EXPLORING EMERGING TECHNIQUES AND TOOLS FOR MALWARE ANALYSIS

*Ankit Jain (Department of ECE, Pranveer Singh Institute of Technology, Kanpur, ankit2483jain@gmail.com)
Anita Shukla (Department of BSH, Pranveer Singh Institute of Technology, Kanpur, shukla.anita27@gmail.com)
Vivek Kumar (Department of ECE, Pranveer Singh Institute of Technology, Kanpur, rastogi0807@gmail.com)
Imran Ullah Khan (Department of ECE, Integral University, Lucknow, imranuk79@gmail.com)
Keshav Kumar (Department of ECE, Pranveer Singh Institute of Technology, Kanpur, keshav@gyancity.com)
Dr. Kamini Simi Bajaj (Directors of Academic Program, Western Sydney University, Australia, k.bajaj@westernsydney.edu.au)

ABSTRACT

One of the largest problems facing the Internet today is the vast volume of data and files that must be analysed for potential malicious intent. according to the constantly changing field of cybersecurity. Malware, or malicious software, is getting more and more complex and frequently uses metamorphic and polymorphic approaches. These features make it possible for malware to alter its code structure while spreading, which makes detection very difficult. Because they frequently miss previously undiscovered or zero-day malware variants, traditional defensive mechanisms especially those that depend on signature-based detection are proving to be inadequate in the fight against these threats. To properly identify and lessen the impact of malware families, sophisticated analytical techniques are required due to their increasing diversity and complexity. Even though malware is always evolving, many versions from the same family have behavioral patterns that are indicative of their origin and underlying purpose. Either static analysis, which examines code without execution, or dynamic analysis, which watches malware run in a controlled environment, can be used to find these behavioral characteristics.

Machine learning (ML) approaches have become effective tools for the categorization and detection of unknown malware by utilizing these behavioral traits. Labelled datasets can be used to train machine learning algorithms to identify common patterns in known malware families. These algorithms can then use this information to identify new, hidden threats. The accuracy and adaptability of detection have been greatly increased by this paradigm shift from signature-based to behaviour-based and machine learning-driven analysis.

An extensive review of the newest methods and resources for malware analysis is provided in this survey article. It focuses on comparing and contrasting modern and traditional approaches, emphasizing their advantages, disadvantages, and suitability for practical situations. Particular attention is paid to how machine learning might improve malware detection skills and how contemporary solutions include these intelligent algorithms to tackle the problems caused by malware that is polymorphic and metamorphic. The significance of integrating static and dynamic analysis techniques to create resilient, hybrid detection models that can successfully combat the constantly shifting malware field is also covered in the article.

KEYWORDS

Malware Analysis, Polymorphic Malware, Metamorphic Malware, Machine Learning, Static Analysis, Dynamic Analysis, Malware Detection, Behavioral Analysis, Signature-based Detection, Cybersecurity Tools

1. INTRODUCTION

The Internet is an essential component of daily living in this day and age. Internet offers services such as online banking, shopping, social networking, finance, and education, have become second nature to people. However, some people utilize the Internet for nefarious purposes and try to enrich themselves through evil actions, which are accomplished with the use of dangerous software known as malware. These are getting more complex, chronic, and unidentified every day. Security systems such as IDS/IPS and antivirus (AV) use manually generated signatures to identify malware, that necessitates a thorough analysis of malware samples. Every minute, McAfee [1] catalogues roughly 69 new malware samples. In 2015, 430 million new malicious samples were found, approximately 36% higher than in 2014, according to the Symantec Threat Report (2016) [2]. Malware specimens are increasing dramatically in volume (the threat environment is expanding), variety (new harmful techniques), and pace, according to a TrendMicro analysis [3]. These are changing, getting more complex, and targeting computers and mobile devices in novel ways. Thanks to easily accessible web tools, malware developers may now produce new dangerous specimens in a matter of seconds. A piece of software [4] that carries out an attacker's malicious aim is called malware. Malware can be divided into many kinds based on their traits, such as how they propagate or infect systems. Worms, viruses, Trojan horses, spyware, backdoors, rootkits, botnets, and adware are among the most common types of malwares. The three most common methods by which malware infects a system are exploiting network services vulnerabilities, social engineering, and drive by download. Violations of Advanced Persistent Threats (APTs) against social, political, economic, and military networks require ongoing awareness in order to reduce risks and provide resilience for national security. Understanding the methods by which malware enters a network, propagates there, and eventually begins transferring data out of it is crucial for spotting its symptoms.

