Static and Dynamic Malware Analysis Using Machine Learning

Conference Paper · January 2019

DOI: 10.1109/IBCAST.2019.8667136

3 authors, including:

Muhammad Ijaz
Pakistan Institute of Engineering and Applied Sciences
3 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Hanif Durad
Pakistan Institute of Engineering and Applied Sciences
35 PUBLICATIONS **117** CITATIONS
SEE PROFILE

Static and Dynamic Malware Analysis Using Machine Learning

Muhammad Ijaz[1], Muhammad Hanif Durad[2], Maliha Ismail[3]

Department of Computer and Information Science
Pakistan Institute of Engineering and Applied Sciences
Islamabad, Pakistan
[1]muhammadijazpk1@hotmail.com, [2]hanif@pieas.edu.pk

CESAT
The National Centre for Physics
Islamabad, Pakistan
[3]maliha.ismaeel@gmail.com

***Abstract*—Malware detection is an indispensable factor in security of internet oriented machines. The combinations of different features are used for dynamic malware analysis. The different combinations are generated from APIs, Summary Information, DLLs and Registry Keys Changed. Cuckoo sandbox is used for dynamic malware analysis, which is customizable, and provide good accuracy. More than 2300 features are extracted from dynamic analysis of malware and 92 features are extracted statically from binary malware using PEFILE. Static features are extracted from 39000 malicious binaries and 10000 benign files. Dynamically 800 benign files and 2200 malware files are analyzed in Cuckoo Sandbox and 2300 features are extracted. The accuracy of dynamic malware analysis is 94.64% while static analysis accuracy is 99.36%. The dynamic malware analysis is not effective due to tricky and intelligent behaviours of malwares. The dynamic analysis has some limitations due to controlled network behavior and it cannot be analyzed completely due to limited access of network**.

Keywords— Malwares, Cuckoo Sandbox, binary, static Analysis, Dynamic Analysis, Machine Learning, obfuscate, evade

I. INTRODUCTION

Malware is the shortest term used for malicious software which is a harmful malicious piece of code. Malware's intention is to harm computer or steal information from system by exploiting vulnerabilities in existing security infrastructure.

Malwares are rapidly increasing with the passage of time and we can categorize malware in to different categories according to their behaviors. The malware can be a script, executable binary or any other piece of code, which have malicious intention. The main aims of malware are to gain access of system, disrupt system services, denial of service, steal confidential information and destruction of resources. Sometime malware is not a defective software but some legitimate software can have malware inside it. Legitimate software often acts as wrapper for malware. Downloading legitimate software from any website may download malicious software itself. Mostly malwares are found in cracked software and pirated software.

Malwares are not only executable codes but sometimes they act as downloader for malware e.g. PDF and PHP link which gains control of system and download more malicious softwares to execute on system. Some softwares gain control of system and do some legitimate work so we cannot classify them malicious.

According to VirusTotal 47.80% of malwares are executable files, so aim of this paper is to analyze the executable binaries. There are many types of malwares, which can be classified into Virus, Trojan horse, Adware, Worm and Backdoor. Some of malwares cannot be classified into one category, because malwares have multiple characteristics which organize them in multiple categories and sometime we called them generalize malware.

Malwares are analyzed on basis of static as well as dynamic features. More than 2300 features are extracted from dynamic analysis and 92 features are extracted statically from binary file using PEFILE. Different dynamic features combinations are used for analysis. Four types of dynamic features are used for malware analysis which are Registry, DLLs, APIs and summary information. Machine learning is applied on these dynamic features combinations.

II. RELATED WORK

A. Static Analysis

In static analysis, the executable file is analyzed on structure bases without executing it in controlled environment. The executable file has many static attributes like different sections and memory compactness. Portable Executable PEFILE is a python library which extracts static features from executable files.

