Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2024 SCPE. Volume 25, Issues 6, pp. 4995–5004, DOI 10.12694/scpe.v25i6.3412

A DYNAMIC SANDBOX DETECTION TECHNIQUE IN A PRIVATE CLOUD ENVIRONMENT

ZHANGWEI YANG* AND JUNYU XIAO†

Abstract. In specific private cloud scenarios, how to defend against malicious software and ensure data security is one of the current research hotspots, and sandbox is an important detection method. This paper proposes a dynamic behavior detection technique based on sandboxing, which real-time monitors and analyzes malicious software behavior. By improving the sandbox behavior weight, integrating virtual resources, and designing fine-grained access control, the detection accuracy and efficiency are enhanced based on zero trust access control system. The simulated attacks are identified on the testing platform, drawing knowledge graphs, achieving effective discovery and tracing. Meanwhile, this paper verified through experiments that the system consumption of the detection method is within an acceptable range, expanding the detection range and reducing the missed detection rate.

Key words: Private cloud, dynamic behavior detection, sandbox escape, access control

1. Introduction. With private clouds, hybrid clouds, and multi-clouds becoming the main form of digital infrastructure, traditional defense mechanisms and security boundaries have been broken. In some specific private cloud application scenarios, its security systems mostly adopt a centralized construction and centralized management approach. This entails the use of firewalls (FW), intrusion detection systems (IPS), web application firewalls (WAF), and database firewalls, forming a security resource pool, and using traditional signature-based detection techniques to identify and block malicious attacks to ensure data security within the cloud [1,2]. However, when attackers or malware exploit 0-day vulnerabilities, signature-based detection methods cannot recognize and defend against them. Hence, there's a need for non-signature-based methods. In response to the defense framework for high-security networks, Li et al. [3] established a defense system architecture that includes APT detection gateways, private clouds within the organization, security management centers, security storage centers, and threat management consoles. Considering the disadvantages of traditional three-tier network architectures when dealing with malicious software attacks, Xu [4] proposed an improved hierarchical centralized network security architecture. By centralizing analysis and control, the internal security components of enterprises form an integrated whole, effectively guarding against malicious attacks.

Sandbox detection technology, based on an environment closely resembling real access entities, can detect malicious activities during the vulnerability exploitation phase. It effectively identifies abnormal behaviors, unknown attacks, and unrecognized malicious code, possessing commendable detection capability . Domestic and foreign scholars' research on sandbox detection technology started with binary program sandboxes. Google's binary sandbox NaCl, designed for X86 architecture, is a double-layer sandbox. The inner layer restricts the control flow of untrusted programs at the instruction level, while the outer layer monitors and validates untrusted program system call behaviors at the system call layer. However, sandbox monitoring consumes significant computational resources. Many malicious attacks and software have integrated sandbox escape features, determining whether a sandbox is running and then employing methods to evade sandbox detection. This has led to a gradual increase in attacks that can bypass sandbox detection, amplifying system risks [6]. Balzarotti et al. [7] conducted research on the different execution paths of malicious samples in simulation analysis environments and real environments, but failed to detect malicious samples with delay types, resulting in a relatively high false alarm rate. Lindorfer et al. [8] proposed a method to detect sandbox escape behavior based on different behavior files, but it was unable to detect some samples that were tested for virtual environments.

*Network and Educational Technology Center, Pingxiang College, Pingxiang, Jiangxi, China. (Corresponding author, yzw@pxu.edu.cn))

†Network and Educational Technology Center, Pingxiang College, Pingxiang, Jiangxi, China.‡

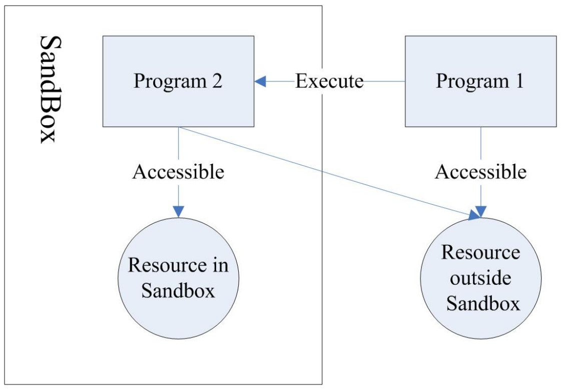

Fig. 2.1: Virtualization-based sandboxes

To effectively respond to new types of attack threats in specific private cloud scenarios, mainly in the form of hybrid cloud and multi cloud, a dynamic sandbox detection technology is implemented to more effectively confront attacks and malware threats with sandbox escape capabilities, which is based on traditional sandbox.In addition, this paper constructs a zero trust access control system, which calculates the trust value for each resource access, by improving the access control mechanism.

