Naif Arab University for Security Sciences
Journal of Information Security and Cybercrimes Research
مجلة بحوث أمن المعلومات والجرائم السيبرانية
https://journals.nauss.edu.sa/index.php/JISCR

JISCR

Cyber Security Incident Response: The Effectiveness of Open-Source Detection Tools in DLL Injection Detection

Ali Abuabid*, Abdulrahman Aldeij
College of Computing and Informatics, Saudi Electronic University, Abha, Kingdom of Saudi Arabia.

Received 11 Jan. 2024; Accepted 25 Mar. 2024; Available Online 02 Jun. 2024.

Abstract

In response to the growing cyber-attack threat, incident response teams have become a critical component of an organization's cybersecurity strategy. These teams are responsible for detecting, analyzing, and responding to security incidents promptly and effectively. However, detecting code injection attacks can be particularly challenging, as they can be difficult to detect and often go unnoticed until it is too late. Cybersecurity professionals use detection tools to detect and respond to DLL injection attacks that monitor system activity and detect unusual behavior. A large portion of the related literature focuses on the use of commercial DLL injection tools. In contrast, little attention has been paid to the effectiveness of using open-source DLL injection detection tools. Thus, this research project aims to evaluate the effectiveness of three widely used open-source tools, VirusTotal, Sysinternals, and Yara, in detecting DLL injection incidents. This study's findings highlight each tool's strengths and limitations, which in turn enables cybersecurity professionals to make informed decisions when selecting the most suitable tool for DLL injection detection. Furthermore, the study emphasizes the importance of continuous tool development and updates to keep pace with evolving malware techniques and emerging threats. By highlighting the effectiveness of the tools, this research enhances the overall security posture of organizations and individuals, empowering them to mitigate the risks associated with DLL injection attacks proactively. The outcomes of this research project also underscore the significance of leveraging advanced tools to fortify cybersecurity defenses and safeguard critical systems and data.

I. Introduction

In recent years, the threat of cyber-attacks has increased exponentially, making it critical for organizations to have effective incident response plans to mitigate the impact of cybersecurity incidents. One such type of cyber-attack is code injection, in which an attacker inserts malicious code into a legitimate application to exploit vulnerabilities and gain unauthorized access to a system. Code injection can occur in many forms, including SQL injection, command injection, and Dynamic Link Library (DLL) injection [1].

Analysis of DLL injection attacks reveals significant consequences, with a majority resulting

Keywords: Open-source, DLL injection, VirusTotal, Sysinternal suite, Yara, Incident response.

Production and hosting by NAUSS

* Corresponding Author: Ali Abuabid
Email: A.abuabid@seu.edu.sa
doi: 10.26735/PNOB5534

in severe issues. Among 103 identified studies, it was found that 90.3% of attacks caused system crashes, while 3.9% resulted in the loss of user request responses. Further investigation into the origins of these attacks indicated that 55.3% were attributed to antivirus software, 19% to hardware vendor drivers, and 10% to malware [2].

DLL injection, in particular, is a widespread technique attackers use to inject malicious code into a system [3]. This technique involves injecting malicious code into a DLL file, which is a file that contains code and data that multiple programs can share. Code injection attacks using DLL files have become increasingly prevalent in recent years, and organizations need to have practical detection tools in place to identify and respond to such attacks. Code injection can lead to severe consequences, including the theft of sensitive information, disruption of operations, and damage to an organization's reputation [1].

In response to the growing cyber-attack threat, incident response teams have become a critical component of an organization's cybersecurity strategy [4]. These teams are responsible for detecting, analyzing, and responding to security incidents promptly and effectively. However, detecting code injection attacks can be particularly challenging, as they can be difficult to detect and often go unnoticed until it is too late. To detect and respond to DLL injection attacks, cybersecurity professionals use detection tools that monitor system activity and detect unusual behavior [5]. Detection tools can help identify the presence of malicious code injected into a DLL file and alert security professionals to act accordingly.

It is observed that the existence of two types of DLL detection tools: commercial and open source. Commercial tools are typically designed for easy expansion and management, and they often come with a graphical user interface to help you visually understand the exact security posture and make faster and better decisions [6]. Despite their powerful capabilities in detecting malware and DLL injection attacks, they are relatively active and require valid proprietary licenses [7]. Some examples of commercial DLL detection tools include:

A. Kaspersky Endpoint Security

It is an endpoint protection platform designed to defend against various cyber threats, including malware, ransomware, and advanced persistent threats. While Kaspersky Lab, the company behind Kaspersky Endpoint Security, provides detailed information about the features and capabilities of their products, specific studies or research papers focusing solely on DLL injection detection with Kaspersky Endpoint Security may be limited [8].

However, Kaspersky Endpoint Security incorporates various technologies and features that contribute to its ability to detect DLL injection attacks, such as:

Behavioral Analysis: Kaspersky Endpoint Security employs behavioral analysis techniques to monitor system processes and behaviors for signs of malicious activity, including suspicious DLL injections. By analyzing the behavior of processes in real-time, the solution can identify abnormal activities indicative of DLL injection attempts [8]. Signature-based Detection: Kaspersky Endpoint Security utilizes a signature database of known malware and malicious DLLs to detect and block threats. When a file or DLL matches a signature in the database, it is flagged as malicious, and appropriate actions are taken to mitigate the threat. Moreover, heuristic Analysis: The solution employs heuristic analysis algorithms to identify previously unseen or unknown threats, including novel DLL injection techniques. Kaspersky Endpoint Security can detect and block suspicious DLL injections based on their characteristics by analyzing file attributes, behavior patterns, and code structures [9]. Integration with Threat Intelligence: Kaspersky Endpoint Security integrates with Kaspersky

Lab's threat intelligence network, providing real-time updates and insights into emerging threats, including DLL injection techniques used by cybercriminals. This integration enhances the solution's ability to detect and respond to evolving threats on time [9].

B. AppGuard

It is an endpoint security solution designed to prevent malware and unauthorized processes from compromising system integrity. While not specifically marketed as a DLL injection detection tool, AppGuard's application control and behavior-based protection approach can indirectly mitigate the risks associated with DLL injections [10]. The key features of AppGuard that contribute to its effectiveness in preventing DLL injection attacks include application whitelisting: it uses application whitelisting to restrict execution to only trusted applications. By maintaining a whitelist of approved applications, AppGuard prevents unauthorized or malicious processes, including those associated with DLL injections, from running on the system [10]. Moreover, isolation and containment: AppGuard employs isolation and containment techniques to prevent malware from accessing critical system resources. By isolating processes and restricting their access to sensitive areas of the system, AppGuard can thwart malware attempts to inject DLLs into legitimate processes or system components [10].

C. Symantec Endpoint Protection (SEP)

Symantec Endpoint Protection (SEP) is renowned as a widely adopted endpoint security solution, adept at thwarting many malware variants, including those employing DLL injection techniques. Several key features bolster SEP's efficacy in DLL injection detection. Firstly, Behavioral Analysis: SEP employs sophisticated behavioral analysis methodologies to scrutinize system processes and behaviors, vigilantly monitoring for any indications of malicious activity, including suspicious DLL injections. Through real-time process behavior analysis, SEP swiftly identifies anomalies suggestive of DLL injection attempts [11].

Secondly, signature-based detection: leveraging an extensive signature database teeming with known malware and malicious DLLs, SEP can swiftly recognize and neutralize threats. Upon detecting a file or DLL that matches an entry in the database, SEP promptly flags it as malicious and takes requisite measures to mitigate the potential threat.

Thirdly, heuristic analysis: SEP integrates advanced algorithms to identify previously unseen or emerging threats, including novel DLL injection techniques. By scrutinizing file attributes, behavior patterns, and code structures, SEP adeptly identifies and obstructs suspicious DLL injections based on their unique characteristics [11].

Finally, integration with threat intelligence: seamlessly meshing with Symantec's expansive global threat intelligence network, SEP remains constantly abreast of emerging threats, including cybercriminals' latest DLL injection methodologies. This seamless integration furnishes SEP with real-time updates and insights, enhancing its capacity to detect and counter evolving threats promptly[3]. Through the amalgamation of these cutting-edge features, Symantec Endpoint Protection stands as a stalwart guardian against DLL injection attacks, safeguarding endpoints with unparalleled efficacy and agility [2].

On the other hand, open-source DLL detection tools are free and often have a more limited feature set than commercial tools [12]. However, they can be a good option if users are on a tight budget or need to customize the tool to meet their needs [13]. Some examples of open-source DLL detection tools include:

1. Process Hacker
2. DLL Hijack Auditor
3. Yara

4. VirusTotal
5. Sysinternal

The effectiveness of open-source detection tools in identifying DLL injection attacks remains an open question. While detection tools have improved over the years, hackers have also become more sophisticated, and they can easily evade detection by using advanced techniques. To overcome such issues, detection tools have been developed to help incident response teams identify code injection attacks early. These tools use various techniques to monitor system activity, identify anomalies, and alert security personnel when suspicious activity is detected [14]. However, the effectiveness of these tools in detecting DLL injection attacks is not well identified, and further research is needed to evaluate their efficacy. This research project is initiated to evaluate the effectiveness of three widely used open-source DLL injection detection tools: VirusTotal, Sysinternals, and Yara. By examining the capabilities and limitations of these tools, the researchers aim to better understand how incident response teams can best leverage them to improve their ability to detect and respond to DLL injection attacks. Furthermore, this study highlights the critical limitation of current open-source detection tools and identifies areas for improvement to enhance the effectiveness of incident response in such attacks. This limitation is that open-source detection tools often require manual operation, which can be time-consuming and potentially prone to human error. To address this limitation, the researcher developed a script that automates detecting DLL injection attacks, reducing the manual effort required by cybersecurity professionals and improving the speed and accuracy of incident response. Using such a script to automate the detection tools provides a solution that can enhance the effectiveness of incident response in detecting DLL injection attacks.

