ISSN: 2454-9940

# INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

E-Mail :
editor.ijasem@gmail.com
editor@ijasem.org

www.ijasem.org

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN2454-9940

www.ijasem.org

Vol 18, Issue 1, 2024

# Analyzing Malware in Both Static and Dynamic Environments using Machine Learning

**Mohammad Abdul Waheed Farooqui [1],Konaparthi Chandrakalavathi [2,] Kammara Uma Naga Sree [3],**

**Associate Professor [1,2,]UG Students[3,]**

**Department of CSE**

**BRILLIANT GRAMMAR SCHOOL EDUCATIONAL SOCIETY'S GROUP OF INSTITUTIONS-INTEGRATED CAMPUS Abdullapurmet (V), Hayath Nagar (M), R.R.Dt. Hyderabad.**

## Abstract:

**The ability to identify malware is crucial to the safety of internet-connected computers. Dynamic malware analysis makes advantage of these smashups of characteristics. APIs, summary data, dynamic link libraries, and modified registry keys all contribute to the many permutations that might be produced. For accurate and flexible dynamic malware analysis, researchers turn to the Cuckoo sandbox. Using PEFILE, we can extract over 2300 features from dynamic malware analysis and another 92 features statically from binary malware. Statistics are gleaned from 39000 malicious binaries and 10000 benign files. Cuckoo Sandbox dynamically analyzes 800 benign files and 2200 malicious files, extracting 2300 characteristics. Static malware analysis is 99.36% accurate whereas dynamic malware analysis only has a 94.64% success rate. Malwares are becoming more sophisticated, rendering dynamic malware analysis ineffective. Some restrictions on network activity mean that even with extensive dynamic analysis, we can't get the whole picture.**

## I. INTRODUCTION

Malicious software, or "malware" for short, is any program designed specifically to do damage. Malware's goal is to cause damage to or steal information from a computer system by taking advantage of flaws in the system's defenses.

The number of malicious programs is growing exponentially, and we can divide them up into several classes based on their characteristics and methods of operation. Malicious software may take the form of a script, an executable binary, or any other kind of code. Malware is designed primarily to obtain access to a system, to cause a disruption in service, to cause a denial of service, to steal information, and to destroy data or other resources. Malware is not always a flaw in software; sometimes it is hidden inside otherwise genuine programs. In many cases, legitimate software is really a cover for malicious code. A dangerous program might be downloaded along with the desired software if you try to download it from an entrusted source. Software cracks and illegal downloads are common breeding grounds for malware.

Malwares are not only malicious executable codes, but also PDF and PHP links that take over the system and download even more malware. To install and run on a computer system. There is software that may Take over a system and yet be considered benign since they provide necessary functions. This paper's Goal is to examine executable binaries; 47.80% of malware samples on Virus Total are binaries. Viruses, Trojan horses, adware, worms, and backdoors are just a few examples of malware. We often refer to malware that has traits with more than one group as "generalized malware" since it defies easy categorization.

Malwares are studied using both static and dynamic characteristics. PEFILE can extract over 2300 features through dynamic analysis and another 92 features statically from a binary file. The analysis employs a wide variety of dynamic feature combinations. Registry, dynamic-link libraries (DLLs), application programming interfaces (APIs), and summary data are four types of dynamic features used in malware analysis. To these ever-changing permutations of features, machine learning is applied.

## II. RELATED WORK

A. Statistical Evaluation

Static analysis involves examining the structure of an executable file rather than running it in a sandbox. Different sections and memory compactness are only

two of the numerous static features of the executable file. Data Execution Format PEFILE is a python library that may be used to parse executables for static features.

Analyses of Change (B)

Analyzing malware's activity in a live, controlled environment is what dynamic analysis is all about.

