IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 8, 2021, accepted August 25, 2021, date of publication September 10, 2021,
date of current version September 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3111735

An Optimal Seed Scheduling Strategy Algorithm Applied to Cyberspace Mimic Defense

ZHUOXING CHEN[1], YIQIN LU[1], (Member, IEEE), JIANCHENG QIN[1], AND ZHE CHENG[2]
[1]School of Electronic and Information Engineering, South China University of Technology (SCUT), Guangzhou 510640, China
[2]School of Computer Science and Engineering, South China University of Technology (SCUT), Guangzhou 510640, China

Corresponding author: Yiqin Lu (eeyqlu@scut.edu.cn)

This work was supported in part by the Key-Area Research and Development Program of Guangdong Province, China, under Grant 2019B010137001, Grant 2018B010113001, and Grant 2020B0101120002; and in part by the National Key Research and Development Program of China under Grant 2020YFB1805300.

ABSTRACT Cyberspace mimic defense (CMD) is an active defense theory that has emerged in recent years. By dynamically constructing and scheduling multiple executors, the CMD can not only effectively defend against security threats caused by unknown cyberspace weaknesses, but also improve the present situation of information asymmetry between attackers and defenders in cyberspace. However, as one of the key technologies of the CMD, the scheduling strategy algorithm still needs to be improved in real-time security, reliability, and universality, which has restricted the development and large-scale deployment of the CMD. To solve this problem, we propose an optimal seed scheduling strategy algorithm (OSSSA) in this paper. After using continuous-time Markov processes to mathematically analyze the model of the mimic defense system in cyberspace, we introduce the key factors and their evaluation methods that affecting the CMD defense performance in the OSSSA, then propose a reliable working mechanism of the OSSSA. Furthermore, we present an evaluation method that can effectively evaluate the defense performance of the scheduling algorithms. Experimental results from software simulations and real experiments show that the OSSSA has better real-time defense performance than the current mainstream scheduling strategy algorithms, and can adapt to different practical application scenarios, which is helpful to the further development of the CMD.

INDEX TERMS Cyberspace mimic defense, cyber security, feedback mechanism, scheduling algorithm.

I. INTRODUCTION

Our daily life has become more and more dependent on the Internet, which makes cyberspace one of the most important spaces for our social functions and activities [1]. Unfortunately, however, cyberspace is facing increasing security risks, which seriously affects the social order. For example, the power grid of Ukrainian was hacked in 2015 [2] and caused a widespread impact at that time. After that, many serious security incidents have occurred one after another, such as the billions of Yahoo user data were leaked in 2016 [3], the CPU of Intel has been found a serious underlying vulnerability in 2018 [4], the cyberattacks that forced a U.S. natural gas operator to shut down its compression facilities in 2020 [5], and the prevalence of ransomware such as WannaCry, Bad Rabbit, and GanCrab [6]. All these information security incidents have one thing in common: the information system or software itself has unknown weaknesses, including vulnerabilities, backdoors, viruses, and Trojan horses [7].

In cyber security, the traditional static defense technologies such as firewall [8], intrusion detection [9], intrusion prevention [10], vulnerability scanning [11] can hardly defend the cyberattacks that effectively utilize the unknown weaknesses of network components. To change the situation of information asymmetry between network attackers and defenders, moving target defense (MTD), as representative active defense technology, has become a hot research technology and received high expectation from scholars as well [12]–[14]. Briefly speaking, MTD introduces dynamics in order to increase the uncertainty of the network surface. It offsets the information advantage of cyberspace attackers by increasing their attack complexity and attack costs. However, with further research on MTD, the inherent problems

The associate editor coordinating the review of this manuscript and approving it for publication was Jiafeng Xie.

of this mechanism have gradually emerged: it is difficult for MTD to meet the performance requirements of defense coverage, MTD is timeliness and uncontrolled [15], and the frequent changes may have a certain impact on the system performance at the same time [16]. In addition, [17] also pointed out that MTD may have the problem of blindness, inefficiency, and unverifiability.

Facing the huge security risks caused by unknown vulnerabilities in cyberspace, as well as the low efficiency and insufficient security of MTD mentioned above, Chinese Academician Wu Jiangxing proposed an innovative active defense theory and mechanism called Cyberspace Mimic Defense (CMD) in 2016 [18]. This active defense theory is called cyberspace mimic defense for it is inspired by the mimicry phenomenon in nature, in which one creature imitates the appearance and behavior of another creature to obtain survival benefits [19].

Based on the premise that "the probability of people with different shortcomings making the same mistake while performing the same task independently at the same time is extremely low" [19], this theory designs a core defense mechanism: Dynamic Heterogeneous Redundancy (DHR) [18]. In this mechanism, a mimic defense system contains multiple redundant heterogeneous executors E_i ($i = 1,2,3,\ldots,m$) that can realize the same network function, and each executor in the system is built by different hardware and software components. When the system is running, it selects n ($n \leq m$) executors based on a certain scheduling strategy at each scheduling moment to form an online executors set, and then obtains the final output of the entire mimic defense system according to the independent working process of each online executor.

For the components of different executors are different, a specific network attack that utilizes a certain vulnerability in an online executor is usually can not affect the rest online executors. For this reason, after adopting the majority voting mechanism, that mimic defense system will not be affected by that attack and will return a correct result, which can actively defend against network attacks. Actually, CMD can defend against most of the network attacks except the resource consumption attacks represented by DDoS attacks. For example, CMD can effectively defend against the attacks that utilize the unknown vulnerabilities or reserved backdoors of the operating system in order to obtain system permissions; the attacks that exploit the vulnerabilities in programming languages; the hijacking and cache poisoning to the domain name servers. Although CMD requires more hardware and software costs than traditional active defense and static defense technologies, it can effectively defend against the network attacks launched by unknown vulnerabilities, backdoors and Trojan horses in the network, so it can endow inherent security characteristic to high-value network targets at the same time. At present, CMD has been applied in many important network fields such as web server [20]–[22], software-defined network [23], cloud service [24], DNS server [25], and so on.

Undoubtedly, any kind of defense mechanism is not absolutely safe, CMD is no exception. By spending huge costs to continuously attack online executors, the network attackers can obtain enough information from the majority of online executors and finally breach the mimic defense system. This kind of attack method is called coordinated attacks by related researchers [17], [22], [26]. In addition, Advanced Persistent Threat (APT) [27] has always been a huge challenge for many active defense methods including MTD and CMD. Cyberspace attackers can learn the system information through long-term network detection and information collection, and then they are able to launch more targeted network attacks on the mimic defense system. In the face of these security threats, the scheduling mechanism in CMD determines when and how to select a part of redundant executors in the executors resource pool to reasonably form an online executors set, and has become the key to whether the mimic defense mechanism can actively defend against the security threat caused by unknown network vulnerabilities. A scheduling algorithm with perfect performance should have the ability to effectively reduce the risk of coordinated attacks and APT faced by the entire mimic defense system.

We deeply research the scheduling strategy in CMD. Specially, the main contributions by us are as follows:

(1) We analyze the key factors that affect the security performance of the mimic defense system, and propose the calculation methods to evaluate them. We model a general mimic defense system, then analyze the key evaluation indicators of dynamic scheduling by introducing a continuous-time Markov process, and propose a specific quantitative calculation method for these indicators. The proposal of these key factors and their evaluation methods provides a mathematically effective way of thinking for the academic community to further study the scheduling mechanism of mimic defense theory.

(2) We present a complete and reasonable feedback mechanism. This mechanism is able to mark the working states of the executors and evaluate their real-time security performance by specific calculation methods. Furthermore, the feedback mechanism presented by us provides critical real-time information for the scheduling decision process, which helps the scheduling strategy algorithm to obtain an online executors set with better security performance.

(3) We propose an optimal seed scheduling strategy algorithm (OSSSA), which can obtain a high reliability and security online executors set at each scheduling moment. OSSSA comprehensively considers the initial performance and the real-time security performance of each executor and the heterogeneity between the executors, so it is able to maximize the endogenous defense performance of the DHR mechanism in mimic defense.

(4) We present an evaluation method including multiple real-time security indicators for scheduling algorithms. How to effectively verify the actual performance of different scheduling strategy algorithms in different application scenarios has always been a research focus in academia.

We put forward the meaning of reliability and anti-attack performance in scheduling strategy algorithms, and give a general evaluation method as well.

The structure of the rest sections of this article is as follows. Section II introduces the related work of our research. Section III first gives a model of a typical mimic defense system, then introduces a continuous-time Markov process to analyze the working states of the mimic defense system. On this basis, section IV proposes an optimal seed scheduling strategy algorithm, and gives a detailed introduction to its key design factors and phased working process. Section V verifies the effectiveness and advancement of the OSSSA proposed in this paper by conducting software simulations and real experiments. In the last section, we summarize our work and give prospects for future study.

II. RELATED WORK

Due to the key role of scheduling mechanism in cyberspace mimic defense, it is a focus problem in cyber security. In general, the research on scheduling mechanism can be classified into two aspects: timing and strategy, or when and how. The scheduling timing focuses on when to carry out dynamic scheduling, and the scheduling strategy focuses on how to carry out dynamic scheduling. As for the research on scheduling timing, there have been substantial academic results [17], [28], [29], so it is not the core issue of this article.

Compared with the scheduling timing that emphasizes the cost and efficiency, the scheduling strategy that emphasizes the defense effect is a more critical problem in CMD. A reasonable scheduling strategy algorithm can not only limit the opportunities for vulnerabilities to be exposed and exploited, but also disrupt the attack process [19]. In the field of scheduling strategy, the related algorithms can be roughly divided into three categories: complete randomness, based on differences or historical confidence, combining differences and historical confidence. In the rest of this section, we will deep analyze and evaluate them.

A. COMPLETE RANDOMNESS

In the field of scheduling strategy, Academician Wu Jiangxing proposed a completely random scheduling strategy algorithm in [18], which generates online executors set through a completely random method. This strategy has strong randomness and low algorithm complexity, so it has been widely used [30]–[32]. However, the completely random strategy has some natural flaws. It may easily schedule the executors that have the same unknown vulnerabilities, which will cause the defense system to be easily broken by network attackers; Besides, the scheduling of the mimic defense system under this strategy can not be managed and controlled by network managers.

B. BASED ON DIFFERENCES OR HISTORICAL CONFIDENCE

Designing a scheduling strategy that can both improve the security of the system and ensure the low complexity of the strategy is a research hotspot in this field. To be specific, this kind of algorithms achieve this goal by considering the differences between the executors or introducing historical confidence in them. Liu *et al.* [33] proposed a random seed scheduling strategy algorithm (RSSSA) based on the minimum similarity between online executors. This algorithm randomly selects an executor as the seed executor and then schedules an executors set with the smallest similarity. On this basis, Jiexin Zhang *et al.* [34] applied the RSSSA to the web server and improved it by considering the heterogeneity between executors and their respective web server quality. Besides, Wang *et al.* [35] proposed a scheduling strategy based on Bayesian Stackelberg game theory. Under the web server scenario, this algorithm can obtain the online executors set that maximizes the security gain of the defense side by calculating the difference between online and offline executors set. In [36], Qingqing Zhang *et al.* added an analyzer in the mimic defense system to learn the historical information of each executor, and then scheduled the executors dynamically according to the results from the analysis and the evolutionary game theory. In addition, Yang *et al.* [37] proposed a strategy algorithm with feedback capability, which calculated the scheduling probability of executors according to a history information table. They verified the defense performance of their algorithm by designing simulated collision experiments. In general, the scheduling strategies proposed in [33]- [35] are able to improve the security of mimic defense system by quantifying the differences between the executors in different ways. Moreover, the scheduling strategy in [36] and [37] is able to utilize the historical confidence of the executors to improve the system security to some extent.

However, the scheduling strategies that based on the differences have determined the scheduling executors set at the time of initialization and will not change it, and the scheduling strategies that considering the historical confidence of the executors ignore the heterogeneity in the theory of CMD. Therefore, this kind of low complexity scheduling strategies are not able to meet the high-security requirements under some practical scenarios.