2. Malware Detection Methods in a Cloud Environment. For malware detection in cloud environments, the prevalent detection methods are static and dynamic analysis of the software. Static analysis involves using techniques like decompilation and pattern matching for binary forensics to analyze its code patterns before the malware executes. It's then compared with known malicious software features, and if there's a match, it's identified as malicious or as an attack. The advantages of static analysis are speed and simplicity in its implementation. However, the shortcoming is that the features of the sample under test must already exist in a previously established feature database. Hence, it's incapable of detecting unknown malware, rendering it defenseless against 0-day attacks [9-10]. Dynamic analysis runs malicious programs in a controlled environment and determines their malice by monitoring their network activities and process calls. Dynamic analysis can detect unknown malicious attacks and effectively block 0-day attacks. Still, it requires considerable computational resources to simulate a controlled environment, making it complex.In a word, the static analysis method is implemented through feature library matching, while the dynamic analysis method determines malicious behavior by detecting it after running the program. The sandbox is one of the mainstream methods for dynamic malware detection. It offers better detection capabilities for unrecognized malicious software attacks. The implementation principle is as follows: (1) directly import suspicious code into a pre-set sandbox environment; (2) monitor the sandbox's filesystem, processes, network behaviors, and registry changes and export the monitoring traffic data; (3) analyze the exported traffic data to determine if the sandbox contains malicious programs. In terms of specific implementation, sandboxes can be constructed based on virtualization or rules [11-13]. Virtualization-based sandboxes can be either system-level or container-level. They provide an encapsulated runtime environment for untrusted programs or resources, securing other trusted data environments while maintaining the untrusted program's original functionality, without affecting the operation of trusted programs outside the sandbox. The implementation process is illustrated in Fig 2.1.

The virtualization-based sandboxes utilizes virtual machine technology to create a secure isolation environment on physical servers for security detection of suspicious files, applications, or websites. It replicate system resources, introducing redundant resources and demanding high system performance. Rule-based sandboxes, on the other hand, mainly operate through access control rule engines and program monitors. They can solve the system resource replication problem but overly rely on the security of safety rules. Typically, these rules are pre-set and complex, lacking flexibility. Their implementation process is depicted in Fig 2.2

In real-world environments, many malicious programs will perform sandbox checks. If they detect that a sample is in a virtual machine or sandbox environment, they won't run, meaning they integrate sandbox escape

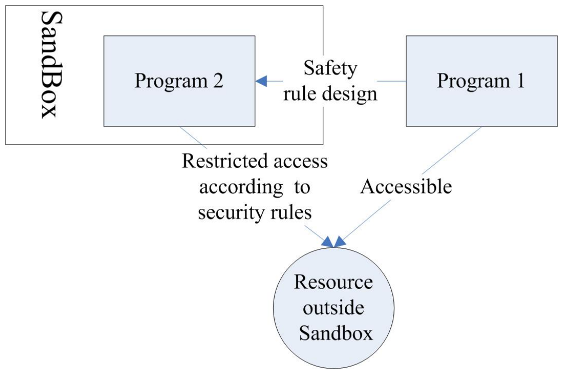

Fig. 2.2: Rule-based Sandbox

features. In private cloud environments, owners can effectively control data and its security. Depending on specific scenario needs and characteristics, they can deploy specific security strategies. Using IaaS according to access entity needs, they can offer virtual resource services formed by integrating computing power, storage, and I/O devices. This simulates more sandboxes and virtual machine environments, allows cloud computing environments to upload detection samples, and improves the success rate and efficiency of sample execution. Additionally, private clouds can adjust different functions and check granularities according to different network environment needs. This allows for more efficient detection of unknown malicious behaviors and codes, compensating for the traditional sandbox's inability to detect malicious attacks with integrated sandbox escape features.

