
ISSN: 2321-2152
IJMECE
International Journal of modern
electronics and communication engineering
E-Mail
editor.ijmece@gmail.com
editor@ijmece.com
www.ijmece.com

ISSN2321-2152 www.ijmece

Vol 8, Issuse.3 July 2020

A Malware Detection Method Based on Machine Learning for Health Sensor Data.

E.Muralidhar Reddy[1],Erugu Krishna[2],G.Shankar Rao[3],Dr.K.Venkata Ramana[4],

Abstract—Small modifications in the malware code are not detected by traditional signature-based techniques. Currently, the vast majority of malware programmes are based on other programmes that have already been developed. In this way, they share certain patterns yet vary in their signatures. It is crucial to identify the malware pattern rather than only detect slight changes in health sensor data in order to get reliable results. As a result, we have proposed a quick detection solution that uses machine learning-based algorithms to identify patterns in malware code. Data from health sensors will be analysed using XGBoost, LightGBM, and Random Forests in particular. Sequences of bytes/tokens or a single byte/token may be supplied into them (e.g. 1-, 2-, 3-, or 4- grams). It's been possible to amass terabytes of software that has been labelled, including both good and bad applications. In order to train and test the dataset, which comprises of health sensor data, we must first pick and get the features, then adjust the three models and assess the features and models. As soon as one model detects an intruding malware programme, its pattern will be broadcast to the other models, thus thwarting the malware program's incursion.

Keywords—Detection of malware, machine learning, health sensor data, and a recurring trend.

INTRODUCTION

All sorts of sensors are being used to gather health sensor data in the Internet of Things era. Health sensor data is inevitably contaminated with malicious code, which is interpreted as an incursion in the target host computer by a hacker. Cybercriminals use a wide variety of malicious code types to infect computers and other electronic devices. Computer systems and networks may be damaged or even destroyed by malware assaults that steal important data. In today's world, it's one of the most serious dangers to computer security there is. Typically, malware analysis is done using one of two methods [4-7]. Static analysis is often performed by displaying the various resources of a binary file without actually implementing it and evaluating each component of the binary file.

Professor[1,2,3,4], Assistant Professor[1,2,3,4], ,Associate Professor[1,2,3,4],
Department of CSE Engineering,
Pallavi Engineering College,
Mail ID:krishna81.reddy@gmail.com, Mail ID:krishna.cseit@gmail.com,
Kuntloor(V),Hayathnagar(M),Hyderabad,R.R.Dist.-501505.

(2) A disassembler may also be used to disassemble (or rewrite) binary files (such as IDA). Often, machine code can be translated into assembly code, and assembly code can be understood by people. It is possible for anti-malware researchers to decipher a program's assembly instructions and create a mental picture of what it should do. In order to evade this sort of examination, some current malware use confusing approaches, such as inserting grammatical flaws in the code. The disassembler may have difficulty deciphering these mistakes, but the code still runs correctly when they are executed. To do a dynamic analysis, the malware must be seen in action on the host system. For example, modern malware may utilise deceptive approaches such as testing virtual environments or active debuggers to avoid dynamic analysis, delaying the execution of harmful code, or necessitating user interaction [8–10]. We concentrate mostly on static code analysis in this research. Feature matching and broad-spectrum signature scanning are the most common methods of early static code analysis. Broad-spectrum scanning scans the feature code and divides the portions that need to be compared and those that do not need to be compared using masked bytes for feature matching. The hysteresis issue is critical since both approaches need malware samples and characteristics to be extracted before they can be identified. As malware technology advances, it starts to deform throughout the transmission process to prevent detection and elimination, and the number of malware variations skyrockets. As a malware signature, it is difficult to extract a piece of code from the many variations due to the frequent changes in their appearance.

II. REALTED WORK

Using current expertise and knowledge to analyse unfamiliar binary code and categorise malware using machine learning-based approaches seems logical in this circumstance. Machine learning-based algorithms and their applications in malware categorization are examined in this work, according to the instructions [11-14]. Classifying malware and genuine software samples is the core of malware detection and classification. As a result, machine learning algorithms are at the heart of the host malware detection technique described in this paper: Gather a sufficient number of samples of malware code and samples of genuine software. Extract the characteristics from the sample using efficient data processing. Identify the most important characteristics for categorization. Use machine learning methods to build a categorization model from the training data. A trained classification model is used to identify unknown samples. The end aim of this practical job is to determine the most effective features and models. Research questions and concepts are introduced in this chapter. After that, we'll explain: How we got our data, what traits are common, and how we're going to deal with them in this study. After conducting the trials, we apply a detection model based on machine learning approaches to summarise and evaluate the data.

