Original Article
ISSN (Online): 3048-894X

Journal of Digital Security and Forensics
July-December 2025 2(2), 88–107

A REVIEW OF BEHAVIOURAL FINGERPRINTING FOR CLOUD RANSOMWARE DETECTION VIA SYSTEM AND API CALL ANALYSIS

Bhavesh Kumar Sharma [1], Dr. Kapil Shukla [2], Dr. Krishna Modi [2]

[1] Student at School of Forensic Science, National Forensic Sciences University, Gandhinagar, Gujarat, India
[2] Assistant Professor at School of Forensic Science, National Forensic Sciences University, Gandhinagar, Gujarat, India

Received 12 October 2025
Accepted 15 November 2025
Published 04 December 2025

Corresponding Author

Dr. Kapil Shukla, kapil.shukla@nfsu.ac.in

DOI
10.29121/DigiSecForensics.v2.i2.2025.69

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

ABSTRACT

The rapid spread of cloud computing has opened profit centres for ransomware attacks. Classical methods of detection are static in nature and signature-based have more and more difficulties with modern ransomware. Ransomware today employs obfuscation and misuses genuine administrative functions, especially in API-centric cloud environments. The paper delivers a structured literature review that focuses on various methodologies for ransomware detection advocating for the central importance of classifying and assessing attacks based on their actions. We argue that behavioural fingerprinting based on extensive studying of cloud workloads and API calls to the cloud control plane is the best approach for early and accurate detection of cloud-native ransomware. This review looks at what is present in the field of malware analysis, we present the fundamental elements of behavioural fingerprinting which we see across the ransomware attack cycle, also we note that which system and API calls are the main data sources for very accurate fingerprints. Also, we report on the machine learning and deep learning tools which we use to automate detection into which we are also putting forward the issue in the real-world setting. Performance issue. We look at what issues bring up as we apply these principles to cloud structures which are also home to new primary data sources in the form of cloud API logs for defenders. We end with a review of what we found out, we also put forth that there is a need for cloud specific data sets and explainable AI which are present research gaps and we also put forth what may prove to be very good areas for future research in what is very much a growing field of cyber security.

Keywords: Ransomware, Cloud Security, Behavioural Fingerprinting, System Calls, API Calls, Machine Learning

1. INTRODUCTION

Ransomware has grown out of the first simple attacks like that of the 1989 AIDS Trojan to become an advanced and large-scale cyber-crime business which also includes the use of complex encryption, Ransomware-as-a-Service models, and large-scale targeted attacks what we see in Big Game Hunting and double extortion. As companies move more and more to the cloud we are seeing a rise in attacks which target cloud-based assets that contain very important information also there are various types of attacks which are a result of issues related to misconfiguration, stolen creds, unprotected APIs, and vulnerable software. Also unlike past end point

attacks today's cloud-based ransomware uses the very functions which are supposed to be secure in a cloud environment against which present signature-based defence solutions do not work and in fact we need to look at different new context aware solutions that are able to detect that which is malicious within what appears to be normal admin actions.

1.1. THE ASCENDANCY OF RANSOMWARE AS A PREMIER CYBER THREAT

Malicious actors get into a target system, put forward your important data to a locked-out state which in turn asks for a ransom in crypto currency for the key which will unlock it. Also, which is the scale of the issue that is brought about by such an attack to which we see from total disruption of services to that of very large financial penalties and very hard to repair reputation damage.

The history of ransomware has seen its very remarkable evolution from basic ideas to a very complex and industrial scale criminal business. In the first reported case of the 1989 AIDS Trojan, also known as the PC Cyborg virus, distribution was via floppy disks to attendees of a World Health Organization conference. After 90 reboots, the malware would encrypt file names and demand users to mail in a check for $189 to a PO box in Panama . While this early foray into digital extortions was technical in its concept it was not so in scale or success.

What we did see over the decades was a gradual growth in tech savvy. By the mid 2000's we had Trojans like Gpcode which used much more robust RSA encryption schemes Zakaria et al. (2022), which in turn made it very hard to decrypt without the proper key. Later variants, such as Gpcode.AK, utilized a 1024-bit RSA key, representing a significant technical escalation . However, the real game changer in ransomware's growth was not so much in tech but in economics.

This economic growth has brought about what you may term a technical and strategic arms race, in which we see the development of ransomware into a business model we may call RaaS Oz et al. (2022). RaaS platforms run like a business franchise at the top you have the skilled developers that put out very complex ransomware and also run the technology which supports it; below that you have the affiliates that are not as tech savvy which pay into a profit-sharing scheme Afzaal et al. (2023), Alexander (2020), Alwashali et al. (2021). This model has reduced what was a high barrier to entry to a low one which in turn has seen a large increase in the number and variety of attacks and has at the same time made the threat landscape much more complex Oz et al. (2022).

Concurrently, ransomware groups have pivoted from indiscriminate, volume-heavy campaigns to a more focused and personally tailored approach termed as "Big Game Hunting" (BGH) . This approach centres around businesses, health institutions, financial entities, and critical infrastructure targets which are more willing to pay large sums to evade operational shut down, and thus, are considered high value targets . This shift in strategy becomes even more pronounced with the use of sophisticated extortion methodologies. "Double extortion" encompasses not just the encryption of data but also the exfiltration of the victim's data Ghani et al. (2022). This exfiltrated data can be held hostage with the intention of ransom, threatening public release or "doxing" Ghani et al. (2022), thus creating a scenario where victims may be forced to pay even if they possess data backups. Ransomware has outgrown its status as a technical inconvenience and has become a very organized economic crime which is constantly evolving to better profit from it.

1.2. THE CLOUD AS A NEW FRONTIER FOR RANSOMWARE ATTACKS

Owing to the strategic advantages of flexibility and cost-effectiveness, firms are migrating to the cloud at an ever-increasing pace. This means, cloud platforms are now the subjects of increasing, and highly sophisticated, ransomware attacks. A cloud environment, akin to an irresistible honey pot, is rich with information and serves the business-critical data and applications which means the ramifications of a successful ransomware attack are unimaginably catastrophic. The operational backbone of organizations hinges on numerous features which include customer databases, sensitive intellectual property, critical financial records, and heavy production workloads.

The shared responsibility frameworks, a pivotal element of cloud security, is too vulnerable in defence and posture, and organizations allocating resources to it, become precariously exposed. Misunderstanding or neglecting these responsibilities can lead to users "unknowingly running workloads that are not fully protected". In the cloud we see a different set of ransomware attack vectors as compared to what we find in traditional on premises endpoints. We are in a fundamental shift in the ransomware model when it comes to the cloud. As opposed to traditional endpoint attacks which may use unpatched software flaws (like the WannaCry which used the EternalBlue exploit) or social engineering through a phishing email to get a user to run a bad binary, in the cloud we see the main vectors as the abuse of legitimate admin functions. This changes the detection game from identifying bad code to that of sorting out bad intent in a series of what are in fact authorized actions.