3. Dynamic Sandbox Detection Technology. When a program in a sandbox environment needs to access resources outside the sandbox that are necessary for its operation, access control rules are needed to restrict the program's behavior. In private cloud scenarios, sandbox technology uses cloud platforms with vast amounts of network security data to deeply analyze computer software and mobile applications. Applications are first executed in the cloud sandbox to capture detailed behavioral information. This allows for a comprehensive check for suspicious activities and security issues in entirely unknown computer software and mobile applications in a short time. Any malicious activities or virus attacks are confined within the cloud sandbox, ensuring the safety of the underlying system used by the accessing entity. However, some malicious software with sandbox escape capabilities can bypass protection and execute malicious code without being detected by network security solutions. These methods mainly include detecting human-computer interaction, detecting system features, loading time, and data obfuscation.

3.1. Human-Machine Interaction Sandbox Escape Detection. One of the technical vulnerabilities inherent in sandboxes is the lack of human-machine interaction. Some malicious attacks determine if they're in a sandbox environment by checking for mouse clicks or pop-up dialog boxes in the invaded system. Such malicious programs remain dormant after infiltrating the target system and only execute malicious code when they detect human-machine interactions, such as mouse movements, clicks, or intelligent reactions to dialog boxes. Using this technique, they continually develop more advanced sandbox escape methods. For example, by setting a delay after a mouse click or adding a loop count feature that waits for several mouse clicks before executing. Some malicious codes even check the direction of mouse movement, only executing when the mouse moves in a specific direction. These covert codes are hard to detect, making it easy to evade sandbox detection. The process of malicious software detecting mouse operations is shown in Fig 3.1.

To counteract these malicious attacks, anti-virtualization techniques are applied on top of the sandbox, incorporating program modules like mouse movements, clicks, and dialog box interactions. This gives the sandbox system the ability to simulate human-machine interactions, making it difficult for malicious codes to discern whether they're in a sandbox or a real environment, hence reducing the chances of evading sandbox detection.

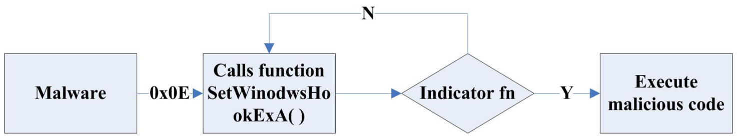

Fig. 3.1: Mouse Operation Detection

3.2. Single File Load Time Detection. Since the sandbox has to inspect a large number of files, the detection time for a single file is limited. Some malicious software sets a sleep timer to delay execution, thereby evading sandbox detection. For instance, certain malware can extend sleep clocks, waiting for sandbox detection to finish before executing. This works by using the timeout variable to call the Sleep Ex() function, ensuring malicious code is only executed after a set period. Typically, the sandbox's inspection time is less than this set period, meaning the malicious code doesn't run during the sandbox's examination. To address this kind of sandbox evasion, the detection method can be improved by setting multiple return checking strategies. When malicious code tries to evade detection by setting a sleep timer, this multi-return detection method will disrupt the set sleep time, increasing the chances of detection during execution.

3.3. Isolation of multiple access subjects. In cloud computing environments, multi-tenant architecture has become one of the core components. In a multi-tenant environment, isolation is key to ensuring the security of data and configuration information for each tenant.In specific applications, we can use methods such as network isolation and computing resource isolation to achieve multi-tenant isolation. Among them, network isolation uses network technologies such as VLAN and VPC to isolate the network traffic between virtual machines of different tenants, making it impossible for tenants to communicate with each other. And computing resource isolation uses container or virtualization technology to isolate computing resources from different tenants.