This research contributes to cybersecurity incident response in several ways. Firstly, it comprehensively reviews the current detection tools used to detect DLL injection attacks and their limitations. This review will serve as a valuable resource for cybersecurity professionals, helping them select the most effective detection tools for their needs.

Secondly, this research evaluates the effectiveness of detection tools in detecting DLL injection attacks through a controlled experiment that simulates real-world attacks. The experiment will test several commonly used detection tools against various DLL injection attacks, including those using evasion techniques. This research provides insights into the effectiveness of these detection tools and the limitations of their current capabilities.

Thirdly, this research identifies the evasion techniques attackers use to avoid detection by detection tools. Identifying these techniques helps cybersecurity professionals develop countermeasures and improve the effectiveness of detection tools. Moreover, this research proposes ways to counteract these evasion techniques, which enhance the accuracy of detection tools and minimize false alarms.

Finally, it provides recommendations for improving the effectiveness of detection tools and cybersecurity incident response, which include enhancing the capabilities of detection tools to detect DLL injection attacks and developing new approaches to mitigate the damage caused by such attacks.

D. Research Objectives

This study assesses the efficacy of detection tools in identifying and responding to DLL injection attacks. The specific objectives of this research are as follows:

- To evaluate and compare the effectiveness of Sysinternals Suite and VirusTotal in detecting DLL injection attacks.

- To assess and compare the effectiveness of Sysinternals Suite and Yara in identifying DLL injection attempts.
- To analyze and compare the effectiveness of VirusTotal and Yara in detecting DLL injection activities.
- To develop a script capable of automating the detection process for VirusTotal and Yara, streamlining the identification of DLL injection threats.

The rest of the study is structured as follows: literature analysis of the open-source DLL detection tools, the research approach employed in this study, research results and discussions, and conclusion, which includes research limitation, contribution, and future work.

II. Literature Review

This section critically analyzes the background literature related to the research questions mentioned above. It consists of four subsections. Each section compares two of the top three open-source DLL Injection Detection tools in terms of detection mechanism, effectiveness, and limitations of each other and formulate a hypothesis after evaluating each.

A. Comparison Between Sysinternal and VirusTotal

Cybersecurity is a critical concern in today's digital age, and cybersecurity tools have become increasingly essential to protect against cyber-attacks. Sysinternal Suites and VirusTotal are two commonly used tools in the field of cybersecurity.

Sysinternal Suites is a suite of more than 70 tools developed by Mark Russinovich and Bryce Cogswell, which are used for analyzing and troubleshooting Windows systems [15]. The suite includes tools for monitoring system activity, managing processes, and diagnosing system issues. Sysinternal Suites is a powerful tool that can be used to identify and mitigate security vulnerabilities in Windows systems.

VirusTotal, on the other hand, is an online tool that analyzes files and URLs for viruses, malware, and different types of malicious content [16]. VirusTotal uses multiple antivirus engines and other tools to scan files and URLs for malicious content. The tool can also provide detailed reports on the behavior of files and URLs, making it a valuable tool for cybersecurity professionals.

Both Sysinternal Suites and VirusTotal are valuable tools for cybersecurity professionals. However, they serve different purposes. Sysinternal Suites is primarily used for analyzing and troubleshooting Windows systems, while VirusTotal analyzes files and URLs for malicious content [16]. Therefore, each tool's effectiveness depends on the user's specific needs. The choice of which tool to use depends on the user's particular needs. Sysinternal Suites is ideal for analyzing and troubleshooting Windows systems, while VirusTotal is ideal for analyzing files and URLs for malicious content [17].

Christopher and Raychaudhuri [18] provided an overview of the Sysinternals Suite and its various tools, including Process Monitor, Process Explorer, and Autoruns. They describe how these tools can analyze virtual machine hard disks, mainly when the virtual machine is offline or in a "frozen" state. One of the key strengths of the Sysinternals Suite in this context is its ability to provide detailed information about system processes and their interactions with other processes. The authors note that tools such as Process Monitor can be used to monitor system processes and identify any unusual or suspicious activity. They also highlight the importance of using the Sysinternals Suite with other digital forensics tools and techniques. While the Sysinternals Suite can provide valuable information, it may not be sufficient on its own to analyze and understand a digital forensic case fully. Overall, it demonstrates the usefulness of the Sysinternals Suite in digital forensics investigations, particularly in the analysis of virtual machine hard disks. It is recommended that digital forensics practitioners become familiar

with the various tools in the suite and learn how to use them effectively in different scenarios.

Sysinternals is not primarily designed for malware analysis, but it can be used in some instances. For example, some of the tools in the Sysinternals suite, such as Process Explorer and Autoruns, can be used to identify and analyze suspicious processes and startup items on a Windows system [15]. However, these tools are not explicitly designed for malware analysis and may not be as effective as dedicated malware analysis tools like VirusTotal.

In contrast, VirusTotal is a specialized tool designed for malware analysis. It uses signature-based and behavior-based detection techniques to identify malware and other types of malicious content [14]. VirusTotal provides detailed information about the analyzed files, including the names of the malware families they belong to, the AV labels assigned to them, and the detection rates of various antivirus engines [16]. VirusTotal is a valuable tool for cybersecurity professionals who need to analyze files and URLs for malicious content.

Cozzi et al. [17] used VirusTotal to extract AV labels for malware samples and fed them to the AVClass tool to obtain a normalized name for the malware family. Moreover, [19] used VirusTotal to detect the classes of malware in their dataset. The key findings of the study were that VirusTotal was effective at detecting both phishing and malware URLs, with a high percentage of URLs being identified as malicious by the scanning engines used by VirusTotal. It was also found that most phishing URLs were hosted on compromised websites rather than standalone pages designed to look like legitimate websites. The study also highlighted some limitations of VirusTotal, particularly when detecting new or previously unknown threats. The authors noted that some URLs initially classified as benign by VirusTotal were later found to be malicious, suggesting that the scanning engines used by VirusTotal may not be able to detect all types of threats. Overall, the study demonstrated the effectiveness of VirusTotal in detecting phishing and malware URLs but also highlighted the need for additional tools and techniques to supplement VirusTotal's capabilities to detect new and emerging threats.

In addition, there have been studies that have used other tools and techniques for malware analysis, such as static feature extraction from ELF binaries [17], deep learning-based detection and classification of Android malware using API-based features [20], and behavioral reports of multi-stage malware. However, these studies did not specifically compare Sysinternals and VirusTotal.

Both Sysinternals and VirusTotal can be used for malware analysis; they serve different purposes and have different strengths and weaknesses. VirusTotal is a specialized tool designed explicitly for malware analysis and provides a wealth of information about the analyzed files [16]. At the same time, Sysinternals is a suite of tools that offers advanced system information and troubleshooting capabilities for Windows-based systems and can be used for malware analysis in some instances [15]. The literature has used VirusTotal for malware analysis and detection, but there have yet to be studies that specifically compare Sysinternals and VirusTotal. Therefore, for this study, it is suggested that Sysinternal is more effective in detecting DLL Injection than VirusTotal. Hence, the following hypothesis is formulated:

H1: Sysinternal is more effective than VirusTotal in DLL injection detection.

B. Comparison Between Sysinternal and Yara

The Sysinternal Suite is a collection of system utilities and tools created by Microsoft for Windows [15]. On the other hand, Yara is an open-source tool used for malware analysis, detection, and classification.

Sysinternals is a suite of system utilities designed to help diagnose and troubleshoot Windows-

based computers. At the same time, Yara is a tool for pattern matching used by malware analysts worldwide [21]. Yara can scan files and process memory, allowing users to define sequences of symbols as text strings, hexadecimal strings, and regular expressions [21]. However, the use of regular expressions is limited because of the concern that it can slow down the scanning process [21]. Moreover, Yara is a tool specifically designed for pattern matching and used by malware analysts to detect and analyze malware [21]. It rules discover malware based on a string-matching technique, which can be customized depending on the specific requirement of an individual or organization to uncover targeted attacks and security threats. The quality and quantity of the Yara rules are crucial for analytic success, as there should be an effective and sufficient number of Yara rules to improve the overall performance of the malware analysis [22].

Regarding usability, both Sysinternals and Yara are easy to use and provide a simple and intuitive interface. Sysinternals provides a unified interface for all its tools, making it easy for users to find the tools they need [23]. In contrast, Yara provides a simple and easy-to-use interface that allows users to view the exported functions of a DLL file quickly.

To sum up, both tools are well-documented and provide helpful information to users. They are two software programs that serve different purposes. Sysinternals is a suite of system utilities designed to help diagnose and troubleshoot Windows-based computers [15]. At the same time, Yara is a tool for pattern matching used by malware analysts worldwide [22]. While both tools are easy to use and provide a simple and intuitive interface, Sysinternals offers a broader range of tools and functionality, while Yara is a more specialized tool that is designed specifically for pattern matching and malware analysis. Sysinternal Suite and Yara are valuable tools in cybersecurity investigations. Although Sysinternal Suite is more effective in monitoring and analyzing system activity, providing valuable insights into malware behavior, Yara is more effective in detecting and classifying malware through its pattern-matching algorithms. Ultimately, the choice between the two tools depends on the specific needs of the investigation and the type of malware being analyzed. Therefore, for this study, it is suggested that Sysinternal is more effective than Yara in detecting DLL injection. Hence, the following hypothesis is formulated:

H2: Sysinternal is more effective than Yara in DLL injection detection.