In the cloud we see primary attack vectors of cloud misconfiguration. This is probably the most common starting point of an attack and the most easily exploited. Publicly reachable cloud storage buckets (e.g. Amazon S3), unprotected databases, and overly permissive storage buckets are frequently identified as one of the most prevalent and perilous threats Sy et al. (2024). These are not software bugs in the traditional sense, rather they are lapses in the security posture management of an organization.

Use of stolen credentials and absence of robust systems of governance on access privileges: Cyber criminals abscond with service and user account credentials using various techniques, including phishing, credential stuffing, and dark market purchases Alquwayzani et al. (2024). This remains a highly effective vector, with one 2024 report associating stolen credentials with 86% of security breaches in cloud networks . An attacker by logging as a user, with little obstruction can, with a purchased credential, log on as a user with little obstruction. If IAM policies are too generalized, they can within the boundaries of the system, masquerade as a legitimate user, and laterally pivot to sponsorship, “execute the payload,” and do a number of other activities. All these are within the boundaries of the system. From the attacker’s standpoint, in respect to the cloud service provider, an account with stolen credentials is, from the perspective of the cloud service provider, a legitimate account user skeletal user account with skeletal utilisation.

Insecure application programming interfaces (APIs). Every cloud service is by design API driven. Deficiencies in any of these APIs within the ecosystem of a cloud service provider or in bespoke applications developed by the cloud customer can be easily abused to gain unauthorized access, manipulate, or even disrupt services. Common API vulnerabilities include broken authentication, excessive data exposure, and injection flaws Faheem et al. (2017).

Software supply chain attacks. The attack surface of any cloud- based environment is dramatically increased by the consolidation of modern applications. An attacker only needs to compromise a third-party software library, container image, or integrated cloud SaaS application, in order for their ransomware to gain a foothold Ladisa et al. (2022). This attack aims to inject malicious code into software components to compromise all downstream users Ladisa et al. (2022). The 2020 SolarWinds attack serves as a canonical example, where hackers targeted a third-party vendor to gain access to customer systems Poddar and Rani (2023). An attacker can embed their malicious payload within a so-called 'trusted' component. The victim deploys this component without suspecting the presence of any payload, thus enabling the ransomware to gain access within the perimeter of the victim's defence.

The new attack vectors greatly disable old means of signature-based defences. A signature-based IDS or antivirus system attempts to answer the question, "Does this file or network packet contain something malicious, and if so, what?" In the cloud, the more pertinent question is, "Does this set of authenticated API calls masked by an authenticated user have malice at its purpose?". Malicious actors using "legitimate AWS SDK calls" can appear as "normal application behavior," rendering traditional rules ineffective . Given the nature of this question, we need to evolve to new security frameworks where context, order, and action matter.

Table 1

Table 1 Analysis of Primary Cloud-Native Attack Vectors

Attack Vector	Description of Attacker Action	Example Vulnerability	Primary Detection Challenge
Cloud Misconfiguration	Exploiting "lapses in the security posture management," not traditional software bugs.	Publicly reachable S3 buckets, unprotected databases, overly permissive storage Sy et al. (2024)	Differentiating malicious discovery from legitimate administrative scanning or configuration errors.
Stolen Credentials / IAM Abuse	Using credentials from phishing or dark markets to "masquerade as a legitimate user."	Overly generalized IAM policies; lack of robust access governance Alquwayzani et al. (2024).	The attacker's actions are "authorized." The challenge is discerning malicious *intent* from a series of legitimate-seeming actions.
Insecure Application Programming Interfaces (APIs)	Abusing deficiencies in API design (provider or customer-built) to gain access or manipulate services.	Broken authentication, excessive data exposure, injection flaws. Faheem et al. (2017)	Attacks are "API driven" and can be masked within a high volume of legitimate API traffic, appearing as "normal application behavior."
Software Supply Chain Attacks	Injecting malicious code into a "trusted" 3rd-party component (library, container image, SaaS app).	Compromised 3rd-party vendor (e.g., SolarWinds) Poddar and Rani (2023), Ladisa et al. (2022).	The malicious payload is deployed "without suspecting" by the victim, bypassing perimeter defenses entirely.

2. RANSOMWARE ANALYSIS: FROM STATIC TO DYNAMIC BEHAVIOUR

Analysing and detecting malware, including ransomware, has been a two-pronged approach, at least so far. These methods are called static and dynamic. Each malware study approach has its pros and cons. Each new generation of ransomware has also driven evolution in analysis methods. As a result, there has been a significant shift from using static signatures toward context-rich and resilient dynamic behavioural analysis techniques Damodaran et al. (2015).

2.1. STATIC ANALYSIS: PRINCIPLES AND LIMITATIONS

Static analysis, or static code analysis, is the assessing and inspecting of the internals of a program's binary code and structure without executing the program. This is an approach which is the basis for traditional AV (Antivirus software) technology, due to much of its execution speed and low performance drain, which is economically sensitive for scanning massive amounts of files. The primary techniques employed in static analysis include:

- **Signature Matching:** This is the simplest form of static analysis, where a file's hash (like an MD5 or SHA-256) or some particular sequence of bytes in the file, is checked against a set of malware signatures. A positive match results in the file being labelled as malicious.
- **PE Header Analysis:** In the case of Windows executables, the Portable Executable (PE) header is a repository of program information which includes imported libraries, function calls, and compilation times Aboaoja et al. (2022). Out of the PE header data anomalies or suspicious entries may be telltales of malicious intent.
- **Opcode Sequence Analysis:** This approach is to disassemble the code and study the series of low-level assembly instructions (opcodes) Sun et al. (2019). Some patterns or frequencies of opcodes may tell us which code is malicious like that which is used in encryption or evasion.

While effective in the identification of known threats where we already have an established signature, we have seen the utility of static analysis drop off sharply in recent times due to what I would term the “arms race” in obfuscation that malware authors have set in motion. What static analysis does is basically see what a given program looks like but does not really look at what is actually done by the program. Today’s ransomware developers are very much aware of this issue and have at their disposal an advanced set of evasion tools that are very much aimed at defeating static analysis and signature detection. These tools include:

- **Packing and Encryption:** Ransomware’s infected payload is put into a harmless looking wrapper. At the time of execution, a small “stub” of code which is part of the ransomware itself runs to decompress or decrypt the payload in memory which then proceeds to run Ki et al. (2015). Static analysis tools only report on the innocent looking wrapper which in fact is a front for the true malicious code .
- **Polymorphism and Metamorphism:** Polymorphic malware changes its binary signature between each infection but has the same base function. Metamorphic malware takes it a step further and rewrites its code with each propagation which includes instruction substitution and code reordering Avhankar (2025). Also, it is guaranteed that no two samples will have the same static signature which in turn makes hash-based detection fail.
- **Code Obfuscation:** Attackers present a challenge to disassembly efforts by which they insert what is essentially "garbage code" into their binary, create complex and non-intuitive flow structures, and also put forth efforts to hide what APIs are actually being called up in the code Afzaal et al. (2023). This can include "instruction substitution," where one instruction is replaced by a semantically equivalent but different sequence of instructions .