Malware, is a fast-increasing hazard for computing industry. In the first quarter of 2017, 48 million different malware samples were created, according to the AV-TEST institute [6]. The sheer volume of malware makes it hard for human engineers to handle it. Malware detection technologies are therefore used by security researchers to find malware. The two phases of detection systems are analysis and detection. To find malware, antivirus software frequently employs a signature-based methodology. This process is of low false positive rate and quick and effective in detecting known malware. However, malware that employs obfuscation techniques can readily circumvent signature-based detection, which is unable to identify new malware. However, another detecting method is behaviour-based, in which questionable files are run under observation in a controlled setting and flagged as malicious if their

actions align with those of recognized malware. Although behaviour-based detection is time-consuming and has a high false-positive rate, it can identify malware that is unknown or that use obfuscation tactics [7]. Because of the abundance of information in the dumped memory that may be utilized to look into malicious activity, memory-based malware detection is also getting more and more prevalent these days [8]. The purpose of this work is to present a summary of methods and tools for malware capture, analysis, and feature extraction, either statically or dynamically.

2. LITERATURE SURVEY

The research articles on malware analysis listed a number of tools and methods that might be used to find and examine malware. Static and dynamic analysis are the two fundamental approaches of malware analysis [9]. The majority of research conclude that dynamic analysis is far more accurate and effective than static analysis [10]. Certain malware cases may simultaneously display the traits of multiple categories. The tools need to be strong enough to effectively identify various types of malwares. Various methods have been identified under the heading of dynamic analysis. Information tracking, process call monitoring, process parameter analysis, instruction trees, and auto-start extensibility. The findings of static and dynamic analysis appear to be inadequate due to the high level of malware obfuscation; therefore, security analysts turn to machine learning, deep learning, and neural network techniques. These days, the field of study on machine learning techniques for malware analysis has grown significantly. Features gathered from both static and dynamic analysis are used to forecast the virus [11]. From the file's raw bytes, malware can even be predicted using neural networks [11]. After the malwares were separated into different families, the dataset was used for machine learning. Different machine learning algorithms were examined using different tools, and the findings showed that Random Forest was the most effective algorithm for dataset analysis [12].

Since malware is becoming more prevalent these days, experts are recommending using data visualization in conjunction with machine learning to improve accuracy. This method, called Visual-AT, reduces false positives by 81.17% and achieves an accuracy of up to 97.73% [13]. Additionally, machine learning can help with the investigation of malware that runs on Linux. Crypto-mining malware was discovered to be infiltrating IOT systems, growing in sophistication, and rapidly proliferating new varieties with no infrastructure investment [14]. Machine learning can be used to forecast which malware cluster is comparable to the unlabelled samples once reverse engineering has been applied to comprehend them [14]. There is a novel technique that uses the n-gram features of the disassembled code and then uses a machine learning model for analysis because the malware's n-gram characteristics are diluted due to its extreme obfuscation [15].

3. MALWARE ANALYSIS

It is looking into and evaluating effects of malicious software. It has no risk of injury because this procedure is carried out in a controlled setting. Various monitoring tools and extraction procedures are used to obtain the data or information from malware. Both run and rest modes are available for the analyses. As a result, Fig. 1 shows three different analysis techniques.

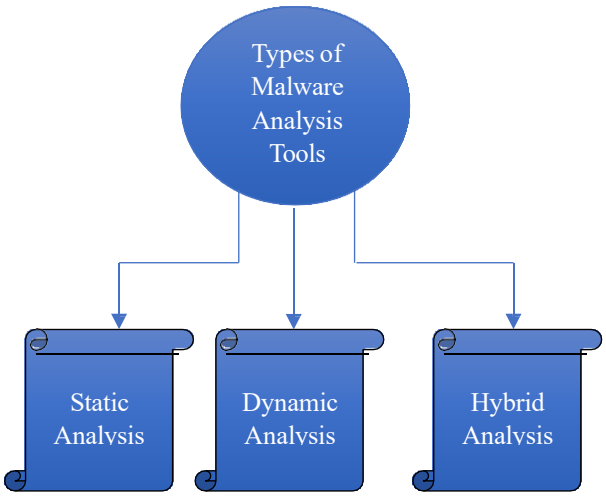

Fig.1. Types of analysis tool

3.1 Static Analysis

In this case, the malware program is not triggered, or in other words, it is not in a running state. Static analysis, on the other hand, looks at the structure and coding of malware without running it. The following are the main procedures and techniques used in static malware analysis:

