Applied Information Technology And Computer Science Vol. 4 No. 2 (2023) 30-45

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

Cuckoo Sandbox VS Virus Total: Categorical Analysis between Sandboxes

Karthic Anna Ravi Maran[1], Nurul Azma Abdullah[1]*

[1]Fakulti Sains Komputer dan Teknologi Maklumat,
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

DOI: https://doi.org/10.30880/aitcs.2023.04.02.003
Received 08 September 2023; Accepted 063 November 202; Available online 30 November 2023

Abstract: This comparative study aims to assess the effectiveness of Cuckoo Sandbox and Virus Total in accurately identifying and categorizing malware samples as either threats or benign files, while optimizing overall performance. The research entails downloading malware samples, establishing the experimental environment, subjecting the files to analysis within the respective sandboxes, and evaluating the results based on specific measurements. These measurements include file scanning time, sandbox system utilization, and file identification accuracy. The findings provide compelling evidence supporting the superior performance of Virus Total, which achieved a perfect accuracy score of 1.0 while maintaining high efficiency which includes low scan time and low system utilization. In contrast, limitations arising from the lack of up-to-date support in Cuckoo Sandbox hindered its performance in all these parameters. This research contributes to advancing the field of sandbox solutions by deepening our understanding of sandbox performance and offering insights for the development of more effective malware analysis tools.

Keywords: Sandbox, Malware, Accuracy, Time, System Utilization

1. Introduction

Sandboxes are essential tools in the field of cybersecurity, allowing users to safely execute suspicious code without endangering their devices or networks. These virtual environments replicate end-user operating systems and provide a secure space for analysing the behaviour of malware.

Understanding the capabilities and limitations of these two solutions is paramount for the cybersecurity research community. This study aims to examine the differences between Cuckoo Sandbox and Virus Total in terms of their ability to analyse and report on malware behaviour. The study will evaluate several parameters including scan time, system utilization and accuracy of reading.

The study aims to achieve three main objectives. Firstly, it seeks to compare the detection rate of Cuckoo Sandbox and Virus Total on the DikeDataset[1], focusing on detection accuracy. By evaluating

*Corresponding author: azma@uthm.edu.my

this factor, the study aims to determine which tool is more effective in detecting malware threats within this specific dataset. Secondly, the study aims to evaluate the system utilization, and scan time. This assessment will help determine which tool is efficient and lightweight, therefore allowing swifter response to a threat if it arises. Lastly, the study aims to analyse the effectiveness of Cuckoo Sandbox and Virus Total in detecting and analysing different types of files, specifically OLE and PE files, using the DikeDataset[1] as a basis for comparison. By assessing Cuckoo Sandbox and Virus Total's performance in detecting and analysing these specific malware types, the study aims to uncover its strengths and weaknesses in combatting various threats.

To ensure that the project can achieve the stated objectives, project scopes have been established. The scopes are that both sandboxes will be run with the same system configuration and parent VM [Ubuntu]. Cuckoo Sandbox will be run with Windows 7 as the OS has the most documentation and support from the Cuckoo Community and the OS used by Virus Total is not mentioned, published, or verified by any source. The same malware will be used for both sandboxes and be taken primarily from DikeDataset[1] from GitHub repository which as a collection of almost 11,000 samples. Using a script, 100 benign and 100 malware samples will be chosen to be executed, studied, and analysed. And finally, the data will be tabulated, and further analysis and charts will be drawn to compare the performance of sandboxes.

By analysing the capabilities and limitations of these two sandboxes, the study hopes to find the discrepancies between these two sandbox solutions. Using the data obtained with drawn charts and explanation, it is expected that one sandbox solution will be more effective at detecting the type of files executed with less time taken and less stress induced upon the host system.

The rest of the paper is organized as follows: Section 2 discusses the literature review of the related work and existing applications. Next, the methodology used to develop the application including the analysis and design is described in Section 3. Section 4 will explain about the experiment conducted and results obtained. Finally, the last section, Section 5, concludes the current work and highlights the future work to be performed in this general topic.

2. Related Works

An overview of malware, malware analysis techniques, sandbox environments and performance metrics used in calculation detection rate.

2.1 Malware Analysis Techniques

As a brief overview, static analysis is a method of analyzing software or code without executing it. It involves examining the code's structure, syntax, and other characteristics to identify potential issues or vulnerabilities[2]. Dynamic analysis, on the other hand, involves executing the software or code in a controlled environment to observe its behavior and interactions. It helps uncover runtime errors, memory leaks, and other runtime issues[2]. Hybrid analysis combines both static and dynamic analysis techniques to provide a more comprehensive understanding of the software's behavior and potential risks. It leverages the strengths of both approaches to enhance the accuracy and effectiveness of the analysis process.

2.1.1 Static Analysis

Static analysis is a technique used to analyze malware without running the code by determining the signature or unique identification of the binary file through calculation of its cryptographic hash and examination of technical indicators such as file names, hashes, and strings including IP addresses and domains[3]. This method is used to reverse engineer malware by disassembling it into human-readable assembly language. Furthermore, since malware source code is generally not available to malware analyst, the common approach to analyze a file is to dismantle the binary of the file [4]. While static

analysis is fast and does not require a controlled environment, it can be thwarted by evasive techniques like obfuscation, packing, and code rearranging and may not be effective against zero-day malware attacks[5].

2.1.2 Dynamic Analysis

Dynamic analysis is a technique to combat the shortcomings of static analysis and is used to observe the behavior of malware in a closed and isolated virtual environment to understand and prevent its spread[3]. This method provides a more in-depth look at the malware and is useful for analysts and incident responders. Dynamic analysis relies on sandboxes and can be aided by automated sandboxing, but some attackers design malware to execute only under certain conditions, making it somewhat dependent on static analysis as well [5].

2.2 Related Work on Sandbox Solutions.

In [4], the authors compare the performance of two sandboxes, Anubis, and Cuckoo, using machine learning engines to analyze data from both sandboxes. The malware dataset they used to conduct this study is the Malheur dataset and for benign files they used Win32 binary executable files. They use k-Nearest Neighbor (kNN) classification and the machine learning engines Malheur and Scikit-Learn to compare the sandboxes. The results show that Anubis has higher accuracy than Cuckoo in kNN classification and in the Malheur and Scikit-Learn engines. In the study it is stated in the system process design section that Cuckoo JSON files were converted to the mist format to be then analyzed by the Malheur machine learning model to be verified; but this posed a challenge in that during this conversion from Cuckoo JSON to MIST file as the conversion suing Cuckoo2MIST has corrupted some of the files resulting in an imbalance and skewed measurement.