C. COMBINING DIFFERENCES AND HISTORICAL CONFIDENCE

The theory of CMD has enormous potential in security performance, while many practical network scenarios need reliable network security. Therefore, many researchers both consider the differences between the executors and the historical confidence in their scheduling strategies, which can further improve the defense capability of the system. Wu *et al.* [38] introduced historical confidence in the RSSSA in order to improve the security of the mimic defense system. They selected the seed executor from the executors which historical confidence satisfy the threshold. However, they did not give a specific calculation method for the historical confidence, which limits the integrity of this algorithm. Zhang *et al.* [39] proposed a dynamic scheduling strategy based on the normal distribution, which calculated the

scheduled probability by comprehensively considering the online duration and security of executors, and standardized the scheduling probability to the normal distribution. However, this scheduling strategy algorithm lacks a scientific basis to prove that an executor is more dangerous while its online duration is longer. Gao *et al.* [40] proposed a scheduling strategy based on the heterogeneity and the security coefficient of the executors set, but the security coefficient is simply calculated by factor scaling, which is not reliable. Focusing on the historical confidence of the executors over a period of time and the heterogeneity between the executors, Zhang *et al.* [41] proposed an optimal scheduling algorithm. They evaluated the historical confidence of the executors by setting a sliding window, which can effectively improve the operating efficiency of the algorithm and the security of the system under non-uniform distributed cyber attacks. Moreover, by introducing dynamic game theory into the scheduling strategy, Chen *et al.* [42] proposed a strategy that can balance the defense costs and system security. In this strategy, the number of executors will be dynamically adjusted according to the historical strategies of the attackers, which can reduce the defense cost of the system.

In general, the mainstream scheduling strategy algorithms in mimic defense are designed from different aspects, and they all have certain feasibility and scope of application. These scheduling strategy algorithms, however, still fail to design from the perspective of effectively improving the security performance of the entire mimic defense system under different scenarios, and are unable to comprehensively consider the randomness, real-time security, and universality, making CMD still facing huge challenges brought by coordinated attacks and APT.

III. MODELING AND ANALYSIS OF MIMIC DEFENSE SYSTEM

In CMD, the design of the scheduling strategy algorithm will be affected by multiple components of the entire mimic defense system, which is a global problem. From this perspective, when designing the algorithm, we should refine the key factors and evaluation indicators of the scheduling strategy algorithm from the overall mimic defense system, so as to design a reliable scheduling strategy algorithm that can fully exploit the security potential of the mimic defense system. This section first builds the structure of a typical mimic defense system and introduces the working principle of it, and then uses continuous-time Markov process to model and analyze the working state of the mimic defense system.

A. STRUCTURE AND WORKING PRINCIPLE OF THE MIMIC DEFENSE SYSTEM

As shown in Fig. 1, a typical mimic defense system has the following modules: input agent, executors resource set, scheduling controller, online executors set, and output agent. To better explain how a typical mimic defense system works, we first state the roles of these modules and their information interaction processes in the system.

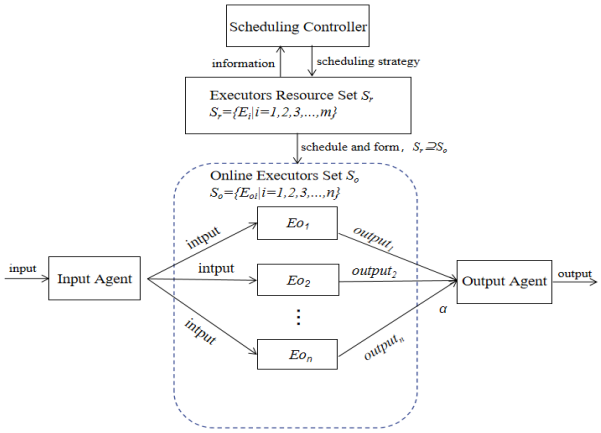


FIGURE 1. A typical structure of mimic defense system.

1) INPUT AGENT

The input agent is the data received module of the mimic defense system, which is responsible for receiving data from clients, copying these data and distributing them to each executor contained in the online executors set.

2) EXECUTORS RESOURCE SET

Executors resource set is a set in logic. It contains multiple executors that can realize the same network function. These executors are built in different ways in different layers, such as operating system, network protocol, and application software. For this reason, these executors are dynamic, heterogeneous, and redundant.

3) SCHEDULING CONTROLLER

Scheduling controller is the key module to ensure that the mimic defense system can actively defend against the unknown network weaknesses. According to the real-time information of each executor, scheduling controller decides when and how to get the online executors set from the executors resource set.

4) ONLINE EXECUTORS SET

Online executors set is a logical set of scheduled executors. Each executor in it will process the data distributed by the input agent independently, and then sends the result to the output agent individually.

5) OUTPUT AGENT

Based on the independent output from each online executor, the voting strategy contained in the output agent is able to obtain the final output of the whole mimic defense system. After that, the output agent will return it to the corresponding client.

In this article, the symbols used to represent the basic elements of the scheduling mechanism in CMD and their respective meanings are shown in Table 1.

TABLE 1. The meaning of the symbols in the scheduling mechanism.

Symbol	Implication
S_r	executors resource set
S_o	online executors set, $S_o \subseteq S_r$
E	the executor, which is heterogeneous and independent to each other
m	the number of executors in executor resource set
n	the number of executors in online executors set, $n \leq m$
α	the set of output from each online executor

After building the structure of a typical mimic defense system, we describe the complete working process of it. At each scheduling moment, scheduling controller will schedule S_o from S_r, which is illustrated in Fig. 2(a). Besides, Fig. 2(b) illustrates the whole process after the mimic defense system receives the data from a client.

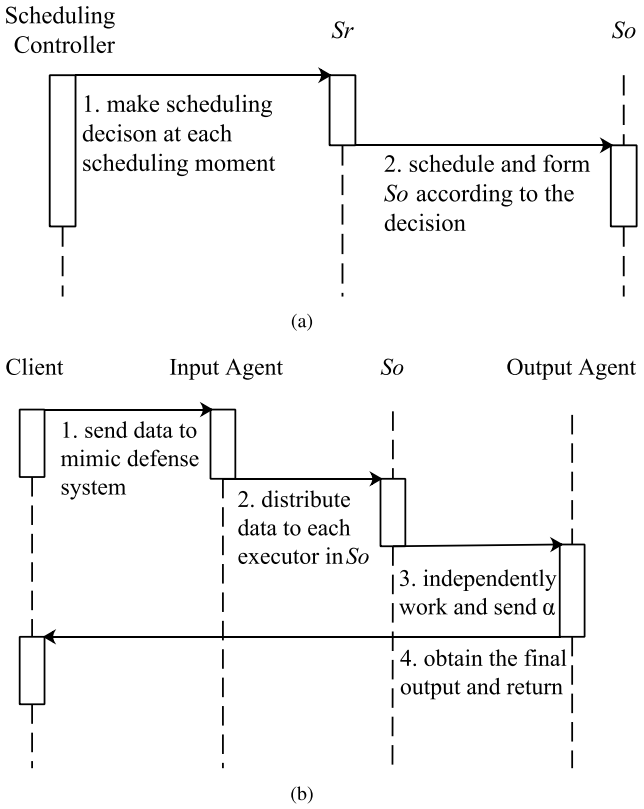


FIGURE 2. Complete working process of the mimic defense system. (a) Scheduling process. (b) Data processing of the system.

In a typical working process of mimic defense system, the input agent will simple copy the data and distribute them to each executor in the online executor set $S_o = \{Eo_i | i = 1, 2, 3, \ldots, n\}$, which is obtained by scheduling controller based on certain scheduling strategy from DHR executors resource pool $S_r = \{E_i | i = 1, 2, 3, \ldots, m\}$. When each online executor $Eo_1, Eo_2, \ldots, Eo_n$ receives the data sent from the input agent, it will independently obtain its running result $output_1$, $output_2, \ldots$, $output_n$, and then sends to the output agent. After that, the output agent will obtain the final output of the mimic defense system from output set α according to a certain voting strategy.

B. CONTINUOUS-TIME MARKOV PROCESS OF THE MIMIC DEFENSE SYSTEM

Based on the model of mimic defense system, we can further clarify the working states and their mutual changes of the mimic defense system through system analysis, which can help us to learn how to keep the entire system in a security state. Among many system analysis tools such as Unified Modeling Language, Finite State Machine, Petri net, data diagram, data dictionary, Petri net [43] use places, transitions, directed arcs, and orders to clarify the structure and function of the system, having strong ability in reliability and security modeling [44]. Academician Wu has established a detailed and specific generalized stochastic Petri net (GSPN) model for the mimic defense system in his work [45]. Based on his study, and since the reachable graph of the GSPN is isomorphic with the continuous-time Markov chain [46], we can utilize the continuous-time Markov process to quantitatively analyze the system states. Different from the [45] focuses on how to prove the effectiveness of the CMD, our mathematical analysis aims to study how to design a scheduling strategy algorithm to improve the security of the mimic defense system.

Generally speaking, if an executor is breached by the network attackers, it can not produce the correct output. We call this working state as ''abnormal state'' in our studies. Besides, the meaning of ''normal state'' in this article is the correct output in accordance with known experience. Before the analysis, we make two reasonable assumptions in the context of continuous-time: the specific moments when different online executors changed to abnormal state are different even if they are compromised by the same network attack, because different online executors are independent and have different running speeds; the specific moments when abnormal executors are restored to normal state are different. Based on these two assumptions, we set the state space of the mimic defense system to $S = \{0, 1, 2, \ldots, m\}$ according to the number of abnormal executors in the mimic defense system at time t. The definition of the Markov process is: if the state at time t is known, then whatever the state before time t is, it has no effect on predicting the state of the system after time t [47]. Obviously, the mimic defense system with the state space S has Markov property. Besides, under the two assumptions mentioned at the beginning of this paragraph, the mimic defense system can only transfer its state from i to $(i + 1)$ or $(i - 1)$, or just remains unchanged in a very short time, so it belongs to the birth and death process.

After the above analysis, we can use the relevant mathematical methods of the continuous-time Markov birth and death process to calculate the stationary distribution of each state of the mimic defense system.

Assuming that the number of executors is m. According to [17], within time Δt, the executors change from normal state to abnormal state following a Poisson process with an intensity of λ. Besides, assuming that the probability of each executor returning from abnormal state to normal state in time Δt is $\mu\Delta t + o(\Delta t)$. Recording $\xi(t)$ as the number of

abnormal executors in the mimic defense system at time t, then $\xi = \{\xi(t), t > 0\}$ is a time-homogeneous Markov chain, the state space $S = \{0, 1, 2, \ldots, m\}$, and the transition probability function of ξ is:

$$P_{ij}(\Delta t) = \begin{cases} i\mu\Delta t + \Delta t, & j = i-1, 1 \le i \le m, \\ \lambda\Delta t + \Delta t, & j = i+1, 0 \le i \le m-1, \\ -[\lambda + i\mu + o(\Delta t)], & 0 \le i = j \le m-1, \\ -m\mu + o(\Delta t), & i = j = m, \\ o(\Delta t), & else, \end{cases} \quad (1)$$

where i or j is taken from the state space S, and represents a specific state of the system. Therefore, we are able to get the transfer rate matrix Q of ξ as formula (2), as shown at the bottom of the page.

Based on the transfer rate matrix Q of the birth and death process ξ, the stationary distribution of each state of the mimic defense system can be calculated as (3):

$$\pi_j = \frac{1}{j!}\left(\frac{\lambda}{\mu}\right)^j \left[\sum_{i=0}^{M} \frac{1}{i!}\left(\frac{\lambda}{\mu}\right)^i\right]^{-1}, \quad 0 \le j \le M. \quad (3)$$

In addition, since ξ is a non-periodic irreducible closed set, its limit distribution is the same as the stationary distribution.

IV. OPTIMAL SEED SCHEDULING STRATEGY ALGORITHM

We propose a reliable scheduling algorithm called optimal seed scheduling strategy algorithm. The OSSSA contains key factors refined from the continuous-time Markov process, which can effectively improve the reliability and real-time security of the mimic defense system. In this section, we first introduce how we refine these key factors, and give their meanings and calculation methods in detail. After that, we introduce the mechanism and working process of the OSSSA by text describing, the introduction of the OSSSA functional modules, and algorithm flow. At the end of this section, we propose a general evaluation method for the reliability and security testing of the scheduling strategy algorithms.

