SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2023, Vol. 66 212304:1–212304:14
https://doi.org/10.1007/s11432-022-3635-2

A DHR executor selection algorithm based on historical credibility and dissimilarity clustering

Sisi SHAO[1,3], Yimu JI[1,2,3,6,7*], Weili ZHANG[4], Shangdong LIU[1,3,6,7], Fei JIANG[5], Zhigang CAO[2], Fei WU[1,3,6,7], Fukang ZENG[1,3], Jun ZUO[1,3] & Longfei ZHOU[1,3]

[1]*School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing* 210023*, China;*
[2]*Purple Mountain Laboratories, Nanjing* 211111*, China;*
[3]*Institue of High Performance Computing and Bigdata, Nanjing University of Posts and Telecommunications, Nanjing* 210003*, China;*
[4]*School of Foreign Languages, Information Engineering University, Zhengzhou* 450006*, China;*
[5]*The 28th Research Institute of China Electronics Technology Group Corporation, Nanjing* 210000*, China;*
[6]*Nanjing Center of HPC China, Nanjing* 210003*, China;*
[7]*Jiangsu HPC and Intelligent Processing Engineer Research Center, Nanjing* 210003*, China*

Received 29 June 2022/Revised 3 October 2022/Accepted 24 November 2022/Published online 12 October 2023

Abstract The security of the dynamic heterogeneous redundancy (DHR) architecture relies on the heterogeneity of its executors, which also defines the vulnerability of the mimic system. In order to select executors with reliable and significant dissimilarity as service executors, we propose a DHR executor selection algorithm based on historical credibility and dissimilarity clustering (HCDC), which adds two metrics of executor historical credibility and dissimilarity. First, to maximize the difference between heterogeneous executor pools, clustering is performed based on the dissimilarity of the executor. Second, the executor with the highest historical credibility is selected from the heterogeneous executor pool as the candidate pool. The historical credibility is dynamically updated by the negative feedback control based on the results of the multi-mode adjudicator. Finally, the dynamic scheduling algorithm selects the executors from the candidate pool to form the set of service executors. The simulation results demonstrate that, in comparison to existing methods, the algorithm reduces the attack success rate and average failure rate while increasing system reliability.

Keywords endogenous security, mimic defense, DHR, executor selection, historical credibility

Citation Shao S S, Ji Y M, Zhang W L, et al. A DHR executor selection algorithm based on historical credibility and dissimilarity clustering. Sci China Inf Sci, 2023, 66(11): 212304, https://doi.org/10.1007/s11432-022-3635-2

1 Introduction

Cyberspace security has become a major issue in the information age [1], where there are always uncertain threats such as unknown vulnerabilities and backdoors. The asymmetric situation between the attacker and defender leads to a very passive defender. To compensate for the insecurity of a system, an unacceptable price is frequently demanded [2]. In view of the above situation, transforming traditional static and passive security defense approaches into dynamic and active defense strategies, (e.g., moving target defense (MTD) [3, 4] and cyber mimic defense (CMD) [5–7]), has become a new network security research direction. MTD employs various diversified techniques to dynamically change the attack surface to enhance the uncertainty of the system and reduce the chance of the vulnerable surface of the system being attacked. However, there is a risk of being breached by attackers with its diversity of variations. Therefore, Chinese academician Wu Jiangxing proposed the theory of CMD. This theory constructs the dynamic heterogeneous redundancy (DHR) architecture by introducing dynamic, random, and diverse defense elements based on closed-loop negative feedback control in the dissimilar redundancy structure (DRS) architecture [8, 9]. Irregularly modifying the internal structure of the system to present generalized external uncertainty provides a new interpretation to address system security vulnerabilities and backdoor threats.

* Corresponding author (email: jiym@njupt.edu.cn)

The DHR architecture satisfies the input-process-output (IPO) [10,11] structure in structured design, in which the input module copies the input and sends it to the n service executors in the processing module, and the processing module sends the computation results of the service executors to the adjudicator in the output module for adjudicating the output. The DHR architecture breaks the pattern of staticity, similarity and determinism of cyberspace security information systems through multiple heterogeneities in different spatial-temporal dimensions. To a certain extent, it tolerates external attacks based on known and unknown vulnerability backdoors as well as penetration attacks by unknown viruses and trojans to achieve the goal of integrating system reliability and attack resistance [12].

The reliability and heterogeneity of executors are the basis and prerequisite for achieving security and diversity in mimic defense. The scheduling module is the core part of the DHR architecture, which selects or replaces the executors in the processing module based on the negative feedback information from the output module, and realizes the operations such as reconfiguration and reconstruction of the executors to make the internal characteristics uncertain [13]. Executors are considered functionally equivalent and structurally distinct independent individuals, although perfect dissimilarity is impossible to attain. Certain similarities exist between them and vary from one executor to another. The higher the possibility of common vulnerabilities among executors with high similarity, the higher the possibility of malicious exploitation by attackers. As a result, executors with high reliability and dissimilarity should be selected to the extent possible, resulting in a greater system security gain. The reliability and dissimilarity among executors mostly determine the upper-security limit of DHR architecture [14].

The current research on the heterogeneity of executors is mainly focused on the quantitative description of heterogeneity and the improvement of scheduling algorithms. Zhang et al. [15] proposed a quantification method for executive heterogeneity based on complexity and dissimilarity. Unlike the Shannon-Wiener index and Simpson index, this method can better distinguish the executors. Wu et al. [16] improved the DHR architecture by introducing an executor partitioning module based on the clustering idea and an improved scheduling algorithm. The improved architecture was analyzed using a probability-based security analysis method. It was also verified to have higher security in the presence of known or unknown vulnerabilities through simulation experiments. Liu et al. [17] proposed a more fine-grained similarity quantification method by quantifying the similarity of each heterogeneous component. The method simultaneously balances the dynamicity and reliability of the DHR system, combines the minimum similarity algorithm to reduce the average failure rate of the system, and improves the comprehensive performance of the system. Ma et al. [18] defined two metrics for the credibility and performance of the executors: the credibility attribute and performance weight attribute values, respectively. Executors with higher weight values are assigned higher scheduling possibilities. Lv et al. [19] proposed a negative feedback scheduling algorithm based on historical data, and the system uses the monitored data for non-consistency checking and non-repetition checking. The scheduling module analyzes this attack type based on the monitored historical data and selects different executors to serve according to the different attack types. Wu et al. [20] used the Jaccard distance to describe the heterogeneity between any two executors combined with the arbiter's history information to propose a random seed scheduling algorithm based on heterogeneity, quality of service, and history information, resulting in a better balance between dynamics, security, and quality of service. Shen et al. [21] defined the credibility and dissimilarity degrees of executors and proposed a scheduling algorithm based on this definition that is capable of adaptive controllers. The algorithm assigns different weights to the degree of dissimilarity between executors and performs polling cleaning periodically to achieve lower overhead and higher security.

In conclusion, this paper proposes a DHR executor selection algorithm based on historical credibility and dissimilarity clustering (HCDC). The executors are classified according to historical credibility and dissimilarity, such that the scheduling module can select executors with high credibility and dissimilarity as service executors to improve the system's defense against known or unknown attacks. The main contributions of this work are as follows.

(1) The conventional DHR architecture is improved by introducing the related definitions of executor historical credibility and dissimilarity, which characterize executor reliability and degree of heterogeneity. The executors are classified and filtered in the heterogeneous executor module based on historical credibility and dissimilarity.