III. MALWARE CODE ANALYSIS

A. Malware Sample Collection

In order to do code analysis, malware samples must be collected in a proper manner. Classification models, when used with machine learning algorithms for detection, may only perform more accurate detection tasks after sufficient training with sample data [38, 40]. Malware samples may be obtained in a variety of methods. Most anti-virus software providers use user-side sampling as their primary strategy. Malware samples are submitted by end users of anti-virus software. The security suppliers typically opt not to freely publish their data, which makes it impossible to access the data directly from this technique. Virus Bulletin, Open Malware, VX Heavens, etc. are all examples of open network databases. The open online sample systems, on the other hand, are restricted at this moment in comparison to the rate of malicious code update, and the sites have issues such as being exposed to assaults. As a result, the need for a malware-sharing mechanism has become more apparent. 3) Additional technological options include: Malware samples may be collected by utilising a capture tool such as a honeypot (such as the Nepenthes honeypot), which is intended to encourage attackers to attack. Other methods of obtaining Trojans and Internet backdoors include using spam traps or security discussion forums. There is a limit to how large a sample size may be obtained via the use of these methods. A business called SecureAge provides the raw data for this research, so we don't have to do any further processing. The static properties of the virus must typically be extracted via code disassembly before any feature extraction can begin. Tools like IDAPro and Hopper, as well as OllyDbg, are common. With IDA Pro, you may build malware assembly code and do additional tasks including identifying functional blocks, retrieving input

functionality and deriving descriptions of functional flow charts. This paper also includes references to these.

B. Feature Selection

A feature may fall into one of three categories: Sample features are most often extracted using feature types depending on sequence. The N-gram is the most prominent example of this technology. Words that appear more than once in a feature sequence are assumed to be linked to those that appear less than once. The N-gram model uses sliding windows to partition a phrase set of length L into L+1 feature sequences. PUSH SUB SAL is one of five distinct 3-gram sequences that may be formed from the word set "PUSH, SUB, SAL, AND," "SAL AND DIV," "AND DIV LDS," and "DIV LDS POP" (L=7 at this time). Three words make up each sequence. Lemmas were selected by Abou-Assaleh [15] using the K-next-neighbor classification approach, a feature extraction framework based on byte sequences. The use of opcodes as a basis for word selection is still another option. New n-gram feature extraction methods were suggested by Henchiri [16]. Using opcode feature extraction, we can better identify malware. Malware detection accuracy was as high as 99 percent when Moskovitch [17] tested more than 3104 files against five opcode-based classifiers. With noisy and inadequate data, to improve classification accuracy. Noise-aware signal combination (NSC) was introduced by Abualsaud [39]. NSC combines k-NN, ANN, SVM, and Bayes models utilising feature extraction depending on their individual performance while maintaining appropriate complexity. Another feature type may be gleaned from the programme code by examining the output string, since the output string reveals a lot about the program's goals. Since the code contains fewer strings, the extracted feature set has fewer dimensions, and less effective control may be obtained with regard to computational costs, than with sequence-based feature sets. As a feature type, a program's call to an application programming interface (API) may also be referred to. It was found that Ding [18] extracted characteristics based on API calls by comparing malicious code and genuine code application programming interfaces and then analysing API calls. Five types of characteristics are selected for this paper: One of the most useful features is the byte count. Binary/hexadecimal coding is used to represent all files on a computer. Counting the quantity of numbers in raw exe files is a natural notion. exe files may be retrieved from the PE header using the following procedure. A label 0/1 is at the start of this string array, which has values ranging from 0 to 255. All strings are counted for the number of 0-255s, and then libsvm files are generated using those numbers.

In machine learning, libsvm files are a typical data format. x:y indicates that the value of the dimension x is the same as the label x:y. There are two labels in the libsvm, one for malware and one for safe software: 0 for malware and 1 for safe software.

The assembly instructions will reveal what the exe file intends to perform, based on the programmer's knowledge and expertise. As a result, each file's assembly code is extracted. In certain cases, it's critical to look at the instructions in their proper context. As a result, we count the number of 1-gram (like test), 2-gram (like test+jnz), 3-gram (like test+jnz+push), and 4-gram command strings and utilise them as features (also for the libsvm format). Features ranging from 1-4 grammes from DAF (Device Assembly Facility). The parameters of the instructions, such as 'test esi, esi,' may be included in the command if the instruction feature is working properly. Memory, register, constant and other types of parameters are divided into five categories and the 1-gram, 2-grams, 3-grams and 4-grams are counted from all instructions with two parameters. The DAF 1-4-gram feature looks like this. Section feature. 4) A disassembled file also includes the section. When a file is opened, the lengths of certain frequent segments are counted and this is the value of the relevant dimension (the segment's name). Functions provided by DLL It is necessary to invoke a system dll function in order to start executable files. There is a good chance that malware will call certain unique dll functions. The additional instruction in the data section of the disassembled file identifies the dll function. Only one dll function will ever be called

from each file, hence the dll function's associated dimension value is 1 (call) and 0. (not call). In order to get malware samples, three typical procedures and three common feature categories are presented. In addition, we'll go through where we got the data for this experiment and which characteristics we'll be using.

