ISSN 2616-5767 Vol.8, Issue 4: 47-52, DOI: 10.25236/AJETS.2025.080406

Design and Implementation of a Mimic Judgment Module Based on Situation Awareness

Li Zhaozhao

Purple Mountain Laboratories, Nanjing, China
lizhaozhao@pmlabs.com.cn

Abstract: *The traditional implementation of mimicry decision-making often adopts majority decision-making mechanisms such as 2-out-of-3 and 3-out-of-5, where each executor is assigned the same decision weight, ignoring the current health status of each executor. It only makes threshold judgments based on the cumulative number of decision errors, and triggers the cleaning and recovery mechanism only when the threshold is exceeded. This approach cannot effectively respond to short-term, sharply deteriorating working conditions and also ignores the guiding significance of historical data for mimicry decision-making. Therefore, this paper proposes a situation-aware-based mimicry decision module, which utilizes the error frequency of executors as an indicator, assigns different decision weights to CPU executors under various working conditions, and fully integrates historical data with current data, thereby making the mimicry defense system more robust. At the same time, the relationship between the window size and the sensitivity of executors is analyzed through simulation experiments.*

Keywords: *Mimic Judgement, Situation Awareness, Mimic Defence*

1. Introduction

Mimic defense aims to form heterogeneous executors through heterogeneous processors, operating systems, and software. It relies on the differences between heterogeneous executors to make up for their inherent defects and backdoor vulnerabilities. This ensures that even when the system is subjected to backdoor attacks via vulnerabilities or experiences functional failures, it can still maintain overall stability, providing dual protection for both network security and the system's functional security [1]. In a normally functioning mimic defense system, data to be processed is uploaded to the mimic scheduler via data acquisition terminals or communication network cards. The mimic scheduler then distributes the pending data to more than three heterogeneous executors for processing. Each heterogeneous executor sends its processing result back to the mimic scheduler, which finally performs mimic decision-making and uniformly outputs the result to the data execution mechanism or network card [2-4].

However, in the traditional mimic defense system, the majority voting method is generally adopted for mimic decision-making [5]. This decision-making method, featuring a simple structure and good universality, has been applied in various scenarios such as cloud platforms, cloud storage, routing, switching, and edge control [6]. Nevertheless, the simple majority decision only utilizes the current output information of executors, failing to effectively make use of the historical experience contained in past data and ignoring the issue that the reliability of output information from different executors varies under different health conditions. As a result, it is prone to causing the phenomenon of common-mode escape in certain cases[7-9]. Therefore, this paper proposes a weighted mimic decision method based on historical error frequency, conducts simulation analysis, and finally carries out application verification in the prototype of the mimic panoramic monitoring terminal.

2. Mimic judgment system design

2.1 Simple majority mimic decision-making method

When the information to be processed is input into the system, the mimic distribution module will copy and distribute the input data to three different CPU executors. These three distinct CPU executors operate on the data. And then their processing results are sent back to the mimic module. In the mimic decision module, a simple majority vote is conducted, and the result that gains majority consensus is selected as the trusted result and output to the outside, as shown in Figure 1.

