ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 5, May 2024

A Comprehensive Survey of Malware Detection Approaches in Cloud Computing Environments

[1] Sheethal Mariya Binoy, [2] Shafc Sulthana, [3] Nandagovind P, [4] Ms. Jomina John
[1] [2] [3] [4] Department of Computer Science and Engineering, Rajagiri School of Engineering and Technology, Kochi, Kerala, India
[4] Assistant Professor, Department of Computer Science and Engineering, Rajagiri School of Engineering and Technology, Kochi, Kochi, Kerala, India
Corresponding Author Email: [1] msheethal7@gmail.com, [2] shafcsulthana@gmail.com, [3] nandagovind.praveen@gmail.com, [4] jominacj@gmail.com

Abstract*— Malicious software or malware is on the rise, with an increasing number of sophisticated variants employing various obfuscation techniques. Detecting malware before it wreaks havoc on computer systems and the Internet is imperative. This paper offers a comprehensive survey of existing malware detection approaches, shedding light on the persistent challenges in this domain. This paper presents a literature survey that delves into cloud-based malware detection methods or models. This thorough analysis looks at four different methods for detecting malware, each designed for a particular environment and taking care of a different set of problems. The Proposed Malware Detection Model (PMDM) and Cloud Deployment Model (CDM) make up the first model, which suggests a malware detection system for cloud deployment. PMDM uses behavioral, symbolic, and DNA sequence detection processes to improve system flexibility and speed. To detect malware in a scalable and easily accessible manner, CDM uses Eucalyptus in an actual cloud setting. The second method uses a Cloud-Based Behavior Centric Model, dynamic analysis tools, and several machine learning algorithms to present an intelligent behavior-based malware detection system for cloud environments. The third technique, called TrustAV, maximizes malware scanning efficiency by utilizing a multimodal strategy inside Intel SGX enclaves. The fourth framework uses the computational power of security labs to simulate end- user environments and presents a cloud-based malware analysis system based on dynamic behavior. A thorough examination of system call interception and proxying, one-way isolation, and integration with currently available malware detectors are included in the paper's conclusion. This review sheds light on various approaches, their advantages, disadvantages, and possible areas for development in the dynamic field of intelligent malware detection.

***Index Terms*—malware detection, cloud computing.**

I. INTRODUCTION

The democratization of cloud computing technology has transformed the way data and applications are collected, processed, and made available to a vast range of users at unprecedented convenience, capacity, and cost efficiency. However, the paradigm shift also opens up new avenues for malicious actors to exploit vulnerabilities in cloud environments. Malware, which has become an ongoing danger to the security of information, is adapting and reconfiguring how it infiltrates and compromises cloud computing systems that could result in significant damage or data breaches. This has become a major and pressing concern with regard to developing effective detection mechanisms for malicious software that are suited to the cloud computing environment.

Malware mainly targeted networks and individual devices in the early days of cloud computing adoption. Cybercriminals realized there was a chance to breach these shared computing environments as companies moved their infrastructure to the cloud. Cloud services were first used by malware developers to host malicious payloads, which allowed them to get around conventional security measures and carry out attacks more quickly. The incorporation of fileless and polymorphic mal- ware into cloud-based assaults is one noteworthy development. The code of polymorphic malware is constantly changing, making it difficult for conventional signature-based detection techniques to stay upto date. Polymorphic malware can quickly adapt to various instances in the cloud, where dynamic scaling is frequently used, making detection more difficult and raising the risk of extensive harm. [9]Another notable change is the use of fileless malware in cloud environments. Fileless malware frequently uses reputable system tools and processes as launchpads for its malicious operations since it runs in memory and leaves little to no trace on disk. Fileless malware can operate without leaving traditional traces in the cloud, where serverless architectures and ephemeral instances are common. This makes it evasive to traditional security measures. [10]Furthermore, a growing emphasis on taking advantage of weaknesses in cloud infrastructure elements like orchestration platforms, containers, and serverless functions characterizes the evolution of malware in the cloud. Malicious actors can launch attacks, obtain unauthorized access, or jeopardize the integrity of cloud-based applications by taking advantage of errors or flaws in these components. [17] The threat landscape in this dynamic environment is changing in tandem with the increasing popularity of cloud computing. Malicious actors use cloud resources to spread sophisticated malware, constantly changing their strategies.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 5, May 2024

Comprehending this progression is essential to formulating efficacious countermeasures that surpass conventional security protocols. The following sections thoroughly examine suggested models, behavior-centric frameworks, enclave-based solutions, and cloud-integrated analysis, highlighting their advantages and possible uses in the continuous fight against malware threats that are constantly changing. [8]