(2) The executors are divided into multiple heterogeneous executor pools with large heterogeneity using the k-means algorithm. The executors with the largest historical credibility are selected from each pool as the candidate pool, with historical credibility dynamically updated by negative feedback control based on the results of the multi-mode adjudicator. The executors are randomly selected from the candidate

pool as the set of service executors.

(3) The effectiveness of this algorithm is evaluated through theoretical analysis and simulation experiments, and the attack success and average failure rates of this algorithm are compared with those of existing algorithms.

The rest of the paper is structured as follows. Section 2 presents the DHR architecture based on historical credibility and clustering algorithms. Section 3 describes in detail the DHR executor selection algorithm based on HCDC. Section 4 evaluates the security of this algorithm through experiments. Section 5 concludes the study and presents avenues for future work.

2 Improved DHR Architecture

In order to increase the reliability and heterogeneity of the scheduling algorithm to select service executors and improve the system's defense against known or unknown attacks, we improve the traditional DHR architecture, as shown in Figure 1. The architecture is mainly divided into input, service executor, output, feedback scheduling policy, and heterogeneous executor modules.

(1) Input/output module. The input agent in the input module is responsible for copying the user input part into n copies and distributing them to the n heterogeneous service executors S_i, $i = 1, 2, 3, \ldots, n$. The adjudicator in the output module performs multi-mode adjudication on the local adjudication results output by each heterogeneous service executor S_i to obtain the global adjudication result, and sends the global adjudication result and historical credibility of the service executor to the negative feedback control. When the adjudication result exists or more than k results agree, the result is sent to the output; otherwise, blocking is performed.

(2) Service executor module. The set of service executors S consists of n executors selected by the dynamic scheduling algorithm from the set of heterogeneous executors E. The heterogeneous service executor S_i is responsible for processing the n sub-inputs sent from the input module and sending the respective processing results to the multi-mode adjudicator of the output module.

(3) Feedback scheduling policy. The feedback scheduling strategy consists of the negative feedback control and dynamic scheduling algorithm. Based on the global adjudication results sent by the multi-mode adjudicator, the negative feedback control periodically replaces the cleaning or adjusts the replacement of the heterogeneous service executor S_i, and sends the historical credibility of the executors to the dynamic scheduling algorithm. Based on the information sent by the negative feedback control, the dynamic scheduling algorithm randomly selects n executors from the candidate pool, ensuring that the selected executors have high reliability.

(4) Heterogeneous executor module. The heterogeneous executor module classifies the executors based on two metrics: historical credibility and dissimilarity. Firstly, the initial set of executors $\{E = E_j, j = 1, 2, 3, \ldots, m\}$ is formed by selecting a certain component from different sets of components according to different construction rules. Secondly, m heterogeneous executors are clustered according to executor dissimilarity values using the k-means algorithm to form heterogeneous executor pools R_p, $p \geqslant n$. The executors in different pools have the largest heterogeneity and contain the smallest common vulnerability. Finally, the historical credibility of each executor in each heterogeneous executor pool R_p is calculated. Further, an executor with the highest credibility is selected from each heterogeneous executor pool to form the executor candidate pool. If more than one executor exists with the highest credibility, anyone is selected to enter the candidate pool. From the above process, there are p executors waiting for selection in the candidate pool. The dynamic scheduling algorithm randomly selects n executors from these p executors to form the set service executors.

3 A DHR executor selection algorithm based on HCDC

3.1 Historical credibility

In the DHR architecture, the reliability of the service executor is a prerequisite for system reliability. Therefore, we use the historical credibility to characterize the vulnerability (i.e., reliability) of an executor as an important metric.

The historical credibility indicates the degree of trustworthiness of an executor. After forming p heterogeneous executor pools according to the clustering algorithm, the executor with the highest historical

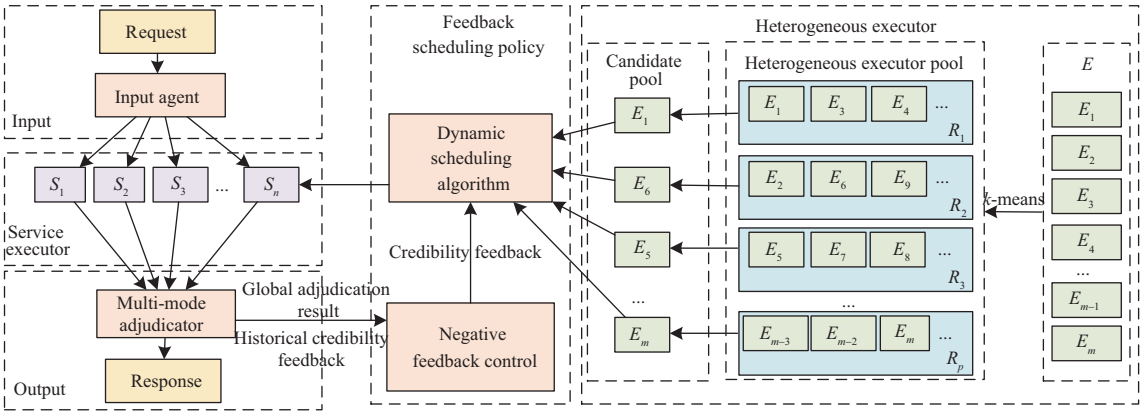

Figure 1 (Color online) The DHR architecture based on HCDC.

credibility in each heterogeneous pool is selected as the executor and will be chosen by the dynamic scheduling algorithm. The historical credibility of the executors is determined by the consistency of the local and global adjudication results. If the local adjudication result of the service executor i is the same as the global adjudication result at time t, then the service executor is considered to have high reliability and the invocation of the service executor can be increased appropriately. Conversely, if the local adjudication result of the service executor i is not the same as the global adjudication result at time t, then the service executor is assumed to have received an attack or an error has occurred at time t. The invocation of the service executor should be reduced and the service executor should be cleansed when necessary.

Let $y_i(t)$, $Y(t)$, and $\zeta_i(T)$ represent the local adjudication result of the server executor i at time t, the global adjudication result output by the multi-mode arbiter at time t, the number of times that executor i is selected as a service executor as of moment T, respectively. So the historical credibility $h_i(T)$ of service executor i as of time T is

$$h_i(T) = \frac{\sum_{t\in\zeta_i(T)} f\left(|Y(t) - y_i(t)|\right)}{|\zeta_i(T)|}, \tag{1}$$

where $f(\cdot)$ is defined as $f(x) = \begin{cases} 1, x = 0 \\ 0, x \neq 0 \end{cases}$. We denote the historical credibility metrics of all service executors in the set by S at moment t by $h(t) = [h_1(t), h_2(t), h_3(t), \ldots, h_n(t)]$. The average credibility $H(t)$ of each scheduling scheme is obtained by averaging the historical credibility of the scheduled executors, as shown in

$$H(t) = \sum_{i=1}^{n} h_i(t). \tag{2}$$

At time t, if executor i is not selected as a service executor, then its historical credibility has the same value as the previous time. Meanwhile, if it is selected as a service executor, then the dynamic update mechanism of the historical credibility is as shown in

$$h_i(t) = \begin{cases} \dfrac{|\zeta_i(t-1)|\, h_i(t-1) + f\left(|Y(t) - y_i(t)|\right)}{|\zeta_i(t-1)| + 1}, & i \in S, \\ h_i(t-1), & i \notin S, \end{cases} \tag{3}$$

where $h_i(0)$ denotes the initial credibility of service executor i and is set to 0.5. The update mechanism of executor credibility changes the historical credibility of executors in real-time based on negative feedback information. Therefore, the probability of selecting malicious executors during algorithm execution, and the influence of malicious executors on the global adjudication results reduce, and the correctness of the system adjudication impoves.

