Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 5, May : 2024

UNMASKING MALICIOUS SOFTWARE WITH MACHINE LEARNING

B. Rajitha, Assistant Professor CSE, Vaagdevi College of Engineering (Autonomous), India
J. Shashank, UG Student, CSE, Vaagdevi College of Engineering (Autonomous), India
K. Bhavishya, UG Student, CSE, Vaagdevi College of Engineering (Autonomous), India
B. Sowmyasri, UG Student, CSE, Vaagdevi College of Engineering (Autonomous), India
M. Abhishek, UG Student, CSE, Vaagdevi College of Engineering (Autonomous), India

ABSTRACT
Android Apps are freely available on Google Playstore, the official Android app store as well as third-party app stores for users to download. Due to its open source nature and popularity, malware writers are increasingly focusing on developing malicious applications for Android operating system. In spite of various attempts by Google Playstore to protect against malicious apps, they still find their way to mass market and cause harm to users by misusing personal information related to their phone book, mail accounts, GPS location information and others for misuse by third parties or else take control of the phones remotely. Therefore, there is need to perform malware analysis or reverse-engineering of such malicious applications which pose serious threat to Android platforms. Broadly speaking, Android Malware analysis is of two types: Static Analysis and Dynamic Analysis. Static analysis basically involves analyzing the code structure without executing it while dynamic analysis is examination of the runtime behavior of Android Apps in constrained environment. Given in to the ever-increasing variants of Android Malware posing zero-day threats, an efficient mechanism for detection of Android malwares is required. In contrast to signature-based approach which requires regular update of signature database.

1. INTRODUCTION
Android is an open source free operating system and it has support from Google to publish android application on its Play Store. Anybody can developed an android app and publish on play store free of cost. This android feature attract cyber-criminals to developed and publish malware app on play store. If anybody install such malware app then it will steal information from phone and transfer to cyber-criminals or can give total phone control to criminal's hand. To protect users from such app in this paper author is using machine learning algorithm to detect malware from mobile app. To detect malware from app we need to extract all code from app using reverse engineering and then check whether app is doing any mischievous activity such as sending SMS or copying contact details without having proper permissions. If such activity given in code then we will detect that app as malicious app. In a single app there could be more than 100 permissions (examples of permissions are transact, API [1] call signature, on Service Connected, API call signature, bind Service, API call signature, attach Interface, API call signature, Service Connection, API call signature, android .os.Binder, API call signature, SEND_SMS, Manifest Permission, Ljava.lang. Class.get Canonical NameAPI call signature etc.) which we need to extract from code and then generate a features dataset, if app has proper permission then we will put value 1 in the features data and if not then we will value 0. Based on those features dataset app will be mark as malware or good ware. In this paper author is using two machine learning algorithms such as SVM (Support Vector Machine) and NN (Neural Networks). App will contains more than 100 features and machine learning will take more time to build model so we need to optimized (reduce dataset columns size) [2] features, to optimized features author is using genetic algorithm. Genetic algorithm will choose important features from dataset to train model and remove un-important features. Due to this process dataset size will be reduced and training model will be generated faster. In this paper comparison we are losing some accuracy after applying genetic algorithm but we are able to reduce model training execution time.