II. RELATED WORKS

A malware detection technique for cloud computing based on convolutional neural networks (CNNs) was presented by Abdelselam et al. [1] Using VM process data obtained from the hypervisor, they trained a conventional 2-D CNN, and then they used a 3-D CNN to increase accuracy even further. The 2-D CNN model had an accuracy rate of 79% in the experimental findings utilizing different malware on virtual machines (VMs); this significantly climbed to 90% with the 3-D model. The research makes recommendations for possible enhancements by increasing the experiment's scope to include a larger variety of malware binaries. The malware detection game that uses cloud computing, in which mobile devices submit application traces to security servers via access points or base stations in dynamic networks was examined by Xiao et al. [18] The malware detection system Q-learned was created with a mobile device in mind. The objective of this research is to attain the best payload transfer ratio while remaining unaware of the radio bandwidth model and trace creation of various mobile devices. They accelerated the reinforcement learning stage with a post-decision learning technique and improved performance with the Dyna architecture. A cloud-assisted model was proposed by Zhou and Yu [19] for the dynamic differential game against malware trans- mission and detection. In the proposed methodology, data is shared on the cloud security platform to develop a malware detection model based on SVM. Second, based on the features of the wireless multimedia system (WMS), the number of malware-infected nodes that physically infect vulnerable nodes is computed. Ultimately, the modified epidemic model describes the states transitioning between WMS devices, and the Hamilton function has been introduced to streamline the saddle point solution. Additionally, a target cost function and dynamic differential game for the malware-WMS system Nash equilibrium have been successively derived. [7] SplitScreen is a brand-new malware detection system that Cha et al. proposed [3].

Before the signature matching stage, this distributed malware detection system performs an additional screening step. Client-server processes make up SplitScreen's two-stage screening procedure. Using half of the memory, the proposed method was implemented as an extension of ClamAV, increasing the scanning throughput more than two times the signature set. As the authors pointed out, SplitScreen performs better in terms of memory savings and acceleration as the number of signatures rises. The suggested approach works with a variety of low-end consumer and handheld devices. Since the cloud side only uses one server, it would be preferable to maximize server efficiency and place some workload on the client. A machine learning-based detection method for the cloud environment was proposed by Indirapriyadarshini et al. [6] After obtaining the worst log loss through random modeling, they employed various modeling techniques like KNN and LR. They next assessed each algorithm's log loss to see if it was a perfect model. The ML model and user interface were then finally deployed on the AWS cloud. To ascertain the legitimacy of the file, the authors claimed to have discovered a novel approach by utilizing cloud computing and machine learning in tandem. Nevertheless, this research can be improved by using new learning models or alternative data mining approaches for feature selection. Using a sizable dataset, Mirza et al. [13] suggested combining several machine learning techniques. The two primary objectives of the paper were low resource consumption and higher DR. To achieve the maximum detection rate, they took a subset of features—both malicious and normal files—and extracted them from the dataset. Then, they applied boosting, SVM, and decision trees to the decision tree.

In the CloudIntell assessment, the decision tree classifier's boost resulted in improved performance. Additionally, they unveiled a cloud- based architecture that is scalable and runs on Amazon Web Services (AWS). They used various scenarios to test the suggested methodology. The authors claim that their methodology yielded excellent results while using the least amount of energy. In an additional study, Mirza et al. [12] proposed an energy-efficient hosting model that enhances a distinct and scalable model by combining various components of Amazon cloud services. This study looked at known antivirus programs and benchmarking data for cloud-based hosting. As per the paper, the suggested method not only worked well for the hosted detection framework but also outperformed conventional antivirus software in terms of optimal performance. However, by including the intrusion detection mechanism to be supported by the cloud-based engine, the malware detection framework and hosting model can be further enhanced. [15]

III. MALWARE DETECTION APPROACHES

Malware, for "malicious software" is a broad class of destructive applications that are purposefully designed to undermine the safety and operation of computer systems, networks, and user information. A wide variety of malicious programs can be used, including worms, Ransomware, trojan horses, spyware and more that seek to destroy or gain unauthorized access. The fact that malware is present on cloud platforms demonstrates the complexity of the environment, with numerous different ways in which the propagation and infiltration can take place.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 5, May 2024

Malware has become a frequent entry point to clouds through infected files and documents. The user can share files containing malicious software, on purpose or unintentionally, causing the virus to spread in a common cloud environment. Moreover, malware has found access points to the cloud's applications, services and infrastructure. The hostile actors are given an opportunity to penetrate and move around the cloud ecosystem by exploiting these vulnerabilities, which may include anything from software bugs to wrong configurations. Phishing also plays a critical part in the spread of cyber threats on cloud platforms. Exploit users to disclose their personal data or logins, attackers can gain unauthorized access and in some cases may launch a malicious program. Another vector is insecure API programming interfaces since hackers can use them to compromise cloud environments by exploiting these poorly secured interfaces. The vulnerability to unauthorized access and the possible dissemination of dangerous con tent is heightened by weak passwords or security weaknesses that compromise user accounts.