ISSN 2616-5767 Vol.8, Issue 4: 47-52, DOI: 10.25236/AJETS.2025.080406

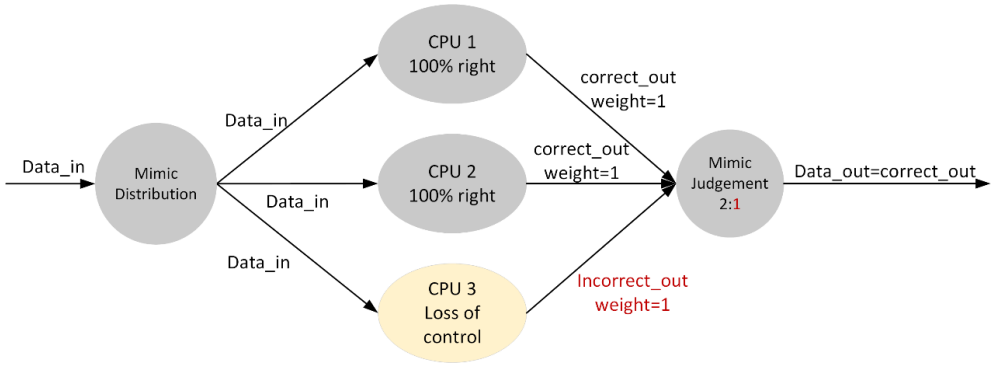

Figure 1: Simple majority mimic decision-making method

However, when some executors are in a sub-healthy state, the simple majority voting mechanism may lead to the common-mode escape problem, as illustrated in Figure 2. Currently, CPU1 and CPU2 are in a sub-healthy state due to issues such as cyber attacks, device aging, and electromagnetic interference, resulting in a 60% probability of outputting incorrect results and a 40% probability of outputting correct ones. If the simple majority voting is still adopted, where each CPU executor is assigned the same decision weight, the mimic defense system will make a misjudgment when CPU1 and CPU2 output incorrect results (with incorrect_out versus correct_out being 2:1), and ultimately output the incorrect result.

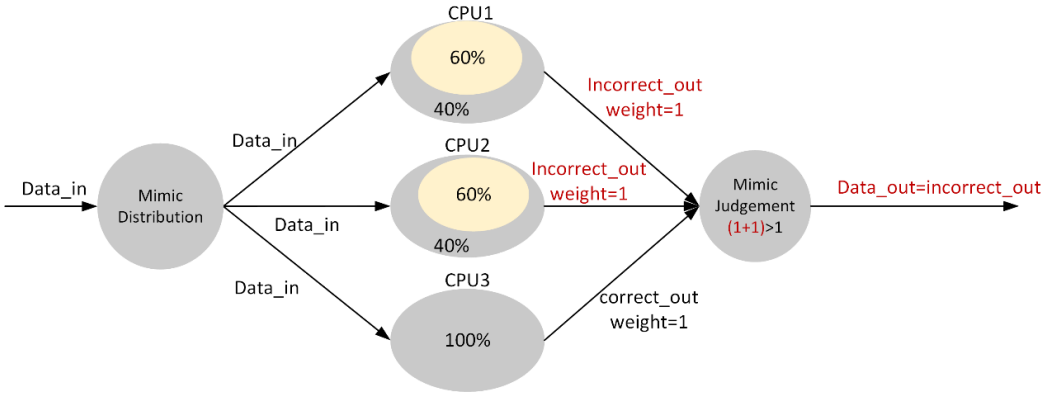

Figure 2: Common-mode escape when two CPU executors are in a sub-healthy state

Therefore, it is necessary to analyze historical data and evaluate the current health status of CPU executors in combination with historical experience, and finally set dynamically configurable weights for them, so as to make mimic decision-making more flexible and applicable to different working conditions.

2.2 Situation-aware-based Mimic Decision-making Method

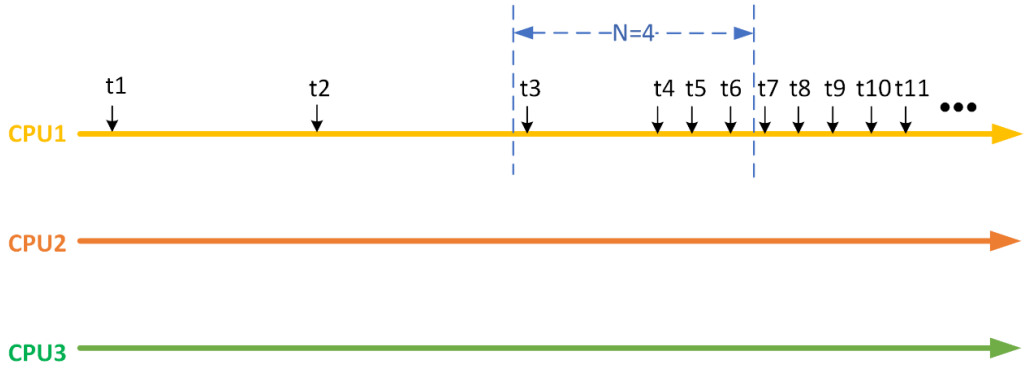

Figure 3: Time series of monomer attack behaviors targeting CPU1

Take the scenario where a single executor is attacked as an example: in a 3-module heterogeneous