To overcome this challenge of this research, work by[6] was taken into consideration as the threat scoring provided by cuckoo can be used to identify the malice of a file. The study [6] highlights that the scoring system of cuckoo relies on API calls and server pings, and that just because a threat score is higher on a file it does not necessarily mean it is more dangerous than a lower scoring file. This is explained because if a malware pings a dead server multiple times for a response cuckoo will give a higher threat score due to the higher frequency of outgoing traffic. For this research, however the indication of a file being classified as malicious or benign is sufficient to proceed as that label is enough for binary classification calculation. The shortcoming of paper [4] and the validity of the threat score in cuckoo by [6] is used as a baseline to measure the performance of Cuckoo Sandbox in its ability to accurately identify and classify files.

In [7], the authors compare the ability of antivirus software to detect malware with that of Virus Total. The authors separate their research into four sub-processes using different file formats and machine learning engines for each sandbox. They create a dataset using two popular shellcodes from the open source Metasploit framework which is Reverse TCP Interpreter and Reverse TCP Shell. Following that they also collected 16 open-source AV evasion tools from GitHub to generate 50 obfuscated malware samples to bypass AV products. This research was divided into two different categories. Malware detection with the VM's that the software runs on disconnected from the internet and 3 of the highly cloud dependent Anti-Virus solutions (Windows Defender, Bitdefender, Avira) being tested with internet connection. They record whether the antivirus programs detect the malware during download, scanning, or execution phases. This study clarifies that the detection method for Virus Total is not known to the author as that information is not made public by Virus Total. It is also stated that Virus Total mainly uses static analysis for quick file detection and behavioral analysis to verify the severity of a file during execution or connecting to the remote server.

Work by [8]also reiterates that Virus Total does not specifically label an app as malicious but returns the total AV engine detection score as its threat label. It was also emphasized by [8] that a lack of standard procedure for interpreting the score opens room for researchers to use their own intuition to

label the file. The authors propose a method called Maat to tackle these issues of standardization and sustainability by automatically generating a Machine Learning (ML)-based labeling scheme, which outperforms threshold-based labeling strategies. They evaluated the applicability of Maat's ML-based labeling strategies using the Virus Total scan reports of 53,000 Android apps that span one year and compared their performance against threshold-based strategies. Both [7], [8] verify that even with its issues, threshold labeling method is still reliable to infer if a file is malicious or benign which is what will be used in this research. Table 1 summarize related work done for comparison of sandbox solutions.

Table 1: Summary of related work on comparison of sandbox solutions

Work	Dataset	Type of Analysis	Metrics
A Comparative Study of Behavior Analysis Sandboxes in Malware Detection [4]	Malhuer Win32 Benign files	Dynamic	Precision Recall F-Measure Confusion Matrix
A Comparative Analysis of Virus Total and Desktop Antivirus Detection Capabilities [7]	Metasploit Framework	Hybrid	Detection Rate via threshold-based labeling
Cuckoo's malware threat scoring and classification: Friend or foe? [6]	Malpedia	Dynamic (Behavioral)	Signature Calculation using threat rating and final threat score for malware.
Maat: Automatically Analyzing Virus Total for Accurate Labeling and Effective Malware Detection [8]	AMD + GPlay AndroZoo Hand-Labeled Hand-Labeled 2019	Hybrid	Matthews Correlation Coefficient (MCC)

2.3 Dataset

A dataset is a structured collection of data generally associated with a unique body of work or study. It can be used for various analysis, machine learning and research purposes. There are multiple types of datasets as explained in the above subsection; Metasplit Framework [7], Malpedia [6], AndroZoo [8] are all types of datasets used in research to validate claims of performance of a system or an algorithm For this specific research, DikeDataset was used because it is a labeled dataset containing benign and malicious PE (Portable Executable) and OLE (Object Linking and Embedding) files [1]. It was created as part of a bachelor's thesis project called *"dike,"* a platform that uses artificial intelligence techniques for malware analysis. The dataset can be used for training AI algorithms to predict whether a PE or OLE file is malicious and to determine its belonging to a malware family.

2.4 Related Work on DikeDataset

Work by [9] introduces a novel method for detecting Advanced Persistent Threat (APT) malware by analyzing system calls. The study utilizes the DikeDataset[1], a well-labeled dataset of benign and malicious PE and OLE files, to develop and evaluate the proposed detection method. The authors demonstrate the method's ability to effectively distinguish between system calls made by APT malware and legitimate software. By monitoring and extracting the APT behavior of malware the author was able to train the ML algorithm to then detect such call signs and functions. DikeDataset was used for this study as it contains common malware belonging to the APT1, APT10, APT19, APT21, APT28, APT29, APT30, Dark Hotel, Energetic Bear, Gorgon Group, and WinNTI.[9] Using this study [9] as reference it is fair to assume that the DikeDataset is trustworthy as usable for another detection-based research.

Meanwhile work by [10] introduces a novel method for identifying malware by utilizing control-flow graphs (CFGs) and a pre-trained language model with a graph isomorphism network using files sourced from the Blue Hexagon Open Dataset for Malware Analysis (BODMAS). The authors

demonstrate the high accuracy of their method in detecting malware using the DikeDataset[1], a labeled dataset of benign and malicious PE and OLE files. The primary contribution of the paper lies in the introduced methodology for malware detection using CFGs and a pre-trained language model. However, the paper would benefit from providing more comprehensive details about the dataset's curation and labeling process, as well as a more extensive comparative analysis with related works. Additional references obtained from the search results offer valuable insights into malware detection and machine learning. Overall, the paper presents a promising approach to malware detection using CFGs and a pre-trained language model, supported by the evaluation using the DikeDataset. Using DikeDataset for evaluation further proofs its validity as a well-rounded malware and benign dataset. Table 2 shows the summary of related work that uses DikeDataset.

Table 2: Summary of related work with the inclusion of DikeDataset

Work	Dataset	Type of Analysis	Metrics
Toward Identifying APT Malware through API System Calls[9]	DikeDataset	Dynamic	Accuracy, Precision, Recall, F1
Malware Detection using Attributed CFG Generated by Pre-trained Language Model with Graph Isomorphism Network[10]	DikeDataset Ember	Static	True Positive Rate, False Positive Rate, Area Under Curve

3. Methodology/Framework

The research framework includes how the malware sample is obtained, how the virtual environment and sandboxes are created, how the obtained malware will be executed on the sandbox and the analysis conducted to obtain a finding. Most of these steps are conducted in a Virtual Machine to ensure that the malware files do not leak out and harm the host PC. The framework is designed to ensure that the security and integrity of the study is preserved while having ample backup to revert to in case of a file or data corruption. The results and findings will be recorded as shown in Figure 1.