3.2 Dissimilarity

The greater the dissimilarity of the selected executors in the DHR architecture, the lower the risk of system failure. Therefore, we characterize the degree of heterogeneity among executors by the dissimilarity of executors, and then select executors with large dissimilarity as service executors whenever possible.

The dissimilarity of executors is reflected in the differences between components. Here, we mainly consider four levels of component differences, which are code, module, transport, and operation levels. The code level considers the differences caused by the programming language and key data structures, and it uses $c_i(t)$ to denote the dissimilarity of the code level components at time t. The module level considers the differences caused by semantic distance and real-time logic tasks, and it also uses $m_i(t)$ to denote the dissimilarity of module-level components at time t. The transport level considers the differences caused by the differences in data transfer protocols. It also uses $t_i(t)$ to denote the dissimilarity of the transfer level components at time t. The operation level considers the differences caused by operating systems, CPU architectures, and compilation methods, and it also uses $o_i(t)$ to denote the dissimilarity of the transport-level components at time t. The dissimilarity of the executor is determined by the dissimilarity of the components at these four levels. We define w_1, w_2, w_3, and w_4 as the weights of these four level components, respectively, which differ for different construction criteria and approaches. Therefore, the dissimilarity $d_j(t)$ of executor j can be expressed as

$$d_j(t) = w_1 c_j(t) + w_2 m_j(t) + w_3 t_j(t) + w_4 o_j(t). \tag{4}$$

Let $d(t) = [d_1(t), d_2(t), d_3(t), \ldots, d_m(t)]$ denote the dissimilarity index of all executors in the set of heterogeneous executors at time t.

3.3 HCDC algorithm

The HCDC algorithm first performs k-means clustering on the calculated dissimilarity $d(t)$ to divide the executors into different heterogeneous pools so that the executors between different pools have the greatest degree of dissimilarity and contain the least common vulnerability. Then the highest one from the heterogeneous pools is selected into the candidate pool based on the historical credibility $h_i(T)$ of the executors. Finally, the dynamic scheduling algorithm randomly selects executors from the candidate pool as the service executor. If the negative feedback control detects a decrease in the executor's credibility, then it assumes the executor has an unknown error or is under attack, and should be replaced using the dynamic scheduling algorithm. The credibility of the replaced executor is restored to its initial state to re-enter the heterogeneous pool of executors. In addition, to prevent the dynamic scheduling algorithm from selecting only a few specific executors with high reputation, this algorithm periodically cleans the service executors and recalculates the historical credibility and heterogeneity to avoid the algorithm from falling into local optimum. The details are shown in Algorithm 1.

Algorithm 1 A DHR executor selection algorithm based on HCDC

Require: The set of executors $E = \{E_1, E_2, E_3, \ldots, E_m\}$, the maximum number of heterogeneous executor pools p_{thresh}, n, $h(0)$, $c_j(t)$, $m_j(t)$, $t_j(t)$, $o_j(t)$.
Ensure: The number of heterogeneous executor pools p, the set of service executors S.
$S \leftarrow 0$, $\zeta_i(T) \leftarrow \emptyset$, $S(t-1) \leftarrow \phi$, w_1, w_2, w_3, w_4;
for t in $\{1, 2, 3, \ldots, Y\}$ **do**
  for i in $\{1, 2, 3, \ldots, n\}$ **do**
    if $i \in S(t-1)$ **then**
      $h_i(t) = \frac{|\zeta_i(t-1)|h_i(t-1)+f(|Y(t)-y_i(t)|=0)}{|\zeta_i(t-1)|+1}$;
    else
      $h_i(t)$=$h_i(t-1)$;
    end if
    if $h_i(t) < h_i(t-1)$ **then**
      $h_i(t) = h_i(0)$;
    end if
  end for
  for p in $\{3, 4, 5, \ldots, p_{\text{thresh}}\}$ **do**
    for j in $\{1, 2, 3, \ldots, m\}$ **do**
      Calculate $d_j(t) = w_1 c_j(t) + w_2 m_j(t) + w_3 t_j(t) + w_4 o_j(t)$;
      Calculate $||d_i(t) - d_j(t)||$;
      Obtain the number of heterogeneous executor pools by k-means algorithm;
      Calculate separately the $h_i(t)$ in the pool of p heterogeneous executors;
      Select the executor with the highest credibility to form the candidate pool;
    end for
  end for
  Randomly select n executors from the candidate pool to form S;
end for

Line 1 of Algorithm 1 initializes the variables, including the set of service executors S, the number of times executor i is selected as a service executor $\zeta_i(T)$, and the construct weights w. Lines 3–12 dynamically update the historical credibility of the executors according to (3), and initialize the executor i with decreasing credibility to restore its credibility to $h_i(0)$. Lines 13–17 calculate the dissimilarity of executors based on (4) and perform k-means clustering based on the dissimilarity to obtain p executor pools so that the pool of different executors contains as few common vulnerabilities as possible. Lines 18 and 19 select the highest executors in the heterogeneous pool into the candidate pool based on the credibility calculated in lines 3–12. Line 22 executes a dynamic scheduling algorithm to randomly select n executors to form the set of service executors S. The time complexity of the algorithm in updating the historical confidence of the executor is $O(n \times Y)$, where Y is the set of times. The time complexity of the algorithm in selecting the candidate pool based on the dissimilarity and historical confidence is $O(p_{\text{thresh}} \times m \times Y)$. Thus, the overall time complexity of the algorithm is $O(p_{\text{thresh}} \times m \times Y)$.

3.4 Theoretical analysis

The conventional DHR architecture selects heterogeneous executors using the random scheduling (RS) algorithm, which selects executors indiscriminately without considering heterogeneity or other metrics across executors. Ref. [22] proposed two dissimilarity executor selection algorithms based on the idea of fault tolerance, namely the maximum dissimilarity (MD) algorithm and the optimal mean dissimilarity (OMD) algorithm. The MD algorithm always selects the executor with the longest dissimilarity distance as the service executor, and the OMD algorithm always selects the executor with the best mean dissimilarity distance as the service executor. Ref. [17] proposed a random seed & minimum similarity (RSMS) algorithm based on randomly selected seed executors to exclude executors whose similarity exceeds a threshold value to constitute the final scheduling scheme. In this subsection, we will analyze and compare the HCDC algorithm with the RS algorithm, OM algorithm, OMD algorithm, and RSMS algorithm theoretically in terms of both system attack success rate and system average failure rate.

