British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 7, Issue 4, 2024 (pp. 8-26)
www.abjournals.org

CUCKOO SANDBOX AND PROCESS MONITOR (PROCMON) PERFORMANCE EVALUATION IN LARGE-SCALE MALWARE DETECTION AND ANALYSIS

Umoh Enoima Essien[1] and Sylvester I. Ele[2*]

[1]Department of Computer Science, University of Cross River State.
Email: enoimaumoh@unicross.edu.ng.

[2]Department of Computer Science, University of Calabar.
Email: elesly@unical.edu.ng

[*]Corresponding Author's Email: elesly@unical.edu.ng

Cite this article:

Essien, U. E., Ele, S. I. (2024), Cuckoo Sandbox and Process Monitor (Procmon) Performance Evaluation in Large-Scale Malware Detection and Analysis. British Journal of Computer, Networking and Information Technology 7(4), 8-26. DOI: 10.52589/BJCNIT-FCEDOOMY

Manuscript History

Received: 13 Jul 2024

Accepted: 24 Sep 2024

Published: 4 Oct 2024

Copyright © 2024 The Author(s). This is an Open Access article distributed under the terms of Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0), which permits anyone to share, use, reproduce and redistribute in any medium, provided the original author and source are credited.

ABSTRACT: *Malware has grown to be an intricate and dynamic threat to cybersecurity. Researchers and cybersecurity specialists use a range of methods to analyze and comprehend malware in order to effectively counter this threat. The malware sandbox is one of the most crucial instruments in this battle. Insights gained by evaluating malware in a sandbox aid in the creation of effective detection. Finding a sandbox that is both highly precise, efficient and affordable is a challenging task. This study compares the effectiveness of Cuckoo Sandbox and Procmon, two of the most popular sandboxes, in the efficient implementation of malware analysis and detection. A Windows 10 Pro window-based computer with a 4 GHz CPU, 16 GB RAM, 8 cores, and a 320 GB hard drive (HDD) was set up. An Oracle virtual machine (VM) for guests was set up and launched. Using the Oracle VM, a virtual operating system (Windows 10 Pro). Furthermore, Yara-Python was deployed and JSON reports, a system built on Python was created. The results show that Cuckoo consistently outperforms Procmon in terms of execution time, completing much more quickly and steadily over each of the ten process runs. Procmon has significantly longer and more fluctuating execution times, peaking at 989 seconds, while Cuckoo maintains execution durations around 530 seconds, suggesting superior efficiency and consistency. Six (6) machine learning-based methods for classifying and detecting malware that used Cuckoo sandbox and process monitor were surveyed. Different performance indicators were found in the six-machine learning-based malware detection and classification studies that Process Monitor was used to survey. A review of six machine learning-based malware detection and classification studies using both Process Monitor and Cuckoo Sandbox indicated that Cuckoo Sandbox consistently delivered better performance. The findings show that machine learning-based malware detection conducted with Cuckoo attained a higher average accuracy of 99.35% compared to 94.48% with Procmon, along with a superior ROC value of 0.97 (97%) versus 0.91 (91%) for Procmon.*

KEYWORDS: Cuckoo sandbox process monitor; Sandboxing; Malware analysis.

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

INTRODUCTION

Malware has developed into a complex and ever evolving threat to cybersecurity. To effectively combat this threat, researchers and cybersecurity experts apply a variety of techniques to examine and understand malware. In this fight, the malware sandbox is one of the most important tools (Raj et al., 2024). Sandboxing is a cybersecurity technique that involves running, observing, and analyzing code within a secure, isolated network environment that mimics an end-user operating system. This method is designed to prevent potential threats from infiltrating the network and is often used to evaluate untested or untrusted code. By constraining the code to a controlled test environment, sandboxing contains any harmful actions or infections, protecting the host machine and operating system from damage (). In order to isolate suspicious files or programs in a secure, segregated environment and prevent them from compromising real systems, malware analysis in a sandbox is performed. Static analysis is usually used to examine file attributes, and dynamic analysis is used to observe behavior while the process is running. To accomplish this, it must be suspended using Docker in a virtual environment where it cannot damage the system (Sinha & Sai, 2023). Sandbox environments capture system calls, network traffic, and modifications to the file system and registry, providing vital information on malware functionalities (Ijaz, Durad & Ismail, 2019). Malware features extracted from the analysis reports (AR) generated by the sandbox captures the behavioral data of each sample. A thorough understanding of the sandbox's functionality and the structure of its reports is essential (Chumachenko, 2017). There are several malware analysis tools and platforms the cybersecurity community have developed to help curb the persistent surge of cyber threats by enabling security analysts to gather and analyze malware samples, unravel their capabilities, and guide investigations. They include Joe Sandbox, Procmon or Process Monitor (windows Sysinternals), Process Explorer (windows Sysinternals), Autoruns (windows Sysinternals), Anubis, CWSandbox, ProDot, x64dbg, Process Hacker, Pestudio, Ghidra, and Fiddler, among others. (Fox, 2021). Of all these sandboxing tools, Cuckoo sandbox and Procmon (process monitor) from Windows Sysinternals are open source and are extensively deployed by malware analysts. The aim of this paper is to evaluate the performance of Cuckoo Sandbox with Procmon in effective implementation of malware analysis and detection.

REVIEW OF RELATED LITERATURE

Malware Analysis

The goal of malware analysis is to investigate the characteristics and features of malicious software, so as to help malware analysts understand better the nature of intended malware attacks and provide protection against future attacks (Mills & Leggs, 2020). According to Alam et al. (2015), there are three approaches in malware detection, known as static analysis, which scrutinize and examine software source code (Egele et al., 2008), and dynamic analysis, which examines system behavior during the execution of malware (Or-Meir et al., 2019), and the third being hybrid analysis. The malware sample is not run during static analysis, whereas during dynamic analysis the malware sample runs (Bragen, 2015).

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

a. **Dynamic Analysis**: Dynamic analysis of malware is achieved by examining the activities and dealings of a program while the program is executing in a secured environment. The indicator, functionality, and behaviors of the program that enable one to determine if a program is benign or malicious, can be identified during dynamic analysis (Pektas, 2015). The procedure for dynamic analysis includes three steps: to set up and cleanse an environment; to run the malware; and to examine and document (log) malware behavior. The log or documented files supply factual information concerning files that are opened, accessed, or called (Arends & Kerstin, 2018). One of the approaches of dynamic analysis is the automatic dynamic analysis, in which the program behavior is traced by using classical tools like Cuckoo and CWSandbox (Cuckoo, 2019; Kim, 2018). These tools take as input, a file, and afterward execute the file using a virtual environment.

b. **Static Analysis**: The static analysis technique refers to the analysis of the PE files (Portable Executable files) without running the code. Binary packers, like ASP Pack Shell and UPX are generally used by malware, in order to keep away from being analyzed (Zaki & Humphrey, 2014). A Portable Executable file has to be unpacked prior to being analyzed. In static analysis, nearly every program employed the Windows API calls to interact with the operating system. A typical example is the OpenFileW, which is a Windows API in Kernel32.dll that generates a novel file or opens an existing file. So, API calls expose the behavior of programs and can be regarded as a crucial mark in malware detection. For example, Windows API calls such as "LoadLibrary" "WriteProcessMemory", and "CreateRemoteThread" are assumed behavior manifested by malware for DLL injection into a process, whereas hardly ever come together in a genuine setting.