Malware, when launched, modifies a registry entry for harmful purposes and runs in privileged mode. If it switches to privileged mode, it may make sweeping modifications to the operating system. In dynamic analysis, the program is allowed unrestricted use of all available system resources. Software may make changes to the registry and execute in debugger mode while it is being tested in a controlled environment. At the completion of malware execution, the environment is reset to the snapshot that was taken at the beginning of the process. The controlled environment agent records the software's activity.

Each of the three components—host, guest virtual machine, and agent—make up the controlled environment known as the Cuckoo Sandbox. Because of its logging capabilities, Cuckoo is used in this work for dynamic malware investigation. Activities performed on the sample file were recorded by the sandbox agent. As soon as they detect a virtual machine in the system, certain malwares are so sophisticated that they switch from malicious to benign behavior. Agent, debugging mode, and virtual machine detection are the three means by which malware may establish its presence in a sandbox.

Malware was investigated by Clemens Kolb itch at el [1] in a sandbox to extract dynamic system calls. The executable file (.exe) is categorized as malicious or safe based on the system calls it makes. Malware often hides its true nature when subjected to dynamic analysis. The structure of executable code was supplied by David at el [2], who also extracted six crucial static properties. They distinguished malicious files from safe ones based on those six characteristics. Compile time, file info, alignment of sections, image size, file alignment, and header size are the six most crucial aspects. To identify novel PE malware, Wang at el [3] suggested an SVM-based malware detection approach. PE characteristics are retrieved through structural static analysis, and then used to train the SVM classifier. The PE file is categorized as harmful or safe based on the malware SVM's training data.

By concentrating on two dynamic features—function call monitoring and information flow tracking—Wabash Amman 2014 [4] presented dynamic malware analysis. Katarina Chumachenkp [5] used Machine Learning to evaluate and categorize malware based on API Calls and API response codes. We analyzed and categorized Malware from 9 distinct families. Millions of characteristics were extracted from malware using Cuckoo Sandbox, and it took between two and three hours for the system to load and process the data. A static and dynamic integrated technique was employed for malware detection by Igor Santos at el. Malware detection was improved because to the integrated methodology. When looking for malware, they employed a combination of dynamic traces and occurrences of static characteristics.

Bojan Kolosnjaji at el [7] used Deep Neural Network to assess the virus, which is superior to traditional machine learning techniques since it offers the highest accuracy in categorization tasks like NLP and MV. Both a convolutional and a recurrent network layer served as the foundation for this neural network. An innovative approach was developed by Mammon Alatza at el [8], which uses the frequency with which APIs are used to accurately and efficiently identify and categorize zero-day threats. They employed a number of data mining algorithms, and outlined numerous techniques for collecting features from enormous datasets. They were able to compare and contrast the merits of different data mining methods by analyzing their respective performance metrics. Mr. C. Mohammed at No. 9. Using Machine learning and Data mining, they built a comprehensive framework for identifying and categorizing malware to safeguard critical information from harmful actors. They analyzed both anomaly and signature based characteristics to develop an approach that is both powerful and efficient for malware detection and classification.

In [10], A. Kumar of el does a static and dynamic analysis of the malware. As a result of combining static and dynamic analysis, the accuracy of malware detection was improved via the use of machine learning.

## III. STATIC AND DYNAMIC MALWARE ANALYSIS

A. Evolving

The Cuckoo sandbox dynamically analyzes malware and extracts its behaviors in real time. Our primary goal in using a sandbox is to separate our live system from the testing environment and to have access to the data we need from malware activity.