4. Private Cloud Access Control Policy. When a program in a sandbox environment needs to access resources outside the sandbox that are necessary for its operation, access control rules are needed to restrict the program's behavior. Private cloud computing environments, based on big data, offer an efficient deployment method for cloud-based sandboxes. Isolated physical computing resources can be used as cloud endpoints to provide detection services, enhancing the accuracy and detection capabilities of dynamic sandbox detection technology. However, the inherent multi-instance permission issues in private cloud environments mean that cloud-based sandboxes must address the isolation of multiple accessing entities. By improving authentication in the private cloud, we can refine access control mechanisms, design access control policies to regulate multiple entity data and service access, prevent side-channel attacks between virtual machines, and implement finer-grained access control mechanisms [14].

4.1. Zero Trust Access Control System. Zero Trust is a trust-building method based on authentication, shifting the security architecture from network-centric to identity-centric. All access requires granular, adaptive access control centered on identity. The core principles of Zero Trust include: (1) Minimum access rights, where each accessing entity can only access permissions necessary for their work [15-17]. By limiting permissions, lateral attacks can be effectively prevented after hackers penetrate the network. This granular authorization is typically managed by data owners who periodically review access rights and member identities. (2) All access requires verification. Every time an accessing entity tries to access shared files, applications, or cloud storage devices, their access to related resources must be verified instead of assuming trust after entering a trusted zone. (3) Log monitoring. All activities in the network are logged, allowing for effective identification of abnormal accounts, ransomware, and malicious actions. In Python language, it can be expressed as:

```
# Simulate access control policies
def Access-Control(username)
# Simulate security auditing and logging
def Audit-log(username, action)
```

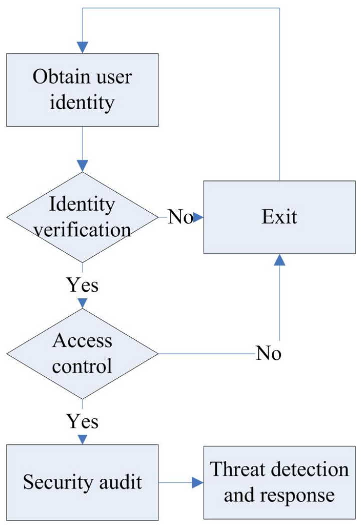

Fig. 4.1: Implementation of Zero Trust Access Control

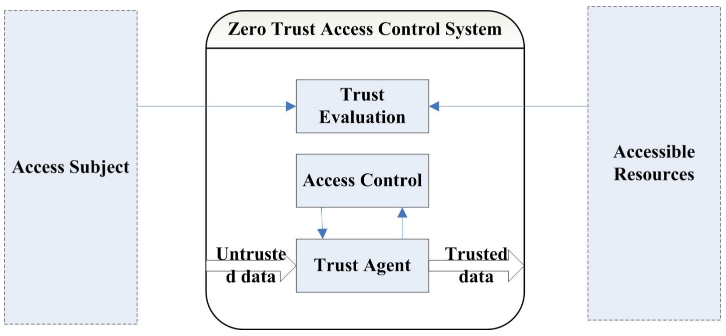

Fig. 4.2: Zero-Trust Access Control System

```
def main()
if Access-Control(username):
Audit-log(username, action)
```

The zero trust access control strategy relies on verifying user identity, verifying each access, conducting security audits and logging, and executing corresponding response measures upon detecting threats. The process is shown in Figure 4.1.

Zero trust has overturned the paradigm of access control, guiding the security system architecture from ”network centralization” to ”identity centralization”. Its essential demand is identity centered access control. In the Zero Trust access control system, no individual, device, system, or application, whether inside or outside the network, is inherently trusted. Trust is rebuilt based on authentication and authorization. Continuous trustworthiness assessments are made on the accessors based on as many data sources as possible. Access control and authorization strategies are then dynamically adjusted based on these evaluations [18-20], as shown in Fig 4.2.