1. Cuckoo Sandbox

It is common knowledge that to engage the machine learning algorithms in any problem, it is critical to represent the data in some form. Consequently, Cuckoo Sandbox was adopted. Malware features extracted from the analysis reports (AR) generated by the sandbox, which describes the behavioral data of every sample, were preprocessed. Understanding the functionality of the sandbox and the report's structure is very essential (Chumachenko, 2017). Cuckoo is one of the most important sandbox environments for malware analysis. Cuckoo Sandbox is an open-source malware analysis sandbox that generates a comprehensive and exhaustive report of the behavior of files within seconds. Per Cuckoo Foundation (2015) supports different categories of file formats, including DLL files, URLs and HTML files, PDF, ZIP files, Microsoft Office documents, CPL files, Visual Basic (VB) scripts, python files, Macro enabled files, Java JAR and PHP scripts. The Cuckoo Sandbox provides platforms for automated analysis of chary files (Cuckoo Foundation, 2015). Cuckoo sandbox has an extremely modifiable flexible architecture, permitting the sandbox to be utilized as a standalone application and can also be embedded into larger frameworks. Figure 1 is the architecture of the Cuckoo sandbox, showing the cuckoo host, Analysis Guests, Analysis VM, n (n = 1, 2, 3), and internet or sinkhole.

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 7, Issue 4, 2024 (pp. 8-26)
www.abjournals.org

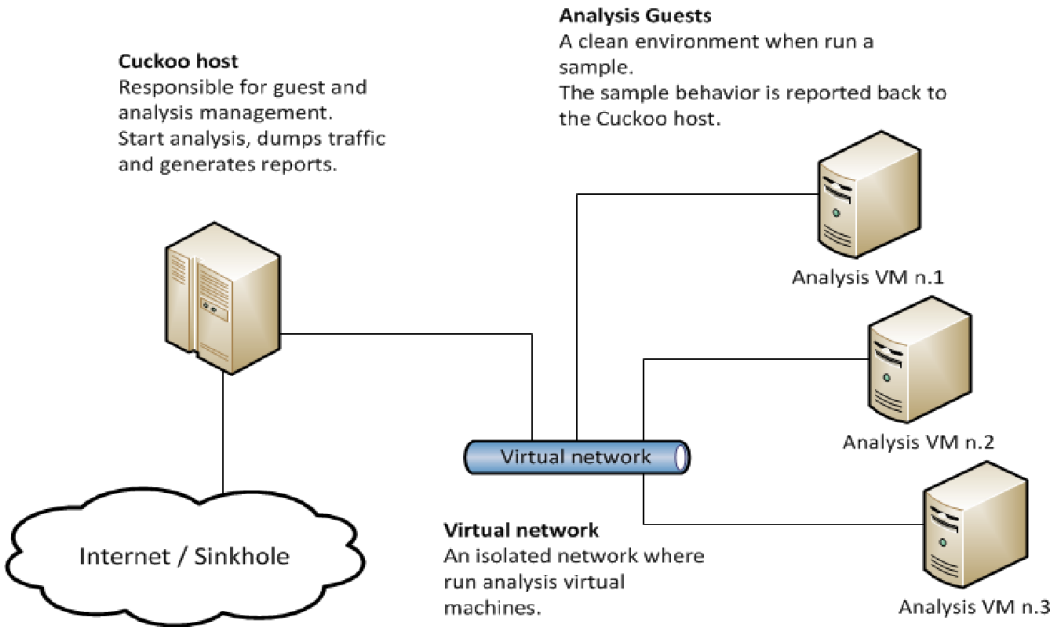

Figure 1: Cuckoo Sandbox Architecture (Cuckoo Foundation, 2015)

After generating the behavior of the file, the Sandbox makes a decision on the severity of maleficence of the file using certain existing signatures. The reports generated from the sandbox are represented as a JSON file. Cuckoo Sandbox affords analysts the opportunity of automating the analysis of suspicious samples. With dynamic (behavioral) analysis, hash comparisons, and other unified tools, identification of malware and determination of the kind of indicators to look for in their production environment to determine the presence of malevolent actions on a system becomes possible. Cuckoo, as an automated malware analysis tool, provides data concerning the behavior of malicious files; however, it does not actively classify malware using these observations. Alternatively, Cuckoo can be set up to allow files to be submitted to VirusTotal (Walker & Sengupta, 2020). Cuckoo sandbox contains built-in logging and monitoring during a malware sample analysis. In the cuckoo setup, as seen in figure 12 above, the cuckoo generates its peculiar network that links the host machine with the virtual machines from which malware analysis is performed. Executing virtual machines in a separate network permits cuckoo to execute malware safely and analyze it without the fear of malware infecting the machine that is external to the isolated virtual machine (Muhovic, 2020). Through Cuckoo, malware files can be directly submitted through a web UI (user interface) from which analysis results can also be viewed in a virtual environment. Cuckoo core can receive malware samples and several options can be selected with regards to what happens with the virtual machine (VM). As formulated by Muhovic (2020), figure 2 illustrates a typical malware analysis process in the Cuckoo sandbox.

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 7, Issue 4, 2024 (pp. 8-26)