2.2. DYNAMIC ANALYSIS

The action of a program dynamic analysis puts the focus on what the code does instead of what the code is. This gives a much more reliable indication of malicious intent. We see the process of creating a unique profile of a malware's actions during run time as behavioural fingerprinting. The data which goes into this fingerprinting is many and varied and gives a complex picture of the program's behaviour which includes: File actions (files which are created, read, written, deleted), registry changes, network traffic (C&C traffic, data exfiltration), which processes ransomware launches or terminates, and what system and API calls ransomware makes to the OS Sihwail et al. (2018), Chhillar et al. (2025).

Dynamic analysis does very well at finding ransomware because the main thing ransomware does is file encryption plays out over a series of visible system interactions which the ransomware can't easily hide. No matter how much a ransomware's code might be concealed, at some point it has to interact with the operating system's kernel to read, encrypt, and write files, then the actions themselves constitute a behavioural pattern which can be recognized regardless of the code. This means that dynamic analysis of a system is much more equipped to handle zero-day attacks and new variants of ransomware, since new variants can be captured based on the abuse of the system and not because of a signature provided.

Still, dynamic analysis has its negative side. It is more costly and requires more time to complete than static analysis. However, we see that dynamic analysis has its issues. It is a more resource intensive and time out process as compared to static analysis Damodaran et al. (2015). Also, we see that advanced malware may put on a different show. It will try to identify if it is in a controlled analysis environment and will change its behaviour to what is expected of a non-threatening agent. This category of "malware dynamic analysis evasion techniques" is a well-documented field Lalejini and Ofria (2018). Malware can actively detect sandboxed environments using timing checks or by looking for specific artifacts Genç et al. (2019). Malware will remain dormant to evade analysis. Though there are these issues, dynamic analysis still gives us the true report of what a program is doing and is an essential tool in our war against ransomware.

2.3. HYBRID ANALYSIS

Taking into consideration the functional advantages and disadvantages of both static and dynamic analyses, a hybrid approach aims to synthesize both methods to produce a single, comprehensive and sophisticated detection pipeline Damodaran et al. (2015). Common practice in hybrid models uses static analysis to filter out low-risk, low-cost files. Files can be scanned at a high velocity for pre-defined signatures and other static components that are static. This "basic static triage" is a low-cost filter Oppliger (2017).

Pieces that fall into the category of ambiguous and cannot be classified for certain are classified as suspicious and are then subjected to a more comprehensive, albeit costly dynamic analysis, albeit in the form of a sandbox. This foster concentrated engagement on analytical resources. With the static analysis approach, the files, the overwhelming majority of files which are benign, can be authenticated within a limited timeframe, and the expensive dynamic analysis, can be allocated and reserved for a minute subset of files that are worthy of deeper inspection. This clinched defence approach fosters the total gain detection payment and fosters an enhancement on system overall detection payment and throughput

on the system, which yields a more pragmatic and scalable answer for real-world deployment Chhillar et al. (2025).

The comparative analysis of the described approaches is presented in the following table to facilitate appropriate understanding of the described methods.

Table 2

Table 2 Comparative Analysis of Ransomware Detection Paradigms

Paradigm	Core Principle	Primary Data Sources	Key Strengths	Critical Weaknesses
Static Analysis	Code and structure review without execution.	PE Headers, Opcode Sequences, Byte Signatures, Strings Aboaoja et al. (2022), Sun et al. (2019).	High speed, low performance overhead, effective for known threats.	Easily defeated by obfuscation Afzaal et al. (2023), packing Ki et al. (2015), polymorphism, and metamorphism Avhankar (2025), ineffective against zero-day threats.
Dynamic Analysis	Code execution in a controlled environment to observe its actions.	System Calls, API Calls, File System I/O, Network Traffic, Memory Dumps Sihwail et al. (2018), Chhillar et al. (2025).	Resilient to obfuscation Damodaran et al. (2015), effective against zero-day threats.	High resource consumption (CPU, memory), performance overhead; potential for sandbox evasion by sophisticated malware Genç et al. (2019), Lalejini and Ofria (2018).
Hybrid Analysis	Combining static and dynamic techniques for a layered approach.	All sources from static and dynamic analysis.	Balances speed and depth; uses static analysis for fast triage Oppliger (2017) and dynamic analysis for deep inspection Chhillar et al. (2025), improving overall efficiency and accuracy Damodaran et al. (2015).	Complexity in implementation; effectiveness depends on the intelligent integration of both static and dynamic components.

Dynamic analysis outperforms static in the identification of modern malware's primary tactics which is what this review reports on. The shift from static to dynamic analysis is not a matter of preference for a different tool but is an evolution in the field of security which is a response to malware's constant improvement at evading detection. This places behavioural fingerprinting forward as the best and most promising base for the development of next generation ransomware detection systems in complex settings like the cloud.

3. BEHAVIOURAL FINGERPRINTING

Behaviour which malware presents is what we base our behavioural fingerprinting on in the fact that while malware can change its visible or static form, it may not at all times change the way it acts. All programs indeed both that are harmless and those which are malicious they interface with the base operating system to perform their tasks like accessing a file, reserving memory, or going out on the network. These interactions, when monitored and analysed in sequence, create a distinct behavioural pattern, or "fingerprint," that can uniquely identify the program's intent Yadav et al. (2020). For ransomware, this fingerprint is defined by the specific sequence of actions required to achieve its objective of data encryption and extortion.

3.1. DEFINING THE BEHAVIOURAL FINGERPRINT

A behavioural profile is not a single out of context event but a complex pattern which is a series of system level actions. In this case the value of this approach is in its ability to tell the story of an attack. Many of the individual actions that ransomware performs are actually very basic and normal. For example, a program

which reads a user's files is a function which a word processor or a backup utility also performs. Also, the use of crypto APIs is a required element in secure web browsing and data protection. What makes ransomware malicious is not any one of these actions in particular, but the atypical set, sequence, and scale of these actions.

3.2. FINGERPRINTING THE RANSOMWARE LIFECYCLE

Developing observable markers that could help create a robust behavioural fingerprint is useful during the different stages of a typical ransomware attack. Focusing the analysis on the ransomware lifecycle allows detection systems to not only map the stages of an attack but more importantly, act at the first possible opportunity. The lifecycle can be divided into three broad phases: Pre-encryption, encryption, and post-encryption.

During the Pre-Encryption Phase, Reconnaissance and Intrusion is done. This is the most important and critical of the early phases of a stage where the ransomware is trying to set itself up to dish out its main payload.

The Destruction stage potentially is still the stage in a pattern of attack where the most damage is done. While all is not lost at this stage in the pattern of detection, highlighted damage control is paramount. There is a critical data loss and rampant damage is visible. Once from the target network, the I/O from the target network can be classified as the internal data stream of the target network. All data streams and connections of the target network are classified as a target user's private resource.