C. Comparison Between VirusTotal and Yara

Two popular tools for malware analysis are VirusTotal and Yara. Cybersecurity experts have widely used both tools for detecting and analyzing malware. VirusTotal and Yara are both tools used for malware detection. VirusTotal is a large-scale malware detection system integrating machine learning with expert reviewers [5]. It treats reviewers as a limited labeling resource and demonstrates that even in small numbers, reviewers can vastly improve the system's ability to keep pace with evolving threats [14]. VirusTotal can scan files and process memory, and it allows defining sequences of symbols as text strings, hexadecimal strings, and regular expressions [21].

On the other hand, Yara is a tool for pattern matching used by malware analysts worldwide [21]. It can use sandbox memory dumps for the identification of malware families and can scan files as well as process memory. Yara allows defining sequences of symbols as text strings, hexadecimal strings, and regular expressions [21]. However, the use of regular expressions is limited because of the concern that it can slow down the scanning process [21]. In terms of performance, [5] found that VirusTotal achieved 72% detection without reviewer assistance. Meanwhile, Yara is a pattern-matching technique using sandbox memory dumps to identify malware families [23]. However, pattern-matching techniques fail silently due to minor code variations, leading to unidentified malware samples [23].

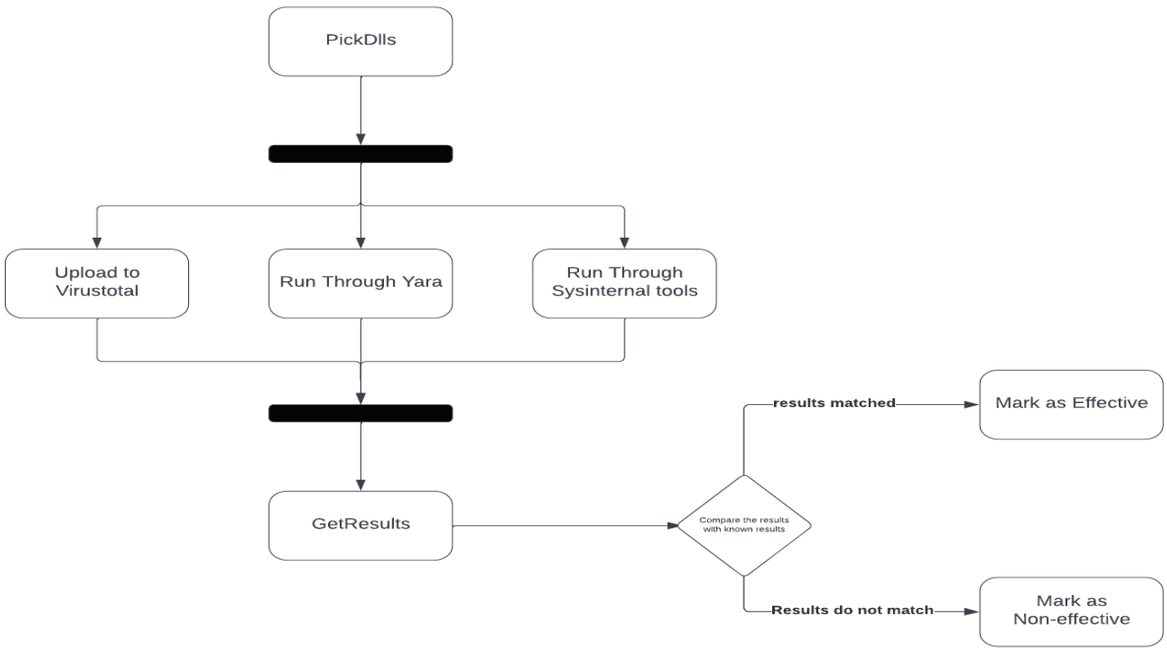

Fig. 1 The Diagram of the Methodology Used.

Both VirusTotal and Yara are valuable tools for malware detection. VirusTotal integrates machine learning with expert reviewers, while Yara is a tool for pattern matching used by malware analysts worldwide. Both tools allow for defining sequences of symbols as text strings, hexadecimal strings, and regular expressions. However, the use of regular expressions in Yara is limited because of the concern that it can slow down the scanning process. While VirusTotal achieved 72% detection without reviewer assistance, pattern-matching techniques in Yara failed silently due to minor code variations, leading to unidentified malware samples.

When comparing VirusTotal and Yara, several factors should be considered, including their effectiveness for detecting known and unknown malware samples, their false-positive rate, and their ease of use [24]. In terms of effectiveness for detecting known malware samples, both tools are effective, with VirusTotal having a detection rate of 80.2% and Yara having a detection rate of 90.2%. However, when detecting new and unknown malware samples, Yara is more effective than VirusTotal, with a detection rate of 82.8% compared to VirusTotal's detection rate of 47.7%. In terms of false-positive rates, VirusTotal has a higher false-positive rate. Therefore, for this study, it is suggested that VirusTotal is less effective in detecting DLL Injection than Yara. Hence, the following hypothesis is formulated:

H3: VirusTotal is less effective than Yara in detecting DLL Injection.

III. Research Method

This study will use an experimental research design. The study aims to evaluate the effectiveness of three open-source detection tools in identifying and responding to code injection cyber security incidents. The research design involves data collection through an automated script to check the injection on DLL files. Fig. 1 shows the diagram of the methodology used in the research project.

A. Data Collection

This research utilizes two datasets: a malicious dataset sourced from MalwareBazaar [25] and a

TABLE I
SUMMARY OF OPEN-SOURCE DLL INJECTION DETECTION TOOLS

	Sysinternals Suite	VirusTotal	Yara
Developed By	Mark Russinovich and Bryce Cogswell	Hispasec Sistemas	Victor Manuel Alvarez
Developed Year	1996	2004	2007
Short Description	It was acquired by Microsoft in 2006.	It was acquired by Google in 2012.	It is available as an open-source tool under the Apache License 2.0.
Underline Theory	Sysinternals includes various utility software tools that use different techniques and theories to help IT professionals and system administrators troubleshoot and diagnose issues with Windows systems. These tools include process monitoring and analysis, file and disk utilities, network utilities, and more.	VirusTotal uses signature-based and behavior-based detection methods to identify malware and viruses. It relies on multiple antivirus engines and other security tools to analyze and identify potential threats in submitted files, URLs, and IP addresses.	Yara uses pattern matching to identify specific characteristics or behavior patterns associated with malware. Analysts can create custom rules based on strings, regular expressions, and other criteria to detect and categorize malware.
Platform	Windows	Windows, Mac, and Linux	Windows, Linux, and Mac

TABLE II
STRENGTHS AND LIMITATIONS OF OPEN-SOURCE DLL INJECTION DETECTION TOOLS

Criteria	Sysinternals Suite	VirusTotal	Yara
Strengths	Provides real-time monitoring of system activity	Utilizes multiple antivirus engines for scanning	Flexible rule-based approach for malware detection
	Offers detailed event information and filtering	Performs behavioral analysis for detecting threatsv	Efficient pattern matching for file analysis
	Enables stack tracing to identify code origins	Provides historical analysis of scanned files	Modular and shareable rules for community use
	Powerful toolset for system troubleshooting	Allows community contributions and comments	Integration with various security systems
Limitations	Not specifically designed for detecting DLL injection	Detection depends on available antivirus engines	Creation of effective rules requires expertise
	Limited to monitoring and analysis, not removal	False positives/negatives possible with some engines	Rules need to be regularly updated and maintained
	Limited support for automated scanning	Limited to file and URL scanning, not real-time	Relies on rule quality and specificity for accuracy

benign dataset sourced from [26]. The malicious dataset contains samples of known malware, while the benign dataset contains clean files that are not associated with any malicious activity. By analyzing these two datasets, the researchers can gain insights into the characteristics and behaviors of malware and develop more effective detection and prevention strategies.

1) MalwareBazaar

It is designed to be a community-driven repository of malware samples where users can upload and share their samples or browse and download samples uploaded by others. The MalwareBazaar website provides a user-friendly interface for browsing and searching the collection of malware samples, as well as for analyzing individual samples

using various tools and services [25]. The website also offers APIs for programmatically accessing and interacting with the malware samples. The file type filter is specified for listing only the required DLL files while searching.

2) DLL-FILES.COM

It is a website that provides an extensive collection of Dynamic Link Library (DLL) files that users can download and use to fix issues related to missing or corrupted DLL files on their Windows operating system. The DLL files uploaded on it are trusted and safe. In this research, therefore, about 80 random DLL files have been used as the dataset of secure files.

B. Open-Source Detection Tools

Three detection tools are tested in this study to evaluate their effectiveness in detecting DLL injection. These tools are selected based on their popularity, availability, and reputation within the industry. The selected tools are Microsoft Sysinternals Suite, Yara, and VirusTotal. Table I shows the background and description of each of these tools. Table II shows the strengths and limitations of each open-source DLL injection detection.

1) Microsoft Sysinternals Suite

Microsoft Sysinternals provides technical resources and utilities for managing, diagnosing, troubleshooting, and monitoring Microsoft Windows environments [15]. Known initially as NTInternals, the suite was founded in 1996 by software developers Bryce Cogswell and Mark Russinovich and was operated by Winternals Software LP in Austin, Texas [27]. Microsoft acquired Winternals and its assets on July 18, 2006. The suite offers a range of freeware tools for administering and monitoring Windows computers, which can now be found on the Microsoft website. Microsoft tools, such as Process Explorer, AutoRuns, Procmon, etc., can be used to find the injected DLL files.