- **Sample Acquisition:** To avoid unintentional system infection, secure a malware sample from a reputable source or controlled environment.
- **Determining the File Format:** Find out if the sample is an executable (EXE), dynamic link library (DLL), script (VBScript, PowerShell), or another sort of file.
- **Disassembly/Decompilation:** Convert the binary code of the malware into a format that is easier to understand. Disassembling is the process of converting machine code into assembly, whereas decompilation is the process of processing the coding languages used in C and Java.
- **Code Examination:** After conducting a thorough analysis, examine the code to find out what it does and take into account any awful tasks it might carry out. The goal of analysts is to identify warning signs and establish standards that would expose detrimental behaviour. **String and Resource Extraction:** By asserting control over the strings and resources of malicious code, we can learn more about its anticipated purpose, the communication channels it uses, and any encoded signatures that are important for detection.
- **Function and API Call Analysis:** Malware can alter files, communicate via networks, and take over the registry, according to its operations and APIs.

- **Packer and Obfuscation Identification:** Packers and obfuscation are common techniques used by malware developers to conceal their objectives. By examining these methods, one can determine the primary motivation behind the original code.
- **Behavioural Rule Creation:** Using the findings, create a set of patterns of conduct that can help identify linked malware that is about to appear.
- **Code Reverse Engineering:** To examine complex algorithms and security techniques frequently present in malware, reverse engineering research is necessary.
- **Sandbox Analysis (Optional):** A type of restricted sandboxing that simulates code snippet execution without requiring the execution of entire malware is available in certain static analysis applications, adding a dynamic element to static analysis. As is typical in daily cyber security operations, improving malware analysis performance typically necessitates the continuous use of both static and dynamic analysis in addition to other techniques like memory analysis and network traffic analysis.

3.2 Dynamic Analysis

The approach is commonly used in cybersecurity to thoroughly examine dangerous software (malware) in a monitored and securely protected environment. This approach deviates from the norm of static analysis, which evaluates the structure and code of malware without putting it to use. A deeper comprehension of malware behaviour results from a review of dynamic analysis, which develops more reliable techniques for identifying and reducing security risks. The following are the main procedures and methods used in dynamic malware analysis:

- **Sample Isolation:** It is executed in a sandbox, also known as a virtual computer, to protect other systems or networks from a malware intrusion.
- **Activity Monitoring:** Throughout the malware's active period, activity is continuously monitored. These processes include a range of system queries, file system events, registry updates, network operations, and adjustments to critical system elements.
- **Network Traffic Analysis:** Analysts can log and examine network traffic produced by malicious software by using dynamic analysis. With the right help, you may learn about exfiltration techniques, the likelihood of command-and-control servers, and their communication methods.
- **Memory Analysis:** To identify potentially harmful code intrusions or system process modifications, malware has been subjected to memory analysis throughout its existence.
- **API Hooking:** In certain situations, dynamic analysis includes the use of API hooking techniques to directly monitor and track malware's API calls. This comprehension of malware behaviours is offered without actually altering the legitimate code of the virus.
- **System Monitoring:** It is understood that the analytical environment offers methods that make it easier to identify any changes to the system, such as the addition of new threads, processes, or erratic changes in characteristics.

- **Runtime Debugging/Dynamic Code Analysis:** A unique technique enables experts to examine malware's memory, determine its ordering, and understand its implementation techniques.

3.3 Hybrid Malware Analysis

Static and dynamic analysis techniques are combined in hybrid malware analysis to give a more comprehensive view of malicious software.

While offering a more thorough examination of the mechanisms and capabilities of malware, this approach aims to maximize the benefits of these two approaches while correcting the inconsistencies present in each. Hybrid malware analysis usually consists of the following steps:

- **Analysis of Statics**

  This first part involves examining the criminal code and its structure without implementing it. At this point, any obfuscation or packaging carried out by the malicious program is recognized, words and images are accessible, and signatures of known malware are understandable.