3.4.1 *System attack success rate*

The system attack success rate [17] reflects the probability that the attacker launches ε attacks on the system and succeeds. The attacker is assumed to be attacking based on a priori information about the system, and the attack without a priori information cannot be successful. The more a priori information is available, the higher the probability of a successful attack. Therefore, it can be accumulated if the attacker obtains the same information as a priori information for each probe; otherwise, a priori information fails. For simplicity of analysis, it is assumed that the attacker can only probe once in one task execution cycle. Thus, when the system information at time t is consistent with the a priori information, and the attacker has obtained ε valid probes before the attack, the probability of the attacker successfully attacking is

$$P_{\varepsilon}(t) = A\left(1 - \mathrm{e}^{-\varepsilon}\right),\ 0 < A < 1. \tag{5}$$

In (5), as the number of effective probes ε increases, the probability of attack success increases and converges to A. The success of the attack based on ε probes indicates that the a priori information of the current system is consistent with the last probe, but the first $\varepsilon - 1$ probes are not certain. Therefore, the attack success rate is

$$P_{\varepsilon} = \sum_{l=0}^{\varepsilon-1} \left(1 - \frac{1}{\Gamma}\right) \frac{1}{\Gamma^{l}} A\left(1 - \mathrm{e}^{-l}\right) + \frac{1}{\Gamma^{\varepsilon}} A\left(1 - \mathrm{e}^{-\varepsilon}\right),\ 0 < A < 1, \tag{6}$$

where Γ is the average period of system execution of tasks (i.e., it is the average period of the scheduling scheme). The average period varies for different scheduling schemes.

Based on security and reliability, the value of k for multi-modal adjudication cannot be less than 3; otherwise, it will not function as a unanimous vote. Thus, the value range of k is $3 \leqslant k \leqslant n$. The range of execution residuals is $(3, 4, 5, \ldots, s)$, $3 < s < n$. For the RS algorithm, no difference exists among all executors and the probability of being selected is the same, so the Γ_{RS} of the RS algorithm is equal to the sum of the number of all scheduling schemes. Therefore, $\Gamma_{\text{RS}} = \sum_{n=3}^{s} C_m^n = \sum_{n=3}^{s} \frac{m!}{n!(m-n)!}$. The HCDC algorithm is also RS, with the difference that the selection from m executors is narrowed down to the selection from p executors with greater heterogeneity and higher credibility, so the range of execution

Table 1 System attack success rate under different algorithms

Algorithm	System attack success rate
RS	$P_{\mathrm{RS}}=\sum_{l=0}^{\varepsilon-1}\left(1-\frac{1}{\Gamma_{\mathrm{RS}}}\right)\frac{1}{{\Gamma_{\mathrm{RS}}}^{l}}A(1-\mathrm{e}^{-l})+\frac{1}{{\Gamma_{\mathrm{RS}}}^{\varepsilon}}A\left(1-\mathrm{e}^{-\varepsilon}\right)$
MD	$P_{\mathrm{MD}}=A\left(1-\mathrm{e}^{-\varepsilon}\right)$
OMD	$P_{\mathrm{OMD}}=A\left(1-\mathrm{e}^{-\varepsilon}\right)$
RSMS	$P_{\mathrm{RSMS}}=\sum_{l=0}^{\varepsilon-1}\left(1-\frac{1}{\Gamma_{\mathrm{RSMS}}}\right)\frac{1}{{\Gamma_{\mathrm{RSMS}}}^{l}}A(1-\mathrm{e}^{-l})+\frac{1}{{\Gamma_{\mathrm{RSMS}}}^{\varepsilon}}A(1-\mathrm{e}^{-\varepsilon})$
HCDC	$P_{\mathrm{HCDC}}=\sum_{l=0}^{\varepsilon-1}\left(1-\frac{1}{\Gamma_{\mathrm{HCDC}}}\right)\frac{1}{{\Gamma_{\mathrm{HCDC}}}^{l}}A(1-\mathrm{e}^{-l})+\frac{1}{{\Gamma_{\mathrm{HCDC}}}^{\varepsilon}}A(1-\mathrm{e}^{-\varepsilon})$

residuals is $(3,4,5,\ldots,p)$, resulting in an average period of $\Gamma_{\mathrm{HCDC}}=\sum_{n=3}^{p}C_m^p=\sum_{n=3}^{p}\frac{m!}{n!(m-n)!}$. The scheduling scheme of the MD algorithm and the OMD algorithm does not change dynamically after it is determined, so the average period is 1. The average period of the RSMS algorithm is $\Gamma_{\mathrm{RSMS}}=\sum_{n=3}^{s}\sum_{f=0}^{n-1}\frac{m}{f=1}C_{n-1}^{f}(P_r)^f(1-P_r)^{n-f-1}$, where $P_r=\frac{C_{m-2}^{n-2}}{C_{m-1}^{n-1}}$ denotes the probability that the scheduling scheme f_d is greater than the other scheduling schemes containing seeds, and f denotes the scheduling scheme f_d in which there are f executors as seed executors in addition to the seed executor P_s is the same as f_d [17]. The system attack success rate under different algorithms is shown in Table 1.

The average scheduling period of the HCDC algorithm is smaller than that of the RS algorithm, which is due to the fact that the HCDC algorithm selects the executors based on the historical credibility and dissimilarity before performing the dynamic scheduling algorithm. As a result, the HCDC algorithm significantly improves the system's security. the MD and OMD algorithms are not dynamic and the system is at risk of going down if an attacker is successful.

3.4.2 *System average failure rate*

The average system failure rate is reflected in the probability of not executing the service correctly when the system is under attack or when an error occurs. The system normal working rate is expressed as the probability that the system obtains the correct output after performing the task. The sum of the system normal working and the system average failure rates is one. Thus, the system reliability is characterized by the system average failure rate. The historical credibility of the executors is a prerequisite for the system reliability in the DHR architecture. Further, the average credibility of the scheduling scheme determines the system average failure rate. The dynamic scheduling algorithm elects different executors as service executors, which will result in different average credibility values $H(t)$, and the higher $H(t)$, the higher the reliability of the system. Assuming that at time t, the dynamic scheduling algorithm selects n executors as service executors from p executors (i.e., the system comprises n residual degrees), there are a total of C_p^n scheduling schemes. Therefore, the set of scheduling schemes satisfying the rule is $F=\{f_1^n,f_2^n,f_3^n,\ldots,f_x^n,\ldots\}$, $1\leqslant x\leqslant C_p^n$, where $f_x^n=\{E_{1x},E_{2x},E_{3x},\ldots,E_{jx}\}$.

It is assumed that the causes of failure of heterogeneous executors are random failures and malicious attacks. When more than k results exist in the multi-mode adjudicator that are identical, the result is considered the system output. If all executors cannot achieve more than k identical results, the system outputs an exception. When an exception occurs, the system does not consider that the executor fails, but re-executes the task. In this paper, we assume that the probability of failure of a heterogeneous executor occurring is distributed from 0 to 1. The failure rate vector is $L=[\alpha_1,\alpha_2,\alpha_3,\ldots,\alpha_m]$, where α_j is the failure rate of the executor E_j. So the failure rate vector of the xth scheduling scheme is $L_x=[\alpha_{1x},\alpha_{2x},\alpha_{3x},\ldots,\alpha_{mx}]$. To make the system output an error result, the attacker must make more than k executors output an error result. If the average credibility value of the overall scheduling scheme is higher, the lower the probability that the executors will make errors. Therefore, the probability of having k errors of the same degree with n residual degrees is related to the average credibility value as in

$$P_k|_{f_x^n}=B\big(H(t)|_{f_x^n}\big)^{-k},\tag{7}$$

where B is a constant, f_x^n denotes the selection of the xth scheduling solution as the final scheduling solution, and $H(t)|_{f_x^n}$ denotes the average credibility of choosing the f_x^n scheduling scheme at time t. From (7), it can be seen that the probability of k identical errors with the same n is negatively correlated with the average credibility of the scheduling scheme, which is because the higher the average credibility, the lower the probability of errors in the executor and the lower the possibility of identical errors.