This section shows a detailed description of Procmon's features, functionalities, and how it can contribute to the detection and analysis of DLL injection incidents:

1. Real-time Monitoring: Procmon captures and displays real-time system activity, providing a comprehensive view of process behavior, file system operations, registry modifications, network activity, and more. This enables users to monitor and analyze the activity of processes and identify any suspicious or unexpected behavior.
2. Filtering and Capture Options: Procmon offers robust filtering capabilities, allowing users to define specific criteria to capture or exclude particular events or processes. This feature helps focus the monitoring on relevant processes or activities related to DLL injection incidents, enhancing the efficiency of analysis.
3. Detailed Event Information: Each captured event in Procmon includes extensive details, such as the process name, process ID, timestamp, operation type, target file/registry key, result, and more. This level of granularity enables users to drill down into specific events, track the sequence of operations, and identify any abnormal or unauthorized DLL-related activities.

2) VirusTotal

VirusTotal is a free online service that allows users to upload files, URLs, or IP addresses to scan them for potential malware and viruses. It uses multiple antivirus engines and other security tools to analyze and identify any threats or malicious behavior in the submitted files. VirusTotal is a website created by the Spanish security company Hispasec Sistemas [14]. In June 2004, it was launched and acquired by Google in September 2012. Later, in January

2018, the company's ownership was transferred to Chronicle, a subsidiary of Google.

A description of VirusTotal's features, functionalities, and how it contributes to the overall analysis of DLL injection incidents:

1. File and URL Scanning: VirusTotal accepts file uploads and URLs, allowing users to scan a wide range of file types, including DLLs, as well as websites or web pages for potential threats. This feature provides a convenient way to assess the safety of specific files or URLs that may be associated with DLL injection attempts.
2. Multiple Antivirus Engines: VirusTotal employs a broad array of antivirus engines and detection mechanisms from different vendors. When a file is submitted for scanning, it is checked against these engines, which collectively provide a comprehensive assessment of potential malware or suspicious behavior. This multi-engine approach enhances the accuracy and effectiveness of detecting DLLs that may be involved in DLL injection incidents.

TABLE III
HARDWARE REQUIREMENTS OF THE ENVIRONMENT

Environment Components	Description
Host Operating System	Windows 11 Home
Processor	12th Gen Intel(R) Core (TM) i7-12650H
RAM	32 GB
Guest Operating system	Windows 10 Home Edition
Virtual Machine Software	Oracle VM VirtualBox 7.0
Tool #1	VirusTotal (Online Version)
Tool #2	Yara (Downloaded on Windows)
Tool #3	Sysinternal (Downloaded on Windows 10 VM)

3. Malware Detection and Analysis: VirusTotal checks files against known signatures and heuristics to identify known malware strains. It also performs behavioral analysis to detect potential threats that exhibit suspicious behavior, even if they are not yet identified by specific signatures. This analysis can help identify DLLs that may be part of a larger malware campaign or used for DLL injection purposes.

3) Yara

Yara is a tool used for malware analysis that allows analysts to create custom rules for identifying and categorizing malware based on specific characteristics. These rules can be based on various factors, including file metadata, code structure, and behavioral patterns. Yara can be used to detect DLL injection by creating custom rules that identify specific characteristics or behavior patterns associated with DLL injection.

This is a thorough explanation of Yara rules, covering their attributes, capabilities, and role in identifying DLL injection incidents:

1. Rule Creation: Yara rules are created using a simple and expressive syntax that allows security researchers and analysts to define patterns and conditions to identify specific characteristics within files. These rules can encompass a wide range of indicators, including file signatures, byte sequences, strings, entropy values, and more. Researchers can create custom Yara rules tailored to their specific requirements, such as detecting DLL injection techniques.
2. String and Byte Sequence Matching: Yara rules can include strings or byte sequences that are indicative of DLL injection. For example, a Yara rule can include specific strings related to injection techniques or known DLLs involved in injection incidents. By scanning files or memory regions for

```python
import os
import hashlib

def get_files_in_directory(directory):
    file_list = []
    for file_name in os.listdir(directory):
        if os.path.isfile(os.path.join(directory, file_name)):
            file_list.append(file_name)
    return file_list
def calculate_md5(file_path):
    with open(file_path, 'rb') as file:
        md5_hash = hashlib.md5()
        while True:
            chunk = file.read(4096)
            if not chunk:
                break
            md5_hash.update(chunk)
    return md5_hash.hexdigest()
def is_prime(number):
    if number < 2:
        return False
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            return False
    return True
def write_to_file(filename, maliciousity):
    with open("output.txt", "a") as file:
        file.write(f"filename: {filename}, Maliciousity: {maliciousity}%\n")

files = get_files_in_directory(location)
totalfiles=len(files)
t=1
for num,i in enumerate(files):
    if is_prime(t):
        vtotal=virusTotalAPI(calculate_md5(location+"\\"+i),0)
        t=t+1
    else:
        vtotal=virusTotalAPI(calculate_md5(location+"\\"+i),1)
        t=t+1
    if vtotal['status']=="Not Found":
        print(str(num)+ " done out of " + str(totalfiles))
        continue
hrome
    write_to_file(i, vtotal['Maliciousity'])
    print(str(num)+ " done out of " + str(totalfiles))
```

Fig. 2 Virustotal Automation Script.

these strings or sequences, Yara rules can flag potential instances of DLL injection.

3. Metadata Extraction: Yara rules can extract metadata from files or memory regions being analyzed, such as file names, sizes, hashes, or version information. This metadata can be used within the rules to filter or further analyze potential DLL injection incidents. For example, rules can be designed to detect DLLs with suspicious or unexpected metadata properties.

C. Hardware Requirements

Table III shows an overview of the hardware requirements. The Windows machine is being used for testing purposes. All the tests are being performed on a Windows Virtual Machine with 16GB of RAM, 12th Gen Intel(R) Core(TM) i7-12650H 2.30 GHz processor. The virtual machine is being used to avoid damage to the host machine in case the malware damages the system.

IV. Implementation

This section identifies the primary tools utilized for effective detection as Yara, VirusTotal, and the Microsoft Sysinternals Suite. As previously discussed, the researcher compiled a collection of both malicious and non-malicious files. Within this implementation section, we will employ the aforementioned tools to analyze the dataset prepared earlier.

A. Using VirusTotal

VirusTotal API is being used to automate the detection of the maliciousness of the DLL files. Writing a script with Python is helpful as it was used to upload the DLL files to the web and automate the process. The VirusTotal API is being requested against all the files in the used dataset, and the result is stored in a text file showing whether the file is malicious or benign.

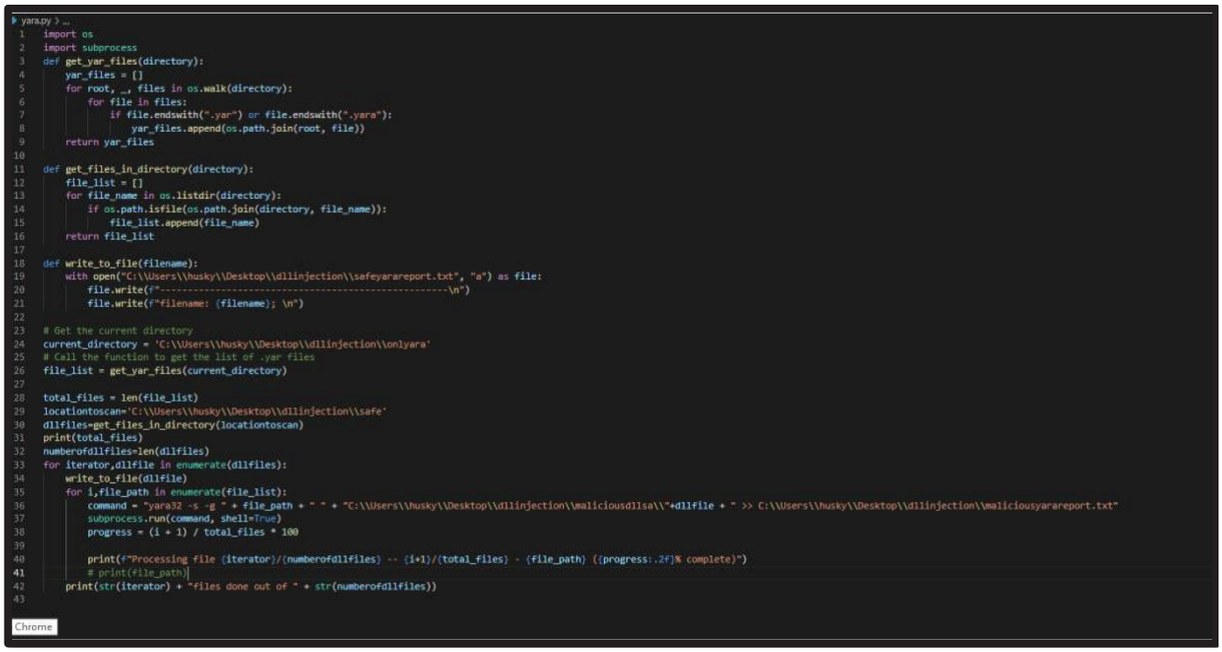

Fig. 3 Yara automation script.