ISSN 2616-5767 Vol.8, Issue 4: 47-52, DOI: 10.25236/AJETS.2025.080406

mimic defense system, only CPU1 is under attack. In the first half of the time period, the intervals between attacks are relatively long, and the frequency of outputting incorrect results is low. However, starting from time t4, the attack intensity increases sharply, leading to a dramatic rise in the frequency of incorrect results output and a rapid deterioration in its health status, as shown in Figure 3. Therefore, it becomes possible to perceive the current attack situation of different CPU executors, calculate their health status, and dynamically assign different decision weights by using the number of breaches per unit time, i.e., breach frequency, as an indicator.

The situation-aware-based mimic decision-making method is shown in Figure 4. With a sliding window of size N as the scale, the time spans Δt_1, Δt_2, and Δt_3 for which the three heterogeneous executors encounter N breach cases are calculated, respectively, as well as the attack densities d_1, d_2, and d_3 of the CPU executors. After normalization, these values serve as the respective decision weights w_1, w_2, and w_3 for the three executors. The calculation formula for the decision weights of the three heterogeneous executors is given in Equation (1).

$$\begin{cases} w_1 = \frac{d_1}{\max(d_1,d_2,d_3)} \times 100\% \\ w_2 = \frac{d_2}{\max(d_1,d_2,d_3)} \times 100\% \\ w_3 = \frac{d_3}{\max(d_1,d_2,d_3)} \times 100\% \end{cases} \tag{1}$$

Among them, the attack density d is calculated in Equation (2).

$$\begin{cases} d_1 = \frac{N}{\Delta t_1} \\ d_2 = \frac{N}{\Delta t_1} \\ d_3 = \frac{N}{\Delta t_1} \end{cases} \tag{2}$$

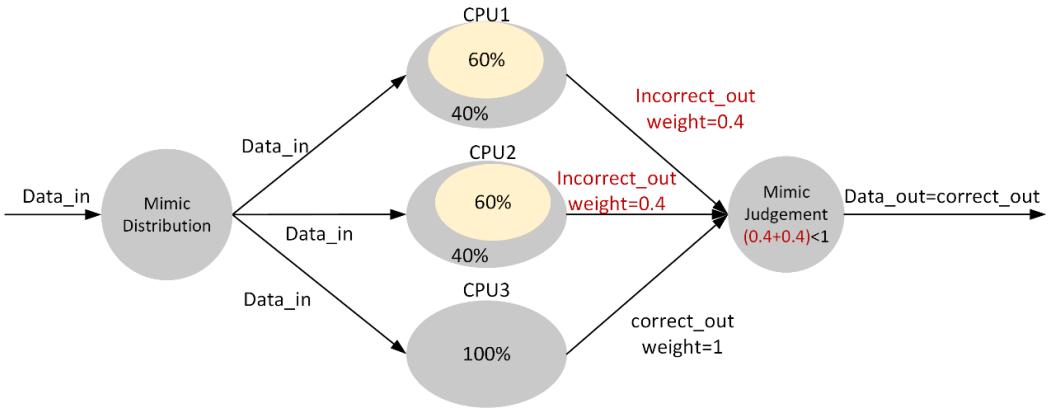

Figure 4: Weighted Mimic Decision-Making Method

As shown in Figure 4, after the mimic arbitrator calculates the respective decision weights of CPU1, CPU2, and CPU3 based on historical data, it outputs the result after weighted arbitration. This avoids the common-mode escape problem when two sub-healthy CPU executors dominate the system in simple arbitration, enabling the system to finally output the correct result.