2. LITERATURE SURVEY

1. We propose a versatile framework in which one can employ different machine learning algorithms to successfully distinguish between malware files and clean files, while aiming to minimise the number of false positives. In this paper we present the ideas behind our framework by working firstly with cascade one-sided perceptron's and secondly with cascade Kernel zed one-sided perceptron's[3]. After having been successfully tested on medium-size datasets of malware and clean files, the Ideas behind this framework were submitted to a scaling-process that enable us to work with very large datasets of Malware and clean files.
2. One of the most significant issues facing internet users nowadays is malware. Polymorphic malware is a new type of malicious software that is more adaptable than previous generations of viruses. Polymorphic malware constantly modifies its signature traits to avoid being identified by traditional signature-based malware detection models[4]. To identify malicious threats or malware, we used a number of machine learning techniques. A high detection ratio indicated that the algorithm with the best accuracy was selected for usage in the system. As an advantage, the confusion matrix measured the number of false positives and false negatives, which provided additional information regarding how well the system worked. In particular, it was demonstrated that detecting harmful traffic on computer systems, and thereby improving the security of computer networks, was possible using the findings of malware analysis and detection with machine learning algorithms to compute the difference in correlation symmetry (Naive Byes, SVM, J48, RF, and with the proposed approach) integrals. The results showed that when compared with other classifiers, DT (99%), CNN (98.76%), and SVM (96.41%) performed well in terms of detection accuracy[5]. DT, CNN, and SVM algorithms' performances detecting malware on a small FPR (DT = 2.01%, CNN
= 3.97%, and SVM = 4.63%,) in a given dataset were compared. These results are significant, as malicious software is becoming increasingly common and complex.
3. In the last decade, Hardware Performance Counters (HPCs) events are increasingly used by Machine Learning (ML) algorithms for malware detection. Modern processors provide a variety of HPCs to measure and monitor processes' events such asmemory accesses, instructions, etc. during their execution. In this paper, an analysis study to categorize the machine learning algorithms based on HPCs that have been used for malware detection is introduced. besides, the most efficient and effective features of HPCs that have been exploited to recognize the abnormal activities on various systems are identified. Furthermore, the Neural Network (NN) [6] Algorithms including Multi- Layer Perceptron (MLP), Convolutional Neural Network (CNN), and Full Order Radial Basis Function (RBF) algorithms are used to simulate several experiments from the literature. The simulation results show that the accuracy of MLP, CNN, and Full Order RBF are 96.95%, 98.22%, and 98.68%, respectively.
4. This research study mainly focused on the dynamic malware detection. Malware progressively changes, leading to the use of dynamic malware detection techniques in this research study. Each day brings a new influx of malicious software programmes that pose a threat to online safety by exploiting vulnerabilities in the Internet[7]-[9]. These records are converted into sparse vector models for use in further machine learning efforts. Classifiers used to synthesise the results of this study included kNN, DT, RF, AdaBoost, SGD, extra trees and the Gaussian NB classifier. After reviewing the test and experimental data for all five classifiers, we found that the RF, SGD, extra trees and Gaussian NB Classifier all achieved a 100% accuracy in the test, as well as a perfect precision (1.00), a good recall (1.00), and a good f1- score (1.00). Therefore, it is reasonable to assume that the proof-of-concept employing autonomous behaviour- based malware analysis and machine learning methodologies might identify malware effectively and rapidly.
5. Malware has become an enormous risk in today's world. There are different kinds of malware or malicious programs found on the internet. Research shows that malware has grown exponentially over the last decade, causing substantial financial losses to various organizations. Malware is a malicious program or software that proves exceedingly harmful to the user's computer. The user's system can be affected in several ways[10]. The proposed solution uses various machine learning techniques to detect whether a file downloaded from the internet contains malware or not. This research aims to use

different machine learning algorithms to differentiate between malicious and benign files successfully. The main idea is to study different features of the downloaded file like MD5 hash, size of the Optional Header, and Load Configuration Size. Based on the analysis performed on these features, the files will be classified as malicious or non-malicious. The models are trained on these different features which enables them to learn how to classify files. The models after proper training will be compared among each other based on various criteria. This comparison is made with the help of the Validation and Test datasets. Finally, the model with the best accuracy will be selected. This process helps in identifying all those types of malware that can have a detrimental impacton the user's system after getting infected. The approach used here will be able to detect malware like Adware, Trojan, Backdoors, Unknown, Multidrop, Rbot, Spam, and Ransomware. After training and testing various machine learning models, the Random Forest Classifier was found to be the most accurate. It's accuracy went as high as 99.99% in the case of the test dataset[11]. This was closely followed by the XGBoost model with an accuracy of 99.68%. The results of five different models have been compared with those obtained in the previous research. These include the Decision Tree Classifier (99.57% accuracy), Random Forest Classifier (99.99% accuracy), Gradient Boosting Model (99.09% accuracy), XGBoostModel (99.68% accuracy), and AdaBoost Model (98.87% accuracy). Four out of five of these models have been found to have accuracies greater than those obtained in previous research works.