- **Significant Use of CPU and Memory Resources:** Increased use of CPU and Memory Resources comes due to employing complex cryptographic algorithms such as encryption during processes like enforcement of security. High CPU usage and high I/O operations are considered hallmark indicators of this phase Sekar et al. (2024).
- **Block of Text with Significant Entropy:** Data sets are random and unstructured during encryption, their nature alarming a user to the presence of ransomware. A sharp increase in entropy being monitored while writing files suggests the presence of non-benign files. Encrypted files are unstructured with entropy close to the peak value while certain benign files like including text or certain types of images are classified as possessing low entropy. This technique, often using Shannon's formula, observes the entropy difference between read and write operations and is a powerful heuristic for detection Sekar et al. (2024)

3.3. THE CRITICALITY OF EARLY DETECTION

The invariable end goal of a behaviour-based detection system is that it goes beyond the basic task of detecting ransomware at all which is to report on it very early. What we mean by early is we see the ransomware attack in the pre-encryption stage Jinmei et al. (2024). The difference between the pre-encryption and encryption stages is what really matters. As academic research emphasizes, "The value of detecting cryptographic ransomware is at the pre-encryption stage. It is useless after the encryption activity is completed because data loss has already happened" Zakaria et al. (2022).

4. SYSTEM AND API CALL INTELLIGENCE FOR ADVANCED MALWARE FINGERPRINTING

The performance of any behavioural fingerprinting system is a function of the quality and detail of the data it looks at. To put together a very accurate fingerprint which in turn is able to tell apart which is normal system activity and what is in fact malicious, we have to look at program behaviour in great depth. System calls and application programming interface (API) calls serve as the primary data sources for this task, providing a detailed, ground-truth record of a program's interactions with the operating system and its resources.

4.1. SYSTEM CALL MONITORING

The interaction between an application and the operating system kernel is first established through system calls. A program needs to perform a privileged operation such as opening a file, creating a new process, or sending data through a network. To do this, the program needs to issue a system call which requests the kernel to open a file and perform the action. Analysing the behaviour of the program in question is then recorded, monitoring system calls Canzanese et al. (2015).

Monitoring system calls is the most effective way to analyse program interaction with the operating system. Usually, data collection for system call analysis is done through the specific applications within the operating system. On for example in Linux based systems the perf tool is used to trace all system calls which are run system wide or by a specific process Udeshi et al. (2025), which it does by capturing info such as the call name (e.g. openat, read, write) its arguments and its return value Chew et al. (2024). This raw stream of system call events is a detailed chronological log of the program’s activity.

To make this data useful for machine learning models it must be transformed from raw trace into a structured feature vector . The simplest technique for counting each distinct system call is to count its occurrences within a defined time window. A ransomware process might indicate the presence of a very high frequency of file-related calls, such as openat, read, write, and unlink Udeshi et al. (2025).

4.2. API CALL TRACING

System calls expose a system at the lowest level while application programming interfaces (API) calls deal with a level of abstraction that more clearly describes a program's intended purpose. APIs, like the Windows API or those of cloud providers, offer proprietary functions that condense complex chains of system calls into one meaningful action. A case in point, one Windows CryptoAPI function call to CryptEncrypt is stronger and more direct evidence for encryption activity than inferring the same intent from a long series of generic system calls to read and write.

Empirical studies show that the vast majority of ransomware leverages widely used, standard cryptographic APIs, such as the Windows CryptoAPI Sajid et al. (2025). Therefore, real-time interception and analysis of these Windows cryptographic API calls is a key strategy for constructing a high-fidelity digital fingerprint Sajid et al. (2025). Analysis of API calls is important in constructing a digitized fingerprint and can be divided according to the surrounding context.

4.3. SYNERGISTIC DATA FUSION FOR A RESILIENT FINGERPRINT

A strong and enduring behavioural fingerprint is not built from a single data source. High skilled adversaries may try to avoid detection by altering or skipping particular types of observed activities. Thus, a defence-in-depth approach to securing digital environments entails the integration of data from several, diverse behavioural streams. Cross correlating signals on multiple layers of the system stack enables the detection model to create a fuller and much more difficult to fabricate impression of the activity of the program.

Such a data source hierarchy is capable of trading evasion potential for semantic richness. To this end, a multi-layered monitoring approach is a necessity to have a robust detection framework. At the highest level are API calls like CryptEncrypt. These are very rich in meaning, they clearly put forth what is intended, but at the same time are the easiest for malware to put a spin on or attach to. Below that are system calls like read and write, which while not as directly telling in terms of what is going on, are very basic to input and output and thus more difficult for malware to avoid. At the very bottom are file system I/O patterns which we see as the result of system calls rather than the calls themselves.

At the most fundamental level are Hardware Performance Counters (HPCs) which track events like CPU cycles, cache misses, and branch mispredictions that are managed directly by the processor Botacin and Grégio (2022). While semantically poor (a cache miss does not explicitly signal "encryption") they are very hard for user space malware to manipulate which makes them a powerful and evasive proof Demme et al. (2013). Research into HPC-based monitoring aims to leverage these registers to monitor application behavior at the hardware level Sayadi et al. (2024).

A robust detection approach should bind these types of signals. For example, a strange crypto API call may be backed up. At the same time see a spike in CPU use and cache misses reported by HPCs. This mix of a rich in meaning but also easy to game signal with a poor in what it says out but very reliable signal puts forth a pattern that is at once meaningful and hard to reproduce. Also, we see value in other data sources for this fusion.

- **Memory Dumps and Forensics:** Examining the memory of processes might uncover coding instructions where the ransomware stealthily embeds its nefarious instructions into a legitimate process to conceal its presence. This "process injection," often accomplished using APIs like CreateRemoteThread, can be detected through memory analysis. Such analysis might also reveal hidden or dynamically loaded modules that evade static analysis,
- **Network Traffic:** Analysing network traffic can reveal both the first communication to a C&C server to retrieve keys for encryption and the data bleeding that is typical of double extortion techniques. This is a standard feature of dynamic analysis Sihwail et al. (2018), Chhillar et al. (2025).

Integrating these disparate data streams allows a detection system to transcend basic event analysis and engage in sophisticated contextualized reasoning, which greatly improves its evasion resistance.

Table 3

Table 3 A Multi-Layered Hierarchy of Data Sources for Fingerprinting

Data Source Layer	Semantic Richness (Clarity of Intent)	Evasion Resistance (Hardness to Fake/Avoid)	Description and Use Case
API Calls	Very High	Low	e.g., CryptEncrypt. "Clearly put forth what is intended" but are the "easiest for malware to put a spin on" or obfuscate.
System Calls	High	Medium	e.g., read, write. Less direct than APIs but "more difficult for malware to avoid." The ground-truth for OS interaction.
Network Traffic	High	Medium	"Reveal CandC server" communication or "data bleeding" Sihwail et al. (2018). Can be encrypted, but patterns are still detectable.
Memory Dumps	High	High	Reveals "process injection" (e.g., CreateRemoteThread) and "hidden or dynamically loaded modules" that evade static analysis.
File System I/O	Medium	High	The *result* of system calls. Captures patterns of read/write operations (e.g., for entropy analysis).
Hardware Performance Counters (HPCs)	Very Low	Very High	e.g., CPU cycles, cache misses. "Semantically poor" (a cache miss doesn't mean "encryption") but "very hard for user space malware to manipulate" Demme et al. (2013).