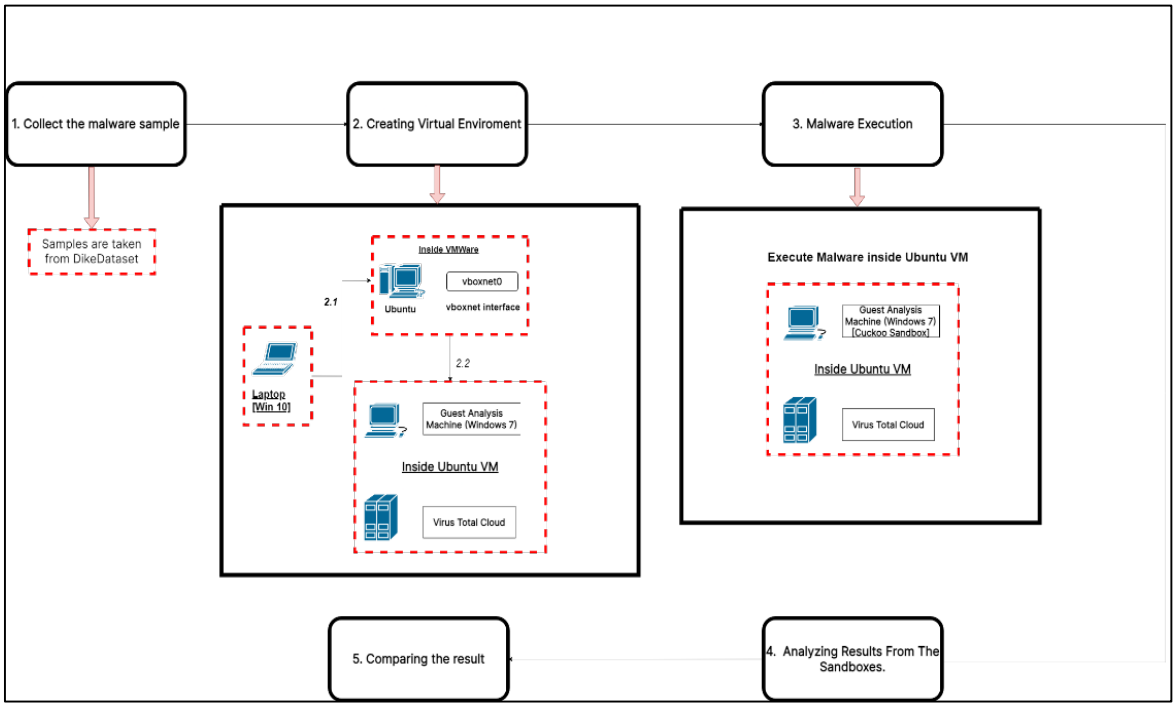

Figure 1: Research Framework

Figure 1 shows the research framework outlining the workflow from collecting the malware sample to creating a virtual environment to malware execution to analyzing the result from the sandbox and finally comparing the results of analysis and documenting them. Each step in this methodology will be explained below starting with obtaining the malware sample.

3.1 Data Collection

The foremost step of conducting a malware analysis is to obtain a set of malware samples. DikeDataset as used in studies [9], [10] is a labeled dataset containing 11,923 benign and malicious PE and OLE files. The family of malware used for this research comes from the Windows PE and OLE files, which are unknown threats - files that look suspicious but do not match any known malware definitions. In this research, the DikeDataset will be used to test the effectiveness of Cuckoo Sandbox and Virus Total in their ability in detecting and analyzing various types of malwares, such as OLE and PE files. Table 3 summarizes the segregation of files in DikeDataset.

Table 3: Summary of segregation of files in DikeDataset [1]

Filetype	Number of benign samples	Number of malicious samples
PE (Portable Execution)	982	8970
OLE (Object Linked and Embedding)	100	1871

200 files from this dataset are then separated into two different folders each labeled benign and malware respectively using a python script to eliminate bias. Table 4 summarizes the segregation of files to be used in simulation phase.

Table 4: Summary of segregation of file for simulation

Filetype	Number of benign samples	Number of malicious samples
PE (Portable Execution)	94	88
OLE (Object Linked and Embedding)	6	22

3.2 Experimental Setup

Once the samples of malware are obtained, the next step is to create a safe environment in which to run them using VMWare and Ubuntu and install two sandboxes, Cuckoo Sandbox and Virus Total. Cuckoo Sandbox is installed directly on the virtual machine, while Virus Total is accessed via the internet. A guest virtual machine is also created using a Windows 7 iso image to run the malware samples for analysis in Cuckoo Sandbox. They test the connection between the guest virtual machine and the host Cuckoo Sandbox. VMware is used to create the Ubuntu VM because it is more stable and provides multiple snapshots and restore options which acts as a failsafe in case of data corruption. Ubuntu 18.04 was chosen because, that release of Ubuntu still supports older python dependencies as Cuckoo Sandbox can only be installed in a Python 2 environment. Table 5 shows the specification of the Ubuntu VM while Table 6 shows the specification of Windows 7 guest VM used by Cuckoo Sandbox.

Table 5: Specifications of Ubuntu VM

Software	Operating System	Specification
VMware	Ubuntu 18.04 [Bionic Beaver]	Memory: 16 GB Processors: 6 Hard Disk: 70GB Network Adapter: NAT

Table 6: Specifications of the Windows 7 VM used by Cuckoo Sandbox.

Software	Operating System	Specification

VirtualBox	Windows 7 Ultimate	Memory: 2 GB Processors: 2 Hard Disk: 20 GB (Virtual) Network Adapter: NAT [vboxnet0]

3.3 Simulation

Files segregated for simulation are first executed in Cuckoo followed by Virus Total to collect the necessary data as defined in the objective.

3.4 Analysis

This study has three main measurement parameters. Those parameters are Scan time, system utilization and accuracy. Scan time each file is recorded using the in-solution time provided by each sandbox during or after analysis which is measured in seconds. System utilization is recorded using the "*System Monitor*" software that is built in Ubuntu. Finally for accuracy, the threat score from Cuckoo and label score from Virus Total are recorded. All these values are recorded in an excel spreadsheet for evaluation.

