chheda, and P. Nerurkar", Malware Detection Using Machine Learning," in Communications in Computer and Information Science, vol. 1232, 10 December 2020.
[8]H. Rathore, S. Agarwal, S. K. Sahay, and M. Sewak, "Malware Detection Using Machine Learning and Deep Learning", in Lecture Notes in Computer Science, vol. 11297, 22 November 2018.
[9]N. B. Muppalaneni and R. Patgiri, "Malware Detection Using Machine Learning Approach", in Lecture Notes in Networks and Systems, vol. 180, 23 March 2021.
[10]Westyarian, Y. Rosmansyah, and B. Dabarsyah, "Malware detection on Android smartphones using API class and machine learning", in 2015 International Conference on Electrical Engineering and Informatics (ICEEI), 10-11 August 2015.