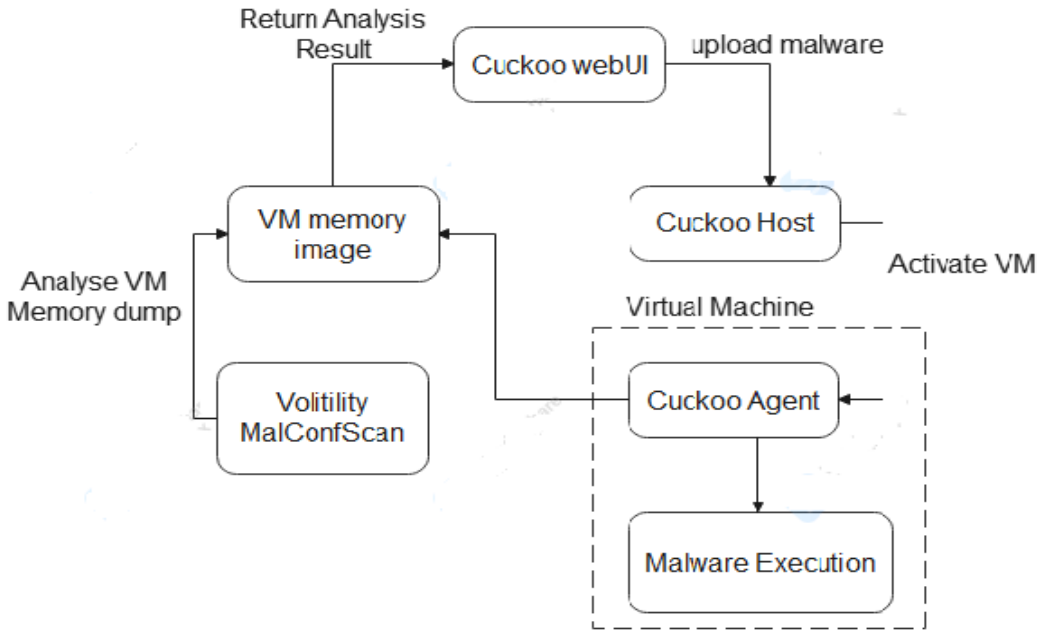

Figure 2: Malware Analysis Process in Cuckoo

2. Process Monitor

Process Monitor is a sophisticated monitoring tool for Windows that provides real-time insights into file system, registry, and process/thread activities. It merges the functionalities of the legacy Sysinternals utilities, Filemon and Regmon, while offering numerous enhancements. These include advanced and non-destructive filtering, detailed event properties like session IDs and user names, accurate process information, complete thread stacks with integrated symbol support for each operation, and the ability to log simultaneously to a file, among other features. These powerful capabilities make Process Monitor an essential tool for system troubleshooting and malware detection (Spiceworks, 2024). The main functionality of Procmon is known as event capture, which allows system administrators to see and monitor all activity in addition to historical event logs in order to identify dangerous activities such as viruses and malware (Harrington, 2021). In the event of malware infiltration to a system, Procmon works in performance with data alert tools to help initiate the appropriate procedures and reactions. It is also similar to open-source malware detection and extraction tools like Process Hacker. Procmon is a potent analysis tool that uses a kernel driver to monitor and capture profiling events, network activity, registry activity, File system activity, and processes, including threads activity on a given system (Bencherchali, 2020). Process Monitor analyzes processes such as update, read-write (WriteFile, ReadFile, and CreatFile), or delete registry records. This action enables the analyst who reviews the file to determine the method in which malware executes its actions and instigates the attack (Bhardwaj, Avasthi, Sastry & Subrahmanyam, 2016).

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

RESEARCH METHODOLOGY

To evaluate the efficiency and accuracy of the cuckoo sandbox with process monitor, we ran a comparative study of the two analysis tools (Cuckoo and Procmon) and their respective execution time was recorded. Accuracy level was also used as a criterion to determine which one of these tools is more reliable in malware detection and classification when applying specific machine learning models.

Experimental Design and Results

To achieve the goal of the study, Cuckoo sandbox lab was set up with ten (10) different runs of malware analysis with different file types and sizes. After each run, the type of malware detected; the severity level of the malware captured by cuckoo threat level score; and the execution time was recorded (table 5.1). A similar lab was set up for Process Monitor (Procmon) that also runs ten (10) isolated processes. The process activities like file system, network communications, and registry keys were monitored, and the process name; operation types performed; operation class (file system, registry keys or network communications); and the duration of process execution were captured and recorded (Table 5.2).

We deployed a window-based PC (Window 10 Pro current version) with 4 GHz CPU, 16 GB RAM, 8 Cores and 320 GB Hard Disk Drive (HDD)

To deploy the host, we set up the experiment as follows:

1. Deploy Windows 10 Pro
2. Launch and configure the guests VM (Oracle VM VirtualBox)
3. Installation of Ubuntu server 20.04LTE (on guest OS)
4. System upgrade
5. Creation of dedicated user for Cuckoo Sandbox
6. Deployment of Cuckoo Sandbox. To configure the Cuckoo Sandbox, we:
    - Added MongoDB
    - Added repository
    - Created database for Cuckoo
    - Deployed Yara-Python
    - Configured of VM

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

For the virtual machine (VM) and guest system, we deployed the Oracle VM VirtualBox which was setup and configured as follows:

Operating System	-	Window 10 (32-bit)
Based Memory	-	904MB
Accelerator	-	PAE/NX, Hyper-V Paravirtualization
Video Memory	-	130 MB
Graphic Controller	-	VBox SVGA
Storage Controller	-	STAT
SATA Port0:	-	Windows 10 vdi (Normal 50.0 GB)
SATA Port1:	-	[Optical Drive] Windows.iso (3.86 GB)
Network adapter:	-	Intel PRO/1000 MT Desktop (NAT)

Cuckoo Sandbox Configuration and Results

The Cuckoo Sandbox and its dependencies were installed using Oracle VM Virtualbox. Its purpose was to execute Portable Executable files (PEs) and generate JSON reports that show the behavior of the executed programs in a controlled and organized environment. To automate the process of extracting program behavior from JSON reports, a Python-based system was developed. For this study, malware samples were obtained from VirusTotal and submitted for dynamic analysis of their behavior via the Cuckoo Sandbox online platform. The resulting reports were extracted and converted to comma-separated value (CSV) file format. Table 1 presents the features extracted from the JSON reports.

Table 1: Raw Features Extracted by Cuckoo

Features	Data type	Features	Data type
e_cp	Integer	hiwater_rss	Integer
e_lfarlc	Integer	total_vm	Integer
e_crlc	Integer	shared_vm	Integer
e_minalloc	Integer	exec_vm	Integer
e_ovno	Integer	shared_vm	Integer
e_cparhdr	Integer	exec_vm	Integer
Machine	Integer	reserved_vm	Integer
e_oemid	Integer	nr_ptes	Integer
e_oeminfo	Integer	end_data	Integer
TimeDateStamp	Integer	last_interval	Integer
map_count	Integer	nvcsw	Integer
lock	Integer	nivcsw	Integer
utime	Integer	min_flt	Integer
stime	Integer	maj_flt	Integer
gtime			
cgtime			

The Cuckoo analysis scoring indicate the degree of maliciousness or malevolence of the analyzed file. The score is determined by measuring how many malicious actions are performed. Cuckoo applies a collection of simplified malicious behavior, known as signatures,

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

to automatically identify malicious behavior of interest. Every signature has a score, which specifies the severity of the completed action. Some instances of what Cuckoo's signatures can be used for are:

1. To recognize a given malware family of interest by separating certain distinct behaviors, such as file names or mutexes;
2. To spot some modifications of interest performed on the system by the malware, like installation of drivers;
3. To identify specific malware groups, like Ransomware or Banking Trojan by separating unique actions frequently carried out by such malware groups; and
4. To categorize samples into malware/unknown.

In the Cuckoo graphical user interface (GUI), there are colors of green, yellow and red indicators to define and specify reliability or severity of the file. The green color is used for samples with a score of 4 and below, the yellow is used for file samples with a score between 4 to 7, while the red indicator defines scores of 7 to 10. Figures 3 and 4 define the cuckoo Color indicators of Report Severity and Scoring Systems, respectively.

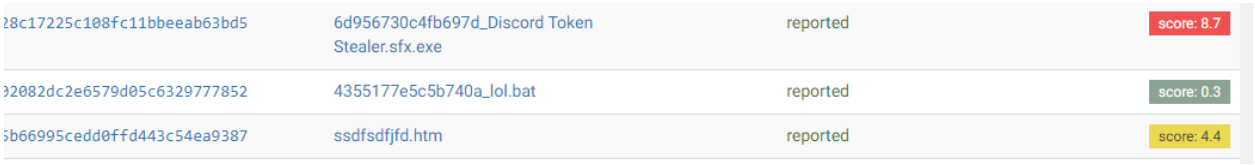

Figure 3: Color indicators of Report Severity

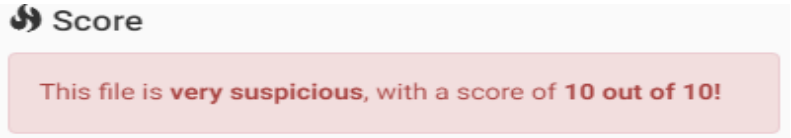

Figure 4: The Cuckoo Scoring Systems

The malware sample files executed by the cuckoo in the cuckoo sandbox were in the format of a PE file with the maximum execution time of each sample set at 432 seconds. For instance, one of the files submitted for analysis on August 8, 2024, at about 2.00 am, using the internet routing, completed its execution on the same day at about 2.47 am, resulting in a total execution time of a total of 432 seconds. As soon as the malware executable was submitted to the cuckoo environment, a software suite known as process monitor traces and monitors the changes in the file in terms of file system activities, registry modifications, and network communications. Cuckoo obtained and captured the screenshots taken during the executions of malware samples; logs all the system calls performed by all the processes generated by malware; and records files that are being deleted or created by the malicious sample that is being executed. Figures 5 and 6 show the analysis and the cuckoo logs, respectively, generated during the malicious sample executions.