- **Behavioural Analysis (Dynamic Analysis)**

  The introduction of malware in a controlled environment, such as a virtual machine or sandbox, encourages surveillance and is a response to static analysis. A thorough analysis of the interactions between malware and the system is conducted at this time, with a focus on file access, network activity, and API calls.

As shown in Table 1, combining static and dynamic analysis yields a comprehensive grasp of malware characteristics that are typically overlooked by standard analysis alone. Dynamic analysis makes it easier to monitor malware targets in real time and have a more comprehensive understanding of how it affects its ecosystem.

Table 1. Benefit comparison of static and dynamic malware analysis

Sr. No.	Static Analysis	Dynamic Analysis
1	Malware may be investigated safely and precisely without running the risk of infection.	It appears to be static at first, but analysis requires a later, more structured setting.
2	To guarantee a more thorough and comprehensive study, this can take place without the need for the internet.	The offline analysis of dynamic analysis might not be available, while static analysis might miss the malware's capabilities.
3	Helps distinguish between known malware and samples that have already been analysed.	Encourages the creation of detection and mitigation techniques by exposing the connections between the malware and the network and system.
4	Key element in determining the virus threat and the possible consequences.	The capacity to watch how the malware behaves in real time, including any strategies used to evade detection or conceal oneself.

As shown in Table 2, hybrid malware analysis has become a popular approach in modern cybersecurity, enabling professionals to keep ahead of evolving malware threats and develop effective defences for systems and sensitive data.

Table 2. A comparison of the drawbacks of static, dynamic, and hybrid malware evaluations

Topic	Static Analysis	Dynamic analysis	Hybrid analysis
Runtime inability	The inability to observe runtime behaviors.	Designed to keep an eye on runtime behaviors, but without runtime, no data can be obtained.	Possess both skills, but the procedure is time-consuming.
Advanced malware handling	Static analysis may be made more difficult by advanced malware using anti-analysis tactics.	When it comes to modern or clever malware that can detect and withstand dynamic assessments, it might not work as well.	In this analysis, there are less opportunities to avoid.
Encrypted Codes handling	It can be challenging to decipher significantly obfuscated or encrypted code statically.	Due to runtime actions, quite good in obfuscated codes.	beneficial for encrypted codes.
Risk of environment	The analysis environment is the most secure of all.	Possible danger of contaminating the analysis environment.	Early on, risk can be avoided.

4. MALWARE ANALYSIS TOOLS

To defend against and anticipate future assaults, analysts utilize malware analysis tools and exchange expertise. When performing such tasks, open-source technologies are frequently the first option. It's no secret that spreading malware is a lucrative industry, and in years to come, the rapidly expanding malware epidemic will only get more capable and effective. When examining the attack life-cycle, researchers will use open-source malware analysis tools to check, identify, and log various malicious trigger types. Proliferation of malware trading sites on the dark web has made it easier than ever to obtain the crypters, botnets, and zero-days required for launching potent attacks. It is becoming more difficult to comprehend and benchmark the particular sort of malware due to its increasing complexity. Finding the appropriate technology to examine each unique attack type is the responsibility of security researchers and analysts. We now offer a few open-source malware analysis tools to aid security engineers and researchers.

4.1 Open-Source Malware Analysis Tools

GRR stands for Google Rapid Response. Google security researchers created the GRR platform, an incident response tool that finds typical malware footprints on workstations used for remote live forensics. This includes a server infrastructure and an application that is installed on the target system to communicate with the agent. Installed on target computers, GRR is a Python client (agent) and server

infrastructure that can communicate with and manage clients. They can become GRR clients and begin receiving messages from the servers after the agent and server side have been deployed. After then, the host computer's incident response team will carry out a number of technical tasks, including examining the RAM, searching for various settings, and managing software selections. In order to facilitate the collection and processing of data from a large number of computers, GRR was created to operate on a scale. The objective of GRR is to facilitate forensics and investigations in a straightforward, adaptable manner that enables investigators to perform remote analysis and quickly triage situations.