3.5 Evaluation

Evaluation is done and both sandboxes have completed executing all 200 samples. The data is then graphed for scan time and system utilization while a confusion matrix is created to calculate the precision, recall, accuracy and F1 score using the threat and label scores obtained from each sandbox respectively.

3.5.1 Scan Time

Scan time refers to the time taken for each solution to execute, analyze and produce a report on a sample. This value is provided in-solution in terms of seconds.

3.5.2 System Utilization

System utilization refers to the amount of CPU, RAM, disk space and network used by each solution to execute, analyze, and produce a report on a sample. Based on study done by [13], the measurement of CPU and I/O bound process (RAM and Hard Disk) is a valid measuring parameter for sandbox security and optimize the sandbox solution to use the CPU and I/O bound devices more efficiently. While the study [11] measures these parameters in Windows using integer values, in this study a percentage value will be used as System Monitor tool for Linux allows the user to monitor individual processes via percentage. The reason system utilization was considered as a parameter is its correlation with scan time where a higher scan time results in higher system utilization either due to inefficient use of resource allocation [11] or more complex processes executed by the malware [12]. Due to the lack of research of sandbox solutions on Linux, the data for system utilization values are interpreted and adjusted based on the study [11] to obtain the percentage values. The values for system utilization are summarized in Table 7.

Table 7 : Summary for System Utilization Values

Values	CPU Utilization	Memory Utilization	Disk Utilization	Network Utilization
Low	Below 50%	Below 50%	Below 50%	Below 50%
Medium	50 % – 70 %	50 % – 70 %	50 % – 70 %	50 % – 70 %
High	70 % – 90 %	70 % – 90 %	70 % – 90 %	70 % – 90 %
Very High	Above 90%	Above 90%	Above 90%	Above 90%

3.5.3 Performance Evaluation.

The evaluation metrics used to evaluate the sandbox performance used in this research are accuracy, precision, recall, and f1-score[9].

1. Accuracy - Indicates how accurate a prediction is. It is the ratio of the number of correct predictions to the total number of predictions made. The formulae accuracy of the reading is shown in Eq 1.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad \text{Eq. 1}$$

2. Precision - Refers to the percentage of actual positive predictions that are correctly identified as positive by the sandbox. The formulae for precision of the reading are shown in Eq.2.

$$Precision = \frac{TP}{TP + FP} \qquad \text{Eq. 2}$$

3. Recall - Refers to the percentage of actual positive predictions that are correctly identified by the sandbox. The formulae for recall of the reading are shown in Eq.3.

$$Recall = \frac{TP}{TP + FN} \qquad \text{Eq. 3}$$

4. F1-Score - This metric is the harmonic means of precision and recall. It provides a single value that balances both precision and recall, making it useful when trying to optimize both metrics. The formulae of F1-score of the reading is shown in Eq.4.

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \qquad \text{Eq. 4}$$

4. Results and Discussion

The focus of this section is on analyzing the performance of both Cuckoo Sandbox and Virus Total. Python code, utilizing "matplotlib," was used to visualize the data and gain insights into the performance of Cuckoo Sandbox and Virus Total. Each test for each parameter is done only once as if conducted multiple times, Virus Total would have an advantage as the cloud server of Virus Total has memory of previously ran results and will output the updated result instantly when the file is uploaded which violated the objective and scope of this study[7].

4.1 Scan Time

Scan time is, as the name suggests, time taken by each solution to complete a scan and provide a report on the analysis. There are several reasons why scan time is important when comparing two sandboxes or security solution. One of it is real time protection; detecting and mitigating threats as quickly as possible is crucial in cyber security, thus a faster scan time ensures that the detected malware can be contained and isolated to prevent it from harming the system as soon as possible. The scan time is recorded based on the information provided by each sandbox. Figure 2(a) shows the scatter plot of Cuckoo Benign Scan Time. There are some anomalies in the data for benign file and scan times. These specific files are shown in Table 8.

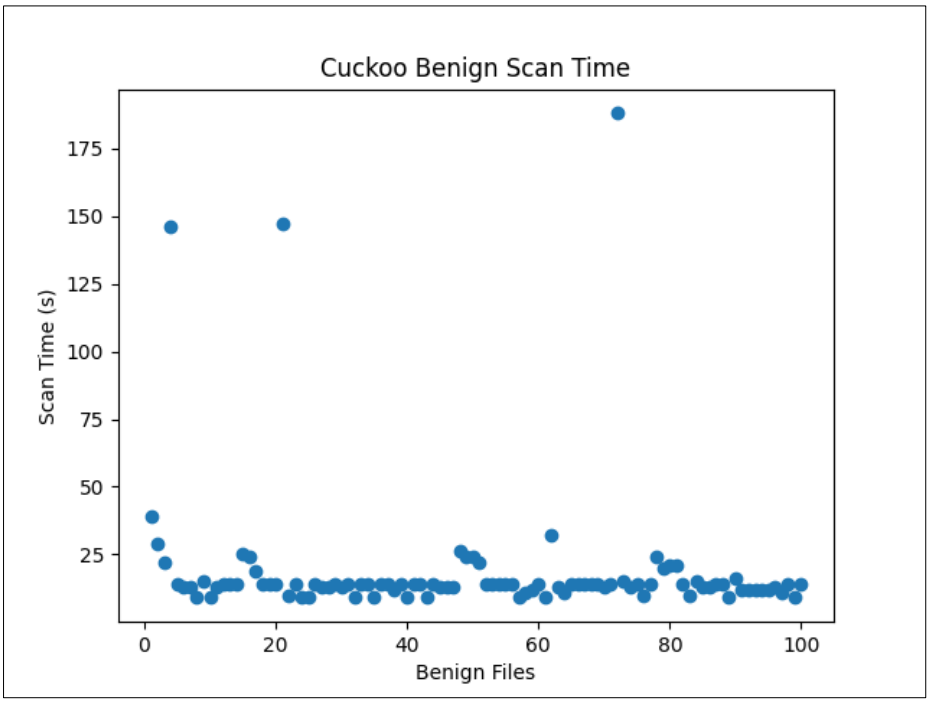

Figure 2 (a): Cuckoo Benign Scan Time

Table 8: Summary of anomaly in scan time with benign files in Cuckoo

Benign Files	Scan time (seconds)	Cuckoo Score	System Utilization
0a2027ea20fd995fd41fbe1a6e6a361dbdc09a83741f1d9e928eddf50030c6b3.exe	146s	2.2/10	Medium
38515483413002d884d7354be2a724117ca9e6b0baecfc9d47292a194996f72b.exe	147s	2.6/10	Medium
aad745af64c86195150c14cd83c41c09dd88bd10f41b4aff65768b7390f637aa.exe	188s	0.4/10	Medium

