A Review of Signature-Based Malware Detection: Mechanisms, Tools, and Evolving Role in Modern Cybersecurity

1st Muhammad Mansoor
Information and Communication Engineering
Islamia University of Bahawalpur
Bahawalpur, Pakistan
Malikmansoor1232@gmail.com

2nd Adnan Hanif
Information and Communication Engineering
Islamia University of Bahawalpur
Bahawalpur, Pakistan
raeesadnan048@email.com

3rd Muhammad Sohail
Information and Communication Engineering
Islamia University of Bahawalpur
Bahawalpur, Pakistan
sohailali162169@gmail.com

4th Ghulam Mohayudin
Information and Communication Engineering
Islamia University of Bahawalpur
Bahawalpur, Pakistan
ighulammohayudin@gmail.com

5th Zain Ul Abideen
Information and Communication Engineering
Islamia University of Bahawalpur
Bahawalpur, Pakistan
engr.zain@iub.edu.pk

***Abstract*—This review provides a comprehensive analysis of signature-based malware detection, it defines the fundamental mechanics from cryptographic hashes to complex byte-pattern rules, integration into tools like ClamAV, Snort, and YARA, and commercial Endpoint Protection Platforms . A central focus is the critical evaluation of the methodology's strengths like speed, high accuracy for known threats, and low computational overhead, and its well-documented weaknesses, particularly its inability to detect zero-day exploits and polymorphic malware. The paper delves into adversarial evasion techniques that are designed explicitly to defeat signature-based controls. Subsequently, the paper situates signature-based detection within the modern security paradigm, examining its role as a component in hybrid architectures that incorporate heuristic, behavioral, sandboxing , and AI/machine learning techniques . The paper concludes that signature-based detection still remains an indispensable and highly efficient first-line filter in a layered, defense-in-depth strategy. Its future lies in AI-based automation and threat intelligence.**

***Index Terms*—Malware Detection, Signature-Based Detection, Intrusion Detection Systems (IDS), YARA, ClamAV, Malware Evasion, Polymorphic Malware, Hybrid Security, Endpoint Detection and Response (EDR).**

I. Introduction

A. The Pervasive Threat of Malicious Software

Malware is an umbrella term for any software code or computer program intentionally written to disrupt computer operations, gather sensitive information, gain unauthorized access to computer systems, or exhibit other malicious behavior.The global financial impact is projected to reach 10.5 trillion USD by 2025.2 Since 1980s, more than a billion malware strains have emerged, ranging from early viruses, worms, Trojans to new modern threats like ransomware, spyware, adware, botnets, fileless malware [1].2 This diverse and evolving threat [2] landscape necessitates the development and continuous refinement of robust detection mechanisms.

B. The Genesis of Malware Detection Paradigms

Signature-based detection is the earliest and most foundational of detection paradigms, which works by identifying known threats by matching their unique characteristics against a pre-compiled database of malicious identifiers [3], [4]. Advanced malwares spurred the development of more proactive and inferential detection paradigms [5] i.e, Heuristic analysis [6] to identify suspicious characteristics in code, even without a known signature [1], [7]–[9]. Behavioral analysis and sandboxing [10] took this a step further, executing and observing suspicious programs in an isolated environment [11]. Artificial intelligence [12] and machine learning have been applied to learn and predict the features of novel threats from vast datasets [13], [14].

DOI: https://doi.org/10.67535/tsp.000001.040
Published under the ICACNC 2025 Proceedings

II. The Mechanism of Signature-Based Detection

A. Defining a "Signature": The Digital Fingerprint

The "signature" is the unique pattern or identifier associated with a malware to allow a security system to recognize it like a digital fingerprint or a DNA sample [15]. Signatures can vary from simplest form like a cryptographic hash, such as an MD5, SHA-1, or SHA-256 hash, to a more robust form of byte-sequence or text signatures found within the malware's code to scan for a particular hexadecimal sequence like 4F 45 20 4D 41 4C 57 41 52 45 21 that is known to be part of a malicious payload in files or network traffic [16], [17]. Finally, behavioral signatures capture predefined patterns of malicious actions,, like failed login or suspicious API calls [18]. Although edging toward behavioral analysis, these remain grounded in matching known patterns [9].