1) Automating the Process

The process of scanning DLL files through VirusTotal is being automated using Python scripting. A Python script is being written that reads all the DLL files from a specified directory (that can either be a directory with all the malicious DLL files or a directory with all the non-malicious files). The Python script goes through every file in the directory. It calculates the hash of the file. Suppose the hash is available in the VirusTotal database. In that case, it just gives the stored results, or if the hash is not in the VirusTotal database, then it calculates the maliciousness of the DLL file by running it against different anti-virus software engines.

The provided code is a program that analyzes files in a specific directory using the VirusTotal API, which checks files for malware. The program calculates the maliciousity percentage for each file and writes the results to an output file. The code is given in Fig. 2.

Below is the list of functions that are being used in this script.

- VirusTotalAPI: This function interacts with the VirusTotal API to retrieve analysis results for a given file hash. It takes the file hash and an API key as parameters and returns a dictionary containing the analysis results.
- Get_files_in_directory: This function obtains a list of file names in a specified directory. It takes the directory path as input and returns a list of file names.
- Calculate_md5: This function calculates the MD5 hash of a file. It takes the file path as input and returns the MD5 hash as a hexadecimal string.
- Is_prime: This function checks whether a given number is prime. It takes a number as input and returns a boolean value indicating whether it is prime.
- Write_to_file: This function appends the filename and maliciously percentage to an output file. It takes the filename and maliciously as input and writes them to the output file.

2) Working on VirusTotal

VirusTotal contains a list of anti-virus software engines. It scans the provided file against all the anti-virus software's. VirusTotal stores all the results

in its database and provides them when needed. It also categorizes the critical features of the file, for example, if the file is malicious and represents some specific malware family or type. It has a community feature where security researchers share information about the specific sample.

B. Using Yara

Yara tool is being automated to find whether the DLL files are malicious or not. In this way, the effectiveness of the Yara tool is being determined. Different Yara rules from the open-source projects are being downloaded to make the results effective. These rules represent the signature of any malware, and these rules print out the rule and matching signature if any of the signatures match.

1) Automating the Process

The process of scanning DLL files through the Yara tool is being automated using Python scripting. A Python script is being written that reads all the DLL files from a specified directory (that can either be a directory with all the malicious DLL files or a directory with all the non-malicious files). The Python script goes through every file in the directory. It will run around 1000 Yara rules against all the malware files individually. If any signature matches, it prints out the Yara rule and matching strings. All the results of all the files are stored in a text file to analyze all the results correctly. Fig. 3 shows the code for the automation of the Yara tool.

This code is a script that scans a directory for Yara (.yar) files and then checks each DLL file in another directory against those Yara rules. It writes the results of the scan to a text file.

- The code first retrieves a list of Yara (.yar) files in a given directory.
- It then retrieves a list of DLL files in another directory and iterates over each DLL file.
- For each DLL file, it iterates over the list of Yara files and executes a Yara scan using the yara32 command. The results are appended to a text file.
- Progress of the scan is printed to the console, indicating the current file being processed and the overall progress.

The following are the functions in Yara automation script:

- *Get_yar_files(directory):* This function takes a directory path as input and returns a list of Yara (.yar) file paths found in that directory and its subdirectories. It uses the os.walk() function to traverse the directory and its subdirectories recursively, and filters the files based on their file extensions (.yar or .yara).
- *Get_files_in_directory(directory):* This function takes a directory path as input and returns a list of file names present directly in that directory (excluding subdirectories). It uses os.listdir() to get the list of all files in the directory and filters out any subdirectories using os.path. isfile().
- *Write_to_file(filename):* This function takes a filename as input and appends information about that file to a text file named "safeyarareport.txt". It opens the file in append mode using the open () function, writes the file information (such as the filename) using file. write(), and then closes the file. This function is used to record the processed DLL files in the report.

2) Working on Yara

Yara is a signature matching language, where a dataset is created with a list of malicious information. Using Yara, that information pattern and rules are being searched in a file. If that information exists in the newly searched file, it is considered malicious as it contains a malicious signature. It can help us identify the static string, pattern, and flow of an attack.

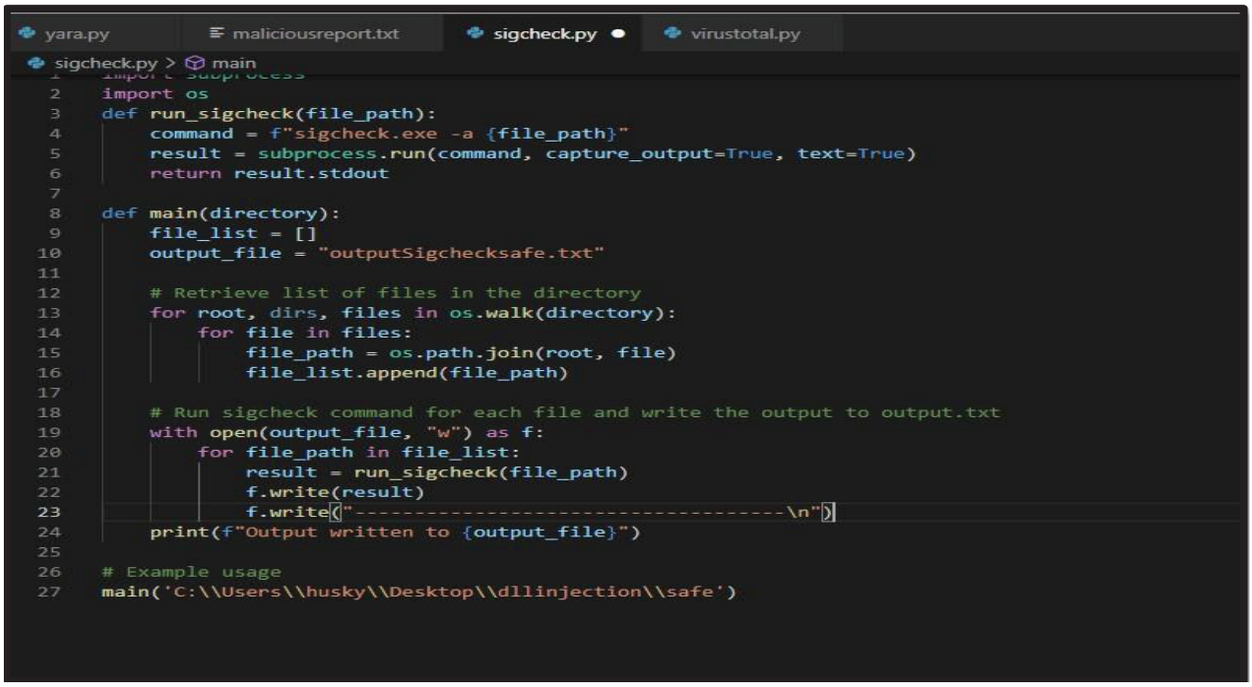

Fig. 4. Sigcheck Automation Script.

C. Using Sysinternal

There are many tools available in the Microsoft Sysinternal suite. These tools can be used for different purposes. A Sigcheck, Autoruns, and Procmon are used in this research project. Procmon and Autoruns are being manually analyzed, and Sigcheck is being automated using Python script.

1) Automating the Process

A Python script is being written for the automation process. This Python script takes a directory and puts all the files in this directory in a list. Then, it runs Sigcheck for each file in the directory and writes the results in the output file. Fig. 4 shows the Python code for the process.

The Python script utilizes the subprocess module to run an external command called Sigcheck.exe and capture its output. The script aims to scan files in a given directory using the Sigcheck command and write the output to "output.txt ". The following are the core functions of the script.

- ***Run_sigcheck****(file_path):* This function takes a file path as input, constructs a command string using the file_path, and runs the sigcheck.exe command with the constructed command. The output of the command is captured and returned as a string.
- ***Main****(directory):* This function is the main entry point of the script. It initializes an empty file_list and sets the output file name as "output.txt". It then retrieves the list of files in the provided directory using os.walk(), adds each file's path to file_list, and executes the run_sigcheck function for each file in file_list. The output of each execution is written to the output.txt file, separating each file's output with a line of dashes. Finally, it prints a message indicating the location of the output file.

2) Working on the Sysinternal

As mentioned earlier, Sysinternals tools are a collection of powerful utilities designed to help users and IT professionals analyze, diagnose, and troubleshoot various aspects of the Windows operating system [27]. Among the tools used in the project, three notable ones are Procmon,

Autoruns, and Sigcheck. Procmon captures real-time system activity, providing detailed information about files, registry, process, thread, and network events [27]. On the other hand, Autoruns identifies and manages auto-starting programs, services, and drivers, enhancing system performance and security. In contrast, Sigcheck is a command-line tool used to verify the digital signatures of files on Windows systems. It provides information about the signer, time stamp, and integrity of executable files, drivers, DLLs, and other system files. Sigcheck helps detect unsigned or tampered files, ensuring the integrity and authenticity of system files. It is valuable for security analysis and identifying potential risks or malware infections. In the research project, this tool is being used, and the sign status is given main priority. The problem with the other Sysinternal tools is being faced in the execution of the DLL files because of tools. like Procmon and Autoruns need the actual execution of the file. Then, they can store and present the activities of that file. However, in this case, the DLL files were not being executed because they cannot be executed stand-alone. An attached .exe file is needed to call the functions inside the DLL files.

V. Research Results and Discussions

This section visually represents the data through graphs, charts, or other visual aids. It thoroughly discusses the findings obtained from the analysis, delving into their significance and implications within the study context. Additionally, it compares these findings to the initial hypotheses or research questions posed at the investigation's outset. Through this comparative analysis, the section aims to evaluate the extent to which the data aligns with the anticipated outcomes and contributes to the overall understanding of the research topic.