2.3 Hardware framework of mimic module

The nodes in current industrial control systems vary greatly in functionality, and the cost of carrying out mimic transformation for each specific system service is enormous. For example, a mimic switch is generally of a rack-mounted structure, with both the switching chip and CPU executors located on the same hardware circuit; a mimic industrial control device is usually in the form of a combination of a backplane and plug-in boards, and the two cannot be used interchangeably. Each time the scenario is changed, a new set of software and hardware systems needs to be rebuilt, which brings inconvenience to the construction of the mimic defense system. Therefore, this paper designs a cascaded universal mimic defense module (Scheduler module), which integrates the mimic scheduler and heterogeneous executors through a cascaded chassis. In different application scenarios, only the data acquisition unit (IO module) needs to be redesigned. The prototype architecture is shown in Figure 5. The situation-aware-based mimic decision system is installed on the mimic scheduling plug-in board of the scheduling module.

ISSN 2616-5767 Vol.8, Issue 4: 47-52, DOI: 10.25236/AJETS.2025.080406

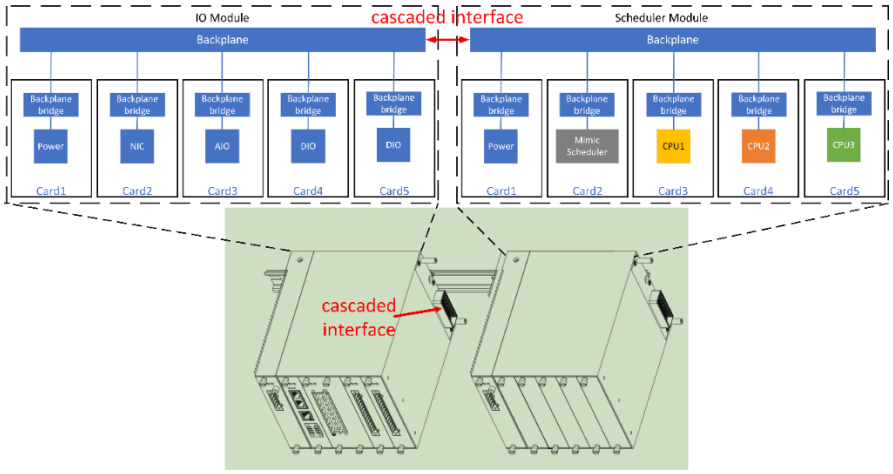

Figure 5: Hardware of Mimic Scheduling System

The working process of the entire mimic scheduling system is divided into an uplink distribution part and a downlink decision part.

The uplink process is as follows: First, the IO module collects external digital and analog information or receives monitoring information from the network through the network interface card (NIC), and then sends the data to be processed to the mimic scheduler module through the backplane cascading interface. Second, in the mimic scheduling board of the mimic scheduler module, the data to be processed is distributed to three heterogeneous executor plug-in boards through a distribution mechanism, namely CPU1 (ARM), CPU2 (Loongson), and CPU3 CARD (RISCV). Then, the application programs deployed in the heterogeneous executors perform computations. This is shown in Figure 6.