Figure 2(a) and Table 8 show the analysis reveals a correlation between the files and their scan times in terms of system utilization. The trend of the graph is scattered. This is because the size and the contents of each file varies from one file to the next. This makes the scan time of each file (which are all benign) scattered and not at all linear. Most benign files have low system utilization, while the files in question have medium utilization. This suggests that the increased demand from Cuckoo Sandbox for analyzing these files contributes to longer scan times. Another correlation observed is between the Cuckoo Rating of the first and second files, which are rated 2.2/10 and 2.6/10, respectively. Although the scan results indicate these files are benign, specific call functions within them require additional analysis to ensure their benign nature. This deeper analysis takes more time compared to other files. These factors can be taken into consideration to reasonably conclude that deeper and more thorough scanning of files is required therefore more time is taken by Cuckoo to complete the report on these files. Apart from these anomalies, the scan times for the remaining benign files range from 10 to 30 seconds, with an average scan time of 18.96 seconds when considering all files, including the anomalies.

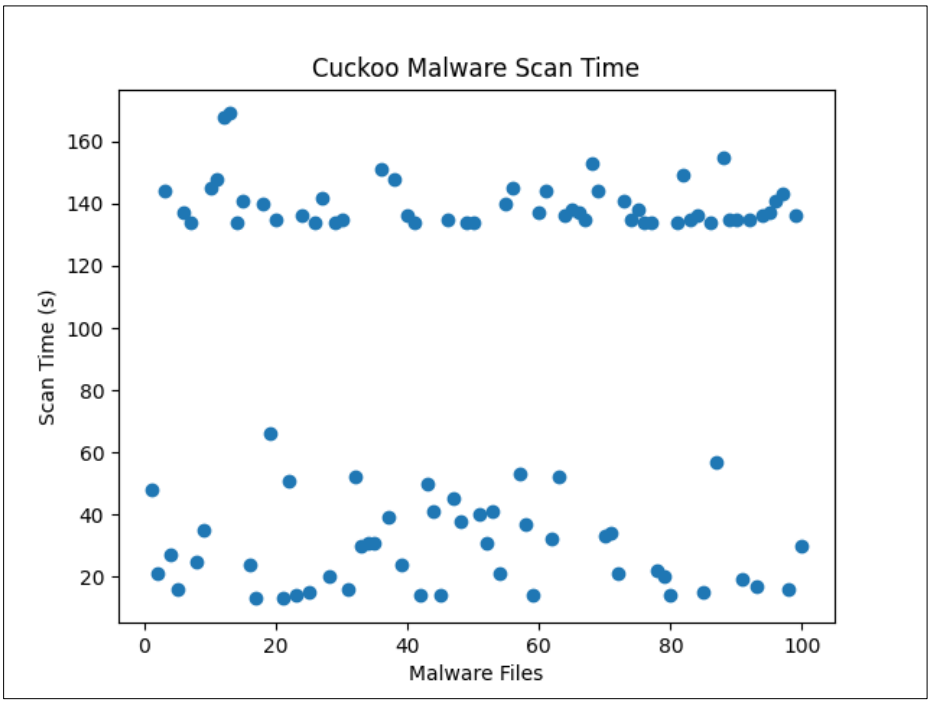

Figure 2(b): Cuckoo Malware Scan Time

Figure 2(b) presents a scatter plot of Cuckoo Malware Scan Time. The trend of the graph is also scattered. This is because the size and the contents of each file varies from one file to the next. This makes the scan time of each file (which are all malicious) scattered and not at all linear. The graph above reveals anomalies in the data for malware files. These anomalies can be easily explained. Most files have scan times either below 80 seconds or above 120 seconds. The files below 80 seconds have fewer call functions or a Cuckoo Rating below 4.0, indicating medium-risk files with some malicious activities but not highly suspicious[6]. Some files are also in this range because Cuckoo was unable to fully scan them, resulting in a score of 0.0/10 and faster analysis. On the other hand, files with scan times above 120 seconds align with the expected scan time for malicious files that have multiple call functions and malicious code strings[12]. These files require more time for thorough analysis and verification, thus increasing the overall scan time. The average scan time for malware files in Cuckoo is 87.12 seconds.

Next is Virus Total. The time taken for analysis can be found during the reanalysis section in the summary page. This is because when a file is submitted Virus Total takes the last scan made by a user or anyone else on the website that scanned the same file. These initial results often are misleading; therefore, a reanalysis is required to ensure the data on a file and its behaviors are up to date.

As shown in Figure 3(a), there are not really any oddities in the scan times for these files but is still scattered. The logic of the above two graphs applies here as well, which is size and content of each individual file acts differently from each other. Therefore, the graph is scattered and not at all linear. Most of the files fall within a few seconds of each other and some increases in time can be reasonably concluded to be caused by a higher number of call functions or higher server response time from the Virus Total servers. The average Scan time benign files in Virus Total are 29.6 seconds.

Based on Figure 3(b), the trend of this graph is also nonlinear, but an anomaly can be seen on the graph and expanded on Table 9, the reason for the higher analysis time can be explained due to the background update on Mozilla Firefox (the default Ubuntu browser) unbeknown to the user. This update has caused the browser to respond slower to the update from the Virus Total server. The other files and their scan time are within expected margins as once again certain files have higher call functions and

the Virus Total server needed more time to fetch and send the data to the user. The average Scan time malicious files in Virus Total are 40.42s.