Model Selection

A harmful code classifier may be generated using the data gathered from the static and dynamic analysis of the malicious code as inputs to the machine learning algorithm training. To design a classifier, you may use the Naive Bayes. The maximum likelihood estimate approach is often used in Naive Bayesian model parameter estimation. Another way of saying this is that the naive bayesian model may function independently of any Bayesian models[39]. Among machine learning algorithms, KNN is one of the most user-friendly. Using "enhanced learning," in which fresh samples of the training set are taught incrementally without retraining the model, is one of the KNN's main advantages. For binary classification, the SVM algorithm seeks a linear hyperplane. As data sets get larger, the SVM and KNN algorithms become computationally inefficient [39]. "Bagging" models like Random Forests are combined prediction models with numerous decision trees. If we use a decision tree to train our model, we will end up with a Random Forest. It is capable of producing high-accuracy classifiers for a broad range of data. It's capable of dealing with a huge number of different inputs. It may evaluate the significance of factors in selecting categories.. Furthermore, the rate at which one picks up new skills is lightning quick. Traditional learning models include naive Bayes, SVM, KNN, and Random Forest. Some novel machine learning models have also been developed in recent years. XGBoost is an open-source package for C++, Java, Python, R, and Julia that offers a gradient boosting framework. With XGBoost, the algorithm's accuracy may be improved automatically by taking use of the CPU's multithreading capabilities. Many of the winning teams in recent machine learning contests used it as their preferred algorithm, resulting in a recent surge in its popularity and interest. (Wikipedia) This is the simplest portion of XGBoost, the CART (regression tree). For each piece of input data, it creates a classification tree based on its properties and previous predictions. The gini index is used to compute the gain and pick the tree's features during development. gini index formula (1) and gini index gain formula (2) are both provided for your perusal (2). [1] [1] [1] [1] Gini In this case, the word "p" stands for "d" and "p" stands for "a." (1) Type k's probability in dataset D is represented by Pk, and a big K indicates that there are a lot of kinds in D. There are two ways to look at this: ( ) D D Assumption of Risk Gini D DD D1 and D2 represent datasets with feature A and datasets without feature A, respectively, and Gini(D1) specifies the gini index of datasets with feature A in the dataset. Second-order Taylor expansions are used to approximate the objective function, which reduces the temporal complexity in XGBoost [19]. Additionally, it specifies the tree's complexity and applies it to the target function, allowing the tree to expand dynamically by splitting and evaluating segments at the split nodes itself. In comparison to other boosting algorithms, these are some of the advantages that XGBoost provides. In addition to LightGBM [20], there are various boosting models. For ranking, classification, and many other machine learning applications based on decision tree algorithms, it is a high-performance gradient framework. It is part of Microsoft's DMTK initiative. The following are XGBoost's drawbacks: In order to do one iteration, the training data must be traversed several times. (2) A split-gain computation is required for each split node, which also takes a long time. This article uses XGBoost, LightGBM, and Random Forest models. As a result of the poor performance of the SVM (Support vector machine), we chose to stick with these three models instead..

CONCLUSION

The use of machine learning methods in the identification of malicious code in health sensor data has been increasingly appreciated by the academic community and various security companies [27-30, 38, 40]. An study of static code using several machine learning methods and code properties is presented in this work, which draws on the theory of machine learning to combine the benefits of many models [31-33, 36-37]. [34] This study may serve as a reference for the design and implementation of machine-learning malware detection technologies. But it is still in the early stages of development in this sector. There are still a slew of responsibilities and obstacles to overcome, which are listed below. It's difficult to train a machine learning algorithm when there aren't tens of thousands of data points [35] to work with. There is no assurance that the collecting of these essential data will be completed in a timely manner [36, 37]. For many features, we merely know that they are successful but have no idea why. This is the internal explanation. In the next years, this problem will be the most difficult one to solve.

REFERENCES

[1] L. Wu, X. Du, W. Wang, B. Lin, "An Out-of-band Authentication Scheme for Internet of Things Using Blockchain Technology," in Proc. of IEEE ICNC 2018, Maui, Hawaii, USA, March 2018.