B. The Core Detection Process

The process of signature-based detection can be broken down into four key stages:

- Signature Generation: Analysts reverse engineer new malwares to extract unique identifiers like hashes, strings, or patterns.
- Database Curation: These signatuures are stored, updated and distributed in massive vendor or community maintained databases.
- Scanning and Matching: The security tool actively scans its environment to compare files or traffic against databases.
- Action on Detection: If a match is found between the scanned data and a known malicious signature in the database, the system flags the file or traffic as a threat.7 It then executes a predefined response to prevent it from any sort of harm.6

C. The Anatomy of a Signature: A Technical Deep Dive

Earlier, malware could be identified by a simple hash of the entire file but attackers learned to make trivial modifications to the file [1], [15]. Defenders then moved to partial-file signatures. When attackers began using packing and obfuscation [19] to counter it, defenders advanced to context-aware rules to analyze file and code structure [20].

1) Hash-Based Signatures: Cryptographic hashes like MD5 or SHA-256 are applied to a malicious file, producing a unique hash value that acts as its fingerprint helping scanners to flag exact matches. Tools like ClamAV's sigtool can be used to automate the process.17 While this method is extremely fast and efficient, it is also exceptionally brittle, any minor non-functional change can invalidate the hash.14 This ease of evasion has rendered full-file hash signatures largely obsolete for altered malware copies.

2) Byte Sequence and String Matching: A more durable approach that identifies byte sequences or string characteristics of a malware family e.g., "UVODFRYSIHLNWPEJXQZA-KCBGMT".20 This method is more robust than full-file hashing because these signatures remain valid despite changes elsewhere in the file [15]. However, this method can still be defeated by code obfuscation techniques [19], [21], [22].

3) Advanced Pattern Recognition with YARA: YARA has emerged as standard for malware researchers [23] seeking to create powerful, flexible, and shareable pattern-matching rules often called "the pattern matching swiss knife for malware researchers" . A YARA rule is composed of three main parts:

- meta: It contains metadata about the rule, such as the author, creation date, description and threat level of the malware it detects .
- strings: It contains patterns such as hexadecimal byte sequences, plain text strings or regular expressions.
- condition: It contains a Boolean expression that defines the logic for a match.

YARA's power lies in its ability to combine multiple indicators of compromise into a single, logical rule, scanning both disk files and memory.

4) Case Study: Crafting Robust.NET Signatures: The creation of effective signatures [15] for modern applications requires a deep, structural understanding of the file format itself. A naive simple searching approach for strings will fail as the binary structure in decompiler is complex. A robust signature must therefore target the underlying metadata structures:

- Stream-Aware Pattern Matching: A.NET executable contains several metadata streams, including #Strings, #US (User Strings), and #Blob (for binary data). Each stream has a unique format like in #Strings heap are UTF-8, #US heap are UTF-16.23 An effective YARA rule must account for this structure of these [23].
- Token Wildcarding in IL Code: IL(Intermediate Language) code references use a 4-byte token made of stable 1-byte token type and 3-byte Record Identifier that changes on code recompilation [24]. Robust signature must wildcard three RID bytes while keeping the stable token type byte e.g, 28?????? 0A for a `call` with a member reference.
- Combining Multiple Weak Indicators: The most resilient signatures are built by combining multiple, contextually-aware patterns. A single rule might look for a specific class name in the #Strings stream, a specific IL code pattern, and a specific method signature in the stream.23

III. A Survey of Signature-Based Detection Tools

The landscape of signature-based detection is populated by a mix of foundational open-source projects [15] that have become industry standards and sophisticated commercial platforms that integrate signatures into a broader security suite [3].

A. Pioneering Open-Source Tools

Several open-source tools have been instrumental in the development and proliferation of signature-based detection.