REMnux: A free Linux toolkit called REMnux was created to help malware analysts with the reverse engineering of malware. Although it might be challenging to find or set up, this aims to make it simple for forensic investigators and accident witnesses to continue using the range of free applications that can analyze ransomware. The purpose of this Linux toolbox is to serve as a one-stop resource for researchers looking for instances of malware reverse engineering. With an emphasis on Ubuntu, REMnux combines multiple tools into a single resource for fast malware analysis of Windows and Linux-based systems. The REMnux Linux system, which is based on Ubuntu, is the project's mainstay. This small distribution offers a number of tools to identify ransomware on Windows and Linux, examine browser-based flaws like obfuscated JavaScript, look into odd text files, and remove other dangerous items. Investigators can use the distro in a lab setting to intercept suspect network communication. Researchers can use it to examine browser-based malware, carry out memory forensics, examine several malware samples, extract and decode questionable objects, and more.

Cuckoo Sandbox: During the Google Summer of Code program in 2010, a team of volunteers created an open-source framework that automates dangerous file analysis for Windows, OS X, Linux, and Android and offers comprehensive and practical information on how each presented file behaves in isolated situations. Additionally, developers are always producing plugins with new features because the software is open-source. Cuckoo is used by malware detection and security firms to lessen the workload associated with manually sifting through mounds of potentially hazardous data. The modular design makes it simple to configure the recording and analysis stages. It is understandable why it has become one of the most popular open-source applications in recent years.

In 2012, Cuckoo introduced Malware, a sandbox-as-a-service that lets users use the data they collected through an easy-to-use graphical user interface. The idea was to provide those who want to take advantage of their intelligence but can't use Cuckoo properly a choice.

Zeek: A flexible network-dependent analytics tool, the Zeek Network Security Monitor (previously Bro) converts network traffic into events that trigger scripts. Using both signature-based and anomaly-based (searches anomalous behaviour) monitoring, it provides users with an overview of their network activity, much like an intrusion detection system (IDS). However, its capabilities are significantly more extensive than those of traditional intrusion detection systems, which can be employed for forensic, network monitoring, and interface research studies. Zeek offers a comprehensive forum for more broad network traffic analysis, despite its primary concentration on network security tracking. With over 20

years of research under its belt, Zeek has been successful in bridging the gap between operations and academia since its founding.

Yara Rules. After being evaluated in Cuckoo, another open-source malware detection tool can recognize malware samples based on binary or textual patterns. To create pattern-based definitions of the malware families, investigators utilize Yara. Since the descriptions are referred to as rules, YARA stands for "Yet Another Recursive Acronym." This can be modified for usage within Cuckoo and aids researchers in recognizing and classifying malware kinds that appear to be identical. Yara, which is compatible with Linux and Windows, is referred as by IBM as the "pattern matching Swiss army knife" of the malware researcher. The developers of Yara launched Yara Rules Analyzer, a new service that is still in development and allows customers to use complete rulesets to analyze files in the cloud. This eliminates the requirement for users to install Yara locally and guarantees that they are always comparing samples to most recent ruleset version. Many Endpoints Detection including Response frameworks have been enhanced with Yara rules to aid in the identification, classification, and subsequent dissemination of their findings to customers and the community.

4.2 Mobile Malware Analysis Tools

APKTool: A reverse engineering tool for closed, third-party, binary Android applications. This may restore and decode resources to nearly their original form by implementing a number of modifications. Additionally, it facilitates the process of handling a device because of the project, including file generation including the execution of repetitive chores like Apk development. With the use of an Apk tool, we can decode APK resources to nearly their original state; we can then reconstruct the decoded resources into APK while making real-time modifications to the source code. They are easy to work with because of their project-like structure. In addition to automating repetitive processes, The Apk tool can handle and manage APKs that depend on framework resources, decode APK resources (resources. arsc, classes. dex, and XMLs), and reconstruct decoded resources back to binary APK.

Smali: Dalvik is an Android Java virtual machine implementation that uses the Smali/baksmali dex format assembler/disassembler. The syntax adheres to the full dex format capabilities (annotations, debug data, line information, etc.) and is roughly based on the grammar of Jasmin / dedexer. Additionally, code developed by the Baksmali is frequently regarded as being in the Smali language. Baksmali is a dex program disassembler for Bytecode. "Smali" and "Baksmali" are just the Icelandic equivalents of "assembler" and "disassembler," respectively. Debugging Smali code used to be difficult, however a fantastic plugin called Smalidea was been released for Intellij IDEA/Android Studio. Dex2Jar. Dex2Jar is a free program for handling Java “.class” and Android “.dex” files. Android programs are compiled into "dex" (Dalvik Executable) scripts, that are essentially compressed into a single. Apk file and packed onto PC. Android will transform the compiled Java programs to automatically create the ".dex" files.