A. VirusTotal

In this research project, the results of all the anti-viruses are being given priority; they are also being used to decide the maliciousness of the sample. The script automates the process and writes the results in the text file. The text file is also shown below. In this example, many of the files are being shown as malicious with 1%, which is considered a false positive, and those are being considered non-malicious because of the very low percentage. Fig. 5 shows the result of the malicious DLL files. While Fig. 6 shows the output of the benign DLL files.

Effectiveness of VirusTotal

Out of 80 malicious files, VirusTotal marked 45 as malicious, and out of 80 non-malicious files, VirusTotal marked 79 as safe. Hence, regarding finding if the file is safe, VirusTotal is very effective for finding malicious files and is around 56% effective in detecting malicious DLLs and around 99% of true positive safe DLLs.

Dependence of Effectiveness

The effectiveness of VirusTotal depends on the popularity of the malware sample. If the malware is being previously known and the same malware, techniques, or signatures used again, then it is very effective. However, if the zero-day attack happened, it would be less effective as the database of the zero-day attack is not well known in the market.

B. Yara

The number of string matching and speed of the string matching are being given priority. The script automates finding if the file contains any of the malicious signatures defined in the malicious signature database. In this case, even a single positive result can lead us toward the conclusion that the DLL is being malicious. Fig. 7 shows the results of the malicious files when the Yara rule is being run on the malicious directory. Whereas Fig. 8 shows the results of Yara rules when the files are safe.

Effectiveness of Yara

Out of 80 Malicious files, Yara marked 42 as malicious, and out of 80 non-malicious files, Yara marked 78 as safe. This makes the tool 52%

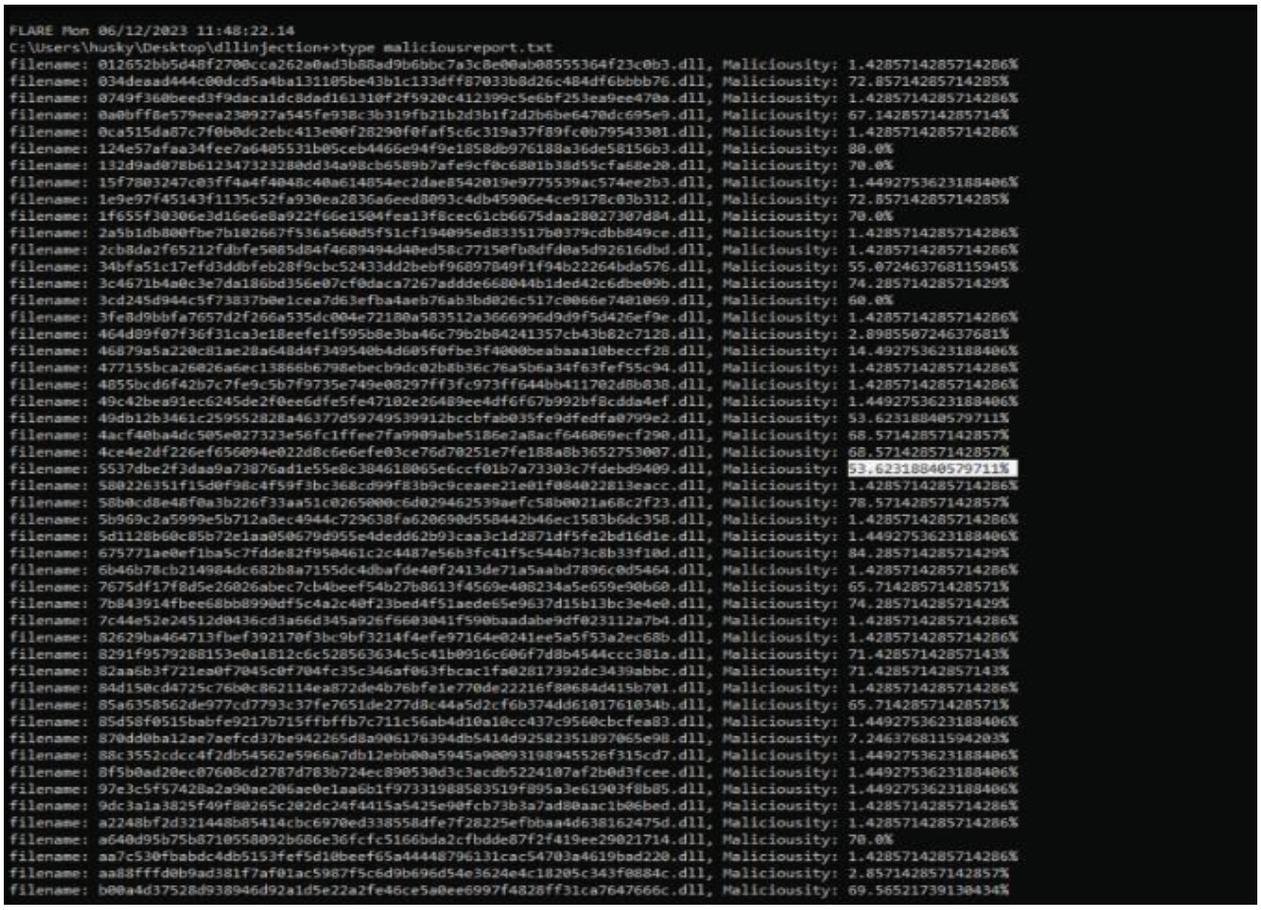

Fig. 5. Output of VirusTotal for Malicious Dataset.

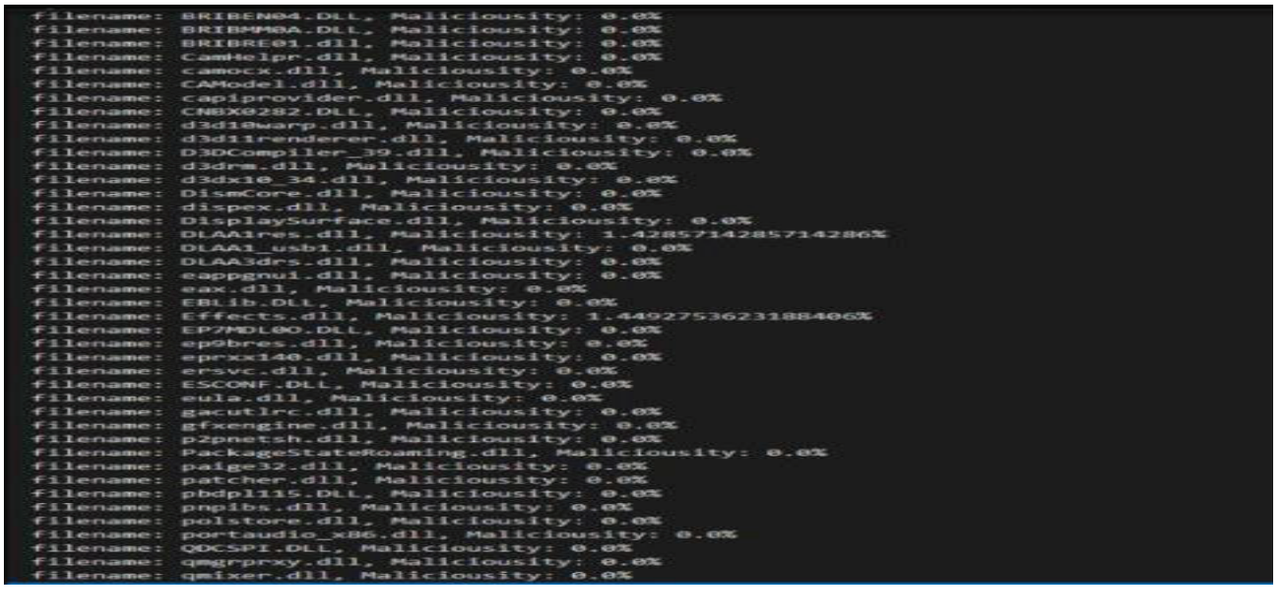

Fig. 6 Output of Virustotal for Safe Dataset.