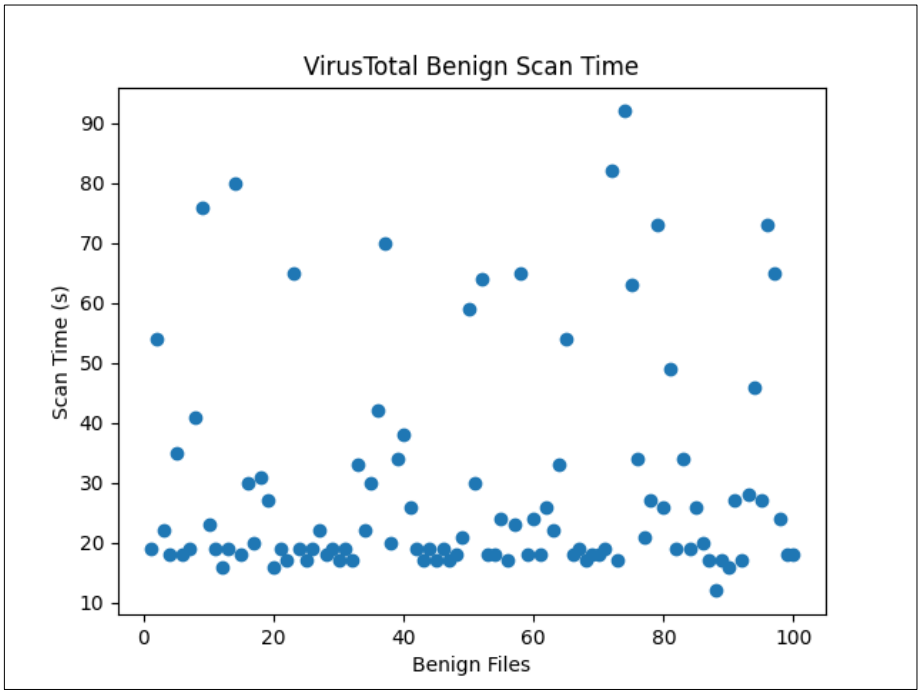

Figure 3(a): Virus Total Benign Scan Time

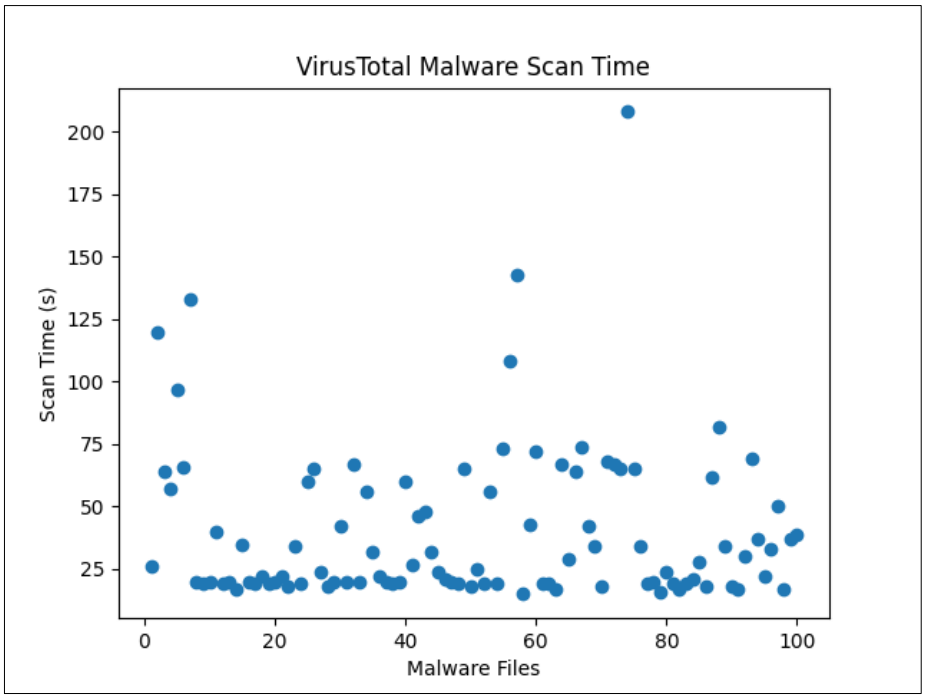

Figure 3(b): Virus Total Malware Scan Time

Table 9: Summary of anomaly in scan time with malware files in Virus Total

Malware File	Scan time (seconds)	Virus Total Score	System Utilization
c4872edb9be964f53cfecc79a682b9ed600b4a5bd5a5eca1a8f5d1625a0e402a.exe	208s	64/71	Low

Table 10 shows Virus Total has the shortest average time in total to scan files with 29.6 seconds for benign files and 40.42 seconds for malicious files. This average time includes all anomalies found during the experiment.

Table 10: Summary of average scan time for both sandboxes

Sandbox	File Type	Average Scan time (seconds)	Anomalies
Cuckoo	Benign	18.96	3
Cuckoo	Malware	87.12	0
Virus Total	Benign	29.6	0
Virus Total	Malware	40.42	1

4.3.2 System Utilization

System Utilization is an important parameter that quantifies the computational workload of a computer system. It provides insights into system performance and resource allocation. By monitoring system utilization, bottlenecks can be identified, resource allocation can be optimized, and system efficiency can be improved. In this research, system utilization is divided into four parts: CPU utilization, memory utilization, disk utilization, and network utilization. Each of these measurements is categorized into low, medium, high, and very high levels based on their respective percentages. For example, low CPU utilization is below 50%, while high CPU utilization is between 70% and 90%. Similarly, the levels are defined for memory, disk, and network utilization. Aggregating these measurements into a single value makes it easier to understand and analyze the overall system utilization. System utilization is an important metric to measure because of the importance of resource allocation [11]. A sandbox with lower resource allocation will consume less resources which allows the system to continue carrying out critical tasks without interruption. Energy efficiency can also be a byproduct of system utilization as lower resource usage directly translates to lower energy consumption which will cost less money to maintain and operate and reduce the carbon footprint on our planet [11].

As described above the aggregation of system utilization is seamless and easier to understand. The score for both Cuckoo and Virus Total in benign file execution is shown below in Figure 4(a).