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

Analyzer Log

```
2024-08-8 15:40:26,015 [analyzer] DEBUG: Starting analyzer from: C:\tmpdrdvpd
2024-08-8 15:40:26,015 [analyzer] DEBUG: Pipe server name: \??\PIPE\iOcPrsftFVkSbRfR
2024-08-8 15:40:26,015 [analyzer] DEBUG: Log pipe server name: \??\PIPE\wqJbKZEyPmVUlqtEvUJKFVuyKNvQE
2024-08-8 15:40:26,265 [analyzer] DEBUG: Started auxiliary module Curtain
2024-08-8 15:40:26,265 [analyzer] DEBUG: Started auxiliary module DbgView
2024-08-8 15:40:26,796 [analyzer] DEBUG: Started auxiliary module Disguise
2024-08-8 15:40:27,000 [analyzer] DEBUG: Loaded monitor into process with pid 508
2024-08-8 15:40:27,000 [analyzer] DEBUG: Started auxiliary module DumpTLSMasterSecrets
2024-08-8 15:40:27,000 [analyzer] DEBUG: Started auxiliary module Human
2024-08-8 15:40:27,000 [analyzer] DEBUG: Started auxiliary module InstallCertificate
2024-08-8 15:40:27,000 [analyzer] DEBUG: Started auxiliary module Reboot
2024-08-8 15:40:27,062 [analyzer] DEBUG: Started auxiliary module RecentFiles
2024-08-8 15:40:27,078 [analyzer] DEBUG: Started auxiliary module Screenshots
2024-08-8 15:40:27,078 [analyzer] DEBUG: Started auxiliary module Sysmon
2024-08-8 15:40:27,078 [analyzer] DEBUG: Started auxiliary module LoadZer0m0n
2024-08-8 15:40:27,171 [lib.api.process] INFO: Successfully executed process from path 'C:\\Windows\\System32\\rundll32.exe' with arguments [u'C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\7596418c84293532ad0596428a7285ac490b65f680cb836a68ac537e36e6bd52.dll,DllMain'] and pid 884
2024-08-8 15:40:27,375 [analyzer] DEBUG: Loaded monitor into process with pid 884
```

Figure 5: Analyzer Log generated by the sandbox during the malware analysis

Cuckoo Log

```
2024-08-8 16:44:07,029 [cuckoo.core.scheduler] INFO: Task #2069709: acquired machine win7x6412 (label=win7x6412)
2024-08-8 16:44:07,030 [cuckoo.core.resultserver] DEBUG: Now tracking machine 192.168.168.212 for task #2069709
2024-08-8 16:44:07,556 [cuckoo.machinery.virtualbox] DEBUG:Starting vm win7x6412
2024-08-8 16:44:08,078 [cuckoo.machinery.virtualbox] DEBUG: Restoring virtual machine win7x6412 to vmcloa
2024-08-8 16:44:20,890 [cuckoo.core.guest] DEBUG: win7x6412: not ready yet
2024-08-8 16:44:21,895 [cuckoo.core.guest] DEBUG: win7x6412: not ready yet
2024-08-8 16:44:25,334 [cuckoo.core.resultserver] DEBUG: Task #2069709 is sending a BSON stream
```

For every malware sample file submitted for analysis, its dynamic features were extracted and stored in a report format. The features extracted were assembled and organized in a particular file, understandable for the machine learning library. A total number of 19,611 datasets comprising 9,806 malware and 9,805 benign files were generated and stored in a comma-separated variable (csv) file format to be used by the machine learning model for classification. The 9,806 malware files were drawn from 10 different malware types' (family) collection counts as described in table 2. The table demonstrates the distribution of the various malware families as identified by 15 AntiVirus engines used by the cuckoo agent as malicious, on 15

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

different events. That is, the feature set generated from the analysis report comprises 5.0% Ransom-Ryuk; 6.3% Trojan.Inject4.9283; 6.7% Win.Ransomware.Ryuk-6688842-0; 4.5% Ransom:Win64/Jabaxsta.B; 5.7% Ransom:Win32/Genasom.ali1000102; 3.6% Packer; 4.5% W64/Ryuk.223E!tr.ranso; 3.7% HEUR/AGEN.1110011; 4.5% Virus:Gen:Variant.Ransom.Ryuk.19; 5.2% Trojan.Generic,; and 49.99% Benign (Goodware), ensuing an equal number of 50% malicious and approximately 50.0% Goodware or benign files. Figure 5 illustrates the distribution of the top eleven malware family's samples in our datasets in terms of sample counts, while figure 37 illustrates the percentage distribution of the counts.

Table 2: Malware Types (Families) Collection Counts

S/N	Malware Sample Type	Sample Counts	Percentage
1	Ransom-Ryuk	981	5.0%
2	Trojan.Inject4.	1236	6.3%
3	Win.Ransomware.Ryuk	1314	6.7%
4	Ransom:Win64/Jabaxsta.B	883	4.5%
5	Ransom:Win32/Genasom.ali	1118	5.7%
6	Packer	706	3.6%
7	W64/Ryuk.223E!tr.ransom	883	4.5%
8	HEUR/AGEN	726	3.7%
9	Virus:Gen:Variant.Ransom.Ryuk.19	883	4.5%
10	Trojan.Generic	1020	5.2%
11	Benign (Goodware)	9806	50.0%
	Total	19,612	100%