B. Dynamic Analysis

In dynamic analysis, malware behavior is analyzed in dynamic contolled enviroment.When the malware executes, it changes the registry key maliciously and takes the privileged mode of operating system. In case when it changes itself to previliged mode, it can change everything of operating system. In dynamic analysis, the software has full access of all the resources to execute in controlled environment. In controlled enviroment software can change registry keys of computer and run in debugger mode. At the end of malware execution, the dynamic enviroment reverts back to its previous snapshot,

which is created at the start of enviroment setup. The agent in controlled environment logs the behaviour of software.
Cuckoo Sandbox is a controlled enviroment which is consisted of three parts one is host, the other is guest virtual machine and third one is agent. In this paper, cuckoo is used for dynamic malware analysis due to logging functionality. The agent of sandbox logged the sample file activities.
Some of malwares are so intelligent when they find any virtual machine in system they change their behaviour from malware to bengin. Malware can detect the controlled enviroment in three ways: by agent, debugging mode and virtual machine
Clemen Kolbitch at el [1] analyzed the malware in controlled environment to extract dynamic system calls. On the basis of system call, the exe file is classified into malware and benign. The malware often obfuscate its behavior in dynamic analysis.
David at el [2] provided the structure of executable code and extracted six important static features. On basis of these six features they classified file into malware and benign. The six important features are CompilationTime, FileInfo, SectionAlignment, SizeOfImage, FileAlignment and SizeOfHeader.
Wang at el [3] proposed a SVM based malware detection method for detecting new PE malware. Using structure static analysis, PE features are extracted and the SVM classifier is trained using the selected features. On basis of trained malware SVM, the PE file is classified into benign and malicious.
Waqas Aman 2014 [4] proposed dynamic malware analysis by focusing two dynamic features: function call monitoring and information flow tracking.
Kateryna Chumachenkp [5] analyzed and classified malware using APIs Calls and return codes of APIs using Machine Learning. Used 9 different families of Malware for analysis and classified them with best Accuracy. The features which were extracted from malware using Cuckoo Sandbox were in millions and 2 to 3 hours are required to load into system and produce result.
Igor Santos at el used [6] static and dynamic integrated approach for malware detection. The integrated approach enhanced the capability of malware detection. They used static features occurances and dynamic traces for malware detection.
Bojan Kolosnjaji at el [7] analyzed the malware using Deep Neural Network, which surpass the limitation of machine learning and provide best accuracy in classification such Natural Language processing and Machine Vision. The neural network was based on convolution and recurrent network layer.
Mamoun Alazab at el [8] proposed a novel method and evaluated techniques to detect and classify zero-day attacks with accuracy and efficiency based on number of times APIs are called. They described different methods for collection of features from large datasets and used various data mining algorithms. Through the performance results of these algorithms, they were able to evaluate and discuss the advantages of one data mining algorithm over the other.
Mozammel C at el [9] They provided a comprehensive framework to classify and detect malware to protect important data against malicious threats using Machine learning and Data mining techniques. They proposed a robust and efficient techniques for malware classification and detection by analyzing both anomaly-based and signature based features.
A. Kumar at el [10] analyzed the malware using both static and dynamic methods. They used an integrated approach of static and dynamic analysis, which enhanced the accuracy of malware detection using machine learning

III. STATIC AND DYNAMIC MALWARE ANALYSIS

A. Dynamic

Cuckoo sandbox is used for dynamic analysis of malwares and extracts their behaviours at run time during execution. Our main aim is to use sandbox to isolate our actual system from testing environment and extract desired information from malware execution.
The cuckoo report provides a summarized information about the malware execution. The report has multiple sections and each section deals with different information. The features which are extracted from Cuckoo Sandbox report are

- Summary information
- Files
- API call during Execution
- Registry Keys
- IP address and DNS queries
- Access URLs

1) Registry Keys

The registry contains information about installed software, hardware device, values and options used by different processes. Registry key is a hierarchical database of setting and information.
The Cuckoo sandbox stores information about registry changed and accessed like registry written, registry deleted, opened and read. Registry information can be used to detect malware very effectively because malware changes multiples registries to bypass the security of window and firewall.
Multiple registry changed have greater possibility that executed file is malicious. All the registry deleted and registry written features are extracted from both malware and benign files. If any file has same registry changed make the entry 1 else the entry should be zero by default. Registry matrix is shown in figure 1.

	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
$Sample_n$	1	0	0	0	1	.....	0

Figure 1. Registry representation in features data

2) Files

The Cuckoo sandbox report contains information about the created file, modified files, deleted file and number of failed files. File creation and modification can be used as a good source of locker and ransomware detection. The file feature is most interested feature for ransomware and locker detection. The number of file accessed, number of file modified and number of file deleted features are used instead of using every file as a feature. Files matrix is shown in figure 2.

	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

Figure 2. File Attributes in Report.json

File data are included in summary section due to easy readability.

a) API calls During Execution

API (Application Programing interface) is a set of subroutine or functions call which are used for communication between two software components or communication between software and hardware components. API can be operating system based, web based hardware based and software library based. Cuckoo sandbox records APIs calls during execution of binary file and provide APIs calls in summarized form with return codes of APIs. The APIs fail and success matrix is shown in figure 3.