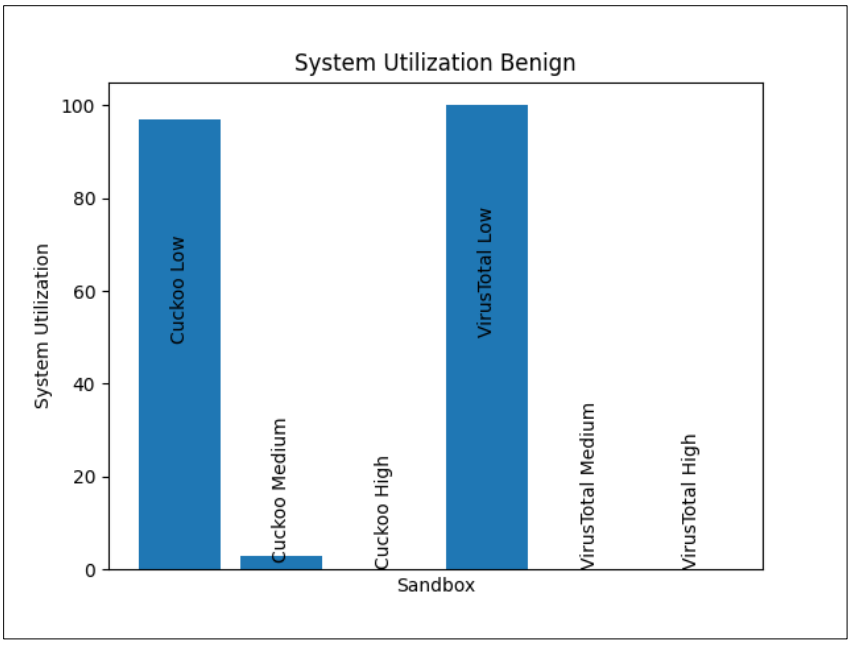

Figure 4(a): System Utilization for Benign files in Cuckoo and Virus Total

As shown in Figure 4(a), Cuckoo has 97 Low and 3 Medium for score system utilization. The Low score is because as benign files they have low system function calls and therefore take less resources to

scan compared to malicious files. The 3 Medium score can be explained in that some benign files have higher amount of intrusive function calls that need to be analyzed deeper and more thoroughly to ensure their validity therefore consuming more system resources. In Virus Total the scores are a full 100 for Low. This is because Virus Total runs the analysis on its own server therefore the only system utilization it uses is disk to upload files and download results and network to send files and receive analysis data. Therefore, there is no pressure on the system to perform heavy duties. The score for both Cuckoo and Virus Total in malware file execution is shown in Figure 4(b).

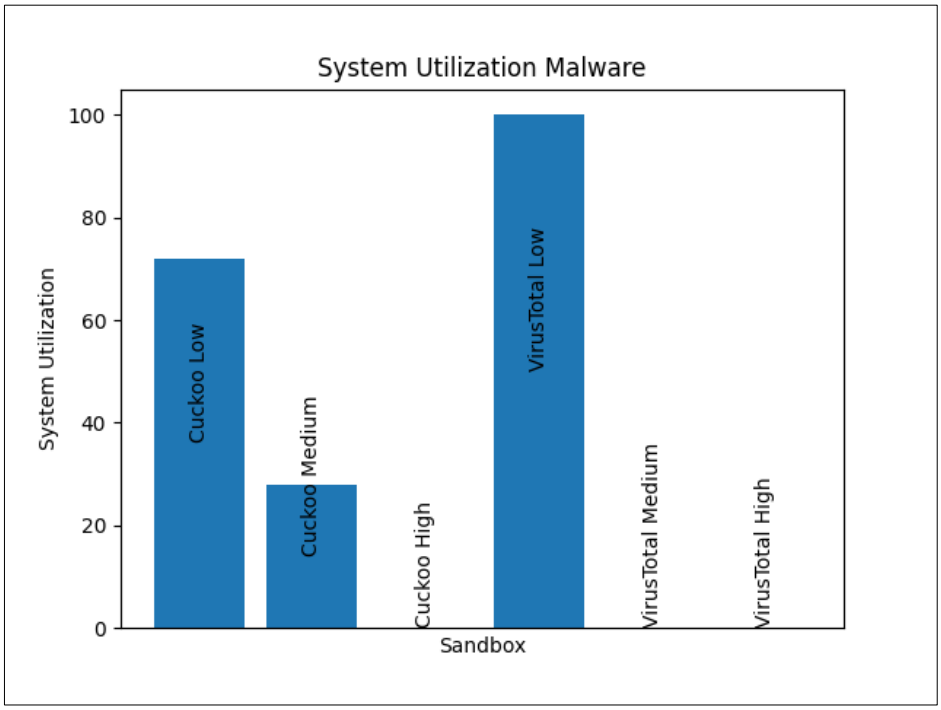

Figure 4(b): System Utilization for Malware files in Cuckoo and Virus Total.

As shown in Figure 4(b), Cuckoo has 72 Low and 28 Medium score for system utilization. The Low score is more complex to explain as some malware files have low system function calls and therefore take less resources to scan compared to other more complicated malicious files. These Low scores indicate that the system call functions in the malicious files are more optimized, and they do not need to leech system resources[6]. The 28 Medium score can be explained in that some malware files have higher amount of potentially malicious function calls that need to be analyzed deeper and more thoroughly to ensure its severity therefore consuming more system resources. In Virus Total the scores are a full 100 for Low. This is because Virus Total runs the analysis on its own server therefore the only system utilization it uses is disk to upload files and download results and network to send files and receive analysis data. Therefore, there is no pressure on the system to perform heavy duties. With that being said, the reason for network utilization to also be low can be attributed to the relatively small size of the uploaded files at a few kilobytes (KB). All these data are summarized in Table 11.

Table 11: Summary of System Utilization

Sandbox	File Type	Low System Utilization	Medium System Utilization
Cuckoo	Benign	97	3
Cuckoo	Malware	72	28
Virus Total	Benign	100	0
Virus Total	Malware	100	0

Table 11 shows all the readings for system utilization for both file types. Cuckoo has higher system utilization value compared to Virus Total, but it has clear and valid reasons for its use as the files are executed natively in the system and require the use of a nested VM which can

be taxing on the system resources. Note that this system utilization is referring to the Ubuntu VM and not the main host of Windows 10. Virus Total meanwhile offloads the taxing system works to its server which excluding network does not utilize the Ubuntu VM system. Nevertheless, Virus Total is less taxing on a system be it low-end hardware or a high-end enterprise hardware to produce results which allows the application to be easier to adopt and use.

4.3.3 Detection Evaluation Measures.

Evaluation of detection performance is crucial in assessing the effectiveness of sandboxing techniques and tools for identifying potential threats. In this experiment, the performance metrics of Accuracy, Precision, Recall, and F1-Score are evaluated for both Cuckoo and Virus Total. To understand these metrics, certain terms need clarification. True Positive (TP) represents the number of correctly identified malware samples, while True Negative (TN) represents the number of accurately identified benign samples. False Positive (FP) refers to the misclassification of benign samples as malware, and False Negative (FN) represents the misclassification of malware samples as benign. These terms provide a basis for evaluating the performance of the sandboxes in accurately detecting malware and benign samples. Table 12 will show a confusion matrix. Results of the experiment can be represented in the form of a table known as confusion matrix.