3. PROBLEM STATEMENT

Machine learning (ML) has revolutionized malware detection systems by providing methods that excel in identifying and neutralizing new and evolving threats beyond the capabilities of traditional signature-based techniques. In these systems, various ML models such as Decision Trees, Support Vector Machines, Random Forests, Neural Networks, and deep learning approaches like Convolutional and Recurrent Neural Networks are employed[12].

These models are trained on labeled datasets comprising both malware and benign software, leveraging classification for direct threat recognition and clustering for anomaly detection which highlights deviations from normal operations that might indicate a threat.

Feature extraction plays a crucial role, involving static analysis of the binaries for extracting metadata, dynamic analysis which observes behavior by running the software in a controlled environment, and hybrid methods that combine both. This comprehensive feature analysis helps in training robust ML models that specialize in different detection techniques including signature-based, behavioral-based, and heuristic-based detection, enhancing the adaptability and effectiveness of malware detection systems.

Static Analysis: Involves extracting features from the binary without executing it, such as the presence of certain opcodes, use of particular APIs, file size, and header information.

Dynamic Analysis: Involves running the software in a controlled environment to monitor its behavior, including system calls made, files written, and network activity.

LIMITATION OF SYSTEM

- Limited to Known Threats: Signature-based systems struggle with new, unseen malware.
- False Positives: Heuristic approaches may flag harmless programs as malicious.
- Scalability Issues: Difficulty handling the growing volume and diversity of malware.
- Polymorphic Challenges: Ineffectiveness against constantly changing malware code.
- Dependency on Updates: Vulnerabilities if users don't regularly update antivirus databases.
-

4. PROPOSED SYSTEM

Two set of Android Apps or APKs: Malware/Good ware are reverse engineered to extract features such as permissions and count of App Components such as Activity, Services, Content Providers, etc.

These features are used as feature vector with class labels as Malware and Good ware represented by 0 and 1 respectively in CSV format.[13],[14] To reduce dimensionality of feature-set, the CSV is fed to Genetic Algorithm to select the most optimized set of features. The optimized set of features obtained is used for training two machine learning classifiers: Support Vector Machine and Neural Network. In the proposed methodology, static features are obtained from AndroidManifest.xml which contains all the important information needed by any Android platform about the Apps. Androguard tool has been used for disassembling of the APKs and getting the static features.

ADVANTAGES

- Security Proposed a novel and efficient algorithm for feature selection to improve overall detection accuracy.
- Machine-learning based approach in combination with static and dynamic analysis can be used to detect new variants of Android Malware posing zero-day threats.

5. IMPLEMENTATION

5.1 Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

5.2 Numpy

Numpy is a general-purpose array-processing package[15]. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