	API_1_Success	API_1_Fail	API_1Total	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

Figure 3. APIs call during Execution of sample

The Cuckoo sandbox logs unique set of APIs in Reports.json, number of times API is called and also provides the return code of API. The return codes show status of APIs, return code 0 means API is executed successfully and non-zero means API is failed during execution.

3) IPS and DNS queries

Cuckoo sandbox logs pcap file for analysis of network traffic, which contains IPs addresses, and DNS queries which were performed by Malware during execution. IPs addresses and DNS queries can be extracted from pcap file using different software e.g. SiLK. Mostly the malwares want to connect with more than one IPs and do multiple DNS queries. The number of IPs accessed by sample during execution and numbers of DNS queries are extracted as features in summary information section.

4) Summary information

The summary information contains all the information about files changed, deleted, created, enumerated and number of registry changed, deleted, accessed, opened and created. It contains all other information like number of IPs and number of URLs accessed.

a) Features Combinations

Different dynamic features combinations are used for malware analysis. The dynamic features are Registry, DLLs, APIs and summary information. The different combinations of dynamic analysis are given in TABLE 1.

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*

B. Static Analysis

In static analysis, the static features are extracted from executable files using python PEFILE library without executing it in controlled environment. It extracts many features from header, optional header and different sections.

Portable Executable (PE) format is used by Microsoft windows executables and Dynamic Link Libraries DLLs. DLLs provide linkage and execution information of code when load into window. It is also used for analysis of imported libraries and the types of linkage are used in execution of executable file.

PE file contains header and sections. Header has many other features groups like DOS, PE, Optional and Sections Table.

Sections part has features like Code, Imports and Data. PE file structure is shown in figure 4.

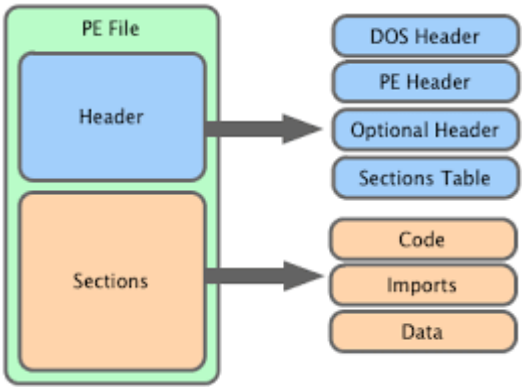

Figure 4. PE File Header and Section

1) File Header

File header gives useful information about PE file, 9 features are extracted from the header.

2) Optional Header

Total 40 features are extracted from Optional header and sections for PE analysis.

3) Sections

Derived features of sections are used in PE file analysis, which are *SectionsMeanEntropy, SectionsMinEntropy, SectionsMaxEntropy, CharacteristicsMean, CharacteristicsMin, CharacteristicsMax, MiscMean, MiscMin, MiscMax, Num_Suspecious_Sections.*

4) Dos Header

Basic 17 features are extracted from Dos Header and two are derived features. The Derived features are length of *e_res2*, *e_res.*

5) Directory Entry Resources

Characteristics, TimeDateStamp, NumberOfNamedEntries, and NumberOfIdEntries features are extracted from Directory Entry Resources Configuration.

6) Directory Entry Load Configuration

Directory Entry Load configuration are used for PE file analysis and 20 basic features are extracted from this Directory.

C. Important Features

Important static features used for static analysis are *Machine, Characteristics, Number of Sections, TimeDateStamp, PointerToSymbolTable, NumberOfSymbols, Magic, MajorLinkerVersion, MinorLinkerVersion, SizeOfCode, SizeOfInitializedData, SizeOfUninitializedData,* CheckSum, BaseOfCode, ImageBase, MajorsubsystemVersion, Subsystem, DllCharacteristics, SizeOfStackReserve, MajorOperatingSystemVersion, Num_Suspecious_Sections, MinorOperatingSystemVersion, SizeOfOptionalHeader, e_lfanew, CharacteristicsMax, SectionsMaxEntropy, SecurityCookie, FileAlignment, SectionsMeanEntropy, *SectionsMinEntropy*.