A summary of the malware execution may be seen in the cuckoo report. Each of the report's many chapters

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN2454-9940

www.ijasem.org

Vol 18, Issue 1, 2024

focuses on a different topic. Cuckoo Sandbox report features include the following:
Information Summaries Files API Execution Call Registry Keys Internet Protocol Addresses Domain Name System Queries Access URLs Registry Keys, Part 1
The registry stores settings and configuration data for the operating system and applications. Registry key stores information and configurations in a tree-like structure. It is possible to write to, delete from, open, and read from the registry, among other actions, in the Cuckoo sandbox. Because malware alters several registers to circumvent the protection of window and firewall, registry data may be utilized to identify malware quite successfully. If the registry is modified several times, it is more likely that the executable file is malicious. Malicious and safe files are analyzed for any feature that modifies or deletes the registry. Make the item 1 if any file shares the same registry changes; else, it should be 0. Figure 1 depicts the registry matrix.

| | $Reg_1$ | $Reg_2$ | $Reg_3$ | $Reg_4$ | $Reg_5$ | ..... | $Reg_n$ |
|---|---|---|---|---|---|---|---|
| $Sample_1$ | 0 | 1 | 1 | 0 | 0 | ..... | 1 |
| $Sample_2$ | 0 | 0 | 1 | 0 | 0 | ..... | 1 |
| $Sample_3$ | 0 | 1 | 0 | 0 | 0 | ..... | 1 |
| $Sample_4$ | 0 | 0 | 1 | 0 | 0 | ..... | 1 |
| . | . | . | . | . | | ..... | 0 |
| $Sample_m$ | 1 | 0 | 0 | 0 | 1 | ..... | 0 |

One example of a registry representation in feature data is shown in Figure 1.

2) Files

Information concerning new files, updated files, deleted files, and failed file counts may all be seen in the Cuckoo sandbox report. It is possible to identify lockers and ransom ware by monitoring the production and change of files. When looking for lockers and ransom ware, the file feature is the most important. Instead of utilizing each and every file as a feature, we utilize the features related to the number of times each file was accessed, edited, and deleted. Figure 2: Matrix of Files.

| | Files Opened | File Written | File Copied | File Exist | File Failed | Number Directory Enumerated |
|---|---|---|---|---|---|---|
| $Sample_1$ | 5 | 5 | 10 | 2 | 1 | 3 |
| $Sample_2$ | 1 | 5 | 0 | 20 | 1 | 0 |
| $Sample_3$ | 0 | 5 | 1 | 22 | 1 | 35 |
| | . | . | . | . | . | . |
| | . | . | . | . | . | . |
| $Sample_n$ | 6 | 50 | 1 | 2 | 1 | 13 |

In this report, the file's attributes are shown in Figure 2. Son

Due of their high readability, file data are presented in the summary section.

Execution-Time API Calls a) A collection of subroutine or function calls used for communicating between two software components or between software and hardware components is known as an API (Application Programming Interface). APIs may be based on the operating system, the web, hardware, or a software library. During the execution of a binary file, Cuckoo's sandbox will keep track of any API calls made, providing a summary of those calls along with the corresponding return codes. Figure 3 displays the API success and failure matrix.

| | $API_1$ Success | $API_1$ Fail | $API_1$ Total | $API_2$ Success | $API_2$ Fail | $API_2$ Total | ..... | $API_n$ Success | $API_n$ Fail | $API_n$ Total |
|---|---|---|---|---|---|---|---|---|---|---|
| $Sample_1$ | 3 | 2 | 5 | 6 | 2 | 8 | ..... | 8 | 6 | 14 |
| $Sample_2$ | 2 | 1 | 3 | 8 | 16 | 24 | ..... | 4 | 17 | 21 |
| $Sample_3$ | 3 | 4 | 7 | 6 | 16 | 22 | ..... | 0 | 07 | 07 |
| | . | . | . | . | . | . | | | . | |
| | . | . | . | . | . | . | | | | |
| $Sample_n$ | 11 | 4 | 15 | 0 | 18 | 18 | ..... | 8 | 6 | 14 |

APIs Called during Sample Execution, Third Diagram in Cuckoo's Reports, and a special collection of API calls is recorded. The API call count in son format is sent together with the API's return code. Status of APIs is indicated by their

return codes; a return code of 0 indicates a successful API execution, whereas a non-zero value indicates an API failure.