1) ClamAV:

- History and Mission: Initially released in 2002 by Tomasz Kojm, ClamAV emerged as one of the first comprehensive, open-source antivirus engines for UNIX-like operating systems.24 It was a personal project that got acquired by Sourcefire in 2007, which was in turn acquired by Cisco in 2013.
- Architecture and Features: ClamAV main components include: clamscan, a command-line scanner; clamd, for high-performance scanning; and freshclam, to automatically update if virus signature database.24
- Signature Formats: ClamAV's supports diverse range of signature formats. These signatures are distributed in ClamAV Virus Database (.cvd) files. The formats include .hdb, .mdb, .ndb, .ldb, and, since version 0.99, support for the industry-standard YARA rule format (.yar).17

2) Snort:

- History and Evolution: Created in 1998 by Martin Roesch, Snort began as a "lightweight" and efficient network intrusion detection system [16], [17], [25], [26]. Snort's development was professionalized through the creation of Sourcefire, which was later acquired by Cisco [27].
- Functionality: Snort operates on IP network traffic, performing protocol analysis, content searching, and pattern matching [16]. These rules can detect a wide array of attacks. Snort can be configured as a simple packet sniffer, as a packet logger for offline analysis [17], or as a full-blown network intrusion detection and prevention system [9].

3) YARA:

- History and Purpose: Developed by Victor Alvarez of VirusTotal and first released in 2013, created for malware researchers.33 Unlike ClamAV or Snort [28], It is a highly specialized tool for creating descriptions of malware families based on textual or binary patterns [29].
- Application: YARA's primary output is a set of rules that can be used to classify malware [23]. The power of YARA lies in its expressive syntax and the ease with which its rules can be shared [30].

B. Commercial Antivirus and Endpoint Protection Platforms (EPP)

Signature-based detection serves as a foundational component within comprehensive security suites, often marketed as EPP, Next-Generation Antivirus [5], or Endpoint Detection and Response solutions [3], [5].

- The Hybrid Approach: Leading commercial vendors e.g, Avast, employ a hybrid detection model using multiple detection techniques for defense-in-depth.36
- Vendor Landscape: ESET combines machine learning with crowdsourced threat intelligence, while Trellix uses behavioral analytics [12].
- The Role of Signatures in Commercial Products: Signature-based detection provides high-speed, low-resource initial filter. Given that the vast majority of malware encounters are variants of known threats, signature scanning is best suited for initial scanning.39

IV. Critical Analysis: Efficacy and Inherent Limitations

The deterministic nature of the approach, its reliance on matching predefined, known patterns, is the root cause of both its high precision and its inability to contend with novelty. This duality is not a flaw that can be engineered away but a fundamental property of the paradigm.

A. Strengths of Signature-Based Detection

The advantages of signature-based detection are:7

- Speed and Efficiency: Signature-based scanner involves comparing data against a database of known patterns [3], [31]. This is a computationally simple and highly optimized process, allowing security tools to scan vast amounts of data, with minimal impact on system performance.43
- High Accuracy for Known Threats: For any piece of malware that has been previously identified, analyzed, and had a signature created for it, signature-based detection is exceptionally effective [29].
- Low False-Positive Rate: A significant advantage over more inferential methods is an extremely low rate of false positives. Signatures are generated from confirmed malicious samples, they are highly specific.
- Simplicity and Proven Track Record: The underlying concept of signature matching is straightforward to understand and implement.7

B. Inherent Weaknesses and Limitations

Despite its strengths, the signature-based model some weaknesses as well [32].

- The Zero-Day Problem: The most profound limitation is its inherent inability to detect novel threats, commonly known as zero-day attacks until they have been captured and analyzed by researchers [32].
- Ineffectiveness Against Polymorphic and Metamorphic Malware: Modern malware authors are acutely aware of how signature-based detection works and actively design their creations to evade it [29]. With an estimated 97 of modern malware employing some form of polymorphism, this is a critical failure point for purely signature-based systems [1], [3].
- Critical Dependency on Updates: The efficacy of a signature-based tool is directly proportional to the currency of its signature database. Any delay in updating the database timely creates a window of vulnerability [31].
- Scalability Challenges and Performance Trade-offs: Scanning a file against a massive database can increase detection time and consume more system resources, presenting a scalability challenge.