5. MACHINE LEARNING AND DEEP LEARNING FRAMEWORKS FOR RANSOMWARE DETECTION

Modern systems generate behavioural data of enormous amounts at incredibly high speed in an intricate manner which makes it impossible to analyse ransomware fingerprints manually. Hence, ML (Machine Learning) and DL (Deep Learning) have increasingly become essential to automate the detection process. These systems learn the subtle patterns that differentiate malicious and benign behaviour from threat analysis.

5.1. ANOMALY DETECTION FOR THREATS (UNSUPERVISED LEARNING)

Here we see that unsupervised learning in particular anomaly detection is the main approach to this issue. At its base is the training of a model only on large sets of benign behavioural data which represents the at large “normal” state of a system or application. The model in this phase learns a compact representation of that normal behaviour. During deployment the model is presented with a new, unseen behavioural pattern, it flags any significant deviation from the learned norm as an anomaly, and thus a potential threat. Several unsupervised algorithms have been successfully applied to ransomware detection based on behavioural data.

- **Clustering-based Methods:** Algorithms such as Local Outlier Factor (LOF) which determine the local density of a data point in reference to that of its neighbours Omar (2022).
- **Boundary-based Methods:** A One-Class Support Vector Machine (OC-SVM) which trains a boundary (or hypersphere) in a high dimensional feature space that includes the great majority of the normal data Al-Qudah et al. (2023). Any new data point which does not fit within this trained boundary is put into the anomaly class. OC-SVM has been

successfully applied to detect anomalies in Windows registry accesses Canzanese et al. (2015).

- **Ensemble-based Methods:** Isolation Forest (IF) is an efficient algorithm with feature selection and random selection of a split value between the maximum and minimum values of that feature. The logic is that anomalous points are the "few and different," and hence easier to isolate. The number of splits required to isolate the data point is used to score its level of anomalies. IF is frequently evaluated alongside LOF and OC-SVM for system call-based anomaly detection Abiodun et al. (2022).
- **Deep Learning Based Methods:** An example of an architecture of a neural network called an autoencoder works by training it to do an identity function, that is, to perform an identity mapping to its inputs and outputs, meaning to reconstruct its inputs. If an autoencoder is trained on data which is benign behaviour, it learns to compress and decompress normal and efficient patterns. If it is trained on anomalous data coming from a ransomware attack which it has not been trained on, it will reconstruct the data poorly Maidamwar et al. (2023). The amount of this 'reconstruction error' is a powerful score for the system which scores anomalies in a system Urooj et al. (2021).
- A high amount of error means that there is a strong correlation to malicious activity.

5.2. CLASSIFICATION FOR FAMILY ATTRIBUTION (SUPERVISED LEARNING)

A large set of supervised algorithms has been applied to this task which report high accuracy.

- **Traditional Machine Learning:** A large set of robust and well-studied classifiers has been proved to do well. These include Decision Trees (DT) Sultana and Jameel (2023), Random Forest (RF), which is an ensemble of decision trees which enhances precision while mitigating overfitting and often outperforms other classifiers Gupta et al. (2022), Support Vector Machines (SVM)s which compute an optimal hyperplane to segregate classes Afzaal et al. (2023) Logistic Regression (LR); k-Nearest Neighbours (kNN); and boosting models such as XGBoost which often rank highly in classification tasks.
- **Deep Learning:** To understand the complex and sequential nature of behavioural data, deep models are becoming very popular. The DNN (Deep Neural Network) architecture is capable of learning deep and highly non-linear features in a given feature space Sekar et al. (2024). For sequential data such as API and system call traces, the Recurrent Neural Networks (RNNs) and its more sophisticated variants are used. LSTM (Long Short-Term Memory) networks are well suited for this purpose due to their capability to model temporal dependencies Brown et al. (2022), Gillard et al. (2023). For example, the control flow of a program can be modelled as a graph, and, for such data, Graph Neural Networks (GNNs) have been very successful in capturing the structural and relational features associated with malware He et al. (2025). GNNs can operate on structures like the Control Flow Graph (CFG) or Inter-Procedural Control Flow Graph (ICFG) to classify malware Herath et al. (2022), Li et al. (2025).

Both paradigms are required for a fully mature security operation. An ideal detection system, as envisioned in some frameworks, might first employ anomaly detection as a stage-one filter to quickly flag any suspicious activity and then pass this data to a more intensive supervised classifier for family attribution and threat intelligence enrichment.

5.3. CRITICAL CHALLENGES

Despite the rigorous research showcasing nearly flawless accuracy with ML inclusion, the reality is far more complicated. The dominant ML systems highlight the operational restraints posed due to excessive resource consumption towards dynamic and detailed behaviour tracking. Monitoring systems in real-time and intercepting every call in a function such as System Call, API Call, and Input Output Operation leads to high latency and significant drain of CPU and memory resources.

This excessive drain and monitoring are not an insignificant issue, as it can ultimately render a System useless. Studies show that the peak of Solid-State drives can monitor systems so that they can lose performance up to 75% of their capacity, while also increasing operation I/O executing time exponentially. One study found that complex monitoring, such as calculating data entropy for I/O calls, can increase execution time by up to 350%. The impact is so large that for most production environments, taking a naive "monitor everything all the time" approach is economically and practically impossible. This is why there is no system monitoring.

In the design of behavioural detection systems, this challenge has brought about a necessary structural evolution. The most realistic approach is the introduction of a Multi-Staged Intrusion Detection System, or MS-IDS Sloun et al. (2025). This is based on the assumption that a large majority of the activity on the system is routine and harmless, and does not require detailed analysis. An MS-IDS evaluates the risk associated with each process and adapts the monitoring level accordingly:

- **Stage 0 (Low Suspicion):** All new processes enter a low overhead monitoring which collects a minimal set of light weight behavioural features (for example counts of file create or write operations).
- **Higher Stages (High Suspicion):** As behaviour of a process in the lower stage crosses a defined threshold of suspicion it is moved to a higher stage. In the higher stages more resource intensive features are enabled which include calculation of written data entropy, deep packet inspection of network traffic, and detailed API call tracing. This in turn allows for in depth analysis to confirm or deny the initial suspicion.

Dynamic analysis renders achievable by this adaptive, multi-staged architecture, and this expertly mixes safety and efficiency. It knows that the high price of complete investigation is a resource that must be applied wisely, specializing on the small percentage of processes that act irrationally. One implementation of an MS-IDS successfully decreased the I/O latency overhead for benign processes from over 180% to under 19% Sloun et al. (2025). The opportunity of high-accuracy conduct detection is finally a reality via this breakthrough architecture, becoming greater than just an improvement.