Converting an APK classes.dex file to classes.jar or the other way around is the main function of Dex2Jar. Therefore, it is feasible to inspect the completely readable source code of an Android

application using any Java decompiler. Instead of the actual Java source code written by the application author, we obtain files from the class here.

Mobile Sandbox. For Android OS cellphones, Mobile-Sandbox offers both static and dynamic malware analysis. Two innovative approaches are used by the system to automatically assess Android software:

a) by combining static and dynamic analysis, whereby the results of static analysis are utilized to expand coverage of executed code and direct dynamic analysis, and

b) by employing various logging techniques for native API calls. To obtain a program overview, it can assess the application using several modules inside the static analysis component.

In order to accomplish this, it parses the manifest file, runs multiple anti-virus scans using the Virus Total service, and then decompiles the application to better detect suspicious code. It can run the application in an emulator and record all of its operations as part of the dynamic analysis; that is, it records both actions taken in the Java Virtual Machine Dalvik and activities taken in native libraries that might be included with the application.

4.3 Other Analysis Tools

Malzilla: An effective malware hunting tool for examining webpages with dangerous code is Malzilla. Exploit-containing websites frequently employ a series of redirects and obfuscated code to make it challenging for someone to follow them. Without going to the website and possibly breaking their gadget, this enables people to access websites and retrieve all of their source code, like we do. With this software, you may choose the user's referrer and change the user agents. This displays every HTTP header along with the complete list of browser webpages. In addition, it offers sophisticated decoders, proxy functionality, and most importantly the ability to deobfuscate JavaScript code in a single program.

Wireshark: Once called Ethereal, Wireshark is a network monitoring program that records packets and displays them in real time in an understandable way. After intercepting communication, it converts the binary data into a format that users can understand. Filters, colour coding, and other tools in Wireshark enable users to examine individual packets and delve deeply into network data. It is the world's top network traffic analyzer and a necessary tool for every competent device administrator or security professional. This free program is frequently the greatest resource for diagnosing issues on any network and enables users to monitor network traffic in real-time. Lost communications, latency problems, and malicious network operations are common problems that Wireshark can troubleshoot. It makes it possible to examine network data closely and provides tools for sorting and analyzing that data in order to pinpoint the problem's underlying cause. Management uses it to identify issues with delay caused by machines sending traffic globally, data exfiltration, and even intrusion attempts against any entity. It also detects malfunctioning network equipment that loses packets.

SysAnalyzer: An open-source program called SysAnalyzer was created to give malware researchers an interactive resource for quickly gathering, examining, and tracking the actions a binary took while running on a network. It is an interactive framework that monitors several aspects of device and method states for the study of malcode runtime. The purpose of SysAnalyzer was to enable analysts to quickly

provide an in-depth report. The primary components of SysAnalyzer work by comparing device snapshots across a specified user time period. A snapshot approach was chosen in order to minimize the volume of data that analysts must sift through while conducting their research, much like in a live monitoring system. By using a snapshot method, audiences can easily display only the persistent changes that have been identified since the first run of the application.

VirusTotal: A service called VirusTotal examines dubious files and URLs and assists in the rapid detection of Trojan horses, worms, viruses, and other malware by antivirus engines. Along with a range of techniques for eliminating signals from the content under study, VirusTotal uses more than 70 antivirus scanners and URL/domain blacklisting services to evaluate products. Anyone can select a file from their device using their browser and upload it to VirusTotal. VirusTotal provides multiple ways to upload data, such as a programmatic API, desktop uploaders, browser extensions, and the basic public web site.

Out of all the application types that are accessible to the general public, the web interface has the highest scanning priority. Any programming language can be used to create the specifications using the public API that is based on HTTP. A number of other features are now available, such as the VirusTotal Community, a network that enables users to report files and URLs and provide comments for one another. This can be useful for both identifying malicious content and false positives, which are ordinary, innocuous things that have been flagged as harmful by one or more scanners.