Thus, the probability that the executor will output at least k error results given that the scheduling

Table 2 System average failure rate under different algorithms

Algorithm	System attack success rate
RS	$P=\frac{1}{s-2}\sum\limits_{n=3}^{s}\left(\frac{1}{C_m^n}\sum\limits_{f_x^n}P_{\geqslant k}\vert_{f_x^n}\right)$
MD	$P=\frac{1}{s-2}\sum\limits_{n=3}^{s}P_{\geqslant k}\vert_{\min(H(t)\vert_{f_x^n})}$
OMD	$P=\frac{1}{s-2}\sum\limits_{n=3}^{s}P_{\geqslant k}\vert_{\min[\sigma(\forall h_{ij}\vert_{h_{ij}\in f_x^n})]}$, h_{ij} is the similarity between executors E_i and E_j
RSMS	$P=\frac{1}{s-2}\sum\limits_{n=3}^{s}\left(\frac{1}{m}\sum\limits_{f_x^n\&P_s\in f_x^n}P_{\geqslant k}\vert_{\min(H(t)\vert_{f_x^n})}\right)$, P_s is the seed executor
HCDC	$P=\frac{1}{s-2}\sum\limits_{n=3}^{s}\left(\frac{1}{C_p^n}\sum\limits_{f_x^n}P_{\geqslant k}\vert_{f_x^n}\right)$

scheme x is determined is

$$P_{\geqslant k}|_{f_x^n}=\sum_{y=k}^{n}\left\{B\big(H(t)|_{f_x^n}\big)^{-y}\sum_{h=y}^{n}\sum_{i_1=1}^{n-h+1}\sum_{i_2=2}^{n-h+2}\cdots\sum_{i_h=h}^{n}\left[\prod_{j=1}^{h}\alpha_{i_jx}\prod_{z\neq i_1,i_2,i_3,\ldots}(1-\alpha_{zx})\right]\right\}. \tag{8}$$

From Subsection 3.4.1, the range of execution residuals is $(3,4,5,\ldots,s)$, so there are a total of $s-3+1=s-2$ types of execution residuals, and there are C_p^n types of scheduling schemes under n residuals. Therefore, the average failure rate of the system is

$$P_{\mathrm{HCDC}}=\frac{1}{s-2}\sum_{n=3}^{s}\left(\frac{1}{C_p^n}\sum_{f_x^n}P_{\geqslant k}|_{f_x^n}\right). \tag{9}$$

This provides the system average failure rate under different algorithms, as shown in Table 2.

As can be seen from Table 2, due to the selection of executors based on historical confidence and phase dissimilarity before performing the dynamic scheduling algorithm, i.e., $p<m$, the average failure rate of the HCDC algorithm is smaller than that of the RS algorithm. This is also confirmed at the experimental level from Subsection 4.2.5. In addition, Subsection 4.2.5 provides a detailed comparison of the five algorithms.

4 Experiment and analysis

In this section, the performance of the algorithm is verified by simulation experiments in terms of heterogeneous executor clustering, average credibility of service executors, and executor selection. The HCDC algorithm is analyzed in comparison with the RS, MD, and OMD algorithms in terms of both system attack success and average failure rates.

4.1 Simulation experiment related settings

The parameters in the simulation experiments are set as follows: the number of executors in the pool of executors is assumed to be $m=70$, only the executors entering the candidate pool enter the ready state, and the remainder of the executors are in the standby state. The number of heterogeneous pools of executors for clustering is set to $p=5$, $p=6$, and $p=7$, respectively. The number of service executors in S can also be considered as the redundancy of the system. Ref. [15] found that the gain of the system is generally greatest when the redundancy of the system is $n=3$. Therefore, we only consider the cases of $n=3$, $n=4$, and $n=5$. Regarding the historical credibility setting, an initial credibility level of 0.5 is assumed. For the setting of the execution dissimilarity, the same security and performance at the module level are also assumed. The remaining weights w_1, w_3, and w_4 at the code level, transport level, and operation level are 1/3, 1/3, and 1/3, respectively. The specific dissimilarity refers to the settings in Table 3.

According to the settings in Table 3, 70 heterogeneous executors are randomly selected from different levels to form the initial set of executors E. The adjudicator uses k-mode adjudication, i.e., when the adjudication result exists or more than k results agree, the result is output; otherwise, blocking is performed. All parameters are set as in Table 4.

Table 3 Settings of the dissimilarity index

Code level		Transport level		Operation level	
Code type	Value	Transmission protocol type	Value	Platform type	Value
Java	0.2	TCP/IP	0.2	VMware Ubuntu	0.2
C++	0.4	IPX/SPX	0.35	Virtualbox Ubuntu	0.35
Python	0.6	NatBRUI	0.5	VMware Centos	0.44
C#	0.8	JSON	0.65	Virtualbox Centos	0.56
		Binary	0.8	VMware Windows	0.68
				Virtualbox Windows	0.8

Table 4 Parameter settings

Property	Values
Number of actuators m	70
Number of heterogeneous pools p	5,6,7
System redundancy n	3,4,5
Initial credibility level $h_i(0)$	0.5
Different level weights w	1/3

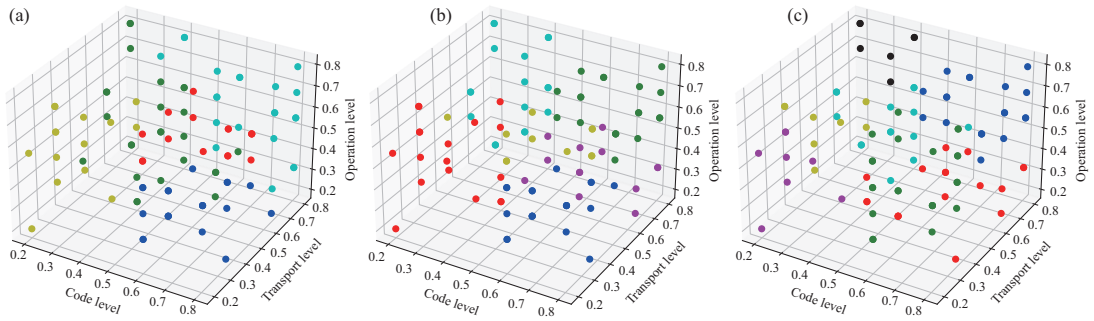

Figure 2 (Color online) Executor clustering results. (a) Clustered into 5 categories; (b) clustered into 6 categories; (c) clustered into 7 categories.

4.2 Analysis of the simulation results

4.2.1 *Heterogeneous executor clustering*

The HCDC algorithm performs k-means clustering to divide the executors into different heterogeneous executor pools based on the executor dissimilarity values, with the executors in each heterogeneous executor pool having less dissimilarity and being more likely to contain the same vulnerability. Conversely, executors in different heterogeneous executor pools have a larger heterogeneity and are less likely to contain the same vulnerability. Selecting executors from different heterogeneous executor pools ensures that the selected executors are more dissimilar to each other and contain fewer identical vulnerabilities. Therefore, the probability of successful attacks by attackers is reduced, and the security of the system further improves. The clustering results are shown in Figure 2.

Figures 2(a)–(c) show the clustering results for the number of heterogeneous executor pools $p = 5$, $p = 6$, and $p = 7$, respectively. The axes in Figure 2 show the dissimilarity values at the code level, transport level, and operation level of the executors, respectively. Evidently, the k-means algorithm can effectively distinguish the executors and classify them into different categories based on the dissimilarity values among the executors.