Thirdly, Domain Name System (DNS) and Internet Protocol Security (IPS) requests The pap file created by Cuckoo's sandbox may be analyzed to learn more about the network activity that Malware initiated while it was running. This activity includes IP addresses and DNS requests. Various programs, such as Silk, can parse pap files and retrieve IP addresses and DNS requests. Most malware is designed to make several connections and do many DNS lookups. In the execution summary, characteristics such as the total number of DNS queries and the total number of IP addresses reached per sample are retrieved. Everything from the number of times the registry was modified, deleted, accessed, opened, and generated to the total number of file modifications is recorded in the summary data. Information such as the total number of IP addresses and visited URLs is also included.

The Integration of Multiple Features

Malware analysis makes use of a wide variety of dynamic feature combinations. Registry, Dynamic Link Libraries, Application Programming Interfaces, and Summaries are examples of dynamic features. Table 1 lists the many permutations of dynamic analysis.

TABLE I. DYNAMIC FEATURES COMBINATIONS

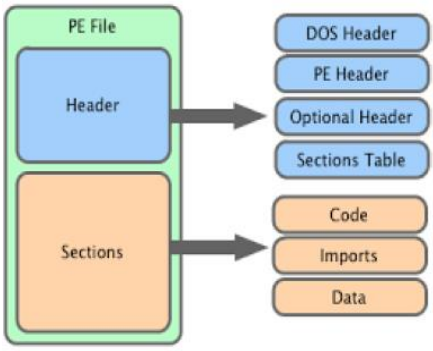
| *S. No* | *Combinations* |
|---|---|
| *1* | *APIs+DLLs* |
| *2* | *APIs+Summary information* |
| *3* | *DLLs+Registry* |
| *4* | *DLLs+Summary information* |
| *5* | *Registry* |
| *6* | *DLLs* |
| *7* | *Registry + summary information +DLLs+APIs* |
| *8* | *Registry+summary information* |
| *9* | *APIs Calls* |

Part B: Static Analysis

Without actually running the executable file, static analysis may extract its static properties using the python PEFILE library. It gathers information from the document's "header," "optional header," and other sections Microsoft Windows executables and Dynamic Link Libraries (DLLs) employ a file format called Portable Executable (PE). When loaded into a window, DLLs contain details about how to link and execute programs. It is also utilized to examine the libraries that were imported and the linking methods that were used in the running of the executable.

A PE file has a header and different sections. The header also includes groups for things like DOS, PE, optional features, and a table of contents.

Sections include options like Code, Imports, and Data. Figure 4 depicts the basic layout of a PE file.




File Extension (PE) and Section 1) File Header (Figure 4)
The 9 retrieved characteristics from the file header provide important information about the PE file. Second, a potential header, if desired. The optional header and sections provide a total of 4characteristics for PE analysis. Three Parts PE file analysis makes use of derived features, such as SectionsMeanEntropy, SectionsMinEntropy, SectionsMaxEntropy, and Characteristics Mean, Characteristics in, Characteristics ax, Miscreant, Miscuing, Micmac, and Num Suspicious Sections.
. Dos Header
Dos Header is mined for a total of 17 properties, 17 of which are fundamental and 2 of which are derived. E res2 and e res length are the derived features.
The TimeDateStamp, NumberOfNamedEntries, and NumberOfIdEntries characteristics are taken from the Directory Entry Resources Configuration.
Adjusting the Loading of Directory Entries (Section

INTERNATIONAL JOURNAL OF APPLIED SCIENCE ENGINEERING AND MANAGEMENT

ISSN2454-9940

www.ijasem.org

Vol 18, Issue 1, 2024

Twenty fundamental properties are gleaned from the Directory using the Directory Entry Load setting for PE file analysis.