6. UNIQUE CHALLENGES AND OPPORTUNITIES FOR RANSOMWARE DETECTION IN CLOUD ARCHITECTURES

The migration of workloads to the cloud necessitates a fundamental re-evaluation of ransomware detection strategies. While the core principles of behavioural fingerprinting remain valid, their application must be adapted to the unique architectural paradigms of the cloud, which range from infrastructure-as-a-service (IaaS) to serverless functions (FaaS). The distributed, ephemeral, and API-driven nature of cloud environments presents both significant challenges and novel opportunities for detection. Critically, the focus of monitoring must shift from traditional OS-level system calls to the cloud control plane API, which has become the primary medium for both legitimate administration and malicious activity.

6.1. TRANSLATING ENDPOINT DETECTION PRINCIPLES TO CLOUD WORKLOADS

- The effectiveness and implementation of behavioural monitoring vary significantly across the different service models in the cloud stack.
- **Virtual Machines (IaaS):** In an IaaS model, where customers manage the guest operating system, the translation of endpoint detection techniques is most direct. A monitoring agent can be deployed within the virtual machine's OS to trace system calls, file I/O, and other host-based indicators, much like on a physical server Brown et al. (2022).
- Alternatively, monitoring could occur at the hypervisor level, providing a more privileged and tamper-resistant vantage point. The behavioural fingerprints developed for on-premises systems are largely applicable here.
- **Containers:** Containerized environments, such as those managed by Kubernetes, introduce additional complexity. Containers share the host OS kernel, which means system calls can still be monitored from the host. However, the ephemeral and single-purpose nature of containers requires a different approach to baselining. A "normal" behavioural profile for a container is typically much narrower and more predictable than for a general-purpose VM. The detection challenge lies in monitoring a large number of short-lived containers and correlating their activities, often across different nodes in a cluster. Modern tools like eBPF are increasingly used to "trace runtime system calls" from monitored containers in Kubernetes Kim et al. (2025).
- **Serverless Functions (FaaS):** Serverless computing represents the most significant departure from the traditional endpoint model . With FaaS, there is no persistent, user-managed operating system to monitor Kelly et al. (2024). The cloud provider entirely abstracts the execution environment away. Consequently, detection methods based on kernel-level system call tracing are not applicable. Ransomware detection in a serverless context must instead rely on analysing the function's inputs and outputs, its execution metadata (e.g., duration, memory usage), and, most importantly, its interactions with other cloud services, which are mediated exclusively through API calls Li et al. (2025).

Table 4

Table 4 Applicability of Behavioural Monitoring Across Cloud Service Models

Service Model	Monitoring Applicability	Key Data Sources	Primary Detection Challenge
IaaS (Virtual Machines)	Direct Translation. Endpoint principles apply.	Guest OS Agent (System Calls, File I/O); Hypervisor-level tracing Brown et al. (2022).	Largely the same as on-premises (e.g., agent performance overhead).
Containers (e.g., Kubernetes)	More Complex. Shared host kernel.	Host OS monitoring (e.g., eBPF for runtime system call tracing) Kim et al. (2025), Container logs.	"Ephemeral" and "short-lived" nature of containers; high-volume monitoring; inter-container correlation.
Serverless (FaaS)	No Translation. Endpoint model is obsolete.	Cloud API Logs (e.g., CloudTrail); Execution metadata (duration, memory).	Kernel-level "system call tracing are not applicable." Detection must rely *exclusively* on API-level interactions Lin et al. (2025).

6.2. CLOUD API CALLS

For cloud native ransomware which attacks cloud resources directly we see that monitoring the cloud control plane API is at least as important if not more so than monitoring OS level events. In the cloud what we have is the convergence of “attacker’s environment” and “defender’s environment” in the control plane which is API driven. An attacker with the compromised credentials uses the same public APIs (e.g., the AWS SDK) which also a legitimate admin uses to manage resources. Which presents a unique challenge for the defenders. Malicious code in a Lambda function using legitimate AWS SDK calls to access S3 buckets, for example, can appear as "normal application behaviour" to traditional detection.

The cloud provider records each of these API calls through services such as AWS CloudTrail, Azure Monitor, and Google Cloud Audit Logs. The logging is thorough and complex, making it near impossible for a malicious actor to remove it without a trace, or at the very least a convincing, very suspicious one. AWS CloudTrail "captures a comprehensive record of all user activities" Marbel et al. (2024) and "records API events" such as CreateUser. This implies that the cloud is, by its very nature, a high-fidelity source of behavioural data. The main problem for the defenders in this case is no longer acquiring the data, but rather the signal processing—and anticipating the huge volume of data API calls that seem to be normal to detect the outliers that are an anomaly, and malicious, in real time.

The identifiable behavioural patterns are based on Ransom and how each instance is accomplished with the use of various different API routines. For instance, a so-called ransomware that is intended to unclip the files and folders that are saved on a cloud drive does behave differently. Rather than using system reads and writes, it will use a series of sequenced routines that are intended for the cloud. These routines can directly be aligned with the different stages of the lifecycle of the ransomware.

6.3. ADDRESSING THE DISTRIBUTED AND EPHEMERAL NATURE OF THE CLOUD

Detected in a distributed microservices environment we see that ransomware presents greater challenges. An attack may not be contained to a single point of entry but may play out over a series of actions in different services. For example a compromised serverless function may be used to get into a database which in turn is used to gain access to a storage bucket.

To detect these types of distributed attacks we must be able to collect, put together, and analyse logs from many sources (for example CloudTrail, VPC Flow Logs, application logs, container logs) as close to real time as possible. The true power of cloud log analysis is "unlocked when different logs are correlated to reveal a comprehensive picture of an attack". For instance, a suspicious login from an identity provider log can be correlated with "abnormal network traffic from VPC flow logs". Threat detection services like Amazon GuardDuty are built on this principle, analysing "AWS CloudTrail management events, Amazon Virtual Private Cloud (Amazon VPC) Flow Logs, and DNS logs" in concert.

Also the transient nature of cloud resources like serverless functions and containers makes it difficult to develop stable long term behavioural profiles. The baseline for “normal” shifted over time to the particular service’s functions, time of day, and the type of event that triggered the anomaly. In particular, this is why we need more sophisticated and agile detection models which will perform dynamic context-based learning quickly. Linking with cloud security native protections.

6.4. INTEGRATION WITH CLOUD-NATIVE SECURITY PLATFORMS

True value of cloud-based fingerprinting behavioural systems depends on their practical implementation context in relation to other tools in the cloud security toolset. Outputs of detector models do not serve a purpose in isolation. Actions of automated monitoring systems must incorporate them and the broader systems must enable the security teams to see the complete picture. SIEM systems can serve as the primary destination for alerts created by the foot printing and scanning behaviour detection model, which enables correlation with other security signals and the automated execution of incident response playbooks. Integrating with these platforms enables operationalized behavioural fingerprinting to functional cloud security.