A. KEY FACTORS IN THE OSSSA

In the quantitative analysis of the continuous-time Markov process, we find that in order to ensure the entire mimic defense system having the ability to get correct output, the value of $\pi_0, \pi_1, \ldots, \pi_{\frac{n-1}{2}}$ should be increased, or the probability of the other states should be decreased. For this reason, when designing the scheduling strategy algorithm with high-reliability and high-security, we should try to reduce λ in the stationary distribution (3), or increase μ from the perspective of security. Actually, different redundant heterogeneous executors have different performance, and network attackers can use the same network weakness between different executors to launch coordinated attacks. Therefore, the scheduling strategy algorithm should give priority to scheduling the executors with better performance and more suitable for specific application scenarios, or avoid scheduling the executors with insufficient differences in software and hardware components at the same time, so as to achieve the design goal of reducing λ in the stationary distribution (3). We use the initial performance of the executor and the heterogeneity between the executors to quantify these two design ideas. Besides, the parameter μ represents the probability of each executor returning from an abnormal state to a normal state within a certain time. So we can introduce a reasonable mechanism called feedback mechanism to schedule offline the abnormal online executors, which can effectively increase the value μ in (3) by allowing the network managers to have enough time to repair or replace the abnormal executors.

Through the above analysis, we have refined three key factors in the OSSSA that are able to improve the reliability and security of the mimic defense system: the initial performance of the executor, the heterogeneity between the executors, and a reasonable feedback mechanism. In the next part, we will introduce their meanings and evaluation methods in detail.

1) INITIAL PERFORMANCE

In CMD, it is necessary to construct multiple executors with the same network function through different software and hardware components. These executors are heterogeneous with each other, redundant in number, reconfigurable in logic, so their initial performance are not the same. Generally, different executors show different initial performance in three main aspects: different security performance, different running performance, and different compatibility to specific application scenarios. Taking DNS servers as an example, the number of vulnerabilities in different DNS software that can be queried on Common Vulnerabilities&Exposures (CVE) is shown in Table 2 (the query date is Oct. 30, 2020).

In terms of the stress test, after performing a stress test of 57,600 queries on different DNS servers, the results are shown in Table 3.

$$Q = \begin{pmatrix} -\lambda & \lambda & 0 & 0 & 0 & 0 \\ \mu & -(\lambda+\mu) & \lambda & 0 & 0 & 0 \\ 0 & 2\mu & -(\lambda+2\mu) & \lambda & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & (m-1)\mu & -(m-1)\lambda & \lambda \\ 0 & 0 & 0 & 0 & m\mu & -m\mu \end{pmatrix} \quad (2)$$

TABLE 2. Number of announced vulnerabilities in different DNS software.

	PowerDNS	Unbound	Bind9	MaraDNS
Number of vulnerabilities	61	19	10	15

TABLE 3. Stress test of different DNS servers.

Performance	PowerDNS	Unbound	Bind9	dnspod-sr
Queries per second	6752.52	6552.52	578.54	54.78
Percentage completed	100%	100%	99.38%	93.78%

In addition, as shown in Table 4, different DNS software support different functions according to Wikipedia.

TABLE 4. Functions supported by different DNS software.

Functions	PowerDNS	Unbound	Bind9	MaraDNS
TSIG	Yes	No	Yes	No
DNSSEC	Yes	Yes	Yes	No
Platforms	Linux,Solaris, BSD, Windows, Mac OS X	Linux,Solaris, BSD, Windows, Mac OS X	BSD,Solaris, Mac OS X, Linux	BSD,Solaris, Mac OS X

Table 2-4 have clearly shown that different executors have different performances, which will undoubtedly affect the working efficiency and security performance of the entire system. For this reason, when an executor is added to the DHR executors resource pool for initialization, the OSSSA will give it a performance value according to the testing results. In addition, we can also find that the data types and value ranges of different testing properties are different. So we propose to use TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) [49] to calculate these different properties, and obtain a performance evaluation result S_p that falls into interval [0,1].

Besides, in different practical application scenarios, there will be some specific requirements or preferences for the executors, such as compiled language, computing speed, operating system, etc. In order to prioritize the executors that are more suitable for practical application scenarios, we propose to introduce a manual evaluation indicator S_m that set by the network managers and falls into interval [0,1]. We obtain the final initial performance evaluation S^i_{per} of each executor E_i according to formula (4):

$$S^i_{per} = 100 \times (w_p S^i_p + w_m S^i_m), \tag{4}$$

where w_p and w_m represent the weight of S^i_p and S^i_m respectively, which can be set and changed by the network managers. The introduction of w_p and w_m can enhance the flexibility of the algorithm and allow the OSSSA to be applied in more specific application scenarios. It is worth noting that the weights in the calculation formulas mentioned in this article can be manually managed by the network managers for these two purposes.

2) HETEROGENEITY

Heterogeneity means that two functionally equivalent executors are different in structural design [26]. The redundant executors in the mimic defense system need to have enough differences in structure, so that different online executors will not be successfully breached for the same network weakness and the λ in the formula (3) can be decreased. The differences include not only software, such as applications, the protocols in the different layers, types of operating systems, types of compiled languages, etc., but also the underlying hardware. As shown in Table 5, multiple network levels of the web server can be different in implementation. Without considering interface compatibility, the data in Table 5 alone can provide up to 400 implementation plans for the formation of an executor:

TABLE 5. Optional structural differences of a mimic web server system.

Network level	Optional
Application script	PHP, ASP, JSP, ASP.net
Server software	Apache, Nginx, IIS, Lighttpd
Virtual operation system	Windows, CentOS, Ubuntu, Red Hat, Solaris
Physical operating system	Windows, CentOS, Ubuntu, Red Hat, Solaris

Generally speaking, the greater the heterogeneity between two executors, the greater the structural differences between the two executors, and the less likely it is to be breached by the same cyberspace attackers. Considering that CMD is a cyberspace security theory, and each executor is composed of multiple software and hardware components, we evaluate the vulnerabilities of each mimic network component layer between the two executors to obtain their heterogeneity. The relevant factors for the evaluation of heterogeneity are defined as follows.

a: VULNERABILITY EVALUATION OF EACH MIMIC COMPONENT LAYER

The multiple component layers in CMD are called mimic component layers because they can be implemented in different manners. For example, the web servers listed in Table 5 have four alternative mimic component layers. When making a fine-grained evaluation of the heterogeneity between two redundant executors, we first evaluate the known vulnerabilities of each mimic component layer in each executor E_i:

$$S^{il}_h = \sum_{k=1}^{q_{il}} e_{ilk}, \tag{5}$$

where S^{il}_h represents the vulnerability evaluation S_h of the mimic component layer l of the executor E_i, q_{il} represents the number of known vulnerabilities in the mimic component layer l of the executor E_i, which can be learned in CVE, e_{ilk} represents threat score of the vulnerability k in the mimic structure layer l of the executor E_i, which can be obtained through the Common Vulnerability Scoring System (CVSS).

b: HETEROGENEITY EVALUATION OF THE MIMIC COMPONENT LAYER

In the same mimic defense system, the number and the division standard of mimic component layers in each redundant

executor are the same. So after calculating S_h^{il}, the degree of structural differences between two mimic components in the same layer l can be reflected by known vulnerabilities:

$$H_{ijl} = 1 - \frac{(S_h^{il} + S_h^{jl})S_h^{ijl}}{2S_h^{il}S_h^{jl}}, \tag{6}$$

where S_h^{ijl} represents the evaluation of the same vulnerabilities that exist in the mimic component layer l between executors E_i and E_j, H_{ijl} represents the heterogeneity of the mimic component layer l between redundant executors E_i and E_j.

The formula of the heterogeneity H_{ij} between redundant executor E_i and E_j is as follows:

$$H_{ij} = \sum_{i=1}^{L} w_l H_{ijl}. \tag{7}$$

Among them, w_l represents the importance of the mimic component layer l, it can be set by the network managers. w_l naturally satisfies the following relationship:

$$\sum_{l=1}^{L} w_l = 1. \tag{8}$$

3) FEEDBACK MECHANISM

As mentioned in the previous mathematical analysis, the security state of each executor may be changed at any time. To express this phenomenon and indicate the real-time security of the executors well, the scheduling algorithm needs to set a feedback indicator S_{fb} for each executor in S_r. A reasonable feedback mechanism is able to make the previous successful attacks launched by network attackers unable to succeed again for a period of time. Moreover, that mechanism allows the network managers have enough time to repair the vulnerabilities or backdoors exposed by previous successful attacks, so that the μ in the stationary distribution (3) will be effectively increased.

Although the introduction of a feedback mechanism into the scheduling strategy algorithm is of great help to improving the defense performance of the mimic defense system, it is pity that there is still lacking a reliable feedback mechanism in the field of mimic defense scheduling algorithms. For example, Wu proposed a qualitative setting standard for feedback indicator in [26], but he did not give a specific quantitative setting method; Gao *et al.* simply increase or decrease the feedback index value by the same factor in [40], which is disable to effectively reflect the real-time security of each executor by that feedback mechanism; In [48], Guoxi Chen *et al.* proposed a feedback method that comprehensively considering the online hours and historical breached times of each executor. Although the mechanism is dynamic to a certain extent, it is still difficult to effectively prevent network attackers from repeating the previous successful attacks to online executors.

Inspired by congestion control ideas in computer networks, we design a reliable feedback mechanism and the evaluation method for feedback indicator S_{fb}, and we apply it to the OSSSA. The feedback mechanism comprehensively considers the initial security performance and the real-time security performance of the executors at each scheduling moment. Specially, we refer to the setting method of the congestion window to formulate the feedback indicator S_{fb}^i of each executor in two aspects: on the one hand, S_{fb}^i will increase in stages when the executor E_i is in normal state, so that the executors with better initial or security performance can be scheduled in priority, and the scheduling closed loop can be avoided; on the other hand, S_{fb}^i will drop sharply when the executor E_i returns to abnormal state, which allowing the network managers to have enough time to repair the abnormal executors. Each time when the output agent of mimic defense system obtains the output, the feedback indicators of each executor in S_r will be updated. The factors included in the feedback mechanism are defined as follows.

a: STATE FLAG

By setting the state flag E_{flag}^i of executor E_i, we can identify whether an executor is scheduled, and mark the changing of its working states. The state flag E_{flag}^i of E_i is defined as follows:

$$E_{flag}^i = \begin{cases} 0, & E_i \notin S_o, \\ 1, & E_i \text{ in normal state and } E_i \in S_o, \\ 2, & E_i \text{ in abnormal state and } E_i \in S_o. \end{cases} \tag{9}$$

b: REMAINING REPAIR TIME

We simplify the vulnerabilities repair process and propose a reasonable assumption: the network managers have the ability to repair the vulnerabilities of an executor or can replace an abnormal executor within a certain period of time, so that the executor in the same number can restore to normal state again. Under this assumption, we use the scheduling time T as the unit, define the remaining time R_i^T for vulnerabilities repair process of executor E_i, and update it at each scheduling time T:

$$R_i^T = \begin{cases} R_{max}^i, & E_{flag}^i = 2, \\ R_i^{T-1} - 1, & E_{flag}^i \leq 2 \text{ and } R_i^{T-1} \geq 1, \\ 0, & E_{flag}^i \leq 2 \text{ and } R_i^{T-1} = 0. \end{cases} \tag{10}$$

where R_{max}^i represents the maximum remaining time for vulnerabilities repair process of executor E_i.

c: FEEDBACK GAIN IN A SINGLE TIME

We set the feedback gain in single time g in the feedback mechanism in order to control the fluctuation ranges of the feedback indicators S_{fb}. g needs to be set by the network managers according to the practical application scenarios and the scale of different mimic defense systems.

d: INITIAL PROBABILITY OF ABNORMAL OUTPUT

In the mimic defense system, since the security performances of the different executors are different, and the degree of network threats faced by different practical application scenarios is also different, so we define the initial probability of abnormal output P_α to describe these two difference. The P_α^i of executor E_i can be obtained by a security test when E_i joins the S_r:

$$P_\alpha^i = \frac{times\ of\ abnormal\ output\ from\ E_i}{times\ of\ data\ input\ to\ E_i}. \tag{11}$$

e: SLOW START THRESHOLD IN POSITIVE FEEDBACK

Similar to the slow start threshold *ssthtrsh* of congestion control in computer networks, we set the slow start threshold in the positive feedback sst_{fb}, which can help to avoid forming the local scheduling closed loop. The sst_{fb}^i of each executor E_i can be calculated by (12):

$$sst_{fb}^i = \frac{g}{P_\alpha^i}. \tag{12}$$

f: BREAK-POINT VALUE

This factor is used to immediately save the value of feedback mechanism when the security state of executor E_i is changed from the normal to the abnormal. When $E_{flag}^i = 2$ and $S_{fb}^{i(t-1)} \geq 0$, we save the S_{fb}^i by formula (13):

$$S_{fbsave}^i = \frac{S_{fb}^{i(t-1)}}{d_1}, \tag{13}$$

where d_1 is a positive integer, set by the network managers; $S_{fb}^{i(t-1)}$ represents the feedback indicator S_{fb}^i updated by feedback mechanism after the mimic defense system gets the output at number *(t-1)* times.