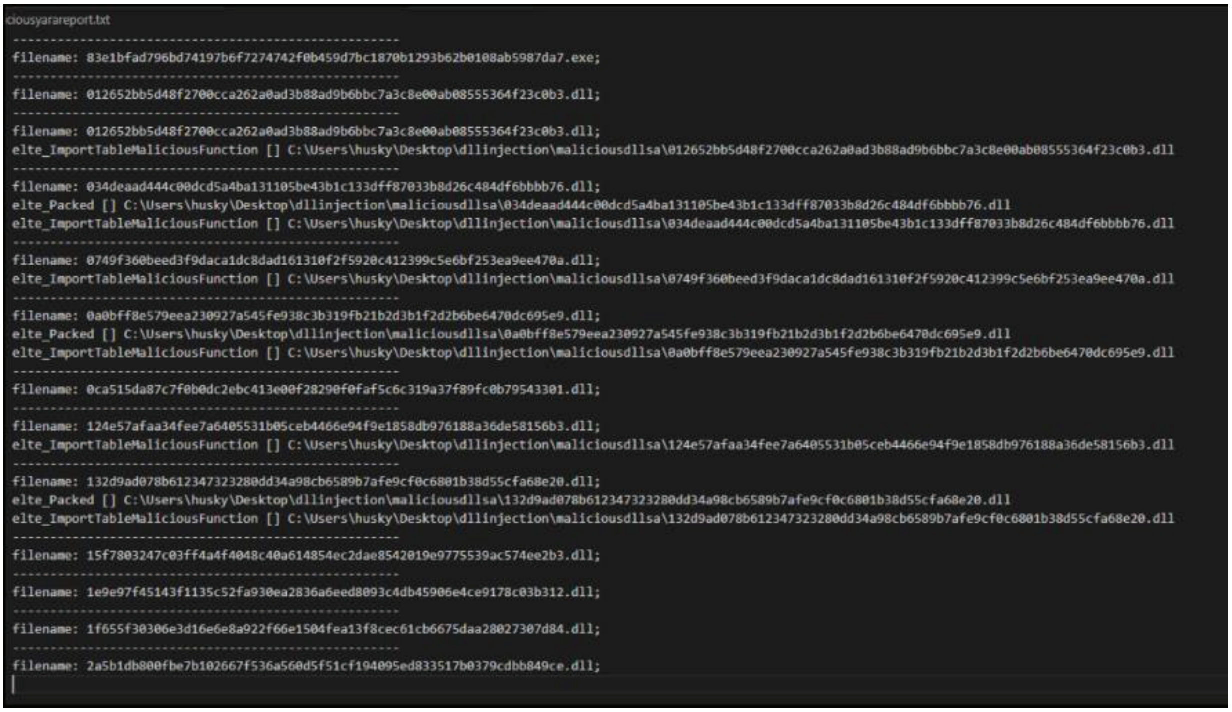

Fig. 7 Output of Yara for Malicious Dataset.

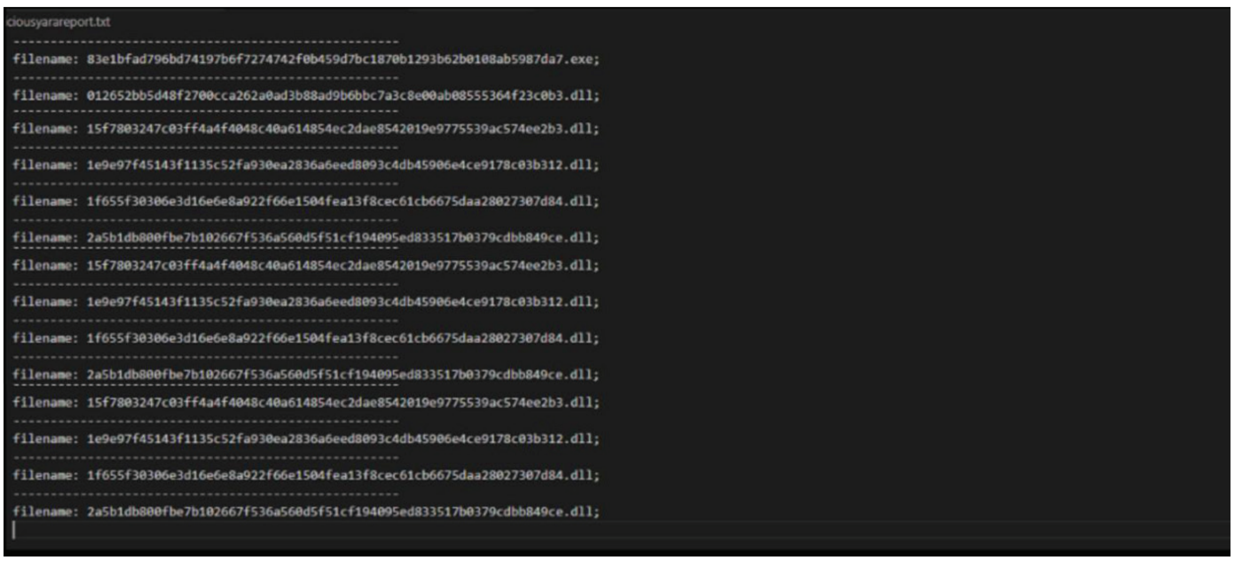

Fig. 8 Output of Yara for Safe Dataset.

effective in detecting malicious files and around 98% of true positive safe DLLs.

Dependence of Effectiveness

The effectiveness of Yara depends on the provided Yara rule and the time it takes to execute it. Because the provided Yara rules contain the information Yara is processing, this processing time is also an essential factor, as there can be thousands of Yara rules. It is also helpful for the malware whose signature or flow is known; it will also not help us detect the zero days based on the signatures.

C. Sysinternal Sigcheck

Out of 80 Malicious files, Sigcheck marked 40 files as unsigned and 40 as signed. In contrast, out of 80 non-malicious files, Sigcheck marked 61 files as signed and 19 files as unsigned, which makes around a 50% success rate of detecting malicious DLLs and roughly 76% of true positive safe DLLs. Fig. 9 and 10 show the results of the output files for both the malicious directory and the safe directory.

Dependence of Effectiveness of Sysinternal Sigcheck

The effectiveness of Sysinternal depends on the hashes and certificates authority database that the Sigcheck is using. The other tools of Sysinternal, like Promon and autoruns, depend on

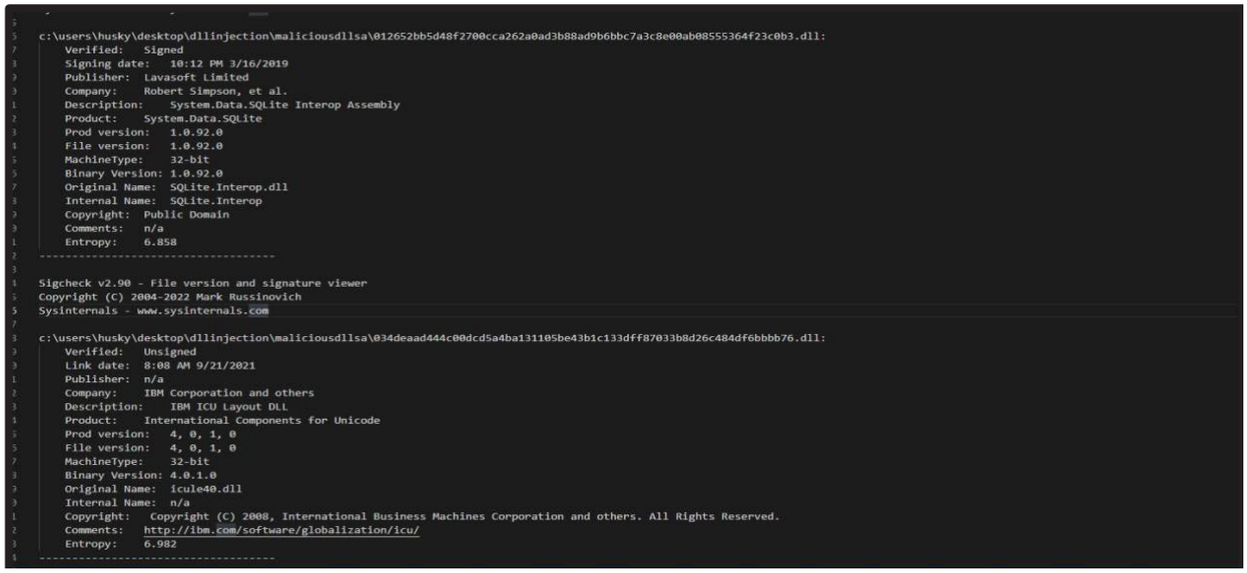

Fig. 9. Output of Sigcheck Malicious Directory.

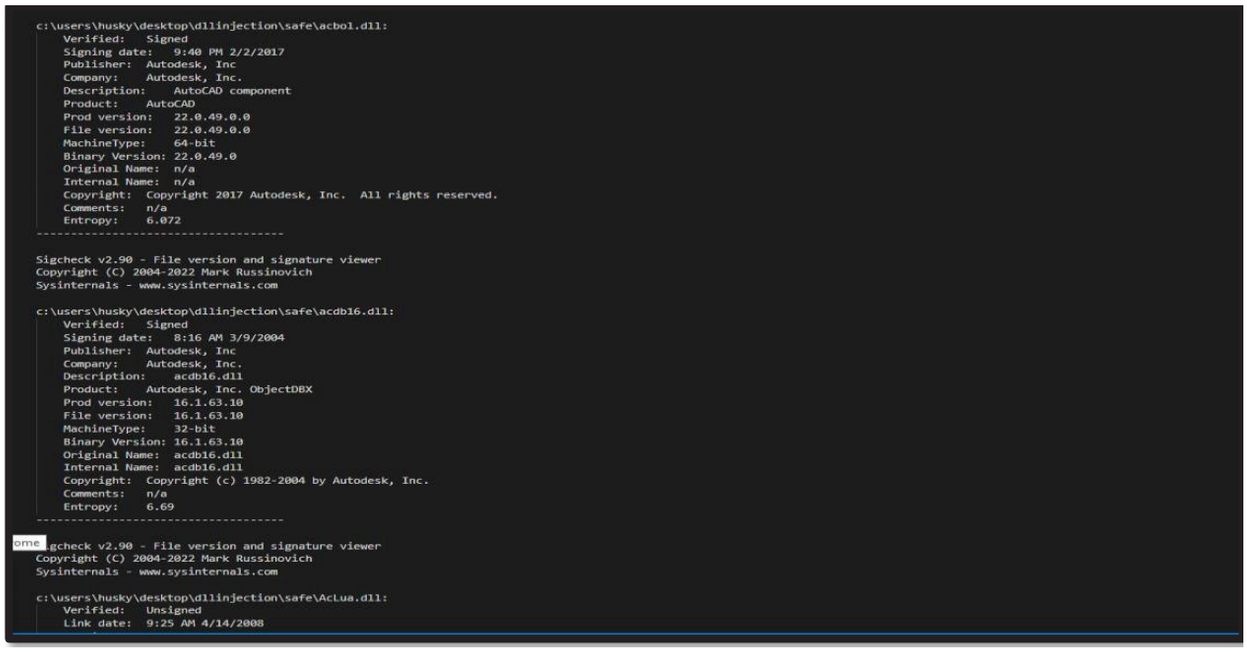

Fig. 10 Output of Sigcheck Safe Directory.

the effectiveness of the manual analysis. The DLL files cannot be executed standalone; that's why it is difficult to manually analyze and use these tools to analyze DLL injection. Hence, the Sysinternal tools are not effective for analyzing DLL files. However, the analysis of the portable executable can be analyzed effectively with Sysinternal tools. The advantage of using Sysinternal tools for the portable executable is that it can analyze the files dynamically. It is beyond the analysis of static or signature-based analyses. Sysinternal can help to identify unknown zero-day attacks.