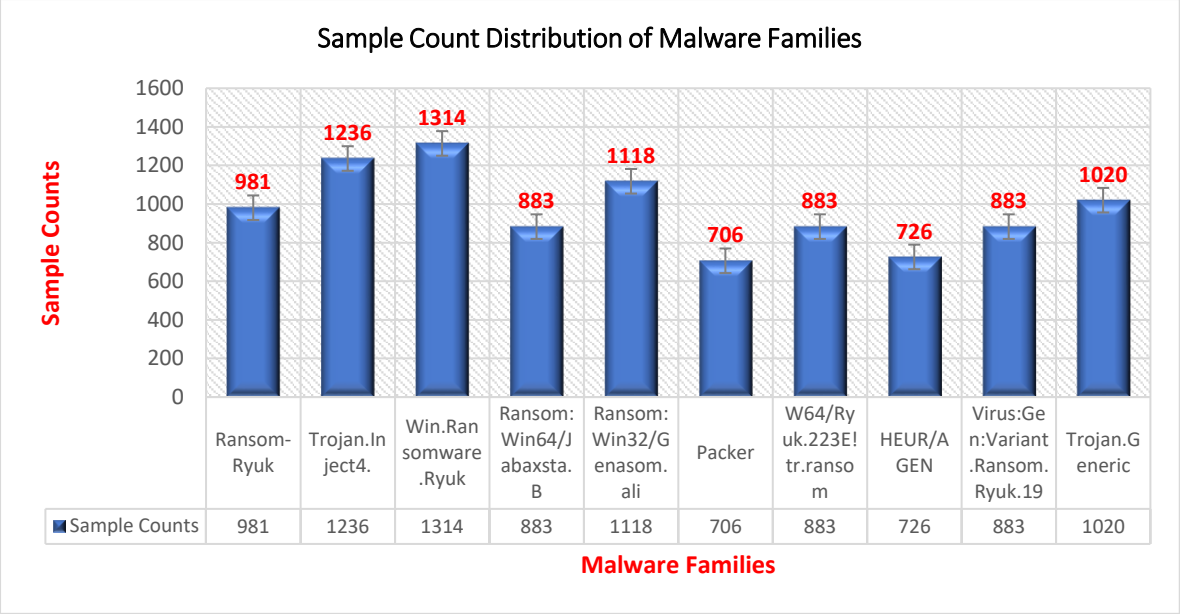

Figure 5: Distribution of top 11 Malware family's samples Counts

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

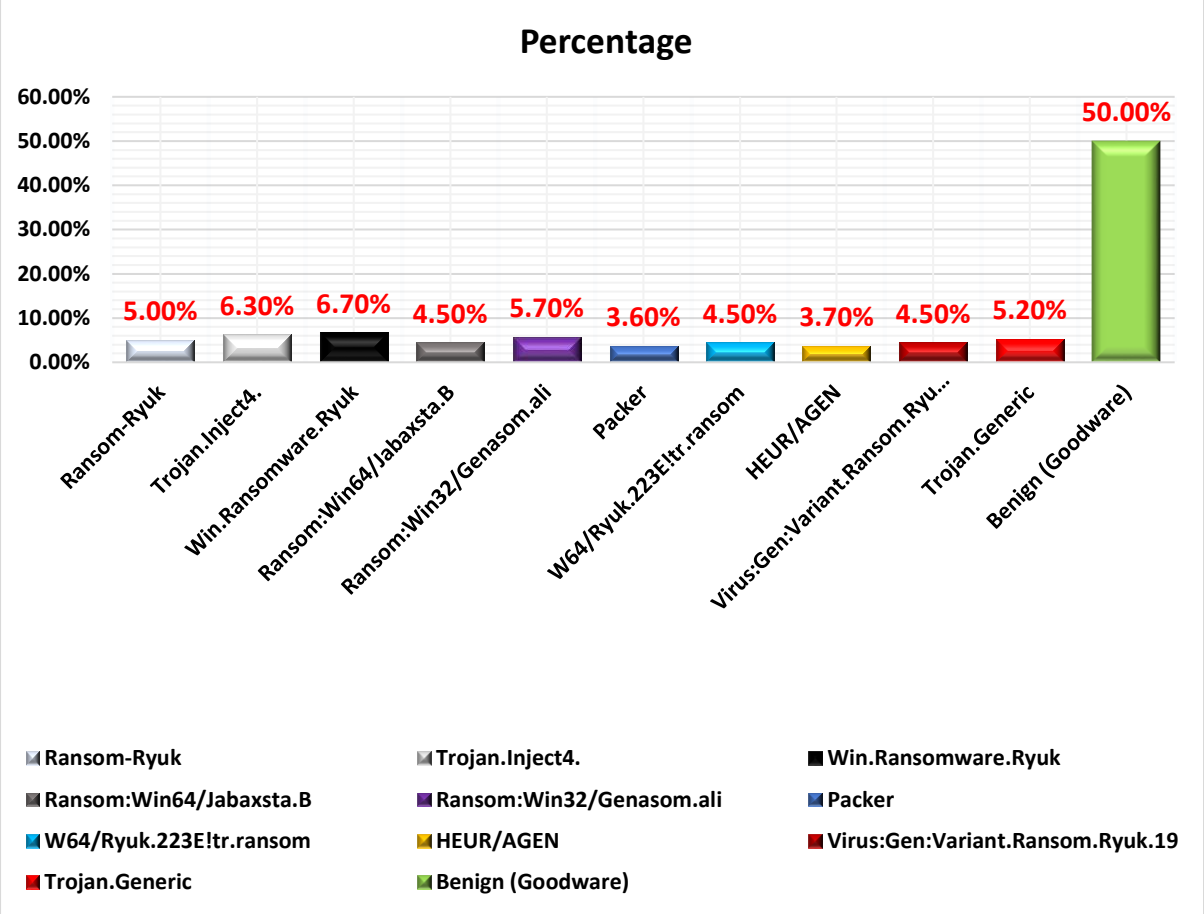

Figure 6: Percentage Distribution of top 11 Malware family's samples Counts

1. Procmon Setup and Results

A similar lab was set up for Process Monitor (Procmon) that also runs ten (10) isolated processes. The process activities like file system, network communications, and registry keys were monitored, and the process name; operation types performed; operation class (file system, registry keys or network communications). The number of experimental runs and duration of process execution was captured and recorded in Tables 3 and 4, respectively. The purpose of the experiment is to use the procmon as well as its functionalities to track and identify the actual activities of malware - the registry keys it modifies, files it creates, its network activities, and more, and connect each action captured by a procmon to an operation. Process Monitor analyzes processes such as update, RegSetValue, RegCreatKey, RegDeletKey, FileSystemControl, read-write (WriteFile, ReadFile, and CreatFile), or delete registry records. Figure 7 shows the GUI for File activities operations during the process execution by the Procmon.

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 7, Issue 4, 2024 (pp. 8-26)
www.abjournals.org