The threats landscape also includes situations in which users get inadvertently infected by driver downloads, when they search for a malicious website or try to engage with undesirable content that triggers automatic download and execution of malware. The risk of the introduction of malware via intentional actions or unintentional usage of compromised devices by employees or contractors with access to cloud platforms is high. Insider threats can be both intentional and inadvertent.

A full strategy is needed to mitigate the risk of malicious attacks in cloud environments. Organizations need to implement strong security arrangements, such as regular audits aimed at identifying and repairing vulnerabilities, training their staff to use new antivirus software or using it most recently. Strict security protocols and encryption methods should be implemented in order to avoid any undesired access or data breaches. For rapid identification and effective containment of possible malware threats, it is essential to have a clearly de- fined Incident Response Plan and Ongoing Cloud Environment Monitoring. In order to keep pace with the constantly evolving landscape of malware, due to its dynamic and interdependant nature, cloud computing calls for an active and continuous security approach.

A. Intelligent behavior based detection

Model architecture, dataset, features, and detection methods are all part of the comprehensive approach presented by the suggested intelligent behavior-based malware detection system, which is tailored for cloud environments. [2] Using a computer network, a user uploads a questionable file to the cloud using this system. The file that has been submitted is executed in multiple Virtual Machines (VMs), and dynamic tools are utilized to collect the execution traces. These traces are then fed into a behavior-based detection agent, which creates features by grouping and generating behaviors according to predefined rules using the Cloud-Based Behavior Centric Model (CBCM).

For efficient feature generation and selection, the subtractive center behavior model is modified and used as the CBCM model. This model uses system calls, paths, resource types, and file types to identify patterns of malicious behavior. The objective is to extract the key characteristics that help differentiate benign samples from malicious ones. By establishing connections between system calls, the behavior creation algorithm generates meaningful activity-based behaviors.

The suggested approach also tackles the difficulties associated with automatically creating datasets, pointing out flaws in the state-of-the-art models like n-gram and offering CBCM as a remedy. By producing fewer, more pertinent features, the system hopes to shorten detection times and increase Detection Rate (DR).

Process Explorer, API Monitor, and Process Monitor are examples of dynamic analysis tools used in the malware analysis process. The CBCM model is used to create behaviors and features in the behavior-detection agent based on the collection of execution traces from various virtual machines. The suggested model is more effective overall because behavior generation, feature extraction, and feature selection are integrated.

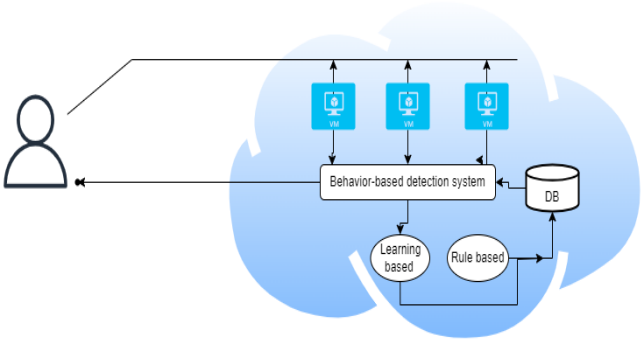

Figure 1. Architecture Diagram

And features in the behavior-detection agent based on the collection of execution traces from various virtual machines. The suggested model is more effective overall because behavior generation, feature extraction, and feature selection are integrated.

A variety of machine learning algorithms, such as logistic model trees (LMT), C4.5 (J48), random forest (RF), simple logistic regression (SLR), sequential minimal optimization (SMO), and k-nearest neighbor (KNN), are used to train specific features in the learning-based detection agent. A dataset is created using the frequencies of the features, and classifiers are then used to identify each sample as benign or malicious. During the training stage, holdout and cross-validation techniques are used, and decision trees like C4.5, LMT, and RF work well with the feature distribution of the dataset.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 5, May 2024

The rule-based detection agent uses a predefined property list that is derived from malware behaviors, so it doesn't require a training phase to function. This list, which is constantly updated with new malware features, groups features according to how frequently they occur. The analyzed program is classified as malicious if its features match those in the list; if not, it is classified as benign. Compared to the learning- based detection agent, the rule-based detection agent functions more quickly and is more effective at identifying different types of malware.

The architecture of the system is given in Figure 1. The suggested system uses the behavior-based detection agent to compare and reconcile classification discrepancies by combining the output from the two detection agents. In order to address the need for automated analysis and feature extraction, the overall methodology offers a comprehensive and flexible approach to intelligent behavior-based malware detection in cloud environments.