4.2.2 *Average credibility $H(t)$*

The dynamic scheduling algorithm selects the executors to construct S, which processes the input for subsequent computations, so the trustworthiness of the selected executors is critical. The average credibility $H(t)$ is determined by the executors selected by the scheduling algorithm and reflects the degree of executor vulnerability. The higher the average credibility of the scheduling scheme at each moment, the higher the system reliability. We define here $p = 5$, $n = 3$, and assume that an attacker can perform only one attack in a scheduling cycle. We observe the change in the average credibility level when the

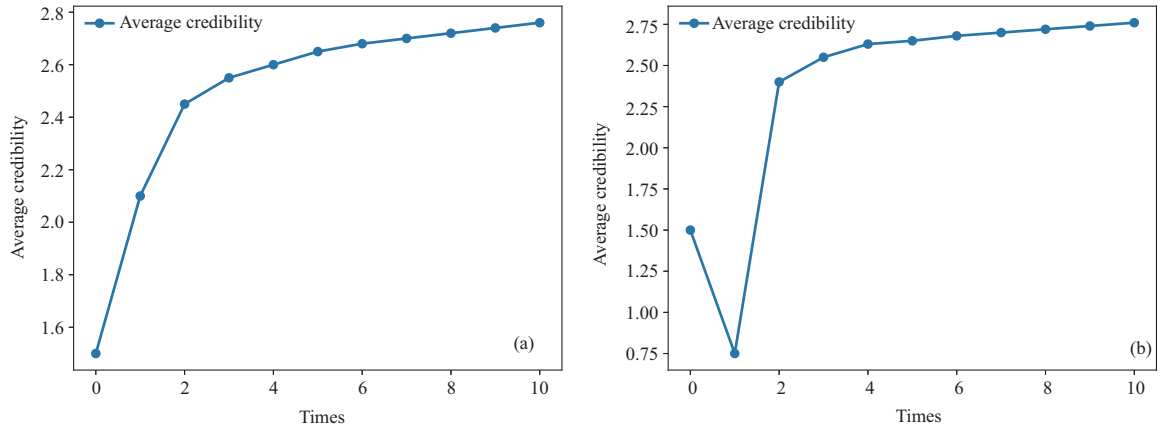

Figure 3 (Color online) Change of the average credibility when (a) not attacked and (b) attacked.

Table 5 Average credibility under different settings

	$p=5$	$p=6$	$p=7$
$n=3$	2.826	2.873	2.932
$n=4$	3.414	3.631	3.721
$n=5$	3.864	4.213	4.431

system is not attacked and when it is attacked as the times of scheduling increase, as shown in Figure 3(a) and (b).

As shown in Figure 3(a), the average credibility tends to increase and gradually stabilizes as the times of system scheduling increase when the system is not under attack. This is because the HCDC algorithm continuously increases the historical credibility of the scheduled executors according to the dynamic update mechanism of (3) when the executors are not under attack. As the times of scheduling increase, the system will schedule executors with high credibility as much as possible, resulting in an increase in average credibility. The credibility of each executor in the candidate pool increases and stabilizes in the later phases of system scheduling since it has not been attacked, so the average credibility will stabilize.

As shown in Figure 3(b), if the system is attacked at the first scheduling, the historical credibility of the attacked executors decreases according to the dynamic update mechanism of (3), so that the average credibility is significantly lower or even lower than the initial value. At this point, the negative feedback control replaces the attacked executors based on the historical credibility feedback from the adjudicator, and cleans all service executors if necessary. The replaced executors are restored to the initial state and returned to the heterogeneous executor pool. Evidently, the average credibility resumes its upward trend after the system is dynamically adjusted to recover.

We conducted 100 simulation experiments on the HCDC algorithm with a different number of heterogeneous executor pools and redundancy to observe the average credibility values after 10 scheduling iterations, and the results are shown in Table 5.

As presented in Table 5, with constant redundancy n, the average credibility $H(t)$ increases as the number of heterogeneous executor pools p increases. This is due to an increase in the number of heterogeneous pools with constant redundancy, resulting in an increase in the available scheduling options for the dynamic scheduling algorithm. When a problem occurs in one executor it is also possible to select the executor with higher credibility from the remaining executors for replacement in time. The average credibility varies due to different settings of the number of heterogeneous executor pools and system redundancy, but they can all maintain a high state. This means that the HCDC algorithm selects executions with high trustworthiness and low vulnerability, which further guarantees the reliability of the system.

4.2.3 *Executor selection*

To verify the effectiveness of the HCDC algorithm, this subsection simulates the change of the executor historical credibility after being attacked during executor selection. We define the heterogeneous executor pool $p=5$ and the number of service executors $n=3$. Assume that the executors may have two types

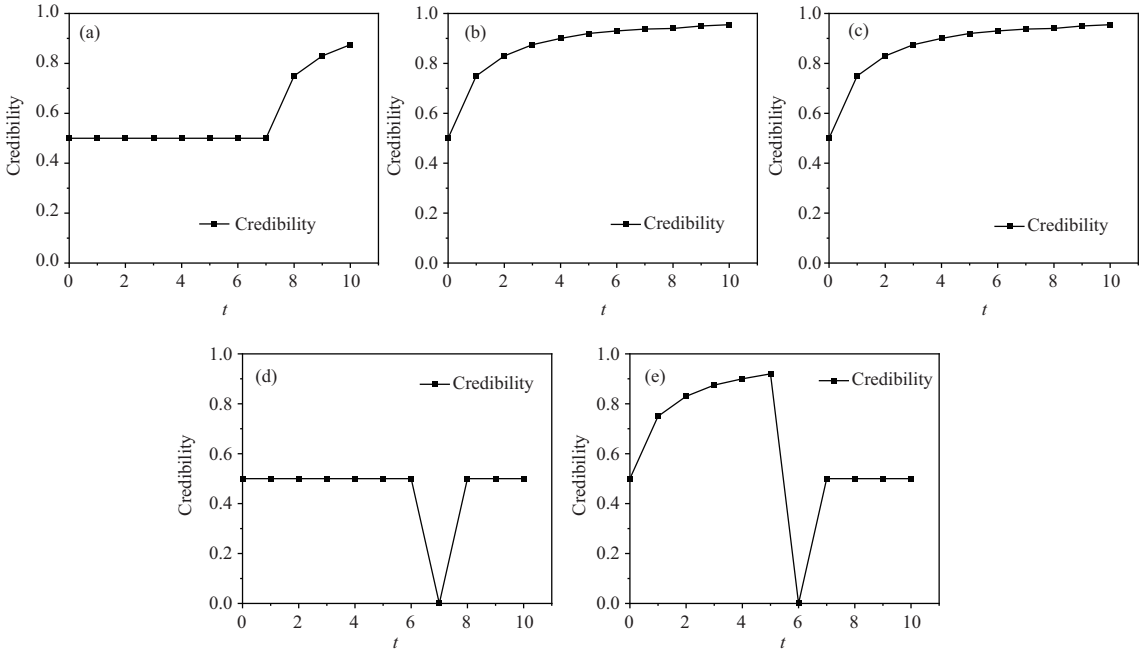

Figure 4 Credibility trend of heterogeneous executors over time. (a) E_0; (b) E_1; (c) E_2; (d) E_3; (e) E_6.

of vulnerabilities v_1 and v_2, both may exist, and the attacker can launch an attack on the executors by using the above two vulnerabilities. Therefore, there are four types of executors $\{s_0, s_1, s_2, s_3\}$, where s_0 means the executor does not have vulnerabilities v_1 and v_2, s_1 means the executor has vulnerabilities v_1, s_2 means the executor has vulnerabilities v_2, and s_3 means the executor has both vulnerabilities v_1 and v_2.