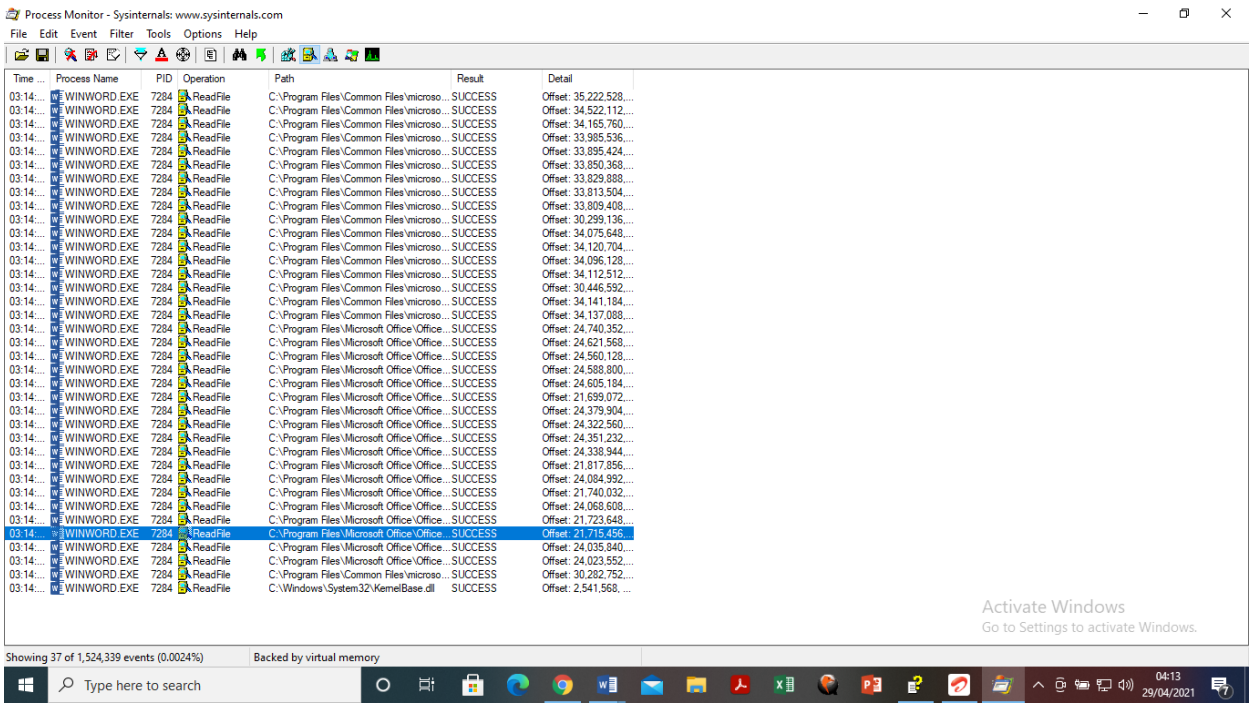

Figure 7: File System Actives Operations during Process Execution

Table 3: Different Runs of Malware Analysis in Cuckoo Sandbox

Sample Count	File Type	File size	Date submitted	Time Cost (in Secs.)	Malware detected	Cuckoo Score (10)
1	ASCII text	164 kb	4-2-2021	169	VB:Trojan.Valyria.3655	8.7
2	MS Word	1.9MB	4-2-2021	297	Trojan.Generic	7.1
3	HTML Doc.	9.6 kb	4-2-2021	521	Trojan.inject	10
4	PE+ executable	432 kb	6-4-2021	432	Packer	10
5		5.0 MB	6-4-2021	76	VB:Trojan.Valyria.3655	10
6	PE executable (dll)	5.0 MB	6-4-2021	76	Ransom-Ryuk	10
7	HTML Doc	9.6 kb	6-4-2021	530	Trojan.HTML.Phishing.Paypal.AF	10
8	PE+ executable	171 kb	6-4-2021	432	HEUR/AGEN.1110011	10
9	PE+ executable	45.7 kb	15-4-2021	118	Virus:Gen:Variant.Ransom.Ryuk.19	10
10	PE+ executable	171.5 kb	6-4-2021	432	Ryuk.Payloader	10

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

Table 4: **Procmon File System and Registry Operations**

Process Count	Date	ProcessName	Operation Type	Class	Result	Time Cost (Secs)
1	29/4/21	MS Word	Readfile	File System	Success	552
2	29/4/21	Explorer.exe	RegReadKey	Registry	Success	989
3	29/4/21	AirtelBroadBand.exe	RegOpenKey	Registry	Success	223
4	29/4/21	Ctfmon.exe	ReadFile	FileSystem	Success	339
5	29/4/21	WinWord.exe	ReadFile	FileSystem	Success	503
6	29/4/21	Svchost.exe	CreatFile	FileSystem	Success	716
7	29/4/21	AirtelBroadBand.exe	CreatFile	FileSystem	Success	588
8	29/4/21	Liveupdate.exe	ReadFile	FileSystem	Success	615
9	29/4/21	MoMpEng.exe	CloseFile	FileSystem	Success	332
10	29/4/21	MoMpEng.exe	FileSystenControl	FileSystem	Success	334

DISCUSSION OF RESULTS

Figure 8 is a graphical interpretation of the divergence in execution time of malware analysis between cuckoo sandbox, and the File system activities and registry operations, which are common operations performed by malware - monitored using Procmon. The target is to correlate the time expended to execute the malware analysis process between the two analyses tools and extrapolate the best option amongst them. The graph shows that it takes more execution time to analyze malware operations in Procmon, whether it be Registry activities, network communication, or file system activities than it does when analyzing malware behavior using the cuckoo sandbox. This implies that the cuckoo sandbox is a more efficient, suitable, and faster malware analysis tool when compared to other advanced tools. The graph shows that Cuckoo consistently outperforms Procmon in terms of execution time, completing much more quickly and steadily over each of the ten process runs. Procmon has significantly longer and more fluctuating execution times, peaking at 989 seconds, while Cuckoo maintains execution durations around 530 seconds, suggesting superior efficiency and consistency.

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

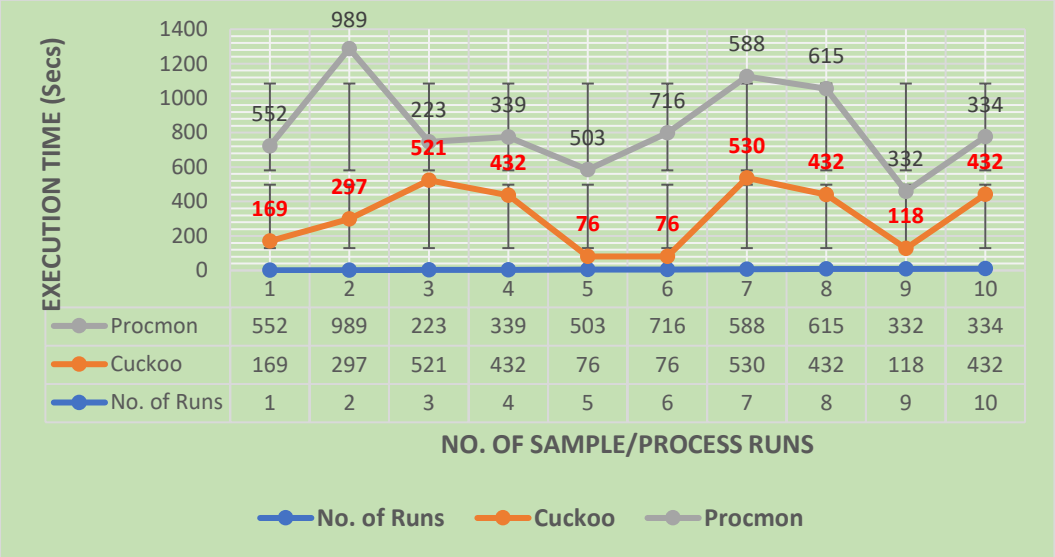

Figure 8: Comparison of Malware Execution Time Cost (Secs) Between Cuckoo Sandbox and Procmon

To further substantiate the above results, a comprehensive research survey on machine learning (ML) based malware detection and classification that deployed cuckoo sandbox was conducted, and the respective accuracy obtained, and Receiver Operating Characteristics (ROC) achieved were recorded. Equivalent survey was conducted separately on ML based detection and classification models that utilized procmon as sandbox. In the survey, the following were found; the work of Zhao, Li, Wu and Yang (2018), which evaluated supervised machine learning techniques for dynamic malware detection using different models, deployed cuckoo sandbox for the analysis, and the result shows that 99.0% accuracy level and ROC of 0.99 (99%) were achieved. Chumachenko (2017), in his Bachelor of Engineering degree research conducted in 2017, tried to find out the best feature extraction, feature representation, and classification methods that could result in the best accuracy when used on the top of Cuckoo Sandbox. The author used different machine learning models and his result indicated that accuracy of 96.8% and ROC of 0.90 (90%) were realized when implemented in SVM. Catak et al. (2020) studied a deep learning-based Sequential model for malware analysis using Windows exe API calls and used cuckoo sandbox for the dynamic analysis. The study which focuses on metamorphic malware, recorded an accuracy level of 97.9% when implemented using RBF-SVM, with no particular emphasis on ROC. Another research by Fui, Asmawi, and Hussin (2020) explored dynamic malware detection in cloud platforms. The authors who tested their framework using different classifiers also deployed a cuckoo sandbox for the malware analysis. The accuracy level of 93.0 and ROC of 0.96 (96%) when implemented using Random Forest. Denzer, Shalaginov and Dyrkolbotn (2020) conducted research on intelligent windows malware type detection based on multiple sources of dynamic characteristics and, deployed cuckoo sandbox for the malware analysis. The authors implemented their detection model using different classifiers and obtained an accuracy level of 98.6% and ROC of 0.98 (98%). A study carried out by Abbadi, Al-Bustanji, Al-kasassbeh (2020), on robust intelligent malware detection using light GBM algorithm to detect IoT botnet attacks fast, in order to end botnet activity before spreading to any new network device, analyzed the malware dynamically using cuckoo sandbox. The study achieved accuracy level of 99.0% and a ROC of 1.00. The summary findings from the survey are recorded in table 5.