D. Hypothesis Results

H1: Sysinternal is more effective than VirusTotal in DLL injection detection.

The empirical findings do not support the findings concerning the research hypothesis (i.e., H1) about Sysinternal being more effective than VirusTotal in DLL injection detection.

H2: Sysinternal is more effective than Yara in DLL injection detection.

The empirical findings do not support the findings concerning the research hypothesis (i.e., H2) about Sysinternal being more effective than Yara in DLL injection detection.

H3: VirusTotal is less effective than Yara in detecting DLL Injection

The empirical findings do not support the findings concerning the research hypothesis (i.e., H3) about VirusTotal being less effective than Yara in detecting DLL Injection.

VI. Conclusion and Future Work

This research project compares the effectiveness of three widely used open-source tools, VirusTotal, Yara, and Sysinternal, in detecting DLL injection incidents in cybersecurity. The findings suggest that VirusTotal proves to be the most effective tool, demonstrating high accuracy and robust detection capabilities. Yara, an open-source pattern-matching tool, exhibits potential in detecting specific injection patterns but falls behind in overall effectiveness compared to VirusTotal. Sysinternal, known for its powerful system analysis capabilities, performs decently in identifying DLL injection occurrences but lags behind VirusTotal and Yara. These findings emphasize the significance of considering tool effectiveness and specific strengths when selecting the most suitable solution for DLL injection detection. Ultimately, VirusTotal emerges as the preferred choice due to its superior performance, while Yara and Sysinternal can be complementary tools in specific scenarios.

The drawback observed in evaluating Sysinternal tools for DLL injection detection is their reliance on executing the DLL files. Unlike VirusTotal and Yara, which analyze the files independently, Sysinternal tools require the DLL files to be executed to detect injection incidents. This approach can be problematic when attempting to run the DLL files standalone, as it may encounter issues or dependencies that prevent their proper execution. This limitation reduces the flexibility and convenience of using Sysinternal tools, especially in cases where executing the DLL files separately is necessary or desired. It highlights the need for alternative detection methods that do not rely on executing the DLL files themselves, ensuring more robust and comprehensive detection capabilities.

Future work in DLL injection detection can involve evaluating new tools that emerge in the field, analyzing advanced malware techniques, integrating multiple detection tools for a comprehensive approach, exploring the application of machine learning algorithms, and developing real-time monitoring and response systems. These efforts aim to stay updated with evolving threats, improve detection accuracy, and enhance the ability to detect and mitigate DLL injection attacks promptly. Furthermore, there could be an investigation into other open-source DLL injection detection tools such as Detours, Volatility, and PeStudio. Detours is a software package

developed by Microsoft Research that allows developers to create applications with DLL injection detection capabilities. It provides a set of APIs and hooks that enable monitoring and interception of DLL injections in real-time. Volatility is a widely used open-source memory forensics framework that helps analyze system memory for forensic investigations. It provides various plugins and capabilities to detect DLL injections and analyze memory artifacts associated with the injected DLLs. PeStudio is a free and lightweight tool for analyzing Windows executable files. It allows users to inspect the imports, exports, resources, and sections of an executable, helping to identify suspicious or unauthorized DLLs that may indicate DLL injection.

Funding

This article did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of Interest

Authors declare that they have no conflict of interest.

References

[1] H. Alnabulsi and R. Islam, "Protecting code injection attacks in intelligent transportation system," in 2019 *18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, IEEE, 2019, pp. 799–806.

[2] L. An, M. Castelluccio, and F. Khomh, "An empirical study of dll injection bugs in the firefox ecosystem," *Empir Softwz Eng*, vol. 24, pp. 1799–1822, 2019.

[3] W. Matsuda, M. Fujimoto, and T. Mitsunaga, "Detection of malicious tools by monitoring DLL using deep learning," *Journal of Information Processing*, vol. 28, pp. 1052–1064, 2020.

[4] A. Ahmad, S. B. Maynard, K. C. Desouza, J. Kotsias, M. T. Whitty, and R. L. Baskerville, "How can organizations develop situation awareness for incident response: A case study of management practice," *Comput Secur*, vol. 101, p. 102122, 2021.

[5] B. Miller et al., "Reviewer integration and performance measurement for malware detection," in *Detection of Intrusions and Malware, and Vulnerability Assessment: 13th International Conference, DIMVA 2016, San Sebastián, Spain, July 7-8, 2016, Proceedings 13*, Springer, 2016, pp. 122–141.

[6] Satyabrata Jena, "Difference-between-open-source-software-and-commercial-software," https://www.geeksforgeeks.org/difference-between-open-source-software-and-commercial-software/.

[7] A. Kleymenov and A. Thabet, *Mastering Malware Analysis: A malware analyst's practical guide to combating malicious software, APT, cybercrime, and IoT attacks*. Packt Publishing Ltd, 2022.

[8] K. E. Cybersecurity, "The Protection Technologies of Kaspersky Endpoint Security."

[9] S. J. Yoo, "Study on improving endpoint security technology," *Convergence Security Journal*, vol. 18, no. 3, pp. 19–25, 2018.

[10] K. Jochem, "Tag Archives: Applocker".

[11] S. E. Protection, "Symantec Endpoint Protection 11.0," *Application and Device Control*, pp. 1–18, 2008.

[12] S. Srinivasa, J. M. Pedersen, and E. Vasilomanolakis, "Open for hire: attack trends and misconfiguration pitfalls of IoT devices," in *Proceedings of the 21st ACM Internet Measurement Conference*, 2021, pp. 195–215.

[13] J. M. Pearce, "Economic savings for scientific free and open source technology: A review," *HardwareX*, vol. 8, p. e00139, 2020.

[14] E. Choo, M. Nabeel, R. De Silva, T. Yu, and I. Khalil, "A large-scale study and classification of Virustotal reports on phishing and malware URLs," *arXiv preprint arXiv:2205.13155*, 2022.

[15] M. E. Russinovich and A. Margosis, *Troubleshooting with the Windows Sysinternals tools*. Microsoft Press, 2016.

[16] S. Zhu, Z. Zhang, L. Yang, L. Song, and G. Wang, "Benchmarking label dynamics of virustotal engines," in *Proceedings of the 2020 ACM SIGSAC Conference*

on Computer and Communications Security, 2020, pp. 2081–2083.

[17] E. Cozzi, M. Graziano, Y. Fratantonio, and D. Balzarotti, "Understanding linux malware," in *2018 IEEE symposium on security and privacy (SP)*, IEEE, 2018, pp. 161–175.

[18] M. G. Christopher and K. Raychaudhuri, "A Digital Forensic Approach for Examination and Analysis of Frozen Hard Disk of Virtual Machine.," *International Journal of Cyber-Security and Digital Forensics*, vol. 8, no. 4, pp. 262–273, 2019.

[19] C. Avci, B. Tekinerdogan, and C. Catal, "Analyzing the performance of long short-term memory architectures for malware detection models," Concurr Comput, vol. 35, no. 6, p. 1, 2023.

[20] E. Ko, J. Kim, Y. Ban, H. Cho, and J. H. Yi, "ACAMA: Deep Learning-Based Detection and Classification of Android Malware Using API-Based Features," *Security and Communication Networks*, vol. 2021, pp. 1–12, 2021.

[21] D. Regéciová, D. Kolář, and M. Milkovič, "Pattern Matching in YARA: Improved Aho-Corasick Algorithm," *IEEE Access*, vol. 9, pp. 62857–62866, 2021.

[22] N. Naik, P. Jenkins, N. Savage, L. Yang, K. Naik, and J. Song, "Embedding fuzzy rules with YARA rules for performance optimisation of malware analysis," in *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, 2020, pp. 1–7.

[23] P. Black, I. Gondal, A. Bagirov, and M. Moniruzzaman, "Malware variant identification using incremental clustering," *Electronics (Basel)*, vol. 10, no. 14, p. 1628, 2021.

[24] J. Zhang, M. F. Khan, X. Lin, and Z. Qin, "An optimized positive-unlabeled learning method for detecting a large scale of malware variants," in *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, IEEE, 2019, pp. 1–8.

[25] abuse.ch, "MalwareBazaar Database," https://bazaar.abuse.ch/.

[26] DLL-Files.com, "DLL-files," https://www.DLL-files.com/ .

[27] M. Halsey and M. Halsey, "Microsoft Sysinternals Suite," *Windows 10 Troubleshooting: Learn to Troubleshoot and Repair Windows 10 Problems Like the Pros Do*, pp. 607–636, 2022.