After giving the definitions and setting methods of the factors included in the feedback mechanism, we propose a complete feedback mechanism and its quantitative evaluation method applied to the OSSSA, which is mathematically expressed as follows:

$$S_{fb}^{it} = \begin{cases} -sst_{fb} \times R_{max}^i, & E_{flag}^i = 2, \\ S_{fb}^{i(t-1)} + g, & E_{flag}^i = 1, 0 \leq S_{fb}^{i(t-1)} < sst_{fb}^i, \\ S_{fb}^{i(t-1)} + \frac{g}{2}, & E_{flag}^i = 1, S_{fb}^{i(t-1)} \geq sst_{fb}^i, \\ S_{fbsave}^i, & R_i^T = 0, E_{flag}^i = 1, S_{fb}^{i(t-1)} < 0, \\ -sst_{fb} \times R_i^T, & R_i^T \geq 1, E_{flag}^i \leq 1, \\ S_{fb}^{i(t-1)}, & R_i^T = 0, E_{flag}^i = 0. \end{cases} \tag{14}$$

where S_{fb}^{it} represents the feedback indicator of executor E_i updated by feedback mechanism after the mimic defense system gets the output at number t times.

In the feedback mechanism defined by formula (14), when Eo_i is in normal working state, its S_{fb}^i will increase at a faster rate before reaching the threshold sst_{fb}^i, and slow down its growth speed after reaching its threshold. Correspondingly, when Eo_i is attacked and becomes insecure, its current S_{fb}^i will first be saved as the break-point value S_{fbsave}^i according to (13), and then be updated to a negative value with a large absolute value. If in the subsequent R_{max}^i scheduling moments, the executor that in the abnormal working state has not been scheduled again, S_{fb}^i will be gradually restored to a negative value with a smaller absolute value. When the network manager successfully repairs or replaces that abnormal executor within R_{max}^i, the feedback indicator will jump to a positive value again and automatically be updated to S_{fbsave}^i. This feedback mechanism aims to achieve real-time differentiated feedback on the security performance of the online executors by designing a complete evaluation method.

B. WORKING MECHANISM OF THE OSSSA

The OSSSA is a scheduling strategy algorithm that is divided into three working phases. After introducing the quantitative methods of the three key indicators, we will describe the working mechanism of each phase and the related calculation methods in the three working phases of the OSSSA.

In the first phase of the OSSSA, it will evaluate the actual performance of each executor E_i, including initial performance S_{per}^i calculated by formula (4) and real-time security performance calculated by formula (14). In this phase, the OSSSA evaluates the actual performance S_A^i of E_i through formula (15):

$$S_A^i = w_1 S_{per}^i + w_2 S_{fb}^i. \tag{15}$$

Similar to formula (4), w_1 represents the weight of S_{per}^i, w_2 represents the weight of S_{fb}^i, and their sum is 1.

In the second phase, the OSSSA first selects an executor from $S_r = \{E_i | i = 1, 2, 3, \ldots, m\}$ according to the method of sorting from small to large, and sets it as the seed executor, which is represented by the symbol $E_s^i (i = 1,2,3,\ldots,m)$. After selecting E_s^i, the OSSSA comprehensively considers the actual performance of each other executor $E_j (j = 1, 2, \ldots, i-1, i+1, \ldots, m)$ in S_r, and the heterogeneity H_{ij} between each executor E_j and the seed executor E_s^i calculated by formula (7). In this phase, the OSSSA obtains the adaptability evaluation S_B^j of each E_j according to formula (16):

$$S_B^j = \begin{cases} w_3 S_A^j + 100 w_4 (w_5 H_{ij} + \frac{w_6}{m-2} \sum_w H_{jw}), \\ \qquad j \neq i \text{ and } w \neq i, j \text{ and } m \geq 3, \\ 0, \qquad i = j. \end{cases} \tag{16}$$

where w_3 represents the weight of S_A^j, and w_4 represents the weight of heterogeneity, the sum of them is 1. Besides, w_5 represents the weight of H_{ij} and w_6 represents the weight of $\frac{1}{m-2}\sum_w H_{jw}$, and their sum also is 1. In addition, $\frac{1}{m-2}\sum_w H_{jw}$ represents the average heterogeneity between the executor E_j and other executors except for the seed executor E_i. After that, the OSSSA selects *(n-1)* executors ranked from high to low according to S_B^j in the *(m-1)* remaining executors E_j, which forming the adapted executors set $S_i (S_i = \{Ek_1, Ek_2, \ldots, Ek_{n-1}, E_s^i\}, S_i \subseteq S_r)$ of the seed

executor E_s^i with E_s^i itself together. We obtain the performance evaluation S_C^i of the adapted executors set S_i by formula (17):

$$S_C^i = \frac{S_A^i + S_B^{k_1} + S_B^{k_2} +, \ldots, + S_B^{k_{(n-1)}}}{n}. \tag{17}$$

In this phase, the OSSSA traverses from the smallest executor number and sets each executor as seed executor E_s^i. If the S_r contains m executors, it can obtain a total of m adapted executors set S_i and their performance evaluation S_C^i. When the OSSSA obtains the set S_i in this phase, it comprehensively considers the actual performance and the heterogeneity of the executors.

The first two phases of the OSSSA have fully considered the security performance and mutual heterogeneity of the online executors. In the third phase, the OSSSA will introduce certain randomness, in order to increase the uncertainty of the mimic defense system showing to the attackers. Specifically, in the third phase of the OSSSA, it will select x candidate executors sets in the all S_i by sorting the performance evaluation S_C^i from high to low, and then randomly select one executor set S_i from them as the final scheduling result of the OSSSA.

By working in phases, the OSSSA can select an online executors set with better real-time security performance at each scheduling time T. From the perspective of mathematical analysis, the OSSSA is helpful to reduce the risks of the executors be breached and increase the possibility that the abnormal executors return to the normal state, so that μ can be increased and λ can be decreased in formula (14).

C. ALGORITHM FUNCTION MODULE

After elaborating the design idea of the OSSSA and the calculation methods of evaluation indicators in each phase, we establish a scheduling controller model as shown in Fig. 3, which aims to further clarify the implementation process of the OSSSA, and the position of the OSSSA in the structure of the mimic defense system as well. The functions of each logical module included in the OSSSA are as follows.

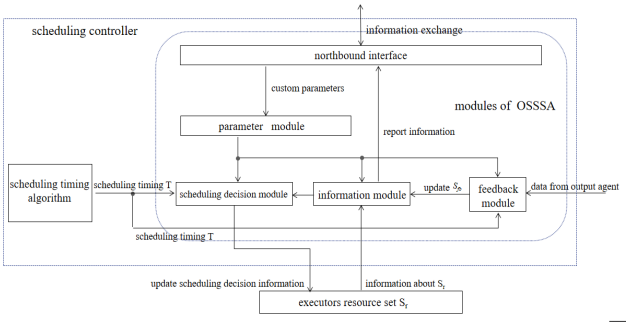


FIGURE 3. OSSSA function modules in the scheduling controller.

1) PARAMETER MODULE

This module stores various parameters that need to be manually set by the network managers in the OSSSA, including m, n, S_m, g, d_1, x, R_{max}^i and all weight parameters have mentioned in this article. The network managers can set these parameters in the parameter module and change them when necessary, so as to effectively manage the entire the OSSSA mechanism and the scheduling process of the mimic defense system.

2) FEEDBACK MODULE

This module is responsible for receiving feedback information from the output agent in mimic defense system, which includes the output results of each online executor Eo_i and the final voting result of the output agent. After judging the consistency between them, the feedback indicator of each executor S_{fb}^{it} will be updated in real-time by the OSSSA feedback mechanism contained in this module. After that, the updated information will be sent to the information module. At the same time, this module will update R_i^T according to the feedback mechanism after receiving the signal at the scheduling time T.

3) INFORMATION MODULE

This module obtains the information of each executor E_i from the S_r, including the number, information of the network components, and working state of E_i. At the same time, this module can calculate and store the actual performance evaluation S_A^i. Besides, information module will regularly report the various information of E_i stored in it to the network managers through the northbound interface module.

4) SCHEDULING DECISION MODULE

When this module receives the scheduling moment signal T generated by the scheduling timing algorithm, it will make scheduling decisions according to the second and third phases of the OSSSA and the information received from the information module. In addition, this module is also responsible for sending the current scheduling decision to the executors resource pool in order to schedule the executors.

5) NORTHBOUND INTERFACE

In addition to the modules related to scheduling, the OSSSA also provides a northbound interface for encrypted information exchange. The network managers can set and modify the key parameters of the OSSSA through the northbound interface, and regularly receive information about each executor E_i returned by the information module.

D. ALGORITHM FLOW

Based on the working mechanism and functional module design of the OSSSA algorithm, we present the algorithm flow of the OSSSA and the feedback mechanism contained in it. Algorithm 1 describes the feedback mechanism, which will receive a copy of the output results produced from the output agent, update the security performance of each online executor in real-time, and provide key information when the OSSSA makes scheduling decisions. Algorithm 2 describes

the OSSSA, which will make a secure and reliable scheduling decision when the signal of scheduling moment T arriving.

Algorithm 1 Feedback Mechanism in the OSSSA

Input: the set of online executors outputs α, the voting result from output agent, scheduling timing T

Output: the feedback indicators of each online executor S_{fb}^{it}

// Initialization phase

initialize parameters n, g, d_1, R_i^{max}

obtain $S_{fb}^{i(t-1)}$, R_i^T, sst_{fb}

if received signal from scheduling decision module, update scheduling timing T // when receive information from output agent

calculate flag E_{flag}^i of each executor E_i

under each scheduling timing T, update R_i^T of each executor E_i

if E_{flag}^i=2 and $S_{fb}^{i(t-1)} \geq 0$: //when the security state of E_i turns to abnormal

    $S_{fbsave}^i \leftarrow S_{fb}^{i(t-1)} / d_1$

    update S_{fb}^{it} by formula(14)

else if $E_{flag}^i \leq 1$ and $R_i^T = 0$ and $S_{fb}^{i(t-1)} < 0$://when the security state of E_i turns to normal

    $S_{fb}^{it} \leftarrow S_{fbsave}^i$

else:

    update S_{fb}^{it} by formula(14)

Algorithm 2 Optimal Seed Scheduling Strategy Algorithm

Input: signal from scheduling timing algorithm, custom parameter settings from network managers, information about each executor from S_r

Output: online executors set S_o

// Initialization phase

initialize parameters m, n, s_m, g, x, d_1, R_{max}^i and all weight parameters

evaluate S_{per}^i when each executor joins S_r // when receive signal from scheduling timing algorithm

calculate S_A^i by formula(15)

for i=1, i++, i≤m:

    set E_i as the seed executor E_s^i

    calculate heterogeneity H_{ij}

    calculate S_B^j by formula(16)

    get the set of suitable executors for E_s^i

    calculate S_C^i by formula(17)

end

$S_o^i \leftarrow$ randomly select a corresponding set from the largest x of S_C^i

E. ALGORITHM PERFORMANCE EVALUATION INDICATOR

After introducing the scheduling mechanism of the OSSSA, this part introduces two evaluation indicators and their calculation methods for evaluating the performance of scheduling strategy algorithm.

1) ANTI-ATTACK PERFORMANCE

The core goal of the scheduling strategy algorithm in CMD is to improve the anti-attack performance of the entire system, that is, the ability to resist external attacks. We calculate the ratio of normal outputs in the mimic defense system, in order to quantify the anti-attack performance of the scheduling strategy algorithm. To achieve this goal and describe the security performance of the executors in different network environments and time processes, it is necessary to first calculate the abnormal output probability P_β^i of executor E_i in real-time.

In CMD, a network attacker can only detect a single online executor at a time, and the rest of the executors in the online executors set are invisible to him. Besides, considering that the initial anti-attack performance of different executors is different, executors implemented with the same network components may have the same unknown weaknesses, and the breached executors have higher security risk before the vulnerabilities are repaired or replaced, we propose to adopt the formula (18) to calculate the abnormal output probability P_β^j of each online executor Eo_j when network attackers detect the online executor Eo_i. On this basis, we can evaluate the anti-attack ability of different scheduling strategy algorithms.

$$P_\beta^j = \begin{cases} P_\alpha^j + \frac{1-P_\alpha^j}{R_{max}^j} R_j^T, & j = i, \\ (1 - H_{ij})P_\beta^i + \frac{1-P_\alpha^j}{R_{max}^j} R_j^T, & j \neq i, \end{cases} \quad (P_\beta^j \leq 1), \quad (18)$$

where H_{ij} indicates that when online executor Eo_j is scheduled, its heterogeneous with online executor Eo_i detected by the network attackers; $\frac{1-P_\alpha^j}{R_{max}^j} R_j^T$ represents the additional risks from the network attackers before the vulnerabilities known by them are repaired or replaced. Before being scheduled offline, the probability of the abnormal executor breached by the same network attack again is 1.