5. CONCLUSION

The continuous fight against the spread of harmful software still heavily relies on malware analysis and categorization, which are essential for recognizing, comprehending, and reducing risks. The methods and tools for identifying and evaluating malware must advance in step with the sophistication and adaptability of cyberthreats. Conventional techniques like hybrid approaches and static and dynamic analysis are still often used. These approaches, however, have some drawbacks, such as the obfuscation strategies used by contemporary malware, the increasing diversity of malware strains, and problems with detection accuracy, particularly false positives and negatives.

Newer approaches are being investigated and implemented to overcome these constraints. Methods based on advanced machine learning, behavioral observation, and heuristic analysis have demonstrated a great deal of promise. Deep learning-based models have shown to be especially successful among them, able to identify intricate patterns and adjust to new threats. Furthermore, a distinct advantage in processing textual data from logs, code, and documentation is provided by the incorporation of natural language processing (NLP) into malware analysis, which improves analytical capabilities.

The growing significance of an integrated approach that combines various analysis tools and approaches to create a more comprehensive defence strategy is one of the study's main conclusions. Tools that facilitate network packet inspection, memory forensics, reverse engineering, and sandboxing are

essential for learning more about the behaviour and spread of malware. Tools for visual analysis also help analysts better understand complex data.

In the end, even if the profession has advanced significantly, there is still a great deal of room for growth. The development of more reliable evaluation frameworks, improved detection accuracy, and improved feature extraction methods should be the main goals of future research. The cybersecurity community can create more robust defences against the constantly changing malware threat landscape by encouraging collaboration amongst different detection techniques and utilizing AI and automation.

REFERENCES

[1]. http://www.scmagazine.com/the-state-of-malware-2013/slideshow/1255

[2]. ttps://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf

[3]. http://www.trendmicro.com/cloud-content/us/pdfs/business/white-papers/wp-addressing-big-data-security challenges.pdf

[4]. Bayer, U., Moser, A., Kruegel, C., and Kirda, E. 2006. Dynamic analysis of malicious code. Journal in Computer Virology 2, 1, 6777.

[5]. Cloppert, M. 2009. Security intelligence: Attacking the kill chain. Retrieved on June 1, 2012.

[6]. AV-TEST, "The AV-TEST Security Report," 2017. [Online]. Available: test.org/fileadmin/pdf/security_report/AVTEST_Security_Report_2016-2017.pdf.

[7]. C. T. Lin, N. J. Wang, H. Xiao, and C. Eckert, "Feature selection and extraction for malware classification," J. Inf. Sci. Eng., vol. 31, no. 3, pp. 965–992, 2015.

[8]. R. Mosli, R. Li, B. Yuan, and Y. Pan, "Automated malware detection using artifacts in forensic memory images," in 2016 IEEE Symposium on Technologies for Homeland Security, HST 2016, 2016, pp. 1–6.

[9]. Egele, M., Scholte, T., Kirda, E., & Kruegel, C. (2008). A survey on automated dynamic malware-analysis techniques and tools. ACM computing surveys (CSUR), 44(2), 1-42.

[10]. Uppal, D., Mehra, V., & Verma, V. (2014). Basic survey on malware analysis, tools and techniques. International Journal on Computational Sciences & Applications (IJCSA), 4(1), 103.

[11]. Gibert, D., Mateu, C., & Planes, J. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. Journal of Network and Computer Applications, 153, 102526.

[12]. Mahajan, G., Saini, B., & Anand, S. (2019, February). Malware Classification Using Machine Learning Algorithms and Tools. In 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP) (pp. 1-8). IEEE.

[13]. Liu, X., Lin, Y., Li, H., & Zhang, J. (2020). A novel method for malware detection on ML-based visualization technique. Computers & Security, 89, 101682.

[14]. Carrillo-Mondéjar, J., Martínez, J. L., & Suarez-Tangil, G. (2020). Characterizing Linux-based malware: Findings and recent trends. Future Generation Computer Systems, 110, 267-281.

[15]. Pektaş, A., Eriş, M., & Acarman, T. (2011, August). Proposal of n-gram based algorithm for malware classification. In The Fifth International Conference on Emerging Security Information, Systems and Technologies (pp. 7-13).