B. Pattern based detection

The Proposed Malware Detection Model (PMDM) and the Cloud Deployment Model (CDM) are the two primary components of the proposed malware detection system, which is intended for cloud deployment. [14] The PMDM seeks to resolve issues with antivirus software, improve system speed, and provide options for a more effective and adaptable work environment. In the PMDM, three distinct processes are employed for malware detection:

1) *DNA Sequence Detection Process:* In this first phase, DNA sequences are extracted from files and converted to binary format. A reversible conversion table is utilized to convert corresponding bits into characters found in the DNA sequence. Following their conversion from binary files to FASTA sequences, the files are combined into a single FASTA sequence file called Malware Sequence Database. (Fig 2) Using the Blast online tool, similarities between the Malware Sequence Database and the FASTA sequence file are analyzed. The file is deemed non-malicious if the BLAST report's identity percentage is less than 80%. In the event that it is between 80% and 90%, the second procedure proceeds with additional detection. A number higher than 90% suggests that the second process has blocked a harmful file.
2) *Symbolic Detection Process:* Files that make it through the first step are grouped according to their file format. In symbolic detection, files are transformed into symbol files and then compared to an existing database table of symbol signatures that include traditional malware signatures. The file may be harmful if the symbols do not match; if they do, it is deemed malicious and is blocked for the third process.
3) *Behavioral Detection Process:* Files that make it past the second step are subjected to behavioral malware detection through the use of a virtual machine, namely the Anubis sandbox. Using API calls, Anubis interacts with the file and observes its activity to detect the existence of malware. Whether or whether the file is harmful is determined by the results.

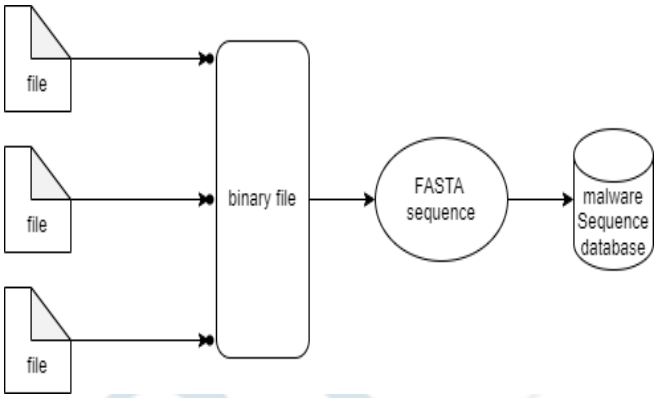

Figure 2. Creating BLAST database

Using the Eucalyptus open-source software, the CDM entails installing the PMDM into a cloud architecture. Implementing PMDM in a real cloud environment and assessing its resilience to known and unknown harmful assaults is the goal of CDM. The Eucalyptus architecture only partially implements PMDM due to architectural constraints. By using the previously covered procedures, the system seeks to identify malicious code. It then issues alerts and prevents malicious files from accessing Guest VMs and Guest Operating Systems. By utilizing the advantages of cloud computing for efficient malware detection, the deployment into the cloud improves scalability and accessibility.

C. Cloud-powered framework for security based assessment

In this paper, a novel cloud-based malware analysis frame- work based on dynamic behavior is presented. The framework attempts to combine end-users' varied and realistic environments with the copious computational resources of cloud- based security labs. The fundamental tenets are as follows: first, security labs have infinite computational capacity and can grow their capabilities by adding new hardware features and advancing research. Second, end-user environments are thought to be more appropriate for analyzing potentially malicious software due to their authenticity and heterogeneity when compared to typical security lab setups.

With the help of the suggested framework, [11] end users can assign potentially malicious program execution and analysis to a security lab, which will simulate the program's behavior as if it were running in the end user's environment. This strategy has two benefits. It gives the security lab the ability to watch a potentially malicious program run in an environment similar to that of an end user, and it gives end users the ability to increase their level of protection by using the security lab's computational resources for in-depth analysis that might not be possible in other circumstances.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 5, May 2024

The framework increases analysis completeness by observing how a program behaves in multiple realistic end-user environments within the cloud, since each end-user's environment is different and a program's behavior heavily depends on the execution environment. A developed mechanism that forwards and executes a portion of the system calls invoked by the analyzed program to a remote end-user's environment and retrieves the computation results is what allows the execution to take place in the cloud. By following this procedure, the analyzed program operating in the security lab is guaranteed to replicate the behavior exactly as if it were run in the user's environment.

1) Executing a program in multiple environments:

a) System calls hooking:: In order to intercept system calls in the program under analysis, the paper uses system calls hooking. It injects a DLL into the suspended process' virtual address space by means of a user-space hooking technique. By hooking into Microsoft Windows' KiIntSystemCall and KiFastSystemCall functions, the DLL makes development and integration with pre-existing malware detection systems easier. By taking this approach, the framework can monitor and intercept system calls with greater effectiveness, which helps with dynamic behavior-based malware analysis.

b) System calls proxying:: In order to control how re-sources are handled within the examined program, the paper presents a technique called system calls proxying. It is predicated on the idea that system calls are necessary to obtain handles for manipulation because user-space applications do not have direct access to system resource data structures. Based on the call's nature and potential security risks, this method- ology seeks to establish whether a given system call should be performed in a remote or local environment.