IV. Results

A. Dynamic Results

A number of machine learning techniques has been applied to Registry, APIs, DLLs and summary information for dynamic analysis. The results of dynamic malware analysis are given in TABLE II. Keeping in view space limitations the ROC curve for dynamic analysis using Gradient Boosting Algorithm is given in figure 5.

.

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	

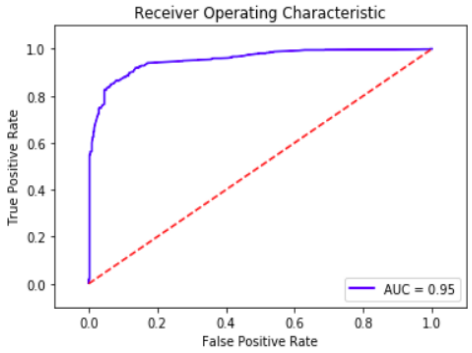

Figure 5. ROC graph using Gradient Boosting

The different combinations results of dynamic features are given in TALBE III.

TABLE III. DYNAMIC FEATURES COMBINATION RESULTS

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. *Static Features Results*

More than 92 static features are extracted from executable files and machine learning is applied on static features. The results of static features are given in TABLE IV. The ROC Curve is shown in figure 6 using gradient Boosting algorithm.

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	

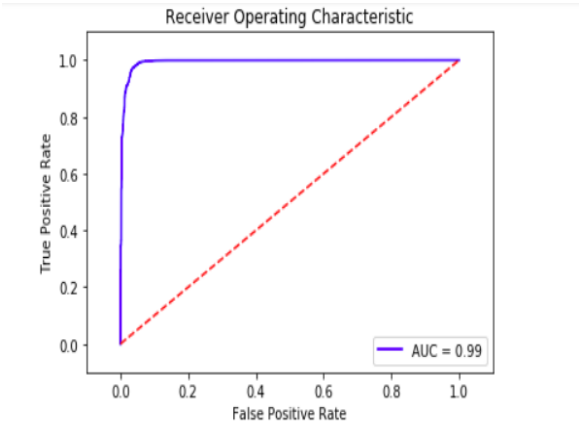

Figure 6. Static ROC graph using Gradient Boosting Classifier

V. CONCLUSION

The accuracy of dynamic malware analysis is not efficient due to intelligent behaviours of malwares. The dynamic analysis has some limitations due to controlled network behavior and it cannot be analyzed due to limited access of network. Controlled environment for malware analysis is not much useful due to tricky nature of malware, the virtualized and debugging mode are quickly detectable by malware. The virtualized environment is not much effective as real system due to many traces, which are easily detectable. The tricky malware executes APIs to detect virtual environment. The virtualized environment is detected by *IsDebuggerPresents* and *GetAdapterAddress* APIs. Sometime malwares access running process and search for the *Agent.pyw which* is python agent for cuckoo sandbox.

The AUC (Area under Curve) of static malware analysis is 99.36% which is better than dynamic analysis. Static analysis has some limitation due to packed nature of malware.

VI. FUTURE WORK

Malwares have intelligent and tricky nature and they can detect dynamic malware analysis very quickly, therefore we need a dynamic analysis controlled environment, which is not detectable. In current era, malwares are packed in nature and it is often able to analyze statically. We need a system, which is initially dynamic and when malware is unpacked then apply the static features extraction on it.

A. *Deep Neural Network*

Machine learning has some limitations like massive store of data, labelling of trained data, bias in train data and algorithm does not collaborate with themselves. These limitations will be overridden by DNN.

B. *Using a Larger Dataset*

The Dataset of malware analysis was small. The larger dataset provides best result due to deep machine learning. The result will be reliable every time due to deep learning.

C. *Static Analysis using Dynamic*

Due to obfuscated and packed nature of malware, the static analysis is not so effective. When malware executes in dynamic environment, it changes its behaviors, so static features can be extracted easily and correctly. The static features extraction in dynamic environment will be very efficient to detect malware.

ACKNOWLEDGMENT

Special thanks to PIEAS ICT Endowment Fund for sponsoring my studies by provision of scholarship and DCIS Grid Lab to provide me a suitable place for research.

REFERENCES

[1] Kolbitsch, C., Comparetti, P. M., Kruegel, C., Kirda, 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., Filiol, E., & Gallienne, 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] Aman, W. (2014). A framework for analysis and comparison of dynamic malware analysis tools. *arXiv 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] Alazab, M., Venkatraman, S., Watters, P., & Alazab, 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., Rahman, 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). Edizioni della Normale, 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.