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

Table 5: **Cuckoo Sandbox based Malware Analysis and Detection Survey**

S/N	Author (s)	Research Title	Year	Accuracy (%)	ROC
1	Chumachenko	Machine Learning Based Malware Detection	2017	96.8	0.90
2	Zhao, Li, Wu and Yang	Evaluation supervised machine learning techniques for dynamic malware detection	2018	99.0	0.99
3	Catak, et al.	Deep Learning Based Sequential model for malware analysis using Windows exe API calls	2020	97.7	NA
4	Fui, Asmawi and Hussin	A Dynamic Malware Detection in Cloud Platform	2020	90.3	0.96
5	Denzer, Shalaginov and Dyrkolbotn	Intelligent Windows Malware Type Detection based on Multiple Sources of Dynamic Characteristics	2020	98.6	0.98
6	Abbadi, Al-Bustanji, Al-kasassbeh	Robust Intelligent Malware Detection using Light GBM Algorithm on IoT	2020	99.0	1
7	The Present (Current) Research	Machine Learning-Based Framework for Malware Detection and Classification	2022	99.94	1.00

NA: Not available; **ROC**: Receiver Operating Characteristics

In the same way, not less than six (6) machine learning-based malware detection and classification that deployed process monitors were surveyed. The findings showed that Kardile (2017) conducted a study on Crypto-Ransomware Analysis and Detection Using Process Monitor and implemented using different machine learning models. Accuracy of 96% was recorded with reference to ROC. Asimov (2919), on the other hand, applied machine learning techniques for Android malware detection and classification using procmon to monitor the malware behavior. The study achieved an accuracy level of 88.0% and a ROC/AUC of 0.79. In another research paper, Gandotra, Bansal and Sofat (2014) investigated malware analysis and classification by applying the process monitor to study the activities of the malware. The authors tested their model and recorded an accuracy of 96.0% with no reference to ROC/AUC. Furthermore, Moussaileb et al. (2019) performed a ransomware Network Traffic Analysis for Pre-Encryption Alert, using procmon as the malware behavioral analysis tool. The study achieved an accuracy of 95.5% and a ROC of 0.95. Singh (2017) classified malware using image representation. Procmon was deployed for the malware analysis and registered accuracy of 95.4% and ROC of 0.95. Barre, Gehani and Yegneswaran (2019) applied mining data provenance to detect advanced persistent threats. To analyze the behavioral characteristics of the malware, the authors deployed the process monitor (Procmon), and achieved accuracy of 85.0% and ROC of 0.90. To conclude the survey, the work of Tobiyama et al. (2016), which seeks to detect malware with Deep Neural Network Using Process Behaviour was reviewed. The results showed that 96% of accuracy was obtained with a ROC of 0.96. Table 4 summarizes the survey. Figures 32 and 33 illustrate the performance indexes of the two analysis tools in terms of accuracy and ROC, respectively. An accuracy level of 90 and above indicates better

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

performance of the machine, while a ROC value close to 1 signals a superior performance of the machine. Proper review and analysis of the results obtained (figures 9 – 10) shows that malware analysis and detection performed using cuckoo sandbox produces a more superior performance, when compared to other sandboxes. As evident in figure 42, cuckoo sandbox recorded a high-performance index with an average accuracy of 99.35%, whereas, the analysis performed using Procmon recorded a moderate performance index with an average accuracy of 94.48%. By the same token, the cuckoo sandbox recorded a higher precision rate with an average of 0.97 ROC, defined by TPR and TNR; meanwhile, Procmon logged an average ROC of 0.91.

Table 4: Process Monitor Based Malware Analysis and Detection

S/No	Author (Year)	Study title	Accuracy	ROC
1	Kardile (2017)	Crypto Ransomware Analysis and Detection Using Process Monitor	96.0	NA
2	Asimov (2919)	Applying machine learning techniques for Android malware detection and classification	88.0	0.79
3	Gandotra, Bansal and Sofat (2014)	Malware Analysis and Classification: A Survey	96.0	NA
4	Singh (2017)	Malware Classification using Image Representation	95.4	0.95
5	Moussaileb, et al (2019)	Ransomware Network Traffic Analysis for Pre-Encryption Alert (2019)	95.5	0.95
6	Tobiyama, et al (2016)	Malware Detection with Deep Neural Network Using Process Behavior (2016)	96.0	0.96
7	Barre, Gehani and Yegneswaran (2019)	Mining Data Provenance to Detect Advanced Persistent Threats.	85.0	0.90

NA: Not available; **ROC**: Receiver Operating Characteristics.

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

British Journal of Computer, Networking and Information Technology
ISSN: 2689-5315
Volume 7, Issue 4, 2024 (pp. 8-26)
www.abjournals.org