After dissimilarity clustering and credibility, E_2, E_1, E_6, E_3, and E_0 are selected to form the candidate pool, where E_6 contains vulnerability v_2, E_0 contains vulnerability v_1, E_3 contains both vulnerability v_1 and v_2, and E_2 and E_1 have neither vulnerability v_1 nor vulnerability v_2, so the types of executors in the waiting pool are $\{s_0, s_0, s_2, s_3, s_1\}$. The dynamic RS algorithm selects E_2, E_1, and E_6 as the initial service executors. In the first 5 time steps, the attacker does not launch an attack against vulnerabilities v_1 and v_2, the system is in a stable and secure state, and the historical credibility of E_2, E_1, and E_6 gradually increases. At the 6th time step, the attacker uses vulnerability v_2 to attack. The local adjudication result of E_6 is inconsistent with the global adjudication result, and the historical credibility drops to 0. The negative feedback control replaces it for cleaning. At the 7th time step, the dynamic scheduling algorithm selects E_3 to enter the service executor module based on the negative feedback information. The attacker will continue to select vulnerability v_2 for the attack based on the a priori information. Executor E_3 contains both vulnerabilities v_1 and v_2, so the local verdict result of E_3 is inconsistent with the global verdict result, and the historical credibility drops to 0. The negative feedback control performs replacement cleaning on it. At the 8th time step, the dynamic scheduling algorithm places E_0 into the service executor module based on the negative feedback information, and the attacker continues to select vulnerability v_2 for an attack. Executor E_0 does not contain vulnerability v_2, and the attacker fails. The historical credibility of E_2, E_2, and E_0 increases, and the system restores stability. At the 9th and 10th time steps, the attacker continues to exploit vulnerability v_2, but the system has an effective defense. The trend of the historical credibility of the five executors over time is shown in Figure 4.

As shown in Figure 4, an attack on the system can manifest itself as a decrease in the historical credibility of the executors. The attacked executors will be replaced and cleaned by the negative feedback module. The cleaned executors will return to the heterogeneous pool and wait to be scheduled. The system's dynamic scheduling algorithm always selects executors with higher credibility and greater dissimilarity to enter the service executor module.

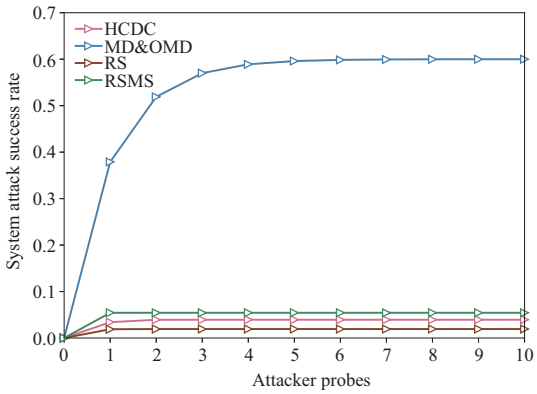

Figure 5 (Color online) Relationship between the attack success rate and number of probes under different algorithms.

Table 6 The average failure rate of the five algorithms with the same failure rate

Algorithm	$n = 3$	$n = 4$	$n = 5$
RS	2.2×10^{-3}	8.0227×10^{-5}	1.8134×10^{-4}
RSMS	7.1909×10^{-4}	2.3656×10^{-5}	9.2103×10^{-5}
MD	3.6751×10^{-4}	1.3350×10^{-5}	6.4437×10^{-5}
OMD	1.3×10^{-3}	8.4503×10^{-5}	1.6772×10^{-4}
HCDC	1.9737×10^{-4}	2.5055×10^{-5}	1.7121×10^{-4}

4.2.4 *System attack success rate*

Subsection 3.4.1 analyzes and compares the attack success rate of the RS, MD, OMD, and HCDC algorithms from the theoretical level, which is calculated as shown in Table 1. In this subsection, the five algorithms are compared at the experimental level, and the relationship between the attack success rate of the four algorithms and the number of attacker probes is simulated, as shown in Figure 5. Here, we define $A = 0.6$, $n = 3$, $p = 5$ in the HCDC algorithm and $s = 3$ in the RSMS algorithm.

As can be seen from Figure 5, the scheduling scheme of MD and OMD algorithms does not change dynamically once determined, and dynamism does not exist. Thus, the attack success rate is higher, and the system faces the risk of being paralyzed once the attacker succeeds. The HCDC algorithm that introduces historical credibility and dissimilarity clustering makes the attacker's attack success rate smaller than that of the MD, OMD, and RSMS algorithms, and only slightly higher than that of the RS algorithm. This is because the scheduling period is a key factor in determining the attack success rate. Further, the HCDC algorithm can select fewer heterogeneous executors than the RS algorithm, and the average scheduling period is smaller than that of the RS algorithm. The scheduling period reflects the dynamics of the system. While having the lowest dynamics, the RS algorithm has a high system uncontrollability. The HCDC algorithm is less dynamic than the RS algorithm. However, the executors selected based on historical credibility and dissimilarity contain fewer common vulnerabilities, making the system more controllable and reliable.

4.2.5 *System average failure rate*

Subsection 3.4.2 analyzes and compares the average failure rate of the RS, MD, OMD, and HCDC algorithms at the theoretical level, which is presented in Table 2. This subsection compares the average failure rate of the four algorithms at different redundancy levels from the experimental level. The system average failure rate is inversely proportional to the reliability of the system; that is, the lower the average failure rate, the higher the reliability of the system. Here, we define $B = 2$, $p = 9$, and $m = 9$ for the RS, MD, and OMD algorithms. A total of 100 simulation experiments were conducted for the five algorithms, and their average failure rates were calculated. First, we consider that the failure rate α_j of each heterogeneous executor is the same, which is 0.1, i.e., the failure rate vector $L = [0.1, 0.1, 0.1, \ldots, 0.1]$, and the results are shown in Table 6.

As can be seen from Table 6, the system average failure rate of the HCDC algorithm is lower than that of the RS and RSMS algorithms. This is because, if the executors selected by the RS algorithm have the same vulnerability among to be exploited by the attacker, the attacker will likely not spend

Table 7 Average failure rates of the five algorithms with different failure rates

Algorithm	$n = 3$	$n = 4$	$n = 5$
RS	7.6647×10^{-4}	1.5152×10^{-5}	3.5186×10^{-5}
RSMS	2.7664×10^{-4}	3.8064×10^{-6}	2.5745×10^{-5}
MD	1.1419×10^{-4}	8.8765×10^{-7}	6.7820×10^{-6}
OMD	3.5989×10^{-4}	1.0583×10^{-5}	1.1182×10^{-5}
HCDC	1.7218×10^{-5}	2.1828×10^{-6}	1.4819×10^{-5}