7. FUTURE RESEARCH DIRECTIONS

Creating cloud tailored datasets, we also see to it that we develop which are at the same time simple in structure and at large very responsive monitoring solutions, also we work on that which is related to Explainable AI in the field of security operations, also we improve the which is related to the resistance of our behavioural models to adversarial attacks, and we do better at that which pertains to generalization across different platforms and environments.

7.1. SUMMARY

The review has meticulously documented the evolution of ransomware from a primitive threat to a developed, economically motivated criminal venture and focused on its evolving manifestation in the cloud. Ransomware is trending towards more complex and cloud focused attacks which in turn sees us shift from reactive and signature-based defences to proactive and behaviour driven strategies. We see the use of system and API level data with machine learning as a way to identify malicious patterns before an attack may succeed. Also we are seeing that in the cloud we have particular issues which include large scale data sets, monitoring issues, model explanation, and adversarial robustness which must be dealt with. To be effective we will need close work between security researches, machine learning

experts, cloud architects, industry players and cloud providers to develop adaptive and robust defence systems for the future.

8. CONCLUSION

In the ongoing and ever-changing battle against ransomware we see that as adversaries have taken their tactics from basic file lock out to very complex multi-pronged extortions which target the core of what makes modern enterprise infrastructure tick in the cloud so too must our defense strategies. This review puts forth that which way forward is through the abandonment of reactive and signature-based approaches in favour of a proactive behaviour centred approach. By use of the in-depth data which is made available via system and API calls and by which we apply the pattern recognition ability of machine learning we are able to develop very detailed behavioural profiles which in turn are used to identify new variants of ransomware before they put out their destructive work.

Threat's migration to the cloud brings forth advanced complexities that need creative data analysis, more refined system architecture, and innovative analytical models. The difficulty of cloud-specific datasets, lightweight monitoring, model explainability, and adversarial robustness will need to be tackled more aggressively in the years to come. In the end, protecting the cloud from ransomware will require a large-scale, interdisciplinary approach involving systems security, machine learning, and cloud architecture researchers, industry, and cloud providers. Only then will we be able to construct the intelligent, adaptive, and resilient defensive systems that will protect our world from ransomware in the digital epoch.

CONFLICT OF INTERESTS

None .

ACKNOWLEDGMENTS

None.

REFERENCES

Abiodun, O. I., Alawida, M., Omolara, A. E., and Alabdulatif, A. (2022). Data Provenance for Cloud Forensic Investigations, Security, Challenges, Solutions and Future Perspectives: A Survey. Journal of King Saud University – Computer and Information Sciences, 34(10), 10217–10245. https://doi.org/10.1016/j.jksuci.2022.10.018

Aboaoja, F. A., Zainal, A., Ghaleb, F. A., Al-Rimy, B. A. S., Eisa, T. A. E., and Elnour, A. A. H. (2022). Malware Detection Issues, Challenges, and Future Directions: A Survey. Applied Sciences, 12(17), 8482. https://doi.org/10.3390/app12178482

Afzaal, H., Imran, M., and Janjua, M. U. (2023). Formal Verification of Fraud-Resilience in a Crowdsourcing Consensus Protocol. Computers and Security, 131, 103290. https://doi.org/10.1016/j.cose.2023.103290

Alexander, R. (2020). Reducing Threats by Using Bayesian Networks to Prioritize and Combine Defense in Depth Security Measures. Journal of Information Security, 11(3), 121–137. https://doi.org/10.4236/jis.2020.113008

Alqahtani, A., and Sheldon, F. T. (2022). A Survey of Crypto Ransomware Attack Detection Methodologies: An Evolving Outlook. Sensors, 22(5), 1837. https://doi.org/10.3390/s22051837

Al-Qudah, M., Ashi, Z., Alnabhan, M., and Al-Haija, Q. A. (2023). Effective One-Class Classifier Model for Memory Dump Malware Detection. Journal of Sensor and Actuator Networks, 12(1), 5. https://doi.org/10.3390/jsan12010005

Alquwayzani, A., Aldossri, R., Frikha, M., and Alabdulatif, A. (2024). Prominent Security Vulnerabilities in Cloud Computing. International Journal of Advanced Computer Science and Applications, 15(2).

Alwashali, A. M. A., Rahman, N. A. A., and Ismail, N. (2021). A Survey of Ransomware as a Service (RAAS) and Methods to Mitigate the Attack. In Proceedings of the 14th International Conference on Developments in eSystems Engineering (DeSE). https://doi.org/10.1109/DeSE54285.2021.9719456

Avhankar, N. M. S. (2025). A Comprehensive Survey on Polymorphic Malware Analysis: Challenges, Techniques, and Future Directions. Communications on Applied Nonlinear Analysis, 32(9s), 2765–2776. https://doi.org/10.52783/cana.v32.4554

Botacin, M., and Grégio, A. (2022). Why we Need a Theory of Maliciousness: Hardware Performance Counters in Security. In Lecture Notes in Computer Science (381–389). https://doi.org/10.1007/978-3-031-22390-7_22

Brown, P., Brown, A., Gupta, M., and Abdelsalam, M. (2022). Online Malware Classification with System-Wide System Calls in Cloud Iaas Environments. IEEE Access, 10, 146–151. https://doi.org/10.1109/IRI54793.2022.00042

Canzanese, R., Mancoridis, S., and Kam, M. (2015). System Call-Based Detection of Malicious Processes. In 2015 IEEE International Conference on Quality, Reliability, and Security (QRS) (177–184). https://doi.org/10.1109/QRS.2015.26

Chew, C. J. W., Kumar, V., Patros, P., and Malik, R. (2024). Real-Time System Call-Based Ransomware Detection. International Journal of Information Security, 23(3), 1839–1858. https://doi.org/10.1007/s10207-024-00819-x

Chhillar, K., Tomar, D., and Verma, A. (2025). A Hybrid Static–Dynamic Malware Analysis Framework Using Interpretable Neural Network. International Journal of Scientific Research in Engineering and Management, 9(9), 1–9. https://doi.org/10.55041/ijsrem52505

Damodaran, A., Di Troia, F., Visaggio, C. A., Austin, T. H., and Stamp, M. (2017). A Comparison of Static, Dynamic, and Hybrid Analysis for Malware Detection. Journal of Computer Virology and Hacking Techniques, 13(1), 1–12. https://doi.org/10.1007/s11416-015-0261-z

Demme, J., Maycock, M., Schmitz, J., Tang, A., Waksman, A., Sethumadhavan, S., and Stolfo, S. (2013). On the Feasibility of Online Malware Detection with Performance Counters. ACM SIGARCH Computer Architecture News, 41(3), 559–570. https://doi.org/10.1145/2508148.2485970

Faheem, M., Akram, U., Khan, I., Naqeeb, S., Shahzad, A., and Ullah, A. (2017). Cloud Computing Environment and Security Challenges: A Review. International Journal of Advanced Computer Science and Applications, 8(10). https://doi.org/10.14569/IJACSA.2017.081025