TABLE I
COMPARISON OF SECURITY TOOLS

Tool	Primary Use	Signature Type	Development
ClamAV	Antivirus engine for servers	Hash and pattern matching	Open Source (Cisco)
Snort	Network intrusion detection	Rule-based patterns	Open Source (Cisco)
YARA	Malware classification	Pattern matching rules	Open Source (VirusTotal)
Microsoft Defender	Endpoint protection	Signatures and AI/ML	Commercial (Microsoft)

V. THE ADVERSARIAL DYNAMIC: MALWARE EVASION TECHNIQUES

The limitations of signature-based detection are not merely theoretical; they are actively and intelligently exploited by malware authors [3], [33].

A. Obfuscation

Obfuscation refers to a set of techniques used to make code unintelligible to analysts and automated tools without altering its core functionality [20], [21].

- String Obfuscation: One of the most common techniques is to hide revealing text strings. Indicators of Compromise (IOCs) like Command & Control (C2) server URLs, file paths, or names of suspicious Windows API functions are often concealed [18].
- Control Flow Manipulation: To confuse reverse-engineering efforts, malware authors disrupt the logical flow of their code [29]. The program's state determines which block of code executes next, making it extremely difficult to trace the execution path manually or with automated tools.53
- Dead-Code Insertion: This technique involves padding the malware's code with a large number of irrelevant or non-functional instructions.

B. Packing and Encryption

Packing and encryption are methods used to fundamentally alter the on-disk representation of a malicious executable.

- Packing: A packer is a tool that compresses or encrypts an executable file and wraps it with a small loader program, often called a "stub".49 When the user runs the packed file, the stub executes first. Its job is to decompress or decrypt the original malicious payload in memory and then transfer program execution to it.54
- Encryption (Cryptors): A cryptor is a specialized tool that focuses solely on encrypting the malware payload [29]. Malware authors may even pack a file multiple times with different packers and cryptors to create multiple layers of obfuscation that must be peeled away by an analyst [20].

C. Polymorphism and Metamorphism: The Ultimate Signature Evasion

These advanced techniques automate the process of creating unique malware variants to ensure that no two infections look the same from a signature perspective.

- Polymorphism: Polymorphic malware uses a built-in "mutation engine" to change its appearance with each replication [29]. Typically, the core malicious payload is encrypted [11].
- Metamorphism: Metamorphism is a more sophisticated and challenging technique. Instead of just encrypting a static payload with a changing wrapper, metamorphic malware completely rewrites its own code with each new infection.48 This presents a significant challenge to all forms of static analysis [4].

D. Bypassing Signatures in Practice

Attackers use several practical methods to defeat specific signature implementations.

- Binary Splitting & Bit-Flipping: To defeat a specific antivirus engine, an attacker can systematically identify the exact byte sequence that is being flagged as a signature. This can be done by splitting a malicious binary into two halves and testing which half triggers the antivirus repeating the process on the triggering half.
- Entropy Manipulation: Attackers embed large amounts of low-entropy data, such as plain text from a book or a dictionary of common words, into their packed binary. This added data lowers the file's overall calculated entropy, making it appear more like a normal, unencrypted program and thus bypassing entropy-based detection heuristics [8], [31].

VI. THE MODERN PARADIGM: INTEGRATION IN HYBRID DETECTION MODELS

The most successful platforms today are those that did not simply replace signatures but integrated them into a layered system, recognizing that the speed of signatures for known threats and the novelty detection of advanced methods are complementary, not mutually exclusive.

A. Beyond Signatures: An Overview of Complementary Techniques

Modern security platforms combine signature-based detection with several other techniques [3].

1) Heuristic Analysis: Heuristic analysis uses rules, algorithms, and expert-based analysis to identify suspicious characteristics or attributes within a file e.g, a heuristic scanner might flag a file because its code contains instructions to directly modify the operating system kernel [7], [8].