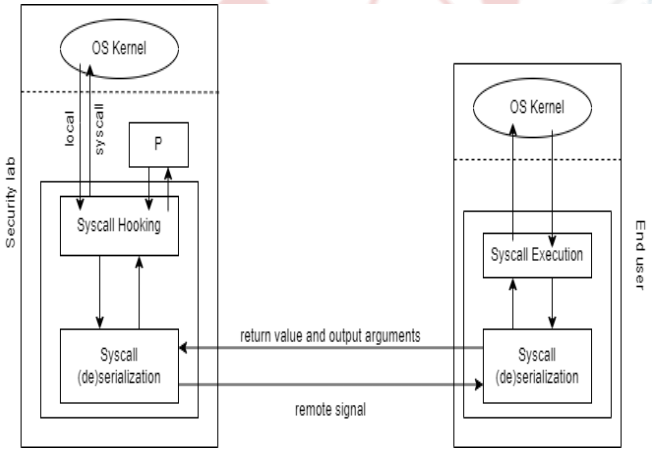

Figure 3. System calls interception and remote execution

Whether a system call creates or uses a handle to manipulate resources determines whether the execution is local or remote. When a call generates a handle—like when a file is opened—the resource being accessed is examined. Should tampering with this resource possibly result in malevolent actions, the call is classified as remote and is carried out in an isolated setting. On the other hand, if the resource manipulation is considered harmless, the call is carried out locally. In addition, the system determines whether a resource is local or remote before intercepting a system call that uses a handle to manipulate it. This assessment is used to determine the appropriate environment in which to execute the call in order to handle any potential security risks related to resource manipulation. A dynamic analysis of the program's behavior is made possible by this methodical system calls proxying technique, which enables the framework to distinguish between local and remote resource manipulations based on any potential security implications. Every system call that the analyzed program (P) makes is intercepted in the suggested framework. While remote system calls are routed to the end-user's system, local system calls are sent straight to the kernel. The local system (L) serializes the system call arguments and sends them to the user environment (U) so that the remote system call can be executed there. After receiving the arguments, U deserializes them, sets up the registers and stack in order to prepare the program for execution, and then executes the system call. U serializes the output arguments and returns them to L after the system call has finished. L then deserializes the output arguments to the anticipated locations in the program's memory, allowing normal operation to resume. It's important to note that P, the analyzed program, is still unaware that some system calls are being run remotely. This is accomplished by giving the program the expected output in memory, which guarantees smooth operation whether the system call was made locally or remotely.

c) Choosing remote system calls:: The system calls that are marked as remote are chosen by the framework using a whitelist. This whitelist consists of a list of system call names and a collection of requirements determined by the callers' arguments. System calls that are considered remote include NtOpenKey, NtCreateKey (in case the arguments indicate that the key is being opened for reading), NtOpenFile, NtCreateFile (in case the arguments indicate that the file is being opened for reading), NtQuerySystemInformation, and NtQueryPer- formanceCounter. The handles returned by these calls are specifically marked as remote by setting the most significant bits (unused bits) in order to make it easier to identify remote system calls. This method guarantees that any further system calls that make reference to these handles will be identified as gaining access to remote resources. Furthermore, this approach ensures that handles referring to local and remote resources do not potentially overlap, offering a reliable way to differentiate between the two kinds of resource access.

d) GUI system calls:: User inputs and graphical user interface (GUI) resources are critical trigger conditions within the framework. In order to facilitate realistic user interaction with the analyzed program, especially with GUI events, the framework makes use of the Windows Terminal Services subsystem. This subsystem makes it easier for the

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 5, May 2024

monitored application's graphical user interface to be automatically forwarded from the security lab to the user's computer. In particular, the prototype makes use of smooth Remote Desktop Protocol (RDP), which makes it possible to export only an application's graphical user interface to a remote host as opposed to the full desktop session. By using this method, the user's computer will display a phony login form without any difficulty if the lab's analyzed program prompts one and requests input from the user. Input events from the user, like keystrokes and mouse clicks, are then relayed back to the lab program. It's crucial to remember that even though this RDP- based solution allows GUI forwarding to a remote system, the session in which the application runs is still owned by the security lab. As a result, inquiries concerning the execution environment would provide data from the laboratory. System calls related to API functions such as GetWindowText and GetForegroundWindow for example, would return windows from the lab session. These functions are used by a sample malware to detect whether the victim is visiting the website of a Brazilian bank. In order to remedy this, these system calls are run remotely in the same way as other remote system calls, making sure that the information they return is about the windows in the real remote environment and not the lab.