Genç, Z. A., Lenzini, G., and Sgandurra, D. (2019). Analysis and Mitigation of a Novel Sandbox-Evasion Technique. In Proceedings of the 2019 Central European Cybersecurity Conference (CECC) (1–4). https://doi.org/10.1145/3360664.3360673

Ghani, W. S. D. W. A. (2022). Exploring System Quality Elements of Mobile Marketplace Application for Textile Cyberpreneurs. Procedia Computer Science, 204, 354–361. https://doi.org/10.1016/j.procs.2022.08.043

Gillard, S., David, D. P., Mermoud, A., and Maillart, T. (2023). Efficient Collective Action for Tackling Time-Critical Cybersecurity Threats. Journal of Cybersecurity, 9(1), tyad021. https://doi.org/10.1093/cybsec/tyad021

Herath, J. D., Wakodikar, P. P., Yang, P., and Yan, G. (2022). CFGExplainer: Explaining Graph Neural Network-Based Malware Classification from Control Flow Graphs. In 2022 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (401–412). https://doi.org/10.1109/DSN53405.2022.00028

Jinmei, G., Zakaria, W. N. W., Bisheng, W., and Ayub, M. A. B. (2024). DeeplabV3+ Model with CBAM and CSPM Attention Mechanism for Navel Orange Defects Segmentation. International Journal of Advanced Computer Science and Applications, 15(9). https://doi.org/10.14569/IJACSA.2024.0150919

Kelly, D., Glavin, F. G., and Barrett, E. (2024). DoWNet—Classification of Denial-of-Wallet Attacks on Serverless Application Traffic. Journal of Cybersecurity, 10(1), tyae004. https://doi.org/10.1093/cybsec/tyae004

Ki, Y., Kim, E., and Kim, H. K. (2015). A Novel Approach to Detect Malware Based on API Call Sequence Analysis. International Journal of Distributed Sensor Networks, 11(6), 659101. https://doi.org/10.1155/2015/659101

Kim, R., Ryu, J., Kim, S., Lee, S., and Kim, S. (2025). Detecting Cryptojacking Containers Using eBPF-Based Security Runtime and Machine Learning. Electronics, 14(6), 1208. https://doi.org/10.3390/electronics14061208

Ladisa, P., Plate, H., Martinez, M., and Barais, O. (2022). Taxonomy of Attacks on Open-Source Software Supply Chains. arXiv. https://doi.org/10.48550/arxiv.2204.04008

Lalejini, A., and Ofria, C. (2018). Evolving Event-Driven Programs with SignalGP. arXiv. https://doi.org/10.48550/arxiv.1804.05445

Li, C., Huang, L., He, D., Wen, Y., Liu, G., and Duan, L. (2025). FAASMT: Lightweight Serverless Framework for Intrusion Detection Using Merkle Tree and Task Inlining. arXiv. https://doi.org/10.48550/arxiv.2503.06532

Maidamwar, P. R., Lokulwar, P. P., and Kumar, K. (2023). Ensemble Learning Approach for Classification of Network Intrusion Detection in IoT Environment. International Journal of Computer Network and Information Security, 15(3), 30–36. https://doi.org/10.5815/ijcnis.2023.03.03

Marbel, R., Cohen, Y., Dubin, R., Dvir, A., and Hajaj, C. (2024). Cloudy with a Chance of Anomalies: Dynamic Graph Neural Network for Early Detection of Cloud Services' User Anomalies. arXiv. https://doi.org/10.48550/arxiv.2409.12726

Omar, M. (2022). Malware Anomaly Detection Using Local Outlier Factor Technique. In SpringerBriefs in Computer Science (37–48). https://doi.org/10.1007/978-3-031-15893-3_3

Oppliger, R. (2017). Disillusioning Alice and Bob. IEEE Security and Privacy, 15(5), 82–84. https://doi.org/10.1109/MSP.2017.3681057

Oz, H., Aris, A., Levi, A., and Uluagac, A. S. (2022). A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions. ACM Computing Surveys, 54(11s), 1–37. https://doi.org/10.1145/3514229

Poddar, A. K., and Rani, R. (2023). Hybrid Architecture Using CNN and LSTM for Image Captioning in Hindi Language. Procedia Computer Science, 218, 686–696. https://doi.org/10.1016/j.procs.2023.01.049

Sajid, M. S. I., Wei, J., and Al-Shaer, E. (2025). RANDeCepter: Real-time identification and Deterrence of Ransomware Attacks. arXiv. https://doi.org/10.48550/arxiv.2508.00293

Sayadi, H., He, Z., Makrani, H. M., and Homayoun, H. (2024). Intelligent Malware Detection Based on Hardware Performance Counters: A Comprehensive Survey. IEEE Transactions on Dependable and Secure Computing, 1–10. https://doi.org/10.1109/ISQED60706.2024.10528369

Sihwail, R., Omar, K., and Ariffin, K. A. Z. (2018). A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid and Memory Analysis. International Journal on Advanced Science, Engineering and Information Technology, 8(4–2), 1662–1671. https://doi.org/10.18517/ijaseit.8.4-2.6827

Sun, Z., Rao, Z., Chen, J., Xu, R., He, D., Yang, H., and Liu, J. (2019). An Opcode Sequences Analysis Method for Unknown Malware Detection. IEEE Access. https://doi.org/10.1145/3318236.3318255

Sy, C. Y., Maceda, L. L., Canon, M. J. P., and Flores, N. M. (2024). Beyond BERT: Exploring the Efficacy of RoBERTa and ALBERT in Supervised Multiclass Text Classification. International Journal of Advanced Computer Science and Applications, 15(3). https://doi.org/10.14569/IJACSA.2024.0150323

Urooj, U., Al-Rimy, B. A. S., Zainal, A., Ghaleb, F. A., and Rassam, M. A. (2021). Ransomware Detection Using Dynamic Analysis and Machine Learning: A Survey and Research Directions. Applied Sciences, 12(1), 172. https://doi.org/10.3390/app12010172

Yadav, P., Feraudo, A., Arief, B., Shahandashti, S. F., and Vassilakis, V. G. (2020). A Systematic Framework for Categorising IoT Device Fingerprinting Mechanisms. In Proceedings of the 4th ACM Workshop on IoT Security and Privacy (62–68). https://doi.org/10.1145/3417313.3429384

Zakaria, W. Z. A., Abdollah, M. F., Mohd, O., Yassin, S. M. W. M. S. M. M., and Ariffin, A. (2022). RENTAKA: A Novel Machine Learning framework for Crypto-Ransomware Pre-Encryption Detection. International Journal of Advanced Computer Science and Applications, 13(5). https://doi.org/10.14569/IJACSA.2022.0130545

Zhang, M., Duan, Y., Yin, H., and Zhao, Z. (2014). Semantics-Aware Android Malware Classification Using Weighted Contextual API Dependency Graphs. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS) (1105–1116). https://doi.org/10.1145/2660267.2660359