2) RELIABILITY

Based on the analysis in section III, we have learned that the reliability of a scheduling algorithm is related to the initial performance S_{per}, historical accumulated feedback indicator S_{fb}, and the heterogeneity H_{ij} between executors. Therefore, we quantitatively evaluate the reliability of the scheduling strategy algorithms with the performance evaluation S_C of online executors set S_o. In addition, counting the scheduling time T of the first abnormal output of the mimic defense system is also helpful to further learn the reliability of different scheduling strategy algorithms in the early running stage of the mimic defense system.

V. SIMULATION

This section verifies the performance of the OSSSA in two ways: conducting software simulation and real experiment.

In the software simulation part, we theoretically verify the performance of different scheduling strategies under various network parameters. Specifically, we simulate the working

processes and defense effects of different scheduling strategy algorithms, then analyzes specific experimental data in detail to respectively verify the reliability performance, the anti-attack performance, the universality performance of the OSSSA mechanism, and the performance of the feedback mechanism contained in the OSSSA mechanism in four group experiments. We will introduce the software simulation experiments in detail from part A to part C of this section.

Besides, we build a mimic domain name server (DNS) system and carry out real network experiments, which can further verify the advantages of the OSSSA in improving the system reliability and anti-attack performance, as well as the defense effect of the CMD.

A. SIMULATION EXPERIMENTS DESIGN

To better focus on the actual performance of the scheduling strategy algorithm in the software simulation experiments, we first simplify some experimental conditions.

1) Since the mimic defense system is mainly applied to high-value key network equipment, and the scheduling timing algorithm is not the research issue of this article, the simulation experiments assume that the scheduling timing algorithm generates a new scheduling time T whenever input agent obtains external data input, which in order to maximize the defense performance of the entire system.
2) We do not consider the information interaction delay between the modules of the mimic defense system in different practical application scenarios, the delay caused by the data interactions and computing ability in the practical hardware, and the delay caused by software and compiled languages with different performance.
3) The main purpose of the scheduling strategy algorithm is to obtain an online executors set with high-security performance at the scheduling time. Therefore, we do not consider the expense of hardware and software resources brought by the introduction of CMD and the adoption of different scheduling strategy algorithms in the practical application scenarios.
4) Considering the compatibility of the network components in each mimic component layer of the executors, we set a single executor as the smallest unit of the experiment. Moreover, the interface compatibility and protocol between the modules in the practical mimic defense system, as well as the difference in the output result format of different online executors, are currently not included in the research scope of scheduling strategy algorithms, so they are not considered in simulation experiments.

The parameter settings in the simulation experiments and their basis are as follows.

1) Initial performance S^i_{per}: the simulation experiments set the S^i_{per} of each executor E_i by taking a random number in (50,100), which can better simulate different practical application scenarios and obtain universal parameter settings.
2) Heterogeneity between executors H_{ij}: since the heterogeneity H_{ij} is usually a key indicator when constructing an practical mimic defense system, it is generally ideal. In our experiments, H_{ij} is set to satisfy the distribution $\beta(7,3)$.
3) Initial probability of abnormal output P^i_α: considering that the initial anti-attack ability of different executors in the same system are different, the simulation experiments obtain P^i_α of E_i by (19):

$$P^i_\alpha = P_{base} \times ramdom[\frac{1}{d_2}, d_2], \tag{19}$$

where P_{base} represents the reference probability of abnormal output, which is used to simulate the degree of network security risk faced by the entire practical mimic defense system. d_2 is an integer factor, allowing P^i_α has a certain degree of randomness based on P_{base}, which means that the initial security risks of different executors composed of different network components are different. In each simulation experiment, $P_{base} = 0.05$, $d_2 = 2$.
4) Custom parameter settings: in the simulation experiments, the settings of the parameters that need to be customized by the network managers are shown in Table 6. Besides, in simulation experiments, each experiment contains 100,000 scheduling moments T, which means that all scheduling strategy algorithms run 100,000 times.

TABLE 6. Custom parameter settings.

Parameter	Value	Parameter	Value
m	7	g	4
n	3	d_1	4
w_1,w_2	0.5	w_3,w_4,w_5,w_6	0.5
x	3	R^i_{max}	3

B. COMPARISON ALGORITHM

In order to verify that the OSSSA is more advanced in security and reliability than the current mainstream scheduling strategy algorithms in the CMD, we set the completely random scheduling strategy algorithm (CRSSA) [18], [30]–[32], and the Random Seed Scheduling Strategy Algorithm with Feedback (RSSSA-F), as the comparison algorithms of the OSSSA.

The contribution of the OSSSA is reflected in two aspects: the introduction of three key factors and their evaluation methods, and a complete working mechanism that including a reasonable selection method for seed executor. Since the CRSSA does not consider the key factors of the executors, we introduce the CRSSA as a comparison algorithm to verify the importance of the key factors in the OSSSA in improving the defense performance of the mimic defense system. Besides, the Random Seed Scheduling Strategy

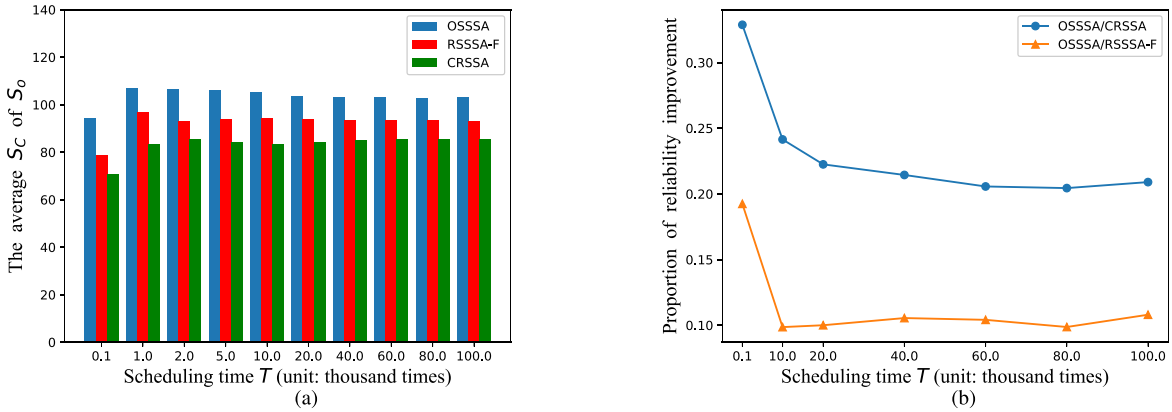


FIGURE 4. The reliability performance of different scheduling strategy algorithms. (a) The average S_C of the online executors sets. (b) Proportion of reliability improvement.

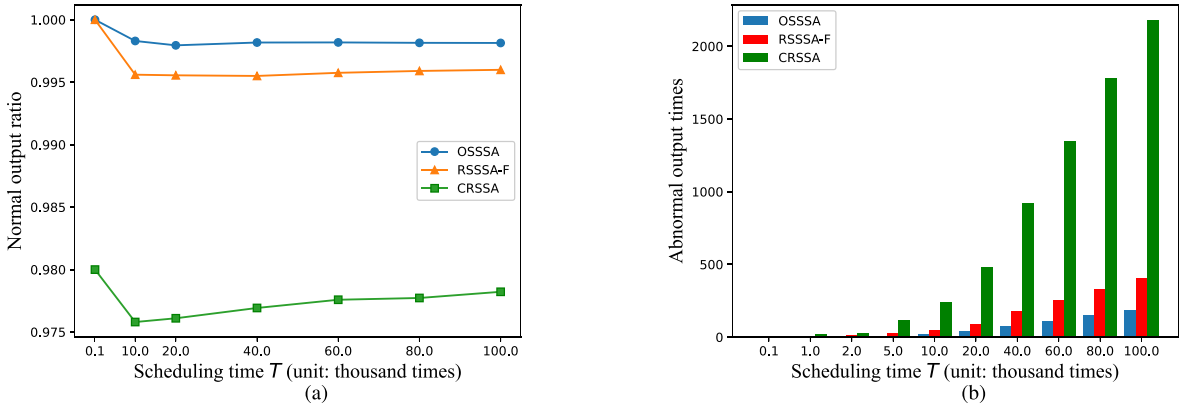


FIGURE 5. The anti-attack performance of different scheduling strategy algorithms. (a) Normal output ratio in 100,000 scheduling moments. (b) Total abnormal output times within 100,000 scheduling moments.

Algorithm (RSSSA) [33]–[35] is another mainstream scheduling strategy algorithm. It considers the running performance and heterogeneity of the executors, but lacks a reasonable seed executor selection method and feedback mechanism. We combine the RSSSA with the feedback mechanism designed by us to form the RSSSA-F algorithm, so that we can verify the necessity and reasonableness of the working mechanism in the OSSSA in improving the defense performance of the system.

C. EXPERIMENTAL RESULTS AND ANALYSES

Under the experimental conditions described in part A of this section, this part shows the results of four simulation experiment groups and makes deep analyses of them.

1) RELIABILITY

The experimental results are shown in Fig. 4. The horizontal axis represents the specific scheduling time T in the timeline with a total of 100,000 scheduling times, and the vertical axis represents the experimental results about the reliability performance.

From Fig. 4(a), it is not difficult to find that the average value S_c of the online executors set S_o obtained under the OSSSA mechanism is higher than that of the CRSSA and the RSSSA-F in each working stage of the mimic defense system. This verifies that due to considerate the key indicators which can improve the algorithm reliability and the adoption of scheduling process, the OSSSA can indeed obtain a more reliable online executors set at scheduling moments. In Fig. 4(b), we can find that with the development of time, the reliability of the OSSSA can be stably improved by about 10% compared with the RSSSA-F, and 20%-25% compared with the CRSSA. In Fig. 4(a) and Fig. 4(b), we can draw the experimental conclusion that the OSSSA has better reliability than the existing mainstream scheduling strategy algorithms.

2) ANTI-ATTACK

The experimental results of the anti-attack experiment for different scheduling strategy algorithms are shown in Fig. 5. Where the horizontal axis has the same meaning as the horizontal axis in Fig. 4, and the vertical axis represents the experimental results about the anti-attack performance.

From Fig. 5(a), we find that when the initial probability of abnormal output is 0.05 and the abnormal output probability P_β of each executor is calculated by the