Table 12: Confusion Matrix

Type of File	Detected as Malware	Detected as Benign
Malware	True Positive (TP)	False Negative (FN)
Benign	False Positive (FP)	True Negative (TN)

Table 12 summarizes the confusion matrix specifically in the context of this experiment involving Cuckoo and Virus Total. The effectiveness of the detection performance was evaluated using several performance metrics. These metrics include:

1. Accuracy
2. Precision
3. Recall
4. F1-Score

Table 13: Summary of each metric for each sandbox

Sandbox	Accuracy	Precision	Recall	F1 - Score
Cuckoo	0.765	0.53	1.0	0.693
Virus Total	1.0	1.0	1.0	1.0

Based on the findings presented in Table 13, it can be concluded that Virus Total outperforms Cuckoo Sandbox in terms of malware detection. Virus Total achieved 100% accuracy, precision, recall, and F1-score, indicating that it correctly classified all combinations of 200 malware and benign files. This can be explained as the collection of anti-virus engines of Virus Total is updated by its vendors and cloud-based systems that are constantly updated with cutting edge detection and analysis method that's available[7] In contrast, Cuckoo Sandbox had an accuracy of 76.5% and a precision of 53%, meaning it accurately identified 53 out of 100 malware samples. The recall for both Cuckoo and Virus Total was 100%, indicating that they correctly classified all 100 benign files. Cuckoo's F1-score was 69.3%, indicating moderate effectiveness in identifying malware and avoiding false positives. However, there is room for improvement to enhance Cuckoo's overall performance in accurately detecting and

classifying malware files. One of the reasons for Cuckoo's lower performance is its inability to accurately identify, analyze, and report on OLE file formats due to outdated Python libraries. The lack of support for newer data analysis techniques in Cuckoo, which is attributed to the development of Cuckoo 3 supporting Python 3 and above but not yet fully stable, may have contributed to this limitation.

5. Conclusion

This study is aimed at finding out which sandbox solution between Cuckoo and Virus Total is categorically better at accurately detecting the malice of a file. A dataset was selected in DikeDataset to ensure that the measurements taken are valid and accurate. This study proposes that Virus Total is a better solution when it comes to accurately detecting the malice in a file. This statement was reached by evaluating the performance of the sandboxes in certain metrics such as scan time, system utilization and accuracy in detection. Virus Total has outclassed Cuckoo in each measurable metric which has landed it as being the better of the two solutions. When broken down by parameters, for scan time, Virus Total is much faster than Cuckoo, in system utilization the result is also the same with Virus Total having much lower system usage due to its cloud-based server compared to the local Cuckoo's system utilization, and finally the measurement parameters cement Virus Total as the better sandbox where it was able to detect every file type with an accuracy score of 1.0 or 100% compared to 76.5% accuracy of Cuckoo sandbox. In the future, this study aims to improve and refine this finding by using a more up to date but unstable version of Cuckoo in Cuckoo 3 which supports the latest version of python which would also by tangent enable the researcher to implement automation in both solutions to explore the wider dataset. Furthermore, future study should focus on more qualitative measurements such as ease of use, quality of documentation and community support and integration between these two solutions.

Acknowledgment

The authors would like to thank the Faculty of Computer Science and Information Technology, University Tun Hussein Onn Malaysia for its support.

References

[1] George-Andrei Iosif, "GitHub - DikeDataset: Dataset with labeled benign and malicious files," *GitHub*, 2022.

[2] Chiradeep Basu Mallick, "What Is Malware Analysis? Definition, Types, Stages, and Best Practices," *Spiceworks*, 2021.

[3] Sagar Khillar, "Difference Between Static Malware Analysis and Dynamic Malware Analysis. Difference Between Similar Terms and Objects.," *DifferenceBetween.net*, 2022.

[4] C. Lim, J. T. Juwono, and A. Erwin, "A Comparative Study of Behavior Analysis Sandboxes in Malware Detection The Implementation of the Aeronautical Telecommunication Network Protocol in the Linux Kernel View project DNS Dataset & Data Mining View project A Comparative Study of Behavior Analysis Sandboxes in Malware Detection," 2016, Accessed: Nov. 14, 2022. [Online]. Available: https://www.researchgate.net/publication/301204666

[5] Kurt Baker, "Malware Analysis Explained | Steps & Examples," *CrowdStrike*, 2023.

[6] A. Walker, M. F. Amjad, and S. Sengupta, "Cuckoo's malware threat scoring and classification: Friend or foe?," *2019 IEEE 9th Annual Computing and Communication Workshop and Conference, CCWC 2019*, pp. 678–684, Mar. 2019, doi: 10.1109/CCWC.2019.8666454.

[7] C. Leka, C. Ntantogian, S. Karagiannis, E. Magkos, and V. S. Verykios, "A Comparative Analysis of VirusTotal and Desktop Antivirus Detection Capabilities," pp. 1–6, Sep. 2022, doi: 10.1109/IISA56318.2022.9904382.

[8] A. Salem, S. Banescu, and A. Pretschner, "Maat: Automatically Analyzing VirusTotal for Accurate Labeling and Effective Malware Detection," *ACM Transactions on Privacy and Security*, vol. 24, no. 4, Jul. 2020, doi: 10.1145/3465361.

[9] C. Wei, Q. Li, D. Guo, and X. Meng, "Toward Identifying APT Malware through API System Calls," *Security and Communication Networks*, vol. 2021, 2021, doi: 10.1155/2021/8077220.

[10] Y. Gao, H. Hasegawa, Y. Yamaguchi, and H. Shimada, "Malware Detection using Attributed CFG Generated by Pre-trained Language Model with Graph Isomorphism Network," *Proceedings - 2022 IEEE 46th Annual Computers, Software, and Applications Conference, COMPSAC 2022*, pp. 1495–1501, 2022, doi: 10.1109/COMPSAC54236.2022.00237.

[11] T. Helmy, I. Keshta, and A. Rashed, "Performance evaluation of system resources utilization with sandboxing applications," *Lecture Notes in Electrical Engineering*, vol. 339, pp. 475–482, 2015, doi: 10.1007/978-3-662-46578-3_56.

[12] M. Alazab, S. Venkataraman, and P. Watters, "Towards understanding malware behaviour by the extraction of API calls," *Proceedings - 2nd Cybercrime and Trustworthy Computing Workshop, CTC 2010*, pp. 52–59, 2010, doi: 10.1109/CTC.2010.8.