4.2. Computation of Subject Trust Value. Within a private cloud’s zero-trust access control system, the trust value of a subject must be recalculated every time they request access to a resource. This is due to the fundamental assumption of zero trust, which inherently deems all subjects as untrustworthy[21-22]. As a result, the calculation of a subject's trust value is pivotal. Given the openness and dynamic nature of the cloud environment, numerous factors can influence trust evaluations within this context. Many models particularly consider aspects like subjective evaluations of the accessing subject, context relevancy, and time decay[23-25]. In a private cloud setting, Fuzzy Analytic Hierarchy Process(FAHP) can accurately reflect the relationship between access subject attributes and trust values, which can be expressed as:

$$F = (e_{ij})_{n*m} \tag{4.1}$$

Where n denotes the number of trust attribute values for the access subject, and indicates the maximum number of trust levels for a certain trust attribute partition. Thus, F is the standardized Analytic Hierarchy Process matrix. Drawing from discrete trust value measurement methods, a subject's behavior can be translated into trust levels defined as: $\{Trust, Somewhat\ Trust, Neutral\ Trust, Somewhat\ Distrust, Distrust\}$. Using model subsets, these trust categories can be articulated as:

$$T_n = \{Trust, Somewhat\ Trust, Neutral\ Trust,\ Somewhat\ Distrust, Distrust\} \tag{4.2}$$

Based on a cloud service provider's trust determination of a subject's behavior, quintuple T_n must simultaneously fulfill the following conditions:

$$\begin{cases} T_{i-1} > T_i & i = (2,3,4,5) \\ T_i \cap T_j & (i \neq j) \end{cases} \tag{4.3}$$

Before ascertaining the trustworthiness of a subject's behavior, one needs to collect behavioral evidence and undertake attribute analysis. Based on the influencing factors of cloud environment security, combined with the AHP model, a scoring method is used to determine the relative importance of each indicator, and thus obtain the weight of each indicator. To ensure relative accuracy of behavioral evidence values, picking the appropriate time granularity within a given time frame is pivotal. Suppose X symbolizes the trust measure value of a subject's behavior, then X should lie within the Tn trust level space, satisfying $X \in [T_{i-1}, T_i]$, where T_{i-1} and T_i represent the upper and lower trust level bounds, respectively. The initial judgment matrix $FQ = (e_{ij})_{m*m}$ can be express as follows, by using the AHP to compare the importance of trust attributes $F = (f_1, f_2, , f_m)$, and compare any two values of matrix F with each other.

$$fe_{ij} = \begin{cases} 0 & f_i < f_j \\ \frac{1}{2} & f_i = f_j \\ 1 & f_i > f_j \end{cases} \tag{4.4}$$

It can calculate the weight vectors $W_n = (W_1, W_2, , W_m)$ of each trust attribute, by converting the initial judgment matrix FQ to a fuzzy consistency matrix $Q = (q_{ij})_{m*m}$ .Similarly, W_n can be transformed into a matrix $W = (w_{ij})_{m*m}$.Then, the weight matrix of the attribute is combined with the judgment matrix to obtain a new matrix Z with the diagonal as the attribute evaluation vector. According to the formula for calculating the trust value, the product of the trust attribute vector and the weight vector is the trust value of the attribute, and the trust value T is obtained as follows:

$$T = 1 - \sum_{i=1}^{n} f_i w_{f_i} \tag{4.5}$$

This trust value becomes void post-resource use, necessitating a recalculation before the next resource request. As time decays, the trust value of the subject will also decrease. Therefore, this paper design a decay factor that dynamically changes based on the time interval of its last trust evaluation, so as to reduce the proportion of historical trust values in the overall trust value calculation process.

Table 5.1: Experimental environment configuration.

Hadoop Cluster	OS	Hadoop Version	IP Address
Master	ubuntu-20.04.4-live-server-amd64	Hadoop 3.2.2	192.168.1.240
Slave1	Ubuntu-20.10-desktop-amd64	Hadoop 3.2.2	192.168.1.245
Slave2	Ubuntu-20.10-desktop-amd64	Hadoop 3.2.2	192.168.1.246

5. Building the Test Environment. To test a series of implementations of dynamic sandbox deployment for the discovery and identification of malicious activities, we have constructed an open-source security sandbox called Cuckoo based on VirtualBox. This sandbox operates in an environment entirely isolated from the host operating system, possessing complete hardware system capabilities[22].The Cuckoo Sandbox is a malware analysis system based on a virtualized environment, used for dynamic analysis of malware samples. It can automatically execute and analyze program behavior, record various dynamic activities of malicious programs, and generate detailed analysis reports.In addition, the Cuckoo sandbox is mainly composed of central management software and various analysis virtual machines. The central management software is responsible for managing the analysis of samples, such as initiating analysis work, recording behavior, and generating reports. And the analysis of virtual machines in Cuckoo sandbox is responsible for executing malicious program samples in isolated environments and reporting the analysis results to central management software. The specific implementation is as follows:

(1) Deploy the Cuckoo sandbox, based on GPLv3, in the cloud to analyze malicious software. Compared to sandboxes that analyze based on static feature codes, Cuckoo has the advantage of dynamic monitoring. Cuckoo requires deployment on both Host and Guest sides. The Host is responsible for managing the Guest's startup analysis, network traffic collection, and receiving tasks (files) from the Host to obtain information post-execution. In the Cuckoo sandbox, the executable files introduced with traffic are executed, monitoring their real-time performance. Cuckoo can analyze traffic content, supporting PE files, DLL files, Office documents, Zip archives, and nearly all other common file formats.

(2) Operate the sandbox within Cuckoo using the python command-line tool. Use cuckoo.py to start the sandbox engine and submit.py to submit applications for analysis to the sandbox. After the Host receives the task, the sandbox engine communicates with the Agent in the virtual machine to run the application, then waits for the analysis results and outputs them to a specified directory.

(3) View the Cuckoo traffic analysis results to discover and identify potential malicious software codes. Cuckoo's application analysis results from traffic mainly include traces of function and API calls, records of application operations on files, memory images of chosen processes, complete memory data of the analysis machine Guest, screenshots during malicious software execution, and the network traffic generated by the Guest. By integrating data analysis from the network security platform, malicious activities can be matched and identified.

We created a Cuckoo sandbox in a cloud computing experimental environment and deployed the SAX2 intrusion detection system and yaahp hierarchical analysis software simultaneously. The experimental environment consists of three servers, and the hardware configuration is as follows: 2*Intel 5218(2.3GHz/16C)/256G DDR3/2*480GB SSD. The experimental environment configuration is shown in Table 5.1.

The SAX2 software in the experimental environment is used to obtain basic data on user behavior, yaahp is used to construct a user behavior trust model, and AHP method is used to construct a judgment matrix to obtain user behavior weights. Meanwhile, a behavior evaluation team consisting of 5 experts was used to evaluate and score the number of times the user ran threat programs N1 and scanned cloud server ports N2 obtained from the SAX2 system, forming the following user behavior evaluation Table 5.2.

Within the Cuckoo sandbox, add program codes to address sandbox evasion tactics, such as human-machine interaction, and determine whether malicious codes call Set Windows Hook Ex A() or other mouse behavior monitoring functions. If such codes are detected, invoke mouse click and movement functions in the sandbox and monitor mouse input. While simulating mouse operations, observe the results of malicious code execution associated with the pointer function fn. Similarly, in the sandbox, simulate clicking the MessageBox() dialogue

Table 5.2: User behavior evaluation form.

N1	N2
7	6
9	8
6	8
5	7
8	9

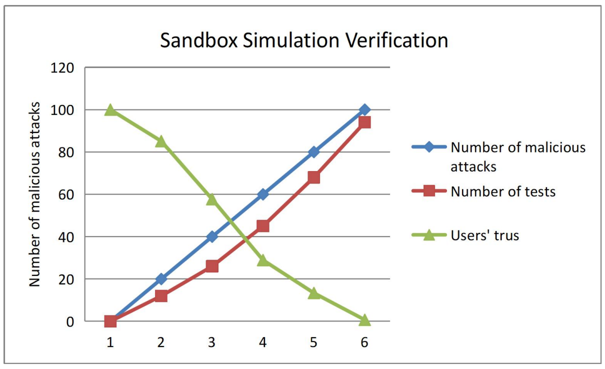

Fig. 5.1: Sandbox Simulation Verification

box operation to activate malicious code and observe its execution results.