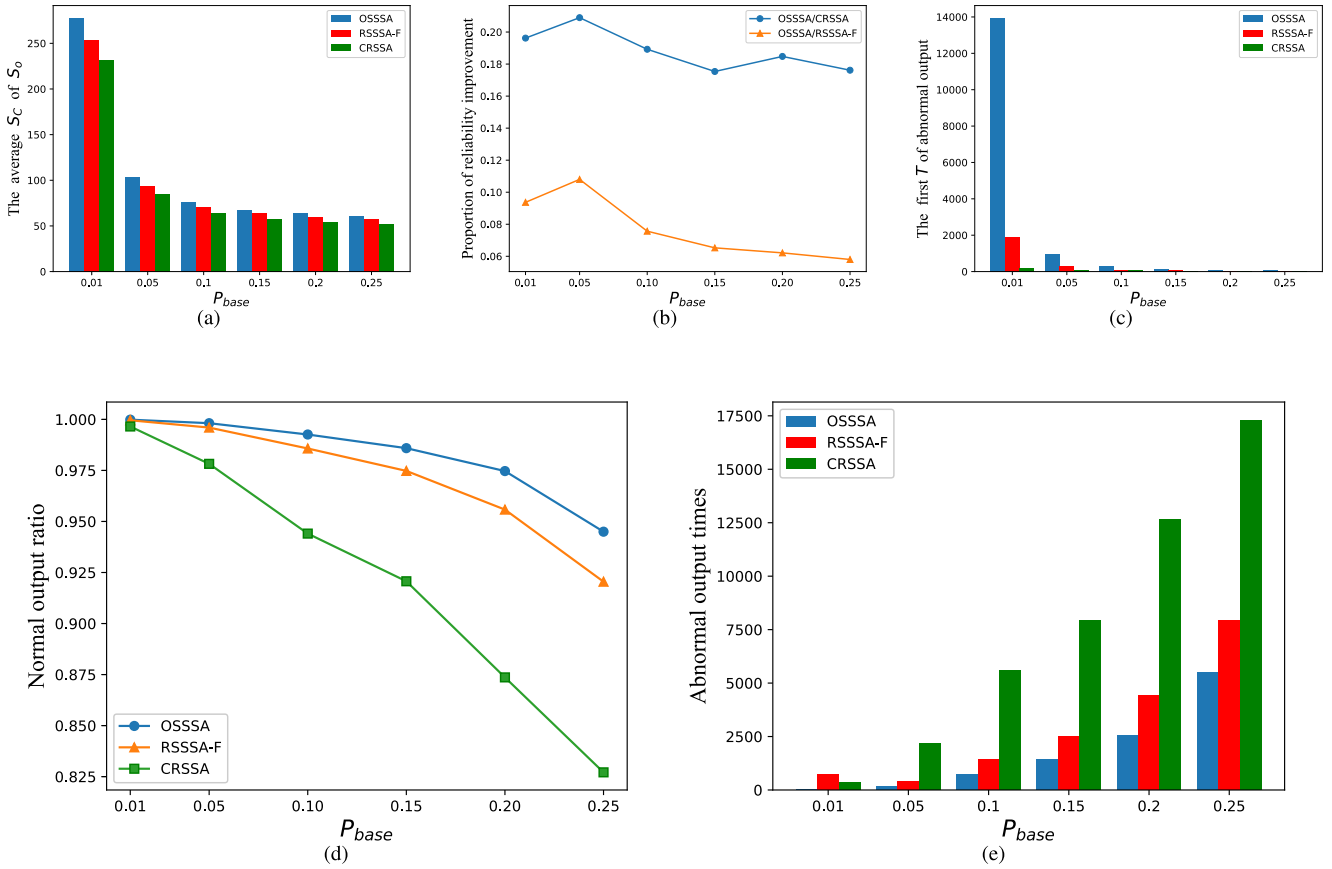


FIGURE 6. Algorithm universality when faced with varying degrees of network security risks. (a) The average S_C of the online executors sets. (b) Proportion of reliability improvement. (c) The first time of abnormal output. (d) Normal output ratio in 100,000 scheduling moments. (e) Total abnormal output times within 100,000 scheduling moments.

formula (18), the normal output ratio in 100,000 scheduling moments of the system adopting the OSSSA mechanism is up to 99.8% when the system gradually stable, which is more than 0.2% higher than the system adopting the RSSSA-F and about 2% higher than the system adopting the CRSSA. When the output agent produces abnormal output, it indicates that the mimic defense system has been breached by network attacks. Therefore, the experiment results show that the OSSSA mechanism has better anti-attack ability than other mainstream scheduling strategy algorithms when facing the same network risk. Fig. 5(b) shows the number of abnormal output produced by the output agent under different working processes, which indicates that the OSSSA can better help the mimic defense system produce less abnormal output. In Fig. 5(a) and Fig. 5(b), it is not difficult to find that the mimic defense system adopting the OSSSA can get a higher ratio of correct output than the comparison groups, thus verifying that the OSSSA has the better anti-attack performance.

3) UNIVERSALITY

We verify the universality of the OSSSA when facing different network risks and in different system scales respectively by changing the reference probability of abnormal output P_{base} and the number of executors in S_r. Furthermore, we can learn whether the OSSSA has better security performance and reliability than the comparison algorithms in different practical application scenarios.

We conduct six experiments with P_{base} set from 0.01 to 0.25, and the rest of the experimental conditions are the same as those in Part A of this section. Besides, each experiment also contains 100,000 scheduling times T. When faced with varying degrees of network security risks, the experimental results of reliability and anti-attack performance of the three scheduling algorithms are shown in Fig. 6.

In Fig. 6(a) and Fig. 6(b), we can find that with the increase in network risks, the average S_C of the online executors sets selected by the three algorithms all tend to stabilize at 50-100 after a sharp drop at the beginning, and the average S_C of the OSSSA is always higher than that of the RSSSA-F by more than 5%, higher than that of the CRSSA by more than 15%. The experimental results not only verify that S_C can indeed reflect the reliability of the online executors, but also indicate that the OSSSA is more reliable than the other two mainstream scheduling strategy algorithms when facing different network security risks. Fig. 6(c) shows that the mimic defense system with the OSSSA can run normally for

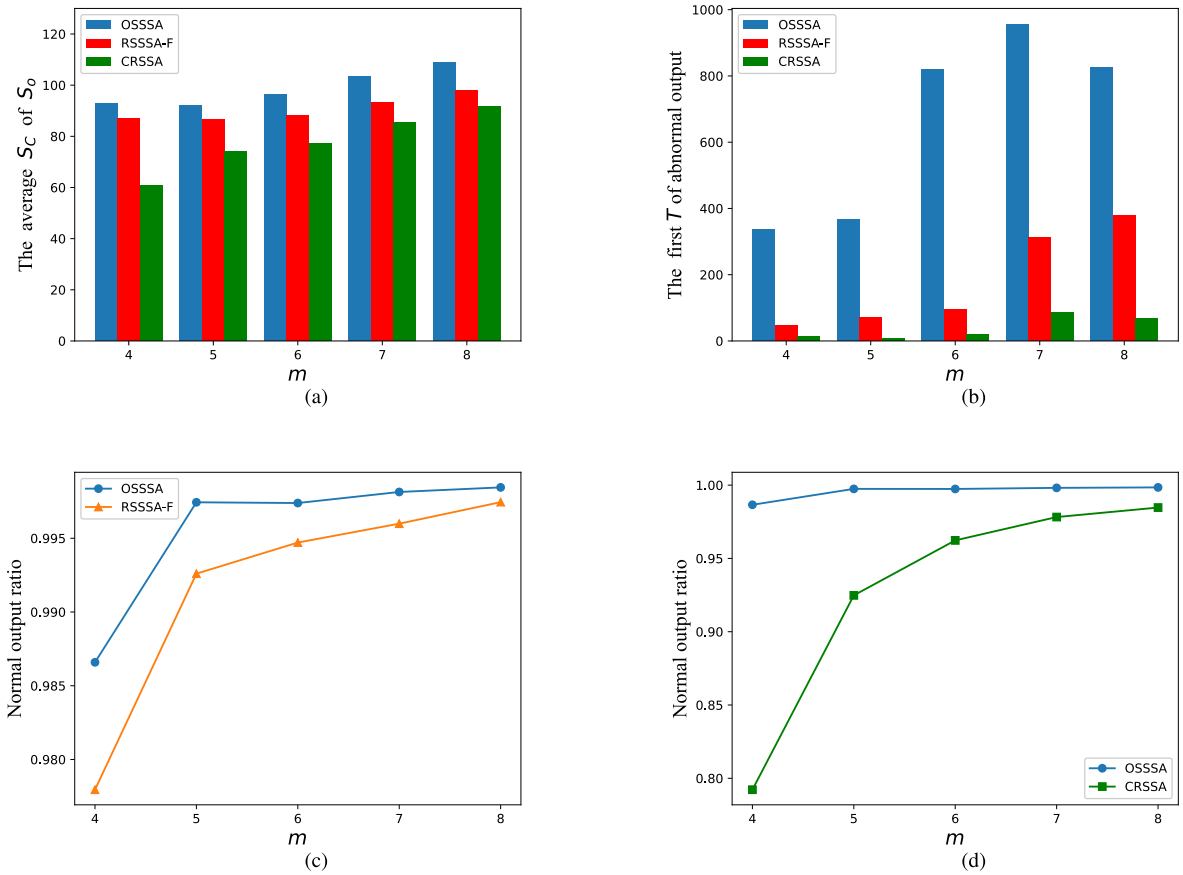


FIGURE 7. Algorithm universality when applied to mimic defense systems with different scales. (a) The average S_C of the online executors sets. (b) The first time of abnormal output. (c) Normal output ratio of the OSSSA and the RSSSA-F. (d) Normal output ratio of the OSSSA and the CRSSF.

a longer time and show stronger reliability at the early stage of system working process than the system with the RSSSA-F or the CRSSA under different network risks. By analyzing the data in Fig. 6(d) and Fig. 6(e), we find that the CMD can effectively improve the security performance of key network equipment in different practical scenarios, and the normal output ratio of the mimic defense system adopting the OSSSA is higher than that of the mimic defense system adopting the RSSSA-F or the CRSSA. For example, when P_{base} is 10%, the experimental data of the OSSSA, the RSSSA-F, and the CRSSA are respectively 99.260%, 98.575%, and 94.406%. This experiment verifies that compared with the RSSSA-F and the CRSSA, the OSSSA can further improve the ability of a mimic defense system to resist network attack under different degrees of network risk.

In order to verify the universality of the scheduling strategy algorithm applied in mimic defense systems of different scales, we conduct 5 experiments with the number of executors m in the executors resource pool S_r from 4 to 8, and the other experimental conditions are the same as the parameters set in Part A of this section. The experimental results are shown in Fig. 7. From Fig. 7(a), we can find that the average S_C of online executors sets obtained by the OSSSA mechanism is stable and always higher than the average S_C obtained by the other two scheduling strategy algorithms in mimic defense systems with different scales. Fig. 7(b) shows that under different system scales, the systems that apply the OSSSA can always safety run for a longer time. Fig. 7(a) and Fig. 7(b) verify that the OSSSA has high reliability in mimic defense systems of different scales. In Fig. 7(c) and Fig. 7(d), we can make a conclusion that when the number of redundant executors m is 4-8, the OSSSA always maintains the advantage of security performance compared to the RSSSA-F and the CRSSA, and its normal output ratio is always higher than 98.5%.

To show more clearly the advancement of the OSSSA in terms of reliability and security performance, we respectively summarize the experimental data of the two main evaluation indicators, the average S_C of the online executors sets, and the normal output ratio of the entire system in Table 7 and Table 8.

4) VERIFICATION OF THE FEEDBACK MECHANISM

This experiment is used to verify the reasonableness of the feedback mechanism in the OSSSA. According to the continuous-time Markov process, a reasonable feedback mechanism is a key factor to improve the reliability and security of the scheduling strategy algorithm. Therefore, this

TABLE 7. The average S_C of the online executors set in experiments.

	OSSSA	RSSSA-F	CRSSA
P_{base}=0.05, m=4	92.9876	86.827762	60.937696
P_{base}=0.05,m=5	91.96162	86.519966	73.976143
P_{base}=0.05, m=6	96.419425	88.093555	77.3079955
P_{base}=0.05, m=7	103.215525	93.154566	85.374101
P_{base}=0.05, m=8	108.982214	97.859388	91.865225
P_{base}=0.01, m=7	277.497593	253.726596	231.97994
P_{base}=0.10, m=7	75.740233	70.411964	63.687778
P_{base}=0.15, m=7	67.864935	63.706689	57.738872
P_{base}=0.20, m=7	63.976865	60.233938	53.99923
P_{base}=0.25, m=7	60.636743	57.312421	51.554529

TABLE 8. The normal output ratio of in experiments.

	OSSSA	RSSSA-F	CRSSA
P_{base}=0.05, m=4	0.98659	0.97795	0.79222
P_{base}=0.05,m=5	0.99744	0.9926	0.9248
P_{base}=0.05, m=6	0.99739	0.99471	0.9623
P_{base}=0.05, m=7	0.99814	0.99599	0.97822
P_{base}=0.05, m=8	0.99845	0.99745	0.9847
P_{base}=0.01, m=7	0.99987	0.99957	0.99653
P_{base}=0.10, m=7	0.9926	0.98575	0.94406
P_{base}=0.15, m=7	0.98591	0.97477	0.9207
P_{base}=0.20, m=7	0.97469	0.95581	0.87362
P_{base}=0.25, m=7	0.94499	0.9205	0.8271

experiment verifies the reasonableness of the feedback mechanism in the OSSSA from the perspective of controllability and real-time differentiated feedback. Under different time scales in the reliability experiment for scheduling strategy algorithms, the fluctuations of feedback indicator S_{fb}^1 of the executor with number *1* in mimic defense system adopting the OSSSA mechanism are shown in Fig. 8.