5.3 Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

5.4 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties,

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 5, May : 2024

etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

6. EXPECTED RESULTS

Fig 6.1 Home Page

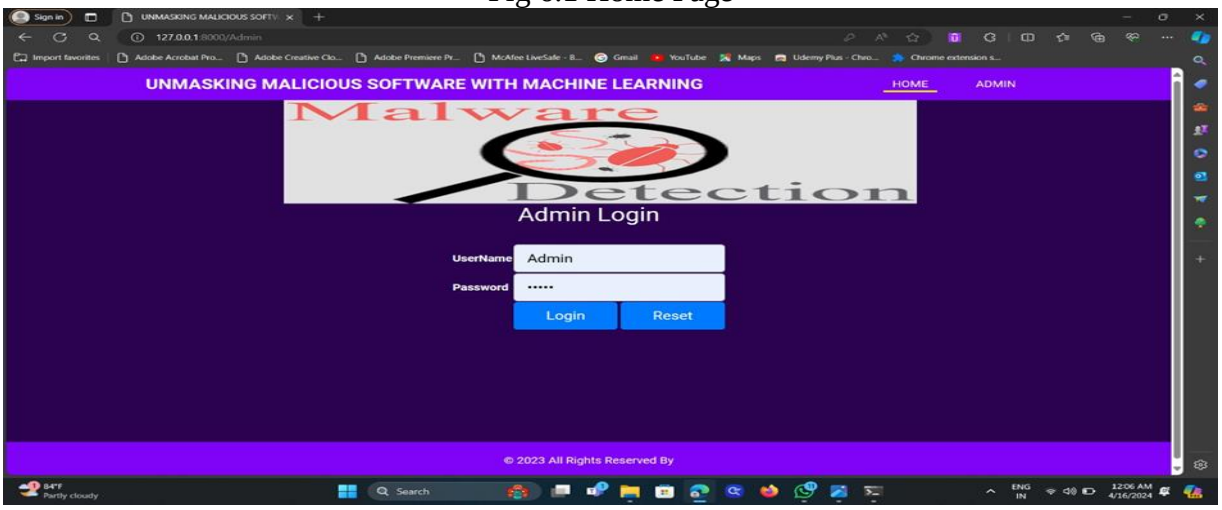

Fig 6.2 Admin Login Page

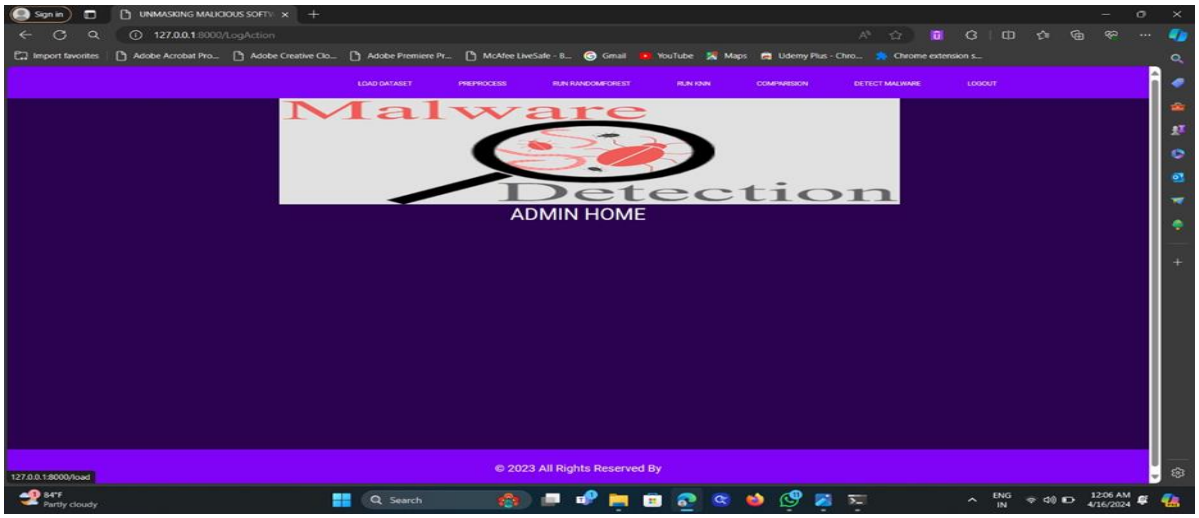

Fig 6.3 Admin Page

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 5, May : 2024

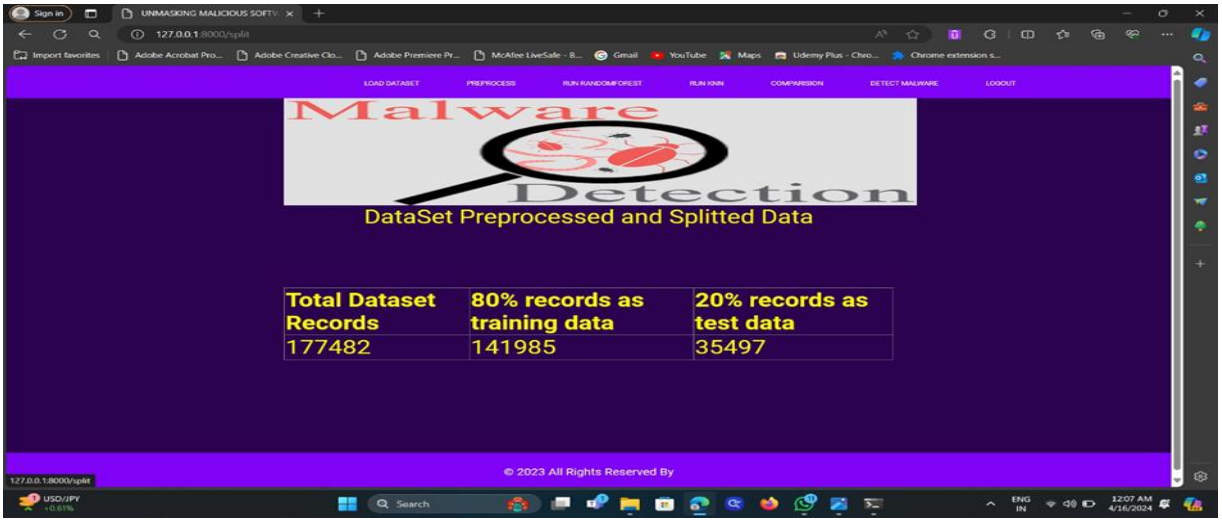

Fig 6.4 Data Preprocessing page

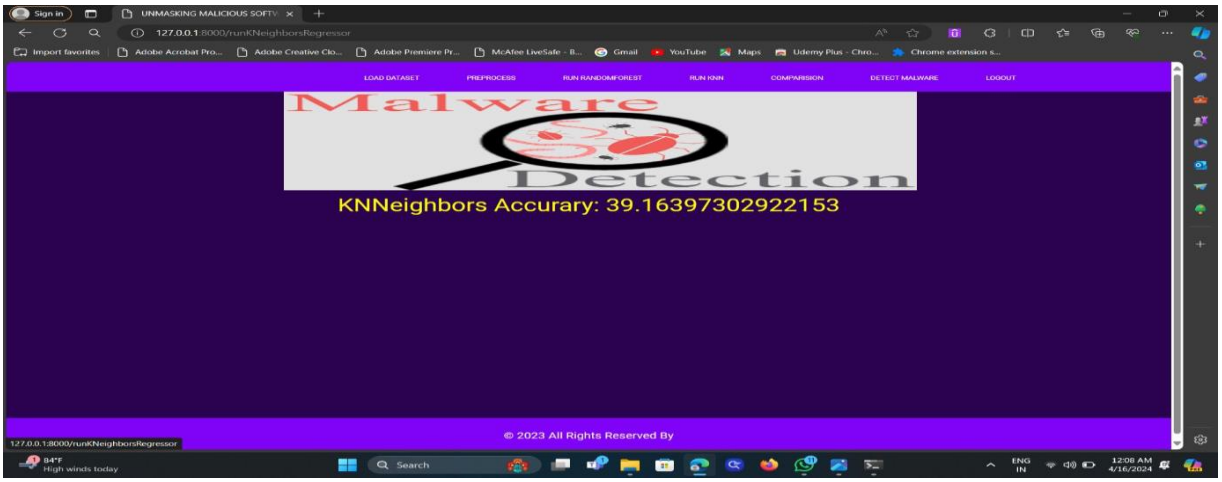

Fig 6.5 Run KNN page

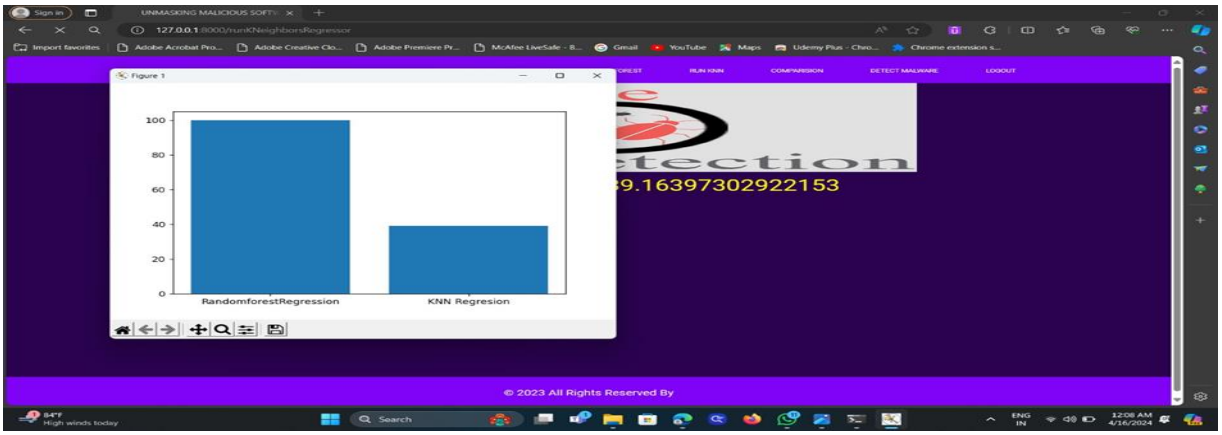

Fig 6.6 Comparison between KNN and Random Forest Page

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 5, May : 2024

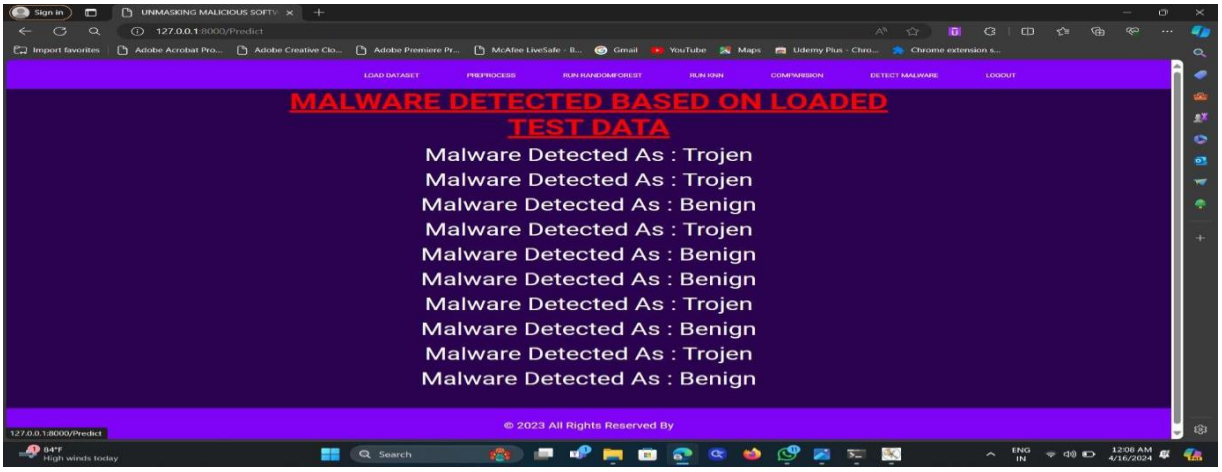

Fig 6.7 Detecting Malware Page

7. CONCLUSION AND FUTURE SCOPE

The goal of this research is to find a machine learning solution to the malware problem. Due to the increasing rise of malware, we require automatic solutions to detect infected files. The data set was constructed using infested and clean executables in the first portion of the investigation, and we used a Python script to extract the data needed for the data set generation. After it has been created, the data collection must be ready to train machine learning algorithms. The algorithms that were used were decision trees, Naïve Bayes, and ADA-Boost. After applying the best accuracy methods, it has a Random Forest algorithm with an accuracy of 99.406012 percent. According to this study, the best method for spotting risky apps is Random Forest. If we add a significantly larger number of files to the data set in the future, we can improve the accuracy. This research gives a thorough examination of the topic. Naïve Bayes is often used in text classification, spam filtering, target recognition, or medical diagnosis. Informational classification techniques assume mutually independent and identically distributed attributes. In other words, each attribute is independent from the others, and the values of the attributes are all drawn from a single distribution. The Naïve Bayes classifier is a probability-based classification technique. It uses Bayes\' theorem to calculate the posterior probability of each class given a particular piece of evidence or feature. Ada-boost is a data processor that improves the performance of people who are searching for things in their own personal or corporate information.