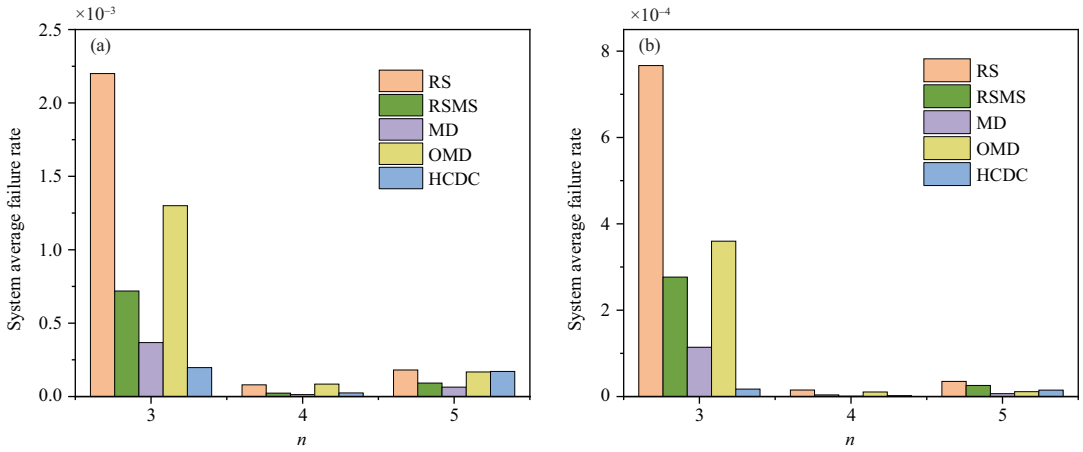

Figure 6 (Color online) Average failure rate of five algorithms with different failure rate vectors. (a) Same failure rate vectors; (b) different failure rate vectors.

a large amount of money to breach the system and make it fail. The RSMS algorithm only considers the heterogeneity between executors and does not consider their vulnerability. The average failure rate of the HCDC algorithm is slightly higher than that of the MD and OMD algorithms in some instances. However, the scheduling scheme of the MD and OMD algorithms is fixed and the attack success rate is much higher than that of the HCDC algorithm as discussed in Subsection 4.2.4.

We also consider the case where the other parameters remain unchanged and the failure rates α_j of the heterogeneous actuators differ. Setting the failure rate α_j in a random uniform distribution between (0, 0.1), the results are shown in Table 7.

As presented in Table 7, when the failure rates of the executors are not equal, the results still have a similar pattern. The system average failure rate of the HCDC algorithm is lower than that of RS and RSMS algorithms, and slightly higher than that of MD and OMD algorithms under certain redundancy degrees. The system average failure rates of the five algorithms are compared for different actuator failure rates, as shown in Figure 6. Evidently, the HCDC algorithm always ensures high reliability of the system.

5 Conclusion

A DHR executor selection architecture based on HCDC was proposed to enhance the conventional DHR architecture and improve the system's defense against known or unidentified threats. First, the executors were divided into multiple heterogeneous pools with large heterogeneity based on the k-means algorithm. Second, the executors with the highest historical credibility were selected from each executor pool as the candidate pool. Finally, the executors were randomly selected from the candidate pool as the set of service executors. The algorithm selects executors with higher credibility and greater dissimilarity as service executors to the extent possible. This minimizes the probability that the set of service executors contains the same error and improves the security and reliability of the DHR system. Theoretical analysis and simulation experiments illustrate that our algorithm has good performance in both attack success rate and average failure rate.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 62076139, 62176264), Natural Science Foundation of Jiangsu Province (Higher Education Institutions) (Grant Nos. BK20170900, 19KJB520046, 20KJA520001), Innovative and Entrepreneurial Talents Projects of Jiangsu Province, Jiangsu Planned Projects for Postdoctoral Research Funds (Grant No. 2019K024), Six Talent Peak Projects in Jiangsu Province (Grant No. JY02), Postgraduate Research &

Practice Innovation Program of Jiangsu Province (Grant Nos. KYCX19_0921, KYCX19_0906), Open Research Project of Zhejiang Lab (Grant No. 2021KF0AB05), and NUPT DingShan Scholar Project and NUPTSF (Grant No. NY219132).

References

1 Wu Z, Wei J. Heterogeneous executors scheduling algorithm for mimic defense systems. In: Proceedings of the 2nd International Conference on Computer and Communication Engineering Technology (CCET), 2019. 279–284
2 Song K, Liu Q R, Wei S, et al. Endogenous security architecture of Ethernet switch based on mimic defense. J Commun, 2020, 41: 18–26
3 Ren Q, Wu J X, He L. Performance modeling based on GSPN for cyberspace mimic DNS. Chin J Electron, 2020, 29: 738–749
4 Wu J X. Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats. Berlin: Springer Science & Business Media, 2011
5 Wu J X. Research on cyber mimic defense. J Cyber Secur, 2016, 1: 1–10
6 Wu J X. Endogenous Safety and Security in Cyberspace: Mimic Defense and Generalized Robust Control. Beijing: Science Press, 2020
7 Wu J X. Development paradigms of cyberspace endogenous safety and security. Sci China Inf Sci, 2022, 65: 156301
8 Wu J X. Cyberspace endogenous safety and security. Engineering, 2022, 15: 179–185
9 Hu H C, Chen F C, Wang Z P. Performance evaluations on DHR for cyberspace mimic defense. J Cyber Secur, 2016, 1: 1–10
10 Tong Q, Zhang Z, Zhang W H, et al. Design and implementation of mimic defense Web server. J softw, 2017, 28: 883-0897
11 Tong Q, Guo Y F. A comprehensive evaluation of diversity systems based on mimic defense. Sci China Inf Sci, 2021, 64: 229304
12 Zhang Z, Ma B L, Wu J X. The test and analysis of prototype of mimic defense in Web servers. J Cyber Secur, 2017, 2: 13–28
13 Zhu Z B, Liu Q R, Liu D P, et al. Research progress of mimic multi-executive scheduling algorithms. J Commun, 2021, 42: 179–190
14 Zhang J X, Pang J M, Zhang Z, et al. Heterogeneity quantization method of cyberspace security system based on dissimilar redundancy structure. J Electron Inf Technol, 2019, 41: 1594–1600
15 Zhang J X, Pang J M, Zhang Z. Quantification method for heterogeneity on Web server with mimic construction. J Softw, 2020, 31: 564–577
16 Wu T, Hu C N, Chen Q N, et al, Defense-enhanced dynamic heterogeneous redundancy architecture based on execution partition. J Commun, 2021, 42: 122–134
17 Liu Q R, Lin S J, Gu Z Y. Heterogeneous redundancies scheduling algorithm for mimic security defense. J Commun, 2018, 39: 188–198
18 Ma H L, Yi P, Jiang Y M, et al. Dynamic heterogeneous redundancy based router architecture with mimic defenses. J Cyber Secur, 2017, 2: 29–42
19 Lv Y Y, Guo Y F, Wang Z P, et al. Negative feedback scheduling algorithm based on historical information in SDN. Chinese J Netw Inf Secur, 2018, 4: 45–51
20 Wu Z Q, Wei J. Heterogeneous executors scheduling algorithm for mimic defense systems. In: Proceedings of the 2nd International Conference on Computer and Communication Engineering, Piscataway, 2019. 279–284
21 Shen C Q, Chen S X, Wu C M, et al. Adaptive mimic defensive controller framework based on reputation and dissimilarity. J Commun, 2018, 39: 173–180
22 Yao W B, Yang X Z. Design of selective algorithm for diverse software components. J Harbin Inst Technol, 2003, 35: 261–264