e) One-way isolation:: The framework's main goal is to protect the end-user's system from any potential harm that the examined program might cause while still permitting the program to run without interruption. The framework uses a tactic called one-way isolation to accomplish this. This method [11] ensures that changes are made locally by allowing "read" accesses to remote system resources but limiting "write" accesses. More specifically, the framework treats the resource as local and does not proxy the call if the analyzed program starts a system call to create or modify a resource that is normally considered remote. For the purpose of keeping the program state consistent, subsequent system calls involving such a resource are additionally performed locally. By blocking potentially damaging changes, this one-way isolation technique safeguards the end-user's system while enabling the examined program to access data from distant sources. Nevertheless, the framework's current prototype does not allow system modifications made in the lab to be committed to the end user's environment. Furthermore, the framework does not support the proper isolation of a program that uses a resource that is being used concurrently by another program. These features point out areas that could be improved in upcoming framework revisions.

2) In the cloud behavior-based malware detector:

The framework is integrated into an existing detector to demonstrate how it works in unison with behavior-based malware detectors. Based on virtual machine introspection, this specific detector can detect data-flow dependencies between system call arguments and provides fine-grained information flow tracking. A specially designed system emulator that supports system call interception and taint analysis with multiple taint labels is the foundation of the malware detector. It took only a small adjustment to incorporate our framework into this already-existing detector so that it could differentiate between system calls made by the suspicious program and those that our prototype for proxying system calls handled, thereby enabling it to ignore the latter. In order to keep track of a suspicious program's execution in multiple end-user environments, we just run multiple instances of the improved malware detector, each of which works with a separate end-user's machine, and then combine the findings. Interestingly, the problem of correlating the outcomes from various analyses has not been tackled up to this point. This integration demonstrates how our framework can coexist peacefully with cutting-edge malware detection methods, improving the system's overall ability to identify and analyze malicious activity in a variety of settings.

D. TrustAV Model

Within the Intel SGX enclaves, TrustAV is being implemented using a multimodal approach to malware scanning. With the help of the well-known Aho-Corasick pattern matching algorithm, which is frequently used in signature-based programs such as ClamAV, TrustAV makes sure that the malware scanning procedure takes place entirely inside of SGX enclaves. The major objectives of this approach are to safeguard executed code, protect user privacy, and uphold the integrity of the signature set. [4] One of TrustAV's unique features is the serialization of the Aho-Corasick DFA into a single-dimensional integer array. This was a crucial decision made to overcome memory constraints and potential performance deterioration from scattered node traversal during pattern matching, which are issues with SGX enclaves.A key element of TrustAV's implementation is the SGX enclave I/O system, which emphasizes safe and effective communication with the enclaves. Since SGX enclaves do not have direct access to system calls, the TrustAV server's non- SGX enabled segment is in charge of managing the network sockets needed to receive client data. To reduce performance overhead, encrypted client data is batched, and a buffer is used for workload-based dynamic optimization, enabling effective data transfer into the enclave. In order to prevent compromised servers from accessing plaintext data or secret keys kept in main memory or the file system, user data is encrypted inside the enclave.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 5, May 2024

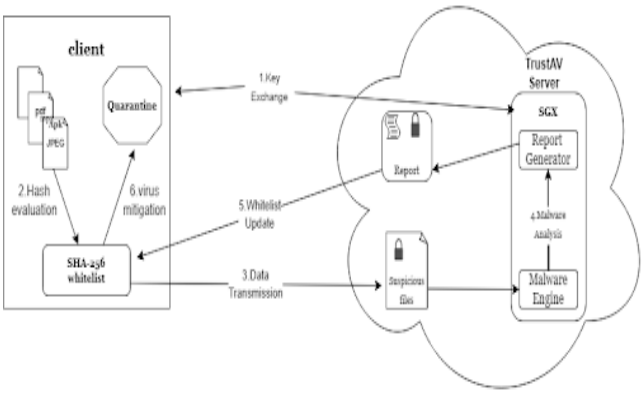

Figure 4. TrustAV design overview

The architecture of TrustAV (Fig 4) places a high priority on performance optimizations, especially when it comes to resolving the memory footprint issues with signature-based solutions. The CPU's constrained cache size is a major obstacle, particularly when automaton size surpasses this limit. In order to reduce access to pages kept outside of the Enclave Page Cache (EPC) and to safeguard big automata on Windows- based platforms, TrustAV implements unique caching techniques. To increase efficiency and minimize memory footprint, the solution incorporates an LRU cache and a configurable-size cache with a maximum capacity of 90MB. To protect cached automaton states, encryption is used by both caching methods. TrustAV expands its use to client apps, supporting desktop and mobile operating systems. By enabling remote attestation, the desktop client improves the security of the remote server connection. Concurrently, the mobile client offers customers detailed configuration choices for scanning intervals, enabling optimization of battery life and network traffic. In addition to implementing secure persistent storage of the white-list, which is exported to the file system in an encrypted format with corresponding checksums for integrity verification, both clients register for services with the server. Because of its modular architecture, TrustAV makes it easier to construct clients for different platforms and guarantees that they will work with a wide range of devices and operating systems.