FUTURE SCOPE

Other data sets can be added to improve accuracy, and more algorithms with greater performance can be added to improve accuracy. It can be hosted on the web for real-time analysis of exe files in the cloud. One could try categorizing data into different malware categories. One can further make a valid set, experiment with different methods. Development of a hybrid approach with two heads. Dynamic analysis will also be carried out, utilizing both automatic and boring methods, as well as a variety of dynamic instruments. Naïve Bayes is a simple, but powerful probability-based classifier that automatically does some of the heavy- lifting for you. You can also use it to calculate some useful parameters for some other machine learning algorithms. Ada-boost is a set of tools to make marketing more predictive with insights from machine learning. It helps with marketing attribution, predicting marketing spend & ROI, and showing marketing ROI over time.
From experimentations, it can be seen that a decent classification accuracy of more than 94% is maintained using Support Vector Machine and Neural Network classifiers while working on lower dimension feature-set, thereby reducing the training complexity of the classifiers Further work can be enhanced using larger datasets for improved results and analyzing the effect on other machine learning algorithms when used in conjunction with Genetic Algorithm.

8. REFERENCES

[1] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," in Proceedings 2014 Network and Distributed System Security Symposium, 2014.

[2] N. Milosevic, A. Dehghantanha, and K. K. R. Choo, "Machine learning aided Android malware classification," Comput.Electr.Eng., vol. 61, pp. 266–274, 2017.

[3] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-An, and H. Ye, "Significant Permission Identification for Machine-Learning-Based Android Malware Detection," IEEE Trans. Ind.Informatics, vol. 14, no. 7, pp. 3216–3225, 2018.

[4] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention," IEEE Trans. Dependable Secur. Comput., vol. 15, no. 1, pp. 83–97, 2018.

[5] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song, and H. Yu, "SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System," IEEEAccess, vol. 6, pp. 4321–4339, 2018.

[6] W. Han, J. Xue, Y. Wang, L. Huang, Z. Kong, MalDAE : Detected as well as explained malware based on correlation and fusion of static and dynamic characteristics, (2019) 208– 233, http://dx.doi.org/10.1016/j.cose.2019.02. 007.

[7] P. Burnap, R. French, F. Turner, K. Jones, Malware classified using machine activity data, and self-organising feature maps Comput. Secur. 73 (2017) 399–410, http://dx.doi.org/10.1016/j.cose.2017.11.016:

[8] A. Damodaran, F.D. Troia, C.A. Visaggio, T.H. Austin, M. Stamp, A comparison between static, dynamic, and hybrid analysis for malware detection is done, J. Comput. Virol. Hacking Tech. (2017) 1–24, http://dx.doi.org/10.1007/s11416-015-0261-z.

[9] E.M. Dovom, A. Azmoodeh, A. Dehghantanha, D.E. Newton, R.M. Parizi, H. Karimipour, Fuzzy pattern tree for edge malicious files detection and categorization in Iot, J. Syst. Archit. 97 (March) (2019) 1–7, http://dx.doi.org/10.1016/j.sysarc. 2019.01.017.55

[10] K. Khan, A. Mehmood, S. Khan, M.A. Khan, Z. Iqbal, W.K. Mashwani, A survey on intrusion detection and prevention in wireless ad - hoc networks, J. Syst. Archit. (2019) 101701,http://dx.doi.org/10.1016/j.sysarc.2019.101701.

[11] AV-TEST, Malware statistics and trends report, AV-TEST, 2020, https://www.avtest.org/en/statistics/malware.

[12] A. Bushby, F. Cybersecurity, How deception can change cyber security defences, Computer Fraud Secure. Bull. 2019 (1) (2019) 12–14,http://dx.doi.org/10.1016/ S1361-3723(19)30008-9.

[13] E. Gandotra, D. Bansal, S. Sofat, Malware analysis as well as classification: A survey, J. Inf. Secur.05 (02) (2014) 56–64, http://dx.doi.org/10.4236/jis.2014.52006,http://www.scirp.org/journal/PaperDownload.aspx?DOI=10.4236/jis.2014. 52006.

[14] S.M. Muzammal, M.A. Shah, S.J. Zhang, H.J. Yang, security risks and authentication techniques for smartdevices:Int. J. Autom. Computer. 13 (2016) 350–363, http://dx.doi.org/10.1007/s11633-016-1011-5.

[15] M. Ficco, F. Palmieri, Leaf : cybersecurity training platform for realistic edge-iot scenarios,J.Syst.Archit.97(September2018)(2019)107129,http://dx.doi.org/10.1016/j.sysarc.2019.04.004.