2) Behavioral Analysis & Sandboxing: The cornerstone of behavioral analysis is the sandbox: a secure, isolated, and virtualized environment where a suspicious file can be safely executed and its actions can be monitored without risk to the host system [10], [11]. Inside the sandbox we look for malicious actions such as:

- Modifying or deleting user files.
- Encrypting files in a manner consistent with ransomware.
- Attempting to disable security software.
- Making unusual network connections to known malicious domains or C2 servers.
- Capturing keystrokes or escalating privileges.

3) AI and Machine Learning (AI/ML): This approach leverages algorithms trained on massive datasets containing millions of both malicious and benign file samples.10 By processing this data, the ML model "learns" to identify the complex, subtle, and often non-obvious patterns and features that differentiate malware from legitimate software [29].

B. Hybrid Architectures in Practice

The hybrid approach is designed to leverage the strengths of each technique while mitigating its weaknesses.

- The Layered Funnel Architecture: A common and effective architecture operates like a funnel. All incoming files are first subjected to a signature scan. This is the fastest and least resource-intensive layer, and it immediately filters out the vast majority of common, known threats.39
- Case Study: Microsoft Defender for Endpoint provides a clear example of this hybrid model in a commercial product On the client device, it performs real-time scanning using local signature databases and lightweight heuristic and behavioral models [7], [8]. If a file is identified as unknown or suspicious, its metadata or the file itself is sent to Microsoft's cloud protection service. In the cloud, a much more powerful suite of tools is brought to bear, including advanced ML models, stacked ensemble classifiers, full-file deep learning analysis, and emulation engines that can dynamically unpack malware to observe its runtime behavior [29].
- Academic Hybrid Models: The academic community is actively researching novel ways to combine these techniques. Proposed models include architectures that transform application files into multiple representations, for example, a Function Call Graph (FCG) to represent code structure and a grayscale image to represent binary texture, and then feed these representations into parallel machine learning classifiers [12].
- **Signature**: Lowest FP rate but reactive
- **Heuristic**: Balance between proactive detection and FP rate
- **Sandbox**: Highest detection efficacy but resource-intensive
- **AI/ML**: Adaptable but requires significant training resources

VII. The Enduring Role of Signatures in a Layered Defense

A. Synthesis of Findings

This review has systematically examined the mechanisms, tools, and efficacy of signature-based malware detection [3], [29]. The analysis confirms that it is a mature technology with a well-defined and dualistic nature. Its primary strengths, unparalleled speed, high accuracy for known threats, and a low rate of false positives, are a direct result of its deterministic, pattern-matching design.7 These attributes have made it a cornerstone of cybersecurity for decades. However, this same deterministic nature is the source of its fundamental weaknesses: an inherent inability to detect novel zero-day exploits and a vulnerability to evasive malware that is polymorphic or metamorphic [29], [33].

TABLE II
Malware Detection Methodologies (Abridged)

Type	Strength	Weakness	Overhead
Signature	Fast known-threat detection	No zero-day protection	Low
Heuristic	Finds new variants	High false positives	Medium
Sandbox	Evasion-resistant	Slow execution	High
AI/ML	Pattern prediction	Data-hungry	Med-High

B. Future Outlook and Enduring Relevance

The future of signature-based detection is not one of obsolescence, but of deeper integration and automation within a broader, hybrid defense framework [3]. Its role will continue to evolve, leveraging new technologies to enhance its capabilities and solidify its position as a foundational layer of security.

- Integration with AI and Machine Learning: The relationship between AI/ML and signature-based detection is becoming symbiotic. Beyond acting as a parallel detection mechanism, AI will increasingly be used to augment and accelerate the signature creation process itself [12]. Machine learning models can be trained to analyze newly discovered malware [1] samples and automatically generate robust, high-quality signatures (such as complex YARA rules). This automation can drastically reduce the "time-to-signature"—the critical window between the discovery of a new threat [2] and the deployment of protection—from days or hours to minutes.61
- Cloud-Based Threat Intelligence: The dependency on periodic database updates is being overcome by cloud-native architectures. As exemplified by modern commercial EPPs, when a new threat [2] is detected and analyzed on a single endpoint anywhere in the world, a corresponding signature or blocking rule can be generated in the cloud and distributed in near real-time to every other protected endpoint in the global network [16].
- The Indispensable Filter: Perhaps its most critical future role is that of a high-performance filter. The volume of daily malware [1] is too immense to be processed entirely by computationally expensive methods like full-system emulation or deep learning analysis. Signature-based detection will continue to serve as the essential first line of defense, efficiently handling the "background noise" of millions of known, common threats with minimal resource consumption [3], [15]. This crucial filtering

function frees up advanced behavioral and AI-powered systems to dedicate their processing power to analyzing the much smaller subset of threats that are truly novel, evasive, and potentially more dangerous.7