TrustAV places a high priority on protecting user privacy by using hardware-assisted enclaves — Intel SGX, in particular, to guarantee the security and privacy of user data even in untrusted environments. This method protects user data from unauthorized access and from being exposed to malicious entities by encapsulating both its movement and processing. The fundamental tactic is to only analyze malware inside of secure areas in order to avoid compromising private data. Furthermore, TrustAV uses secure offloading, which enables malware analysis to be carried out on a distant server inside hardware-enabled secure enclaves. This preserves productivity while offering a high degree of security for managing personal user information in possibly untrusted environments.

IV. COMPARISON

In comparison to current state-of-the-art malware detection methods, the proposed approaches in the paper present notable advancements. The integration of the Proposed Malware Detection Model (PMDM) and Cloud Deployment Model (CDM) introduces a versatile solution that effectively balances adaptability and scalability. PMDM's incorporation of behavioral, symbolic, and DNA sequence detection processes enhances system flexibility and speed, addressing challenges posed by the dynamic nature of evolving malware. Additionally, the CDM's utilization of Eucalyptus in cloud settings ensures scalability and accessibility, overcoming issues associated with deploying detection systems in diverse cloud environments. [16] This contrasts with some existing methods that may lack agility or prove resource-intensive in dynamic cloud settings. The Cloud-Based Behavior Centric Model represents a departure from conventional detection methods prevalent in the current state of the art. [5] By integrating dynamic analysis tools and machine learning algorithms, it aligns with the industry's shift towards more sophisticated and adaptive detection mechanisms, providing a forward-looking perspective. TrustAV further augments the proposed models by incorporating hardware-level security measures, a departure from prevalent software-centric strategies.

Table 1. Comparison of existing methods

Paper	Method	Advantage	Disadvantage
Ref 8	Intelligent behavior based malware detection in cloud	Utilizes dynamic analysis tools for comprehensive malware analysis	Rule-based agent lacks adaptability to emerging threats
Ref 9	Pattern based malware detection	Multifaceted Detection Approach by integrating DNA sequence, symbolic, and behavioral detection processes	CDM faces Limitations in fully implementing the PMDM due to architectural constraints within the Eucalyptus framework
Ref 10	Framework based on dynamic behavior 2	Enhanced security through undetectable systems thwarts evasion attacks	Privacy risks and incomplete implementation compromise system integrity
Ref 11	Cloud based malware detection model called TrustAV	Protects the transmission and processing of user data in distrusted networks	Intel SGX Size limits hinder malware scanning engine sophistication.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)

Vol 11, Issue 5, May 2024

Utilizing Intel SGX enclaves, TrustAV maximizes malware scanning efficiency, addressing a critical gap in the state of the art and fortifying the detection process with advanced hardware features for a more resilient defense against sophisticated threats. Overall, these proposed approaches contribute to the ongoing evolution of intelligent malware detection, offering comprehensive and effective solutions against the increasing sophistication of malicious software. The common objective of all the four papers is to detect the presence of malware in the cloud platform. Although the papers widely differ in their methodologies, they have proved to achieve their objective. Various advantages and disadvantages of the four papers are given in the Table 1.

V. CONCLUSION

We have examined four different methods for detecting malware in this thorough literature review, each of which is designed to take into account the changing threat landscape in various computing environments. By utilizing DNA sequence detection, symbolic detection, and behavioral detection processes, the suggested malware detection system for cloud deployment demonstrates a comprehensive defense strategy. An efficient solution for cloud environments is provided by the intelligent behavior-based malware detection system, which emphasizes the integration of virtual machines, ma- chine learning algorithms, and dynamic tools. Using Intel SGX enclaves for malware scanning and placing a strong emphasis on performance optimization in the face of memory limitations, TrustAV presents a novel approach. Furthermore, to improve analysis scalability and completeness, the cloud- based malware analysis framework innovates by fusing cloud- based security labs with end users' actual environments.

In the future, the focus of malware detection research will be on improving these strategies to stay up with the constantly changing threat landscape. Malware detection systems need to change to keep up with the rapidly evolving cloud computing landscape, which includes new technologies like edge computing and the Internet of Things. The incorporation of artificial intelligence, specifically through advanced machine learning and deep learning models, has the potential to improve malware detection accuracy and efficiency. Furthermore, investigating cooperative strategies that make use of threat intelligence exchanges between institutions can fortify the group's defenses against malware strains that are evolving quickly and becoming increasingly sophisticated. As the field develops, addressing privacy issues, ethical issues, and guaranteeing the smooth integration of these detection systems into various cloud architectures should also be top priorities. Future malware detection systems will be more resilient, adaptable, and privacy-aware thanks to the synthesis of these developments and considerations

REFERENCES

[1] Mahmoud Abdelsalam, Ram Krishnan, Yufei Huang, and Ravi Sandhu. Malware detection in cloud infrastructures using convolutional neural networks. In *2018 IEEE 11th international conference on cloud computing (CLOUD)*, pages 162–169. IEEE, 2018.