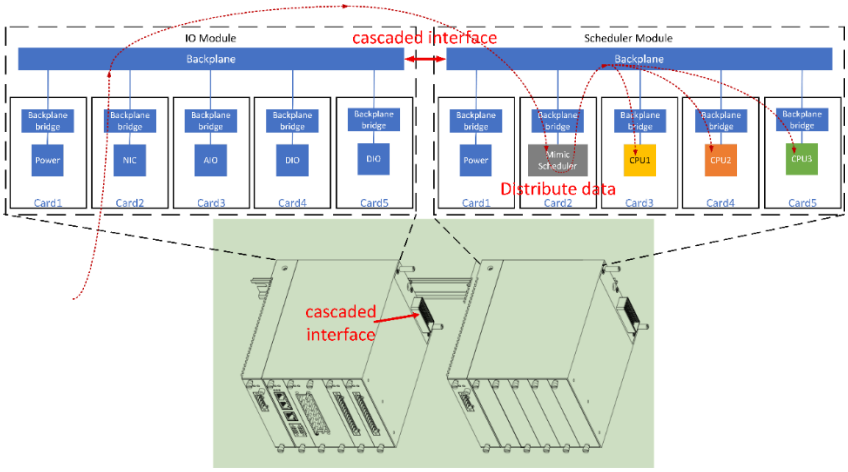

Figure 6: Uplink Data Process

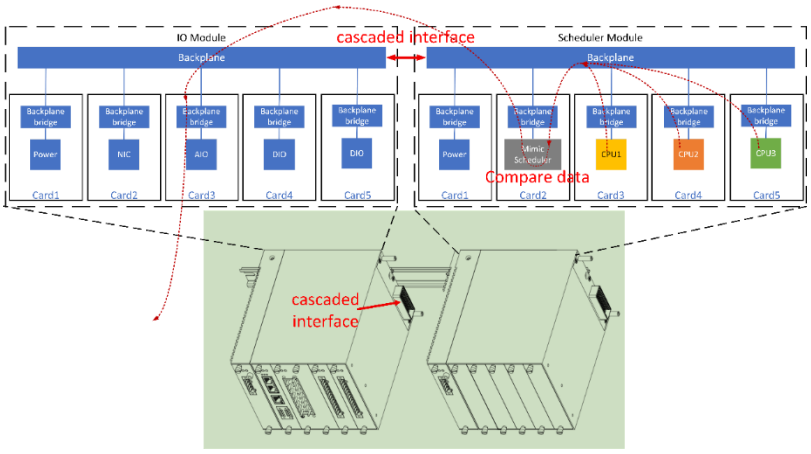

Figure 7: Downlink Data Process

ISSN 2616-5767 Vol.8, Issue 4: 47-52, DOI: 10.25236/AJETS.2025.080406

The downlink process is as follows: First, the three heterogeneous CPU executors respectively output their respective calculation results to the mimic scheduler board. Second, mimic decision-making is performed in the mimic scheduler board, and the calculation results that obtain the majority in the same batch are output as trusted results to the IO module. Finally, the calculation results are output to the NIC or IO board according to the controlled object, and the safe and trusted results are ultimately output to the network or digital/analog actuators, thus completing the entire mimic control process. As shown in Figure 7.

2.4 Parameter Analysis

The situation-aware-based mimic decision-making method can also promptly issue alarm information when the system deteriorates sharply, instead of triggering an alarm only when the cumulative output failures reach a threshold, which enhances the system's ability to respond to environmental changes. The sliding window size N is crucial to the sensitivity of the mimic defense system. A larger N will smooth the abrupt system changes with the historical previous values in the window, reducing the slope of the attack density curve and thus delaying the alarm time. Meanwhile, a larger N can smooth out minor disturbances and prevent false alarms. A smaller N makes the system more sensitive when the system deteriorates sharply and the attack density increases drastically, enabling rapid alarms. However, a smaller N has relatively poor ability to cope with short-term disturbances and is prone to false alarms. As shown in Figure 8, the changes in attack density are detected by the mimic scheduler when the sliding window N takes different values; the smaller N is, the more drastic the fluctuations, and the more sensitive the system. Figure 9 shows the changes in mimic decision weights when N takes different values. When a change occurs at time t_3 as shown in Figure 3, a smaller N leads to a faster convergence of the mimic decision weight to a lower level, with faster weight updates, allowing the system to adapt to new changes in a shorter time and reduce attack escape behaviors. Nevertheless, a smaller N is also prone to misjudgment during small-range fluctuations. Therefore, when deploying the weighted mimic decision-making mechanism, it is necessary to balance the impact between anti-interference and response speed to determine an appropriate window size.