C. Concluding Remarks

In the complex and ever-evolving landscape of cybersecurity, a defense-in-depth strategy is paramount. The debate is no longer a matter of choosing between signature-based, behavioral, or AI-driven detection [3], [15]. The evidence overwhelmingly indicates that an effective defense requires the intelligent integration of all these methods. By combining the deterministic certainty and efficiency of signatures for known threats with the adaptive, probabilistic power of behavioral and AI systems for unknown threats, organizations can build a security posture that is both robust and resilient [12]. Far from being a relic of the past, signature-based detection has secured its permanent and vital role as the foundational bedrock upon which modern, hybrid cybersecurity architectures are built.

REFERENCES

[1] H. Alasmary, A. Khormali, A. Anwar, J. Park, J. Choi, D. Nyang, and A. Mohaisen, "Analyzing and detecting emerging internet of things malware: A graph-based approach," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8977–9888, 2019.

[2] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "Intelligent os x malware threat detection with code inspection," *Journal of Computer Virology and Hacking Techniques*, vol. 14, pp. 213–223, 2018.

[3] M. Goyal and R. Kumar, "The pipeline process of signature-based and behavior-based malware detection," in *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, 2020, pp. 497–502.

[4] R. Sihwail, K. Omar, and K. A. Z. Ariffin, "A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 8, pp. 1662–1671, 2018.

[5] R. Muthuregunathan, S. Siddharth, R. Srivathsan, and S. R. Rajesh, "Efficient snort rule generation using evolutionary computing for network intrusion detection," *International Conference on Intelligent Agent & Multi-Agent Systems*, 2009.

[6] F. A. Bhuiyan, K. E. Brown, M. B. Sharif, Q. Dai, and D. A. Schneider, "Assessing modality selection heuristics to improve multimodal machine learning for malware detection," *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, 2020.

[7] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," *The 5th Conference on Information and Knowledge Technology*, pp. 113–120, 2013.

[8] E. M. Alkhateeb and M. Stamp, "A dynamic heuristic method for detecting packed malware using naive bayes," *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pp. 1–5, 2019.

[9] M. Graziano, D. Canali, L. Bilge, A. Lanzi, and D. Balzarotti, "Needles in a haystack: Mining information from public dynamic analysis sandboxes for malware intelligence," *USENIX Security Symposium*, pp. 1057–1072, 2015.

[10] O. Alrawi, M. Y. Wong, A. Avgetidis, K. Valakuzhy, E. Chan, E. Karlin, R. Ensafi, K. Borgolte, and M. Antonakakis, "Sok: An essential guide for using malware sandboxes in security applications: Challenges, pitfalls, and lessons learned," *arXiv preprint arXiv:2302.01134*, 2023.

[11] M. Belaoued and S. Mazouzi, "Statistical study of imported apis by pe type malware," *2014 International Conference on Advanced Networking Distributed Systems and Applications*, pp. 82–86, 2014.

[12] E. de Oliveira Andrade, J. Viterbo, C. Nita-Rotaru, J. Guérin, and F. C. Bernardini, "A model based on lstm neural networks to identify five different types of malware," *Procedia Computer Science*, vol. 159, pp. 1821–1830, 2019.

[13] A. Bensaoud, J. Kalita, and M. Bensaoud, "A survey of malware detection using deep learning," *Machine Learning with Applications*, 2023.

[14] R. A. Yunmar, S. S. Kusumawardani, Widyawan, and F. Mohsen, "Hybrid android malware detection: A review of heuristic-based approach," *IEEE Access*, 2024.