From Fig. 8(a), it is not difficult to find that the feedback indicator S_{fb}^1 obtained by the feedback mechanism in the OSSSA has upper and lower bounds, that is, the numerical range is controllable. From Fig. 8(b), we can see that the feedback indicator S_{fb}^1 is continuous in the scheduling moments and can reflect the practical security states of executor E_1 in real-time. In addition, the trends in Fig. 8(a) and 8(b) verify that when an executor is selected into the online executors set S_o and works normally, its feedback indicators will increase, whereas its feedback indicator will become negative when it is breached by attackers. This experiment verifies that the practical trend of S_{fb} conforming to the design idea of the feedback mechanism in section IV, and the complete and reliable feedback mechanism proposed by us can better differentiate the real-time security of the executors, which helps mimic defense system resist the threats from coordinated attacks and APT.

D. EXPERIMENTS UNDER MIMIC DNS SYSTEM

In this part, we will introduce the real experiments based on our own mimic DNS system in detail.

1) BASIC INFORMATION OF THE EXPERIMENTS

For describing the real experiments in detail, we first introduce the basic information about the experiments and mimic

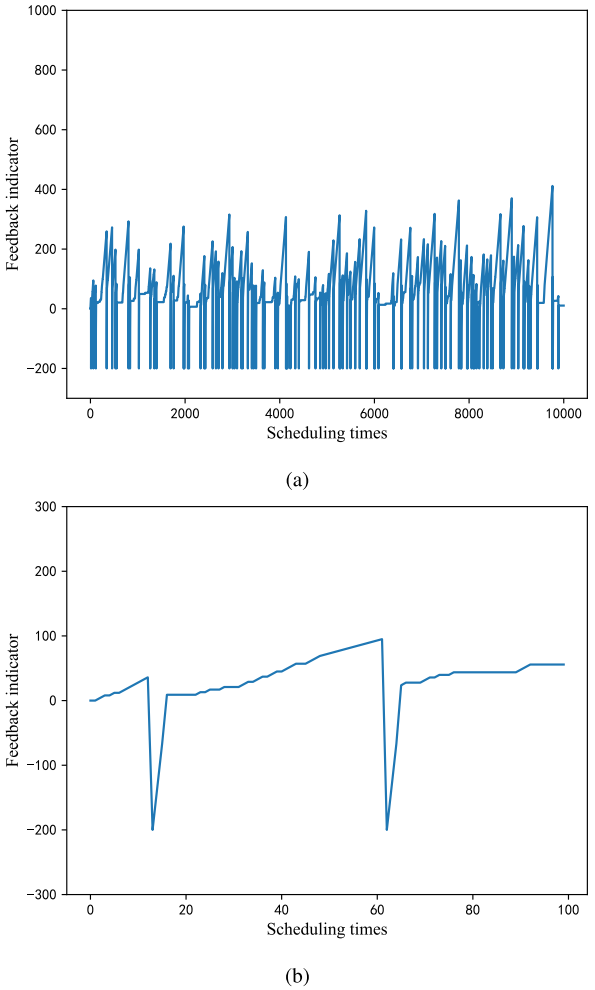


FIGURE 8. The feedback indicator S_{fb}^{1t} under different time scales. (a) 10,000 scheduling moments. (b)100 scheduling moments.

DNS system. The topology of the real network attack experiments is shown as Fig. 9. In the mimic DNS system, the agent includes input module, output module, scheduling controller, and voting module, which forms the system with five DNS executors and provides reliable DNS service to clients.

Besides, The information of each component in the mimic DNS system is shown in Table 9.

TABLE 9. The information of the mimic DNS system.

	Application	Operating System	Kernel Version	Ip Address
Agent	–	Ubuntu 18.04	5.4.30	116.57.115.98
Executor 1	Bind 9	Windows 7	6.1.7601	116.57.115.80
Executor 2	dnspod-sr	Ubuntu 16.04	4.15.0	116.57.115.81
Executor 3	PowerDNS	Ubuntu 18.04	5.4.0	116.57.115.82
Executor 4	Unbound	CentOS 7.8	3.10.0	116.57.115.83
Executor 5	Bind9	Ubuntu 18.04	5.4.0	116.57.115.84

We set the feedback gain g contained in the feedback mechanism as 2, the other parameters contained in the OSSSA that need to be initialized are set to the same value as the corresponding parameters in Table 6. Furthermore, in order to verify the defense performance of the different algorithms

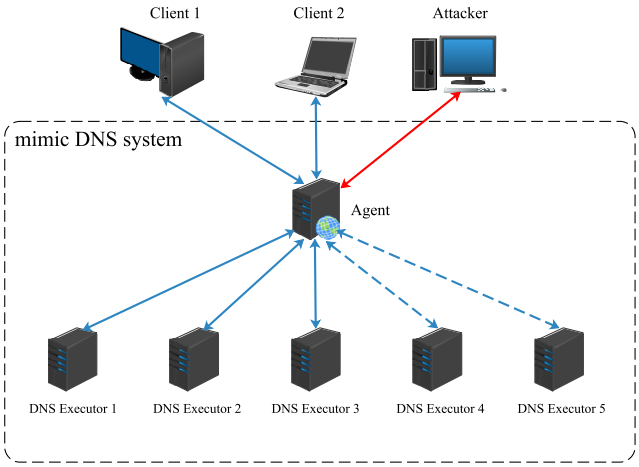


FIGURE 9. The topology of the experiments.

when the mimic DNS system faces different network risks, we set the attack traffic sent by the attacker as 5% and 10% and conducted two groups of experiments respectively. The attack traffic in the experiments refers to the queries that can modify the results of one or more online executors.

2) EXPERIMENTAL PROCESSES AND RESULTS

In the specific experiment process, we first detected the information of each online executor by using the agent as a springboard and continuously penetrating the system, and then exploited the vulnerability of each executor as shown in Table 10 to poison the cache of the corresponding DNS executors. Finally, we launched coordinated attacks randomly and successfully breached the mimic DNS system by tampering the system response with error ip address.

In the real network experiments, we set the CRSSA and the MTD mechanism as the comparison algorithms. We can verify the advantage of OSSSA in improving the real-time security performance by comparing it with the CRSSA. In addition, we can verify that whether CMD is an active defense mechanism with better security performance by comparing with a MTD system degraded from CMD. In each group of the real experiments, the total number of domain name queries is 10000 for each algorithms.

Moreover, we introduce the defensive efficiency η_d expressed by formula (20) to better show the improvement of system defense capability brought by different algorithms and defense mechanisms. The improvement is caused by the redundancy of the executors and the dynamics of scheduling.

$$\eta_d = 1 - \frac{abnormal\ output\ times}{attack\ traffic}. \tag{20}$$

Table 10 and Table 11 respectively shows the experimental results of real experiments when the attack traffic accounts for 5% and 10% of the total queries.

Table 11 and Table 12 clearly show that compared with the mimic defense system deployed the CRSSA, the mimic DNS system deployed the OSSSA has higher average S_C, normal output ratio, and defensive efficiency. The experimental results prove that deploying the OSSSA in a practical mimic system can significantly improve the reliability and anti-attack of the system. In addition, the experimental results of the OSSSA and the CRSSA are also better than those for the MTD, which shows that the CMD theory can indeed improve the network defense capability of network key facilities compared with the mature MTD. Furthermore, it can be concluded from Table 12 that the OSSSA is able to guarantee the mimic DNS system with high-security performance even when the system faced huge network risk, which verifies that the OSSSA has strong robustness.

In summary, by deploying OSSSA in a practical DNS system and conducting real experiments, we not only verify the advancement of OSSSA in enhancing the defense capability of the system, but also prove the reliability of CMD theory.

TABLE 10. The vulnerability of each executor.

	Vulnerability	Method of attack
Executor 1	CVE-2017-0148	By scanning port 445 and sending message, the code is executed remotely to modify the DNS cache.
Executor 2	The source ports of DNS packets are determined	Forge DNS response message to launch Kaminsky attack and modify DNS cache.
Executor 3	CVE-2019-3807	Exploit this vulnerability can bypass DNSSEC validation.
Executor 4	CVE-2018-14634	Execute the SUID-root program and cause integer overflow of the Linux kernel, which can obtain the root permission of the system.
Executor 5	CVE-2020-25705	Extend the attack window and utilize the ICMP error reply mechanism, which can launch cache poisoning attack on DNS [50].

TABLE 11. The experimental results when the attack traffic is 5%.

	The average S_C of online executors	Normal output ratio	η_d
OSSSA	125.6478	0.9961	0.9220
CRSSA	104.2502	0.9920	0.8400
MTD	97.7798	0.9854	0.7080

TABLE 12. The experimental results when the attack traffic is 10%.

	The average S_C of online executors	Normal output ratio	η_d
OSSSA	107.8040	0.9937	0.9370
CRSSA	97.7798	0.9824	0.8240
MTD	72.8334	0.9700	0.7000

VI. CONCLUSION AND FURTHER STUDY

This paper proposes the OSSSA mechanism for the mimic defense system, which has strong reliability, security, and universality performance, and also verifies its performance by conducting simulation experiments and real experiments.

We first introduce the phenomenon of information asymmetry between attackers and defenders in cyberspace security, and a new active defense method called cyberspace mimic defense theory. After that, we elaborate on the importance of the scheduling strategy algorithm in mimic defense theory and its current research achievements. In order to make a deep study on the scheduling strategy algorithm, we model a mimic defense system, and then use a continuous-time Markov process to mathematically analyze the working states

of the mimic defense system. In section IV, we propose the OSSSA mechanism on the basis of the mathematical research, and describe it in detail by giving its key design factors, calculation methods, functional modules, and algorithm flow analysis. Besides, we also describe the working process of feedback mechanism included in the OSSSA in detail. In the experimental part, by conducting software simulations and real experiments, we not only compare the performance of the OSSSA with the current mainstream scheduling strategies in detail, but also further deploy the OSSSA in a practical mimic DNS system and launch real attacks.

The experimental results show that the scheduling strategy algorithm proposed by us indeed has stronger reliability and better security performance than the current mainstream algorithms in this field, and has the ability to improve the defense effect of mimic defense systems in different network conditions. In addition, by comparing with the CRSSA and the RSSSA-F, we verify the necessity and effectiveness of the three key factors and the working mechanism in the OSSSA. Moreover, in section V, we also verify that the feedback mechanism contained in the OSSSA can indeed differentiate feedback the security of the executors in real-time. Furthermore, in the real experiments, the experimental results are close to the results of simulation experiments, which verifies the reliability of the simulation experiment, the advancement of the OSSSA, and the reliability of CMD.

In general, the OSSSA comprehensively considers and evaluates the different key factors of the mimic defense system in different phases. Therefore, the OSSSA can achieve an optimal combination of executors during the scheduling moments by improving the security and reliability of the entire mimic defense system. Besides, the OSSSA mechanism contains some weight parameters, which can not only adjust the importance of each key factor, but also enhance the flexibility of the algorithm in different practical application scenarios. It should be pointed out that for the OSSSA is a complete and reliable scheduling mechanism that focuses on defense performance, it requires higher resource expense than other mainstream scheduling strategy algorithms. In some practical scenarios with low security requirements, the advantage of the OSSSA in defense performance may not be so obvious.

Looking forward to the future, we will do further research based on the OSSSA mechanism proposed in this paper, and put forward a more reasonable and concise scheduling mechanism to improve security performance and operational efficiency of mimic defense system. Furthermore, we will integrate the research results with more cyberspace security scenarios closely, and help promote CMD to the industry. We firmly believe that our research work, the OSSSA and its feedback mechanism, are helpful to the academic research on the scheduling mechanism of the mimic defense system and the development of CMD, and are also helpful to reverse the grim situation of information asymmetry between the attackers and defenders caused by unknown network weaknesses in cyberspace.

REFERENCES

[1] G. Li, W. Wang, K. Gai, Y. Tang, B. Yang, and X. Si, "A framework for mimic defense system in cyberspace," *J. Signal Process. Syst.*, vol. 93, nos. 2–3, pp. 169–185, Mar. 2021, doi: 10.1007/s11265-019-01473-6.

[2] A. Bindra, "Securing the power grid: Protecting smart grids and connected power systems from cyberattacks," *IEEE Power Electron. Mag.*, vol. 4, no. 3, pp. 20–27, Sep. 2017, doi: 10.1109/MPEL.2017.2719201.

[3] S. Thielman, "Yahoo hack: 1bn accounts compromised by biggest data breach in history," *Guardian*, vol. 15, p. 2016, Dec. 2016.

[4] J. Zhu, W. Song, Z. Zhu, J. Ying, B. Li, B. Tu, G. Shi, R. Hou, and D. Meng, "CPU security benchmark," in *Proc. 1st Workshop Secur.-Oriented Designs Comput. Archit. Processors*, Jan. 2018, pp. 8–14.