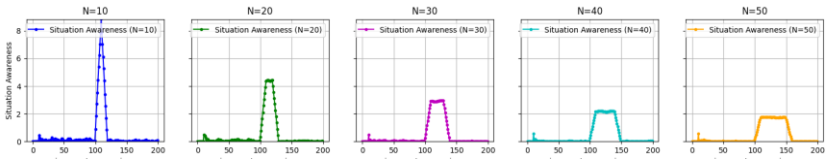

Figure 8: Changes in attack density with different values of N when CPU1 is individually under attack

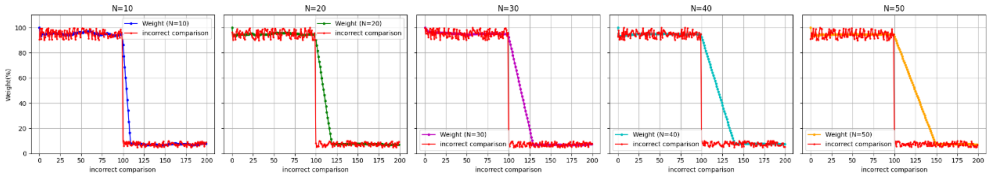

Figure 9: Changes in vote weight with different values of N when CPU1 is individually under attack

3. Conclusions

This paper proposes a situation awareness method for mimic defense systems based on historical data. Based on situation awareness, the health status of each CPU executor is obtained, and different mimic decision weights are assigned to each CPU executor according to their health status. This combines historical data with current results to jointly output corresponding results, avoiding the escape of some common-mode vulnerability attacks and achieving better robustness. Meanwhile, a universal hardware architecture of the mimic scheduling module is proposed, providing new ideas for the deployment and implementation of mimic defense.

Acknowledgements

This work was supported by the State Grid Corporation of China: Research on the Theory and Technology of Endogenous Security and Reliable Operation of Power Information System Against Unknown Attacks (Project Code: 5700-202458225A-1-1-ZN).

ISSN 2616-5767 Vol.8, Issue 4: 47-52, DOI: 10.25236/AJETS.2025.080406

References

[1] Hu, H., Wu, J., Wang, Z., & Cheng, G. (2018). Mimic defense: a designed-in cybersecurity defense framework. IET Information Security, 12(3), 226-237.
[2] Ma, H. L., Yi, P., Jiang, Y. M., & He, L. (2017). Dynamic heterogeneous redundancy based router architecture with mimic defenses. Journal of Cyber Security, 2(1), 29-42.
[3] Xu, J. (2021). Research on cyberspace mimic defense based on dynamic heterogeneous redundancy mechanism. Journal of Computer and Communications, 9(7), 1-7.
[4] Zhang, B., Xi, Z., Wang, Y., & He, C. (2023, March). Research on the main elements of mimic platforms. In Fifth International Conference on Computer Information Science and Artificial Intelligence. 12566, 148-154.
[5] Wang, Y., Xi, Z., Zhang, B., & He, C. (2023, March). A protection framework based on dynamic heterogeneous redundancy architecture. In Fifth International Conference on Computer Information Science and Artificial Intelligence. 12566, 127-134
[6] Ouyang, L., Song, K., Zhang, W., & Wei, S. (2023). Microcontroller design based on dynamic heterogeneous redundancy architecture. China Communications, 20(9), 144-159.
[7] Shao, S., Ji, Y., Zhang, W., Liu, S., Jiang, F., Cao, Z., ... & Zhou, L. (2023). A DHR executor selection algorithm based on historical credibility and dissimilarity clustering. Science China Information Sciences, 66(11), 212304.
[8] Hu, J., Li, Y., Li, Z., Liu, Q., & Wu, J. (2024). Unveiling the strategic defense mechanisms in dynamic heterogeneous redundancy architecture. IEEE Transactions on Network and Service Management, 21(4), 4912-4926.
[9] Kang, Y., Zhang, Q., Jiang, B., & Bu, Y. (2024). A Differentially Private Framework for the Dynamic Heterogeneous Redundant Architecture System in Cyberspace. Electronics, 13(10), 1805.