[15] I. Santos, Y. K. Penya, J. Devesa, and P. G. Bringas, "N-grams-based file signatures for malware detection," *International Conference on Enterprise Information Systems*, 2009.

[16] S. K. Patel and A. Sonker, "Rule-based network intrusion detection system for port scanning with efficient port scan detection rules using snort," *International Journal of Future Generation Communication and Networking*, vol. 9, no. 6, pp. 339–350, 2016.

[17] R. Velea, C. Ciobanu, L. Margarit, and I. Bica, "Network traffic anomaly detection using shallow packet inspection and parallel k-means data clustering," *Studies in Informatics and Control*, 2017.

[18] A. Sami, B. Yadegari, H. Rahimi, N. Peiravian, S. Hashemi, and A. Hamzeh, "Malware detection based on mining api calls," *ACM Symposium on Applied Computing*, 2010.

[19] K. Bakour, H. M. Ünver, and R. Ginhoux, "A deep camouflage: evaluating android's anti-malware systems robustness against hybridization of obfuscation techniques with injection attacks," *Arabian Journal for Science and Engineering*, 2019.

[20] S. Banescu, T. Wüchner, A. Salem, M. Guggenmos, M. Ochoa, and A. Pretschner, "A framework for empirical evaluation of malware detection resilience against behavior obfuscation," *IEEE Computer Security Foundations Symposium*, pp. 326–338, 2015.

[21] A. Bacci, A. Bartoli, F. Martinelli, E. Medvet, and F. Mercaldo, "Detection of obfuscation techniques in android applications," *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018.

[22] P. Faruki, A. Bharmal, V. Laxmi, M. S. Gaur, M. Conti, and M. Rajarajan, "Evaluation of android anti-malware techniques against dalvik bytecode obfuscation," *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, 2014.

[23] P. S. Patel, R. S. Kunwar, and A. Thakar, "Malware detection using yara rules in siem," pp. 287–306, 2023.

[24] G. Fortino, C. Greco, A. Guzzo, and M. Ianni, "Sigil: A signature-based approach of malware detection on intermediate language," *Computer Security. ESORICS 2023 International Workshops*, pp. 256–266, 2023.

[25] G. K. Bada, W. K. Nabare, and D. K. K. Quansah, "Comparative analysis of the performance of network intrusion detection systems: Snort, suricata and bro intrusion detection systems in perspective," *International Journal of Computer Applications*, vol. 176, pp. 29–36, 2020.

[26] M. Masdari and S. Ahmadzadeh, "A survey and taxonomy of the fuzzy signature-based intrusion detection systems," *Applied Soft Computing*, 2020.

[27] G. V. Nadiammai and M. Hemalatha, "Handling intrusion detection system using snort based statistical algorithm and semi-supervised approach," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 6, pp. 2914–2922, 2013.

[28] A. Garg and P. Maheshwari, "Performance analysis of snort-based intrusion detection system," *2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 01, pp. 1–5, 2016.

[29] S. P. Niraj and A. K. Tiwari, "Performance analysis of signature based and behavior based malware detection," *Research Journal of Engineering Technology and Medical Sciences*, 2020.

[30] Q. Si, H. Xu, Y. Tong, Y. Zhou, J. Liang, L. Cui, and Z. Hao, "Malware detection using automated generation of yara rules on dynamic behaviors," *Science of Cyber Security*, 2017.

[31] W. S. Bhaya and M. A. Ali, "Review on malware and malware detection using data mining techniques," *Journal of University of Babylon for Pure and Applied Sciences*, vol. 25, no. 5, pp. 1–8, 2017.

[32] S. Venkataraman, A. Blum, and D. Song, "Limits of learning-based signature generation with adversaries," *Department of Electrical and Computing Engineering*, 2008.

[33] M. Fan, X. Luo, J. Liu, M. Wang, C. Yin, W. Feng, Z. Tian, and Q. Guo, "Effectiveness of android obfuscation on evading anti-malware," *Proceedings of the eighth ACM conference on data and application security and privacy*, 2018.