Crucial Characteristics Static features such as Machine, Characteristics, Number of Sections, TimeDateStamp,PointerToSymbolTable, NumberOfSymbols, Magic, MajorLinkerVersion, MinorLinkerVersion,SizeOfCode,SizeOfInitializedData, SizeOfUninitializedData, Checksum, BaseOfCode, Image Base, MajorSubsystemVersion, Subsystem, DllCharacteristics, SizeOfStackReserve, Major

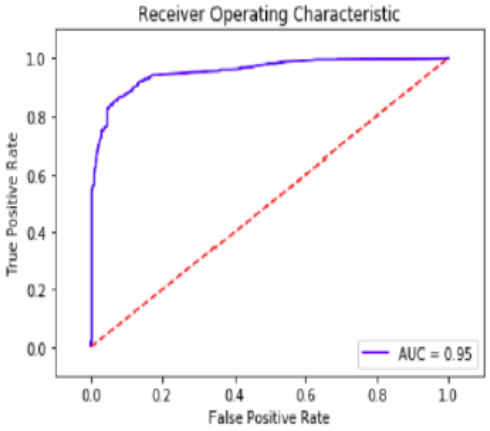
## IV. RESULTS

Insights that Change Over Time

The Registry, APIs, DLLs, and summaries have all been subjected to various machine learning approaches for real-time analysis. TABLE II displays the outcomes of a dynamic malware analysis. Figure 5 displays the ROC curve for space-constrained dynamic analysis using the Gradient Boosting Algorithm...

***TABLE II DYNAMIC RESULTS***

| Algorithm | % | | AUC % |
|---|---|---|---|
| *Logistic Regression* | FP | *11.07* | 87.44 |
| | FN | *13.64* | |
| *Decision Tree* | FP | 13.81 | 86.99 |
| | FN | 12.19 | |
| *Random Forest* | FP | 11.47 | 85.67 |
| | FN | 17.17 | |
| *Bagging Classifier* | FP | 11.70 | 87.72 |
| | FN | 12.89 | |
| *AdaBoost Classifier* | FP | 11.94 | 93.84 |
| | FN | 10.91 | |
| *Tree Classifier* | FP | 9.60 | 86.61 |
| | FN | 17.18 | |
| *Gradient Classifier* | FP | 5.85 | 94.64 |
| | FN | 13.96 | |




5. Gradient Boosting ROC Graph

In TALBE III, we see the outcomes of several dynamic feature combinations.

The results of combining the dynamic features are shown in Table III.

| Combination | Algorithm | AUC % |
|---|---|---|
| APIs+DLLs | AdaBoost Classifier | 84.60 |
| APIs+Summary information | Gradient B. Classifier | 95.86 |
| Registry | Gradient B. Classifier | 86.10 |
| DLLs+Summary information | Gradient B Classifier | 94.84 |
| Registry+APIs | Gradient B Classifier | 84.87 |
| DLLs | Logistic regression | 80.42 |
| Registry+summary information | Gradient B. Classifier | 94.64 |
| APIs Calls | Gradient B Classifier | 94.44 |

B. Outcomes from Non-Moving Features

Machine learning is used to more than 92 static characteristics retrieved from executable files. TABLE IV displays the outcomes of the static characteristics analysis. Figure 6 illustrates the ROC Curve generated by the gradient Boosting technique.

TABLE IV. STATIC FEATURES RESULTS

| Algorithm | % | | AUC % |
|---|---|---|---|
| *Linear Ridge Classifier* | FP | *10.56* | 94.91 |
| | FN | *6.67* | |
| *Decision Tree* | FP | 5.27 | 96.92 |
| | FN | .665 | |
| *Random Forest* | FP | 3.09 | 97.36 |
| | FN | 2.17 | |
| *Bagging Classifier* | FP | 4.75 | 97.32 |
| | FN | 0.60 | |
| *AdaBoost Classifier* | FP | 4.92 | 99.20 |
| | FN | 3.65 | |
| *Tree Classifier* | FP | 3.25 | 97.55 |
| | FN | 1.62 | |
| *Gradient Classifier* | FP | 3.25 | 99.36 |
| | FN | 2.73 | |