To validate the effectiveness of the detection method, a simulation is conducted where random users run malicious programs within the Cuckoo sandbox. As the experiment begins, random users interact with the cloud server. As the number of interactions increases, the frequency of running malicious programs also grows. Experiments are conducted on detection accuracy and false alarm rates to measure the precision of the detection method, as shown in Fig 5.1.

With the gradual increase in malicious software attack behaviors, the dynamic sandbox detects more instances, and the trust level of the malicious software rapidly decreases. This process quickly restricts the malicious software from running in the sandbox, marking it as malicious software, preventing its entrance into the actual private cloud environment, thereby reducing risks in the private cloud environment.

The calculation of user trust values during the interaction process is completed in a trusted proxy in a zero trust access control system. As the number of visits increases, the access to resources also gradually increases. In the zero trust mechanism, each visit requires recalculating the trust value, resulting in an increase in the consumption of system resources, as shown in Fig 5.2, which shows the change in CPU usage.

It can be seen that compared to the resource consumption of traditional access control, the trust value in the zero trust mechanism increases by about 3% per calculation, which is within an acceptable range. The sandbox detection technology proposed in this paper, can achieve dynamic sample detection in virtual machine environments , expanding the detection range based on Lindorfer's detection method. Furthermore, it is possible to detect delayed malicious samples and reduce the missed detection rate, based on the detection technology proposed by Balzarotti.

6. Conclusion. This paper proposed a dynamic sandbox detection technology based on a private cloud environment, aimed at detecting malicious software attacks in private cloud environments and enhancing their security. This technology leverages the advantages of dynamic sandbox technology, incorporating features to counteract sandbox evasion tactics such as human-machine interactions. It's optimized for the characteristics of a private cloud environment, ,and a model of trust value decay over time was designed based on the FAHP method, realizing sandbox access control in a private cloud based on the zero-trust concept. By judging the

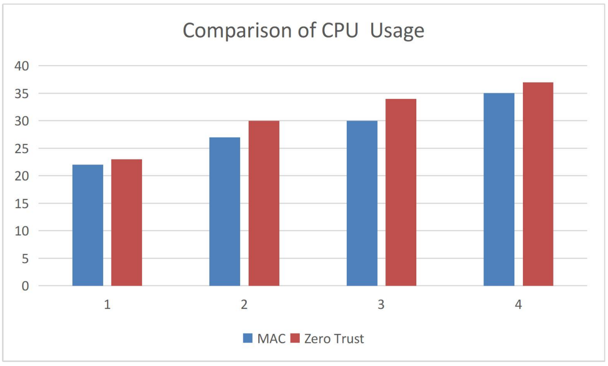

Fig. 5.2: Comparison of CPU Usage

trustworthiness through the trust value of each accessing entity, a Cuckoo test environment was constructed for verification, and observed that the changes in system resource consumption were within an acceptable range. In future research, this technology can be further refined to improve the efficiency and accuracy of malicious software detection.

Acknowledgments. This work was supported in part by Science and Technology Program of Education Department of Jiangxi Province, China (GJJ219307), Science and Technology Program of Pingxiang (201800002), and Science and Technology Programs of Education Department, China (2018A01017).

REFERENCES

[1] Hassan Takabi, JamesB.D. Joshi,Gail-Joon Ahn , *Security and Privacy Challenges in Cloud Computing Environments.*IEEE Security Privacy, 2010,8(6):24-31.

[2] Yu Nenghai, Hao Zhuo, Xu Jiajia, et al, *A review on cloud security research progress.*Journal of Electronics, 2013, 41(02):371-381.

[3] Li Fenghai, Li Shuang, Zhang Bailong, et al , *Research on high-level security network anti-APT attack scheme.*Computer Security Professional Committee of the Chinese Computer Society. Proceedings of the 29th National Computer Security Academic Conference. Editorial Department of "Information Network Security",2014(9):109-114.

[4] Xu Ting, *A network security architecture effectively defending against APT attacks.*Information Security and Communication Confidentiality, 2013(06):65-67.

[5] Diao Mingzhi, Zhou Yuan, Li Zhoujun, et al, *Simulation of Windows security mechanism based on ine and sandbox system implementation.* Computer Science, 2017, 44(11):246-252,267.