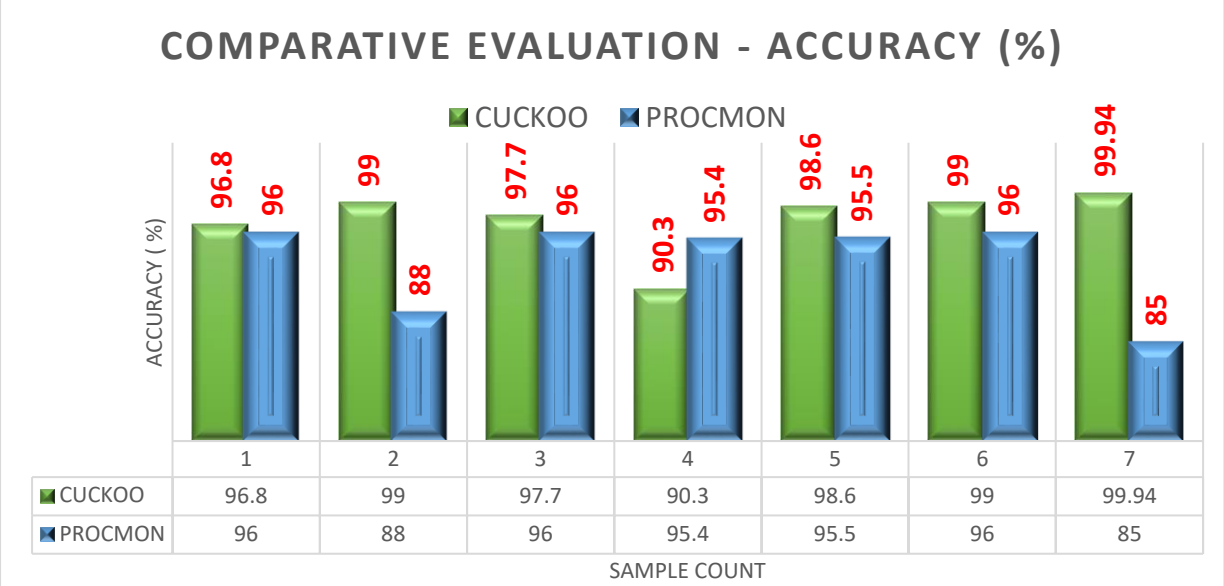

Figure 9: Comparison of Accuracy between Cuckoo and Procmon

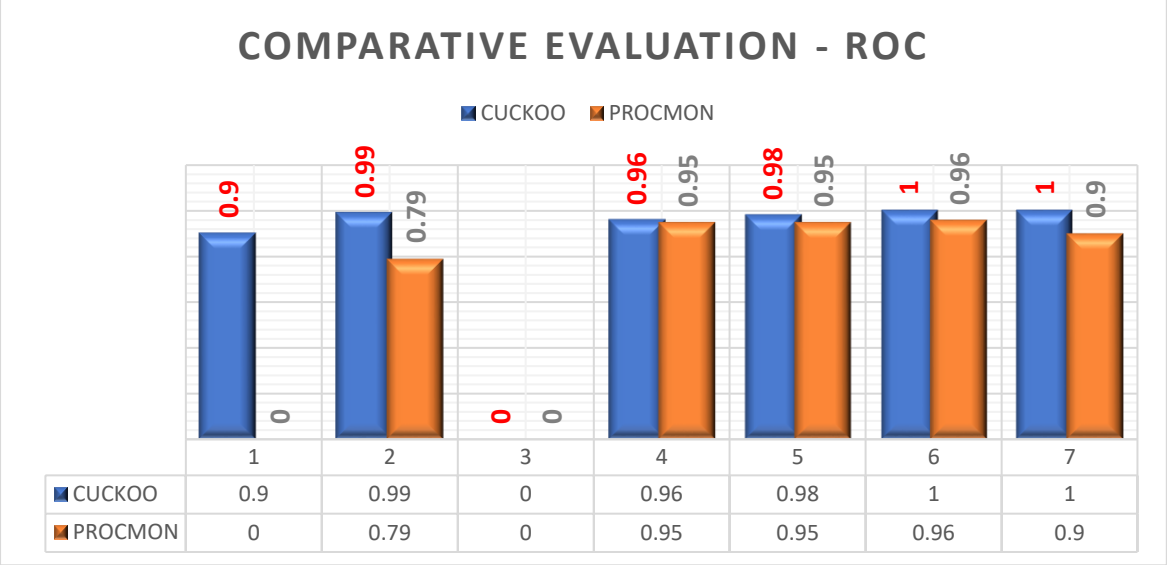

Figure 10: Comparative Performance Assessment between Cuckoo and Procmon in terms of ROC

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

CONCLUSION

To combat the evolving threat of malware, researchers use various techniques, with sandboxing being one of the most crucial. Sandboxing involves running and analyzing code in a secure, isolated environment to prevent potential threats from infiltrating networks, typically through static and dynamic analysis. This method captures essential data like system calls and network traffic to understand malware behavior. The findings in this study show that Cuckoo consistently outperforms Procmon in terms of execution time, completing much more quickly and steadily over each of the ten process runs. Procmon has significantly longer and more fluctuating execution times, peaking at 989 seconds, while Cuckoo maintains execution durations around 530 seconds, suggesting superior efficiency and consistency. A further survey of six machine learning-based malware detection studies comparing Process Monitor and Cuckoo Sandbox found that Cuckoo consistently outperformed Procmon, achieving higher accuracy (99.35% vs. 94.48%) and a superior ROC value (0.97 vs. 0.91).

REFERENCES

Alam, S., Horspool, R. N., Traore, I. and Sogukpinar, I. (2015). A framework for Metamorphic Malware Analysis and Real Time Detection. Computers & Security Journal. 48, 212–233.

Bencherchali, N. (2020). Troubleshooting with the Windows Sysinternals Tools, 2ed. eBook 978-0-13-398653-2. https://nasbench.medium.com/hunting-malware-with-windows-sysinternals-process monitor-e67476f44514. Accessed on April 29, 2021 at 17.05 pm.

Bhardwaj, A., Avasthi, V. Sastry, H. and Subrahmanyam, G. V. B. (2016). Ransomware Digital Extortion: A Rising New Age Threat. Indian Journal of Science and Technology, 9(14), 1-5. DOI: 10.17485/ijst/2016/v9i14/82936.

Bragen, S. R. (2015). Malware Detection Through Opcode Sequence Analysis Using Machine Learning. Master's Thesis, Gjøvik University College.

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

CISCO (2018). What Is the Difference: Viruses, Worms, Trojans, and Bots? CISCO Security Portal. Online at https://tools.cisco.com/security/center/resources/virus_differences. Retrieved on September 4th, 2019 at 5:10 pm.

Cuckoo Foundation (2015). Automated Malware Analysis - Cuckoo Sandbox.. Available: http://www.cuckoosandbox.org/. Retrieved on 13- February- 2021 at 12.51 am.

Cuckoo Foundation (2019). Automated Malware Analysis - Cuckoo Sandbox.. Available: http://www.cuckoosandbox.org/. Retrieved on 13- February- 2021 at 12.51 am.

Egele, M., Scholte, T., Kirda, E., Kruegel, C. (2008). A Survey on Automated Dynamic Malware-Analysis Techniques And Tools. ACM Computing Survey 2008, 44 (6).

Fox, N. (2021). 11 Best Malware Analysis Tools and Their Features. Varonis Threat Detection online at: https://www.varonis.com. Retrieved on April 29, 2021: 14.04 pm.

Harrington, D. (2021). The Ultimate Guide to Procmon: Everything you need to know. Varonis. https://www.varonis.com/blog/procmon#:~:text=Procmon's%20main%20functionality%20is%20known,other%20forms%20of%20malicious%20activity.

Ijaz, M., Durad, M. H., & Ismail, M. (2019). Static and Dynamic malware analysis using machine learning. In 2019 16th International Bhurban Conference on Applied Sciences

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY

and Technology (IBCAST) (pp. 687-691). IEEE. https://doi.org/10.1109/IBCAST.2019.8667136.

Kim, C. W.(2018). NtMalDetect: A Machine Learning Approach to Malware Detection Using Native API System Calls. arXiv preprint arXiv:1802.05412 (2018).

Mills, A. and Leggs, P. (2020). Investigating Anti-Evasion Malware Triggers Using Automated Sandbox Recognition Techniques. Journal of Cybersecurity and Privacy, MDPI. 1, 19-39.

Muhovic, T. (2020). Behavioural Analysis of Malware Using Custom Sandbox Environments. Unpublished Master's Degree Thesis. Networks and Distributed Systems Aalborg University https://www.aau.dk.

Or-Meir, O., Nissim, N., Elovici, Y., Rokach, L. (2019). Dynamic Malware Analysis in the Modern Era - A state of the Art Survey. ACM Computing Survey 2019,(52),1–48.

Pektas, A. (2015). Behavior Based Malware Classification Using Online Machine Learning. (Unpublished DoctoralThesis), Grenoble Alpes (2015).

Raj, R., Naveen, S., Subhikshan, R., & Tarun, S. (2024, January 27). Malware analysis using sandbox. SSRN. https://ssrn.com/abstract=4708146 or http://dx.doi.org/10.2139/ssrn.4708146.

Sinha, A. K., & Sai, S. (2023). Integrated Malware Analysis Sandbox for Static and Dynamic Analysis. In *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (pp. 1-5). IEEE. https://doi.org/10.1109/ICCCNT56998.2023.10306805.

Walker, A. and Sengupta, S. (2020). Malware Family Fingerprinting Through Behavioral Analysis. In the IEEE International Conference on Intelligence and Security Informatics (ISI), 1-5.

Article DOI: 10.52589/BJCNIT-FCEDOOMY
DOI URL: https://doi.org/10.52589/BJCNIT-FCEDOOMY