[5] A. M. Sawas, H. Khani, and H. E. Z. Farag, "On the resiliency of power and gas integration resources against cyber attacks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 5, pp. 3099–3110, May 2021, doi: 10.1109/TII.2020.3007425.

[6] N. Naik, P. Jenkins, N. Savage, and L. Yang, "Cyberthreat hunting—Part 1: Triaging ransomware using fuzzy hashing, import hashing and YARA rules," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, New Orleans, LA, USA, Jun. 2019, pp. 1–6.

[7] J. Wu, "Security risks from vulnerabilities and backdoors," in *Cyberspace Mimic Defense*. Berlin, Germany: Springer, 2020, pp. 3–38.

[8] M. J. Ranum, "Thinking about firewalls," in *Proc. 2nd Int Conf. Syst. Netw. Secur. Manag.*, vol. 8, 1993, pp. 1–10.

[9] J. P. Anderson, "Computer security threat monitoring and surveillance," James P. Anderson Company, Fort Washington, MD, USA, Tech. Rep., Feb. 1980.

[10] D. Sequeira, "Intrusion prevention systems: Security's silver bullet?" *Busin Commun. Rev.*, vol. 33, no. 3, pp. 36–41, 2003.

[11] C. Phillips and L. P. Swiler, "A graph-based system for network-vulnerability analysis," in *Proc. Workshop New Secur. Paradigms*, Charlottsville, VA, USA, 1998, pp. 71–79.

[12] The White House, "Trustworthy cyberspace: Strategic plan for the federal cybersecurity research and development program," Executive Office President Nat. Sci. Technol. Council, Washington, DC, USA, 2011.

[13] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: Transparent moving target defense using software defined networking," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Helsinki, Finland, 2012, pp. 127–132.

[14] R. Zhuang, S. A. DeLoach, and X. Ou, "Towards a theory of moving target defense," in *Proc. 1st ACM Workshop Moving Target Defense (MTD)*, Scottsdale AZ, USA, 2014, pp. 31–40.

[15] T. Hobson, H. Okhravi, D. Bigelow, R. Rudd, and W. Streilein, "On the challenges of effective movement," in *Proc. 1st ACM Workshop Moving Target Defense (MTD)*, Scottsdale AZ, USA, 2014, pp. 40–50.

[16] G. Cai, B. Wang, T. Wang, Y. Luo, X. Wang, and X. Cui, "Research and development of moving target defense technology," (in Chinese), *J. Comp. Dev.*, vol. 53, no. 5, pp. 968–987, 2016, doi: 10.7544/issn1000-1239.2016.20150225.

[17] W. Guo, Z. Wu, F. Zhang, and J. Wu, "Scheduling sequence control method based on sliding window in cyberspace mimic defense," *IEEE Access*, vol. 8, pp. 1517–1533, 2020, doi: 10.1109/ACCESS.2019.2961644.

[18] J. Wu, "Research on cyber mimic defense," (in Chinese), *J. Cyber Secur.*, vol. 1, no. 4, pp. 1–10, Oct. 2016, doi: 10.19363/j.cnki.cn10-1380/tn.2016.04.001.

[19] H. Hu, J. Wu, Z. Wang, and G. Cheng, "Mimic defense: A designed-in cybersecurity defense framework," *IET Inf. Secur.*, vol. 12, no. 3, pp. 226–237, Apr. 2018, doi: 10.1049/iet-ifs.2017.0086.

[20] Q. Tong, Z. Zhang, W. Zhang, and J. Wu, "Design and implementation of mimic defense web server," (in Chinese), *J. Softw.*, vol. 28, no. 4, pp. 883–897, 2017.

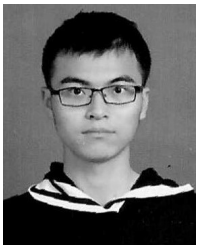
[21] J. Zheng, G. Wu, B. Wen, Y. Lu, and R. Liang, "Research on SDN-based mimic server defense technology," in *Proc. Int. Conf. Artif. Intell. Comput. Sci.*, Beijing, China, Jul. 2019, pp. 163–169.

[22] Y. Cai, D. Pan, and Y. Wang, "The framework study on mimic defense technology in power web service system," *J. Phys., Conf. Ser.*, vol. 1616, Aug. 2020, Art. no. 012099.

[23] Z. Wang, H. Hu, and G. Cheng, "Design and implementation of mimic network operating system," *J. Comp Res. Dev.*, vol. 54, no. 10, pp. 2321–2323, 2017, doi: 10.7544/issn1000-1239.2017.20170444.

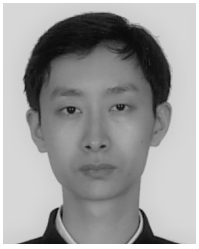
[24] Y.-W. Wang, J.-X. Wu, Y.-F. Guo, H.-C. Hu, W.-Y. Liu, and G.-Z. Cheng, "Scientific workflow execution system based on mimic defense in the cloud environment," *Frontiers Inf. Technol. Electron. Eng.*, vol. 19, no. 12, pp. 1522–1536, Oct. 2018.

[25] Q. Ren, J. Wu, and L. He, "Performance modeling based on GSPN for cyberspace mimic DNS," *Chin. J. Electron.*, vol. 29, no. 4, pp. 738–749, Jul. 2020.
[26] J. Wu, "Examples of mimic defense application," in *Cyberspace Mimic Defense*. Berlin, Germany: Springer, 2020, pp. 597–641.
[27] C. Tankard, "Advanced persistent threats and how to monitor and deter them," *Netw. Secur.*, vol. 2011, no. 8, pp. 16–19, Aug. 2011.
[28] Z. Lu, F. Chen, and G. Cheng, "Design and implementation of the controller scheduling-time in SDN," (in Chinese), *Chine J. Netw. Inf. Sec.*, vol. 4, no. 1, pp. 36–44, Jan. 2019, doi: 10.11959/j.issn.2096-109x.2018003.
[29] Z. Gu, X. Zhang, and S. Wei, "Adaptive dynamic defense mechanism based on reinforcement learning," (in Chinese), *J. Chin. Comput. Syst.*, vol. 40, no. 2, pp. 401–406, Feb. 2019.
[30] X. Sun, Q. Li, S. Zhou, and C. Sun, "Research on mimic defense technology and security test method of electric power web service system," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 569, no. 4, 2019, Art. no. 042011.
[31] Q. Ren, J. Wu, and L. He, "Research on mimic DNS architectural strategy based on generalized stochastic Petri net," (in Chinese), *J. Inf. Sec.*, vol. 4, no. 2, pp. 37–52, 2019, doi: 10.19363/J.cnki.cn10-1380/tn.2019.03.05.
[32] J. Wu, "Testing and evaluation of the mimic defense principle verification system," in *Cyberspace Mimic Defense*. Berlin, Germany: Springer, 2020, pp. 643–681.
[33] Q. Liu, S. Lin, and Z. Gu, "Heterogeneous redundancies scheduling algorithm for mimic security defense," *J. Commun.*, vol. 39, pp. 188–198, Jul. 2018.
[34] J. Zhang, J. Pang, Z. Zhang, M. Tai, H. Zhang, and G. Nie, "Executors scheduling algorithm for web server with mimic structure," (in Chinese), *Comp. Eng.*, vol. 45, no. 8, pp. 14–21, 2019.
[35] X. Wang, W. Yang, W. Zhang, and Z. Yang, "Research on scheduling strategy of mimic web server based on BSG," (in Chinese), *J. Commun.*, vol. 39, no. Z2, pp. 112–120, 2018.
[36] Q. Zhang, H. Tang, W. You, and L. Pu, "Dynamic scheduling strategies of NFV mimic defense architecture based on evolutionary game," (in Chinese), *Comp. Eng.*, to be published, doi: 10.19678/j.issn.1000-3428.0061282.
[37] L. Yang, Y. Wang, and J. Zhang, "FAWA: A negative feedback dynamic scheduling algorithm for heterogeneous executor," (in Chinese), *Comput. Sci.*, vol. 48, no. 8, pp. 284–290, Aug. 2021, doi: 10.11896/jsjkx.200900059.
[38] Z. Wu and J. Wei, "Heterogeneous executors scheduling algorithm for mimic defense systems," in *Proc. IEEE 2nd Int. Conf. Comput. Commun. Eng. Technol. (CCET)*, Beijing, China, Aug. 2019, pp. 279–284.
[39] Z. Zhang, "Research on dynamic scheduling strategy for mimic defense," (in Chinese), Ph.D. dissertation, Zhengzhou Univ., Zhengzhou, China, 2018.
[40] M. Gao, J. Luo, Y. Zhou, H. Jiao, and L. Ying, "A differential feedback scheduling decison algorithm on mimic defense," (in Chinese), *Telecommun. Sci.*, vol. 36, no. 5, pp. 73–82, 2020.
[41] W. Zhang, S. Wei, L. Tian, K. Song, and Z. Zhu, "Scheduling algorithm based on heterogeneity and confidence for mimic defense," *J. Web Eng.*, vol. 19, nos. 7–8, pp. 971–997, Dec. 2020, doi: 10.13052/jwe1540-9589.19783.
[42] Z. Chen, G. Cui, L. Zhang, X. Yang, H. Li, Y. Zhao, C. Ma, and T. Sun, "Optimal strategy for cyberspace mimic defense based on game theory," *IEEE Access*, vol. 9, pp. 68376–68386, 2021, doi: 10.1109/ACCESS.2021.3077075.
[43] C. A. Petri, "Communication with automata," Rome Lab., New York, NY, USA, Tech. Rep. RADC-TR-65-377, 1966.
[44] Z. Lu, Z. Zhang, L. Zhuang, and J. Zhou, "Reliability model of the fly-by-wire system based on stochastic Petri net," *Int. J. Aerosp. Eng.*, vol. 2019, pp. 1–12, Nov. 2019, doi: 10.1155/2019/2124836.
[45] J. Wu, *Introduction to Cyberspace Mimic Defense*, (in Chinese), vol. 2. Beijing, China: Science Press, 2018, pp. 593–664.
[46] J. Wu, "Revelation of the heterogeneous redundancy architecture," in *Cyberspace Mimic Defense*. Berlin, Germany: Springer, 2020, pp. 207–271.
[47] C. He, *Applied Stochastic Processes*, (in Chinese). Guangzhou, China: South China Univ. of Technology, 2017, pp. 144–212.
[48] G. Chen, G. Shi, L. Chen, X. He, and S. Jiang, "A novel model of mimic defense based on minimal L-order error probability," *IEEE Access*, vol. 8, pp. 180481–180490, 2020, doi: 10.1109/ACCESS.2020.3024847.
[49] C. L. Hwang and K. Yoon, "Methods for multiple attribute decision making," in *Multiple Attribute Decision Making*. Berlin, Germany: Springer, 1981, pp. 58–191.
[50] K. Man, Z. Qian, Z. Wang, X. Zheng, Y. Huang, and H. Duan, "DNS cache poisoning attack reloaded: Revolutions with side channels," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Oct. 2020, pp. 1337–1350.



ZHUOXING CHEN was born in Guangzhou, Guangdong, China. He received the B.E. degree in communication engineering from Shandong University (SDU), Weihai, Shandong, China. He is currently pursuing the Ph.D. degree in information and communication engineering with South China University of Technology (SCUT), Guangzhou. His research interests include software-defined networking, information security, and time-sensitive networking.

YIQIN LU (Member, IEEE) was born in Zhaoqing, Guangdong, China. He received the Ph.D. degree in electronic circuits and systems from South China University of Technology (SCUT), Guangzhou, China, in 1996. In 2006, he was appointed as a Professor with SCUT. He is currently a Ph.D. Supervisor with the School of Electronic and Information Engineering, SCUT. He is also the Dean of the Leading Group for Cyberspace Affairs, SCUT, the Director of the Network Center of Southern China Region of China Education and Research Network (CERNET), and the President of the Computer Information Network Safety Association of Guangdong Province. His research interests include communication networks, network security, error-correcting code, and the Internet of Things.



JIANCHENG QIN received the bachelor's degree in computer engineering from the School of Computer Science and Technology, Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 1999, the master's degree from the School of Software Engineering, BUPT, in 2008, and the Ph.D. degree from the School of Computer Science, BUPT, in 2011. His major research interests include information security and mobile computing.

ZHE CHENG received the M.S. degree from Central South University, Changsha, Hunan, China. He is currently pursuing the Ph.D. degree with South China University of Technology, Guangzhou, China. His current research interests include computer networks and information security.