[6] Yuning C ,Yi S ,Zhaowen L , *DroidHook: a novel API-hook based Android malware dynamic analysis sandbox.*Automated Software Engineering,2023,30(1).

[7] Balzarotti D,Cova M,Karlberger C,et al, *Efficient Detection of split personalities in malware.* Proceedings of Network and Distributed System Security Symposium,San Diego,California,USA,2010.

[8] Lindorfer M,Kolbitsch C,Comparetti P M., *Detecting Environment-Sensitive Malware.* Proceedings of International Conference on Recent Advances in Intrusion Detection,2011:338-357.

[9] Songsong L ,Pengbin F ,Shu W , et al., *Enhancing Malware Analysis Sandboxes with Emulated User Behavior.*Computers & Security,2022,115, 102613..

[10] Evgeny N ,Dmitry I ,Pavel K , et al , *Isolated Sandbox Environment Architecture for Running Cognitive Psychological Experiments in Web Platforms.*Future Internet,2021,13(10),245.

[11] Shun Y,Yuzo T,Youki K, et al , *Design and implementation of a sandbox for facilitating and automating IoT malware analysis with techniques to elicit malicious behavior: case studies of functionalities for dissecting IoT malware.*Journal of Computer Virology and Hacking Techniques,2023,19(2),149–163.

[12] Zhang Juntao, Wang Yijun, Xue Zhi., *Research on adversarial malicious code sandbox detection method based on angr.* Computer Applications and Software, 2019, 36(2):308-314.

[13] Wang Zhihua, Pang Haibo, Li Zhanbo , *An access control scheme suitable for Hadoop cloud platform.*Journal of Tsinghua University (Natural Science Edition), 2014, 54(1):53-59.

[14] Gousiya Begum;S.Zahoor Ul Huq , *Sandbox security model for Hadoop file system.*Journal of Big Data,2020,7(1).

[15] Jiang Ning, Fan Chunlong, Zhang Ruihang, et al, *Model-based zero-trust network security architecture.* Small and Micro Computer Systems, 2023, 44(8):1819-1826.

[16] Mark B, *Zero trust computing through the application of information asset registers.*Cyber Security: A Peer-Reviewed Journal,2021,5(1), 80-94.

[17] Astakhova L. V , *Zero Trust Model as a Factor of Influence on the Information Behavior of Organization Employees.*Scientific and Technical Information Processing,2022,49(1), 60–64.

[18] Mingyang X,Junli G,Haoyu Y, et al, *Zero-Trust Security Authentication Based on SPA and Endogenous Security Architecture* Electronics,2023,12(4):782.

[19] GUO Baoxia, WANG Jiahui, MA Limin, et al, *Research on Dynamic Access Control Model of Sensitive Data Based on Zero Trust.*Netinfo Security, 2022, 22(6): 86-93.

[20] Luo J Yang Zhangwei, *A Behavior Trust Model based on Fuzzy Logic in Cloud Environment.*International Journal of Performability Engineering,2018,14(4).

[21] Tian Liqin, Lin Chuang, *User behavior trust evaluation mechanism based on dual sliding window.* Journal of Tsinghua University: Natural Science Edition, 2010(5): 763-767.

[22] Shang Y ,Shang L , *Trust model for reliable node allocation based on daily computer usage behavior.*Concurrency and Computation: Practice and Experience,2018,30(6).

[23] Indu S ,Rajni J , *Trust factor-based analysis of user behavior using sequential pattern mining for detecting intrusive transactions in databases.*The Journal of Supercomputing,2023,79(10), 11101–1113.

[24] Deng Sanjun, Yuan Lingyun, Sun Limei, *Research on IoT access control model based on trustworthiness.* Computer Engineering & Design, 2022, 43(11):3030-3036.

[25] Rushdan A H ,Shurman M ,Alnabelsi S , *On Detection and Prevention of Zero-Day Attack Using Cuckoo Sandbox in Software-Defined Networks.* The International Arab Journal of Information Technology,2020,17(4A),662-670.

Edited by: Jingsha He
Special issue on: Efficient Scalable Computing based on IoT and Cloud Computing
Received: Mar 1, 2024
Accepted: May 6, 2024