[2] Omer Aslan, Merve Ozkan-Okay, and Deepti Gupta. Intelligent behavior-based malware detection system on cloud computing environment. *IEEE Access*, 9:83252–83271, 2021.

[3] Sang Kil Cha, Iulian Moraru, Jiyong Jang, John Truelove, David Brumley, and David G Andersen. Splitscreen: Enabling efficient, distributed malware detection. *Journal of Communications and Networks*, 13(2):187–200, 2011.

[4] Dimitris Deyannis, Eva Papadogiannaki, Giorgos Kalivianakis, Giorgos Vasiliadis, and Sotiris Ioannidis. Trustav: Practical and privacy preserving malware analysis in the cloud. In *Proceedings of the tenth ACM conference on data and application security and privacy*, pages 39–48, 2020.

[5] Himanshu Gupta, Akashdeep Bhardwaj, et al. Securing the cloud: An in-depth exploration of conceptual models, emerging trends, and forward-looking insights. 2023.

[6] P Indirapriyadarsini, Mohammed Uzair Mohiuddin, Mohammed Taqueeuddin, Ch Srikanth Reddy, and T Koushik. Malware detection using machine learning and cloud computing. *Int. J. Res. Appl. Sci. Eng. Technol.*, 8(6):101–104, 2020.

[7] R Jayanthi and K John Singh. A public key-based encryption and signature verification model for secured image transmission in network. *International Journal of Internet Technology and Secured Transactions*, 9(3):299–312, 2019.

[8] Jomina John and Jasmine Norman. Analysis of worm attack detection methods in cloud find a better solution for malware in cloud. *International Journal of Applied Engineering Research*, 10(92):2015.

[9] Jomina John and Jasmine Norman. Major vulnerabilities and their prevention methods in cloud computing. In *Advances in Big Data and Cloud Computing: Proceedings of ICBDCC18*, pages 11–26. Springer, 2019.

[10] Osama Khalid, Subhan Ullah, Tahir Ahmad, Saqib Saeed, Dina A Alabbad, Mudassar Aslam, Attaullah Buriro, and Rizwan Ahmad. An insight into the machine-learning-based fileless malware detection. *Sensors*, 23(2):612, 2023.

[11] Lorenzo Martignoni, Roberto Paleari, and Danilo Bruschi. A framework for behavior-based malware analysis in the cloud. In *Information Systems Security: 5th International Conference, ICISS 2009 Kolkata, India, December 14-18, 2009 Proceedings 5*, pages 178–192. Springer, 2009.

[12] Qublai K Ali Mirza, Irfan Awan, and Muhammad Younas. A cloud-based energy efficient hosting model for malware detection framework. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2018.

[13] Qublai K Ali Mirza, Irfan Awan, and Muhammad Younas. Cloudintell: An intelligent malware detection system. *Future Generation Computer Systems*, 86:1042–1053, 2018.

[14] Sagar Shaw, Manish Kumar Gupta, and Sanjay Chakraborty. Cloud based malware detection technique. In *Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications: FICTA 2016, Volume 1*, pages 485–495. Springer, 2017.

[15] Vaishali Ravindra Thakare and John Singh K. Computational trust evaluation algorithm for cloud models using fuzzy logic

approach. *International Journal of Ad Hoc and Ubiquitous Computing*, 38(1-3):127–140, 2021.

[16] KARIM USMAN. *DEVELOPMENT OF A MODEL FOR USER-CENTRIC CYBER DISASTER RECOVERY*. PhD thesis, FACULTY OF PHYSICAL SCIENCES, NNAMDI AZIKIWE UNIVERSITY, AWKA, 2019.

[17] Devi Priya VS, Sibi Chakkaravarthy Sethuraman, and Muhammad Khurram Khan. Container security: Precaution levels, mitigation strategies, and research perspectives. *Computers & Security*, page 103490, 2023.

[18] Liang Xiao, Yanda Li, Xueli Huang, and XiaoJiang Du. Cloud-based malware detection game for mobile devices with offloading. *IEEE Transactions on Mobile Computing*, 16(10):2742–2750, 2017.

[19] Weiwei Zhou and Bin Yu. A cloud-assisted malware detection and suppression framework for wireless multimedia system in iot based on dynamic differential game. *China Communications*, 15(2):209–223, 2018.