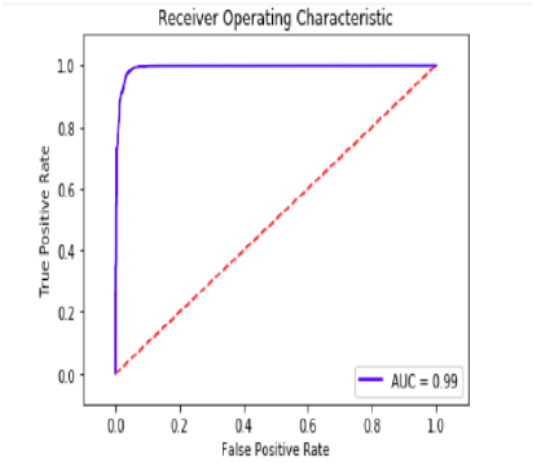



Figure 6. Static ROC graph using Gradient Boosting Classifier

## V. CONCLUSION

Intelligent malware behaviors reduce the efficacy of dynamic malware analysis. Limitations on dynamic analysis are introduced by the fact that restricted network access prevents the examination of free-flowing network activity. Malware is notoriously difficult to analyze in a controlled setting, since its presence may be readily detected even in virtualized or debugging mode. Since the virtual system leaves behind easily-detectable traces, it is not as efficient as the genuine thing. This devious virus uses application programming interfaces (APIs) to identify a virtualized host. The IsDebuggerPresent and GetAdapterAddress APIs may tell whether a machine is running in a virtualized setting. Malware may access processes in progress and look for Agent.pyw, a python agent for the cuckoo sandbox.

Static malware analysis is superior to dynamic analysis with an AUC (Area under Curve) of 99.36%. Because malware often comes in compressed forms, static analysis has various restrictions.

## REFERENCES

[1] Kolbitsch, C., Comparetti, P. M., Krueger, C., Kilda, E., Zhou, X. Y., & Wang, X. (2009, August). Effective and Efficient Malware Detection at the End Host. In USENIX security symposium (Vol. 4, No. 1, pp. 351-366).

[2] David, B., Filial, E., & Galliano, K. (2017). Structural analysis of binary executable headers for malware detection optimization. Journal of Computer Virology and Hacking Techniques, 13(2), 87-93.

[3] Wang, T. Y., Wu, C. H., & Hsieh, C. C. (2009, August). Detecting unknown malicious executables using portable executable headers. In INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on (pp. 278-284). IEEE.

[4] Amman, W. (2014). A framework for analysis and comparison of dynamic malware analysis tools. Arrive preprint arXiv: 1410.2131.

[5] Chumachenko, K. (2017). Machine Learning Methods for Malware Detection and Classification.

[6] Santos, I., Devesa, J., Brezo, F., Nieves, J., & Bringas, P. G. (2013). Opem: A static-dynamic approach for machine-learning-based malware detection. In International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions (pp. 271-280). Springer, Berlin, Heidelberg.

[7] Kolosnjaji, B., Zarras, A., Webster, G., & Eckert, C. (2016, December). Deep learning for classification of malware system call sequences. In Australasian Joint Conference on Artificial Intelligence (pp. 137-149). Springer, Cham.

[8] Alatza, M., Venkatraman, S., Watters, P., & Alatza, M. (2011, December). Zero-day malware detection based on supervised learning algorithms of API call signatures. In Proceedings of the Ninth Australasian Data Mining Conference-Volume 121(pp. 171-182). Australian Computer Society, Inc.

[9] Chowdhury, M., Raman, A., & Islam, R. (2017, June). Malware analysis and detection using data mining and machine learning classification. In International Conference on Applications and Techniques in Cyber Security and Intelligence (pp. 266-274).

Edition Della Normal, Cham.

[10] Jain, A., & Singh, A. K. (2017, August). Integrated Malware

Analysis using machine learning. In 2017 2nd International

Conference on Telecommunication and Networks (TEL-NET) (pp. 1-

8). IEEE.