[2] M. Shen, B. Ma, L. Zhu, R. Mijumbi, X. Du, and J. Hu, "Cloud-Based Approximate Constrained Shortest Distance Queries over Encrypted Graphs with Privacy Protection", IEEE Transactions on Information Forensics & Security, Volume: 13, Issue: 4, Page(s): 940 – 953, April 2018, DOI: 10.1109/TIFS.2017.2774451.

[3] P. Dong, X. Du, H. Zhang, and T. Xu, "A Detection Method for a Novel DDoS Attack against SDN Controllers by Vast New Low-Traffic Flows," in Proc. of the IEEE ICC 2016, Kuala Lumpur, Malaysia, 2016.

[4] Z. Tian, Y. Cui, L. An, S. Su, X. Yin, L. Yin and X. Cui. A Real-Time Correlation of Host-Level Events in Cyber Range Service for Smart Campus. IEEE Access. vol. 6, pp. 35355-35364, 2018. DOI: 10.1109/ACCESS.2018.2846590.

[5] Q. Tan, Y. Gao, J. Shi, X. Wang, B. Fang, and Z. Tian. Towards a Comprehensive Insight into the Eclipse Attacks of Tor Hidden Services. IEEE Internet of Things Journal. 2018. DOI: 10.1109/JIOT.2018.2846624.

[6] Z. Wang, C. Liu, J. Qiu, Z. Tian, C., Y. Dong, S. Su Automatically Traceback RDP-based Targeted Ransomware Attacks. Wireless Communications and Mobile Computing. 2018. https://doi.org/10.1155/2018/7943586.

[7] L. Xiao, Y. Li, X. Huang, X. Du, "Cloud-based Malware Detection Game for Mobile Devices with Offloading", IEEE Transactions on Mobile Computing, Volume: 16, Issue: 10, Pages: 2742 – 2750, Oct. 2017. DOI: 10.1109/TMC.2017.2687918.

[8] https://en.wikipedia.org/wiki/Malware_analysis

[9] Z. Tian, W. Shi, Y. Wang, C. Zhu, X. Du, et al., "Real-Time Lateral Movement Detection Based on Evidence Reasoning Network for Edge Computing Environment", IEEE Transactions on Industrial Informatics, Volume: 15, Issue: 7, Page(s): 4285 – 4294, March 2019.

[10]L. Xiao, X. Wan, C. Dai, X. Du, X. Chen, M. Guizani, "Security in mobile edge caching with reinforcement learning", IEEE Wireless Communications Volume: 25, Issue: 3, pp. 116-122, June 2018, DOI: 10.1109/MWC.2018.1700291.

[11]S. Su, Y. Sun, X. Gao, J. Qiu* and Z. Tian*. A Correlation-change based Feature Selection Method for IoT Equipment Anomaly Detection. Applied Sciences.

[12]X. Yu, Z. Tian, J. Qiu, F. Jiang. A Data Leakage Prevention Method Based on the Reduction of Confidential and Context Terms for Smart Mobile Devices. Wireless Communications and Mobile Computing, https://doi.org/10.1155/2018/5823439.

[13]Y. Sun, M. Li, S. Su, Z. Tian, W. Shi, M. Han. Secure Data Sharing Framework via Hierarchical Greedy Embedding in Darknets. ACM/Springer Mobile Networks and Applications.

[14]Y. Wang, Z. Tian, H. Zhang, S. Su and W. Shi. A Privacy Preserving Scheme for Nearest Neighbor Query. Sensors. 2018; 18(8):2440. https://doi.org/10.3390/s18082440.

[15]ABOU-ASSALEH T , CERCONE N , KESELJ V ,et al. N-gram-based detection of new malicious code[C] The 28th Annual Int. Computer Software and Applications Conference (COMPSAC). 2004: 41-42.

[16]Henchiri O, Japkowicz N. A feature selection and evaluation scheme for computer virus detection[C] Data Mining, 2006. ICDM'06. Sixth International Conference on. Hong Kong, Chian IEEE, 2006: 891-895.

[17]Moskovitch R, Feher C, Tzachar N, et al. Unknown malcode detection using opcode

representation[C]//European conference on intelligence and security informatics. Springer, Berlin, Heidelberg, 2008: 204-215.

[18]Y. Ding , X. Yuan , K. Tang, et al. A fast malware detection algo-rithm based on objective-oriented association mining[J]. Computers & Security, 2013,39: 315-324.

[19]T. Chen, C. Guestrin. XGBoost: A Scalable Tree Boosting System[C] KDD '16 Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Pages 785-794.

[20]LightGBM: A Highly Efficient Gradient Boosting Decision Tree[C] Advances in Neural Information Processing Systems 30 (NIPS 2017)
[21]Park N H, Lee W S. Grid-based subspace clustering over data streams [C] //Proc of the ACM Conf on Information and Knowledge Management. New York: ACM, 2007: 801-810