CHINA COMMUNICATIONS

A Function-aware Mimic Defense Theory and its Practice

Jiajun He[1], Yali Yuan[2,4,*], Sichu Liang[3], Jiale Fu[1], Hongyu Zhu[2], Guang Cheng[2,*]

[1] School of Mathematics, Southeast University, Nanjing 211189, China
[2] School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China
[3] School of Artificial Intelligence, Southeast University, Nanjing 211189, China
[4] State Key Laboratory for Novel Software Technology, Nanjing University, P.R. China
[*] The corresponding author, email: yaliyuan@seu.edu.cn, chengguang@seu.edu.cn

Abstract: In recent years, network attacks have been characterized by diversification and scale, which indicates a requirement for defense strategies to sacrifice generalizability for higher security. As the latest theoretical achievement in active defense, mimic defense demonstrates high robustness against complex attacks. This study proposes a Function-aware, Bayesian adjudication, and Adaptive updating Mimic Defense (FBAMD) framework for addressing the current problems of existing work including limited ability to resist unknown threats, imprecise heterogeneous metrics, and over-reliance on relatively-correct axiom. FBAMD incorporates three critical steps. Firstly, the common features of executors' vulnerabilities are obtained from the perspective of the functional implementation (i.e., input-output relationships extraction). Secondly, a new adjudication mechanism considering Bayesian theory is proposed by leveraging the advantages of both current results and historical confidence. Furthermore, posterior confidence can be updated regularly with prior adjudication information, which provides mimic system adaptability. The experimental analysis shows that FBAMD exhibits the best performance in the face of different types of attacks compared to the state-of-the-art over real-world datasets. This study presents a promising step toward the theoretical innovation of mimic defense.

Keywords: mimic defense, Bayesian theory, functional implementation, confidence

Received: Dec. 02, 2022
Revised: Apr. 29, 2023
Editor:

I. INTRODUCTION

With the rapid development of mobile terminals and information technology, cyberspace has emerged as a powerful platform for the Internet of Things. At the same time, the main challenge faced by diversified cyberspace is differentiated security threats. Namely, increasingly complex security scenarios require defense strategies to sacrifice generalizability for higher security, making it difficult to migrate defense strategy within different domains. From the defender's perspective, deploying a targeted defense strategy is essentially an Attack-Defense Game, and it is easier to attack than to defend, especially with unknown attackers. However, the security research currently conducted, i.e. firewalls [1], malware detection [2–4], intrusion detection [5–7], intrusion tolerance [8], and other passive defense techniques, can only cope with attacks whose characteristics are known.

In fact, there is no absolutely secure system. It's unfeasible to prove that the defense technology imposed on the system is trustworthy and free of vulnerabilities. On the premise of the inability to circumvent endogenous flaws in the system, the common challenge encountered in cybersecurity is how to deal with security threats based on unknown vulnerabilities, backdoors, and viruses in the absence of prior knowledge [9]. To solve the above problem, it is necessary to change the defense notion from the traditional "external" to "endogenous", in other words, only by improving the inherent dynamics and uncertainty of the system can we turn "passive" into "active" in network attack and defense.

In recent years, academia and industry have been trying to develop innovative, active defense technologies. By arranging decoys in the system and tricking attackers to perform the attack, the honeypot technique [10–12] achieves the purpose of collecting prior attack knowledge and inferring attack intent. Moving Target Defense (MTD) [13] introduces dynamics in defense systems to increase the dynamics of the attack surface and the uncertainty within the system, which attenuates the attacker's information advantage to a great extent. In practice, MTD has several outstanding results in the areas of power systems [14–16], web servers [17], and cloud environments [18]. However, the honeypot technique requires a large amount of prior knowledge from the attacker [10], MTD is time-sensitive and uncontrolled, and the high-frequency variability specifically makes the system performance degraded [19]. As an emerging active defense technique, Cyber Mimic Defense (CMD) [20] additionally considers the heterogeneous and redundant characteristics of the system and introduces adjudication and negative feedback mechanisms to improve the stability of the system. CMD theory has been studied and practiced in distributed storage architectures [21], IoT [22], and cloud [23], showing better defense results.

The core structure of mimic defense is "Dynamic, Heterogeneous, Redundancy" (DHR), implying the following basic ideas: (1) Heterogeneous. The protected object is abstracted into a functional region. Several heterogeneous executors are constructed through diversely redundant design principles to achieve the same function. (2) Dynamic. A scheduling strategy is used to select the set of executors that realize the function at the current moment, shifting and changing the set over time to hide internal structure. (3) Redundancy. A voting-based adjudication mechanism is used to decide which executor has the correct output and whether the mimic system is under attack.

There are several momentous problems in the current research [24–27] on the traditional DHR. (1) The division of functional regions of executors is homogeneous. Most studies in the field of mimic defense only focused on representing functional regions by a symbol set. Such expositions are unsatisfactory because they cannot complete detailed metrics. (2) The research to date has tended to achieve a high degree of dynamic rather than validity. Besides, almost no researchers use historical information (usually inferring the attacker's propensity) to improve DHR's performance. (3) The voting-based adjudication mechanism based on a relatively-correct axiom [9] has limitations in several settings. The traditional voting strategy does not take account of an even set of executors, nor does it resist the attacker's concerted attack.

To solve the aforementioned problems, this paper proposes a novel mimic defense framework FBAMD based on Bayesian theory. Our contributions mainly include the following aspects:

- In view of the functional implementation, we extracted the common features of vulnerabilities from the input-output relationships through executors, which provides a fine-grained definition for the executors' functions.
- Considering the intersection and difference of the mapping sets (i.e., input-output relationships), the heterogeneity metric is defined in conjunction with the Common Vulnerability Scoring System (CVSS), in addition, the confidence metric that can adaptively update with tasks is designed.
- An adjudication mechanism independent of the relatively-correct axiom is proposed by introducing the concept of Bayesian theory into the arbiter while the posterior confidence is calculated by leveraging the advantages of historical information to obtain trust results.
- We examined the effectiveness of our FBAMD on real-world datasets, e.g., the Darknet2020 dataset [28]. Numerous experimental results show that FBAMD has the highest classification accuracy. We also analyze the causes of the jitter phenomenon and introduce the method of setting parameters to make the system more stable. Our code is available at https://github.com/dbsxfz/FBMAD.

The paper is organized in the following way. Section II gives the related works. Section III introduces the new definition of DHR from the perspective of functional implementation. Section IV presents the calculation methods of heterogeneity and confidence respectively. Section V begins by laying out the application of Bayesian theory and looks at how to update posterior confidence. Section VI analyzes the security, stability, and efficiency of FBAMD. Finally, the conclusion is drawn in section VII.

II. RELATED WORK

Relevant prior work focuses on the current research status of different defense concepts in cyberspace, including both passive and active defense.

2.1 Passive Defense Concept

Most of the research on security up to now has been passive in nature. "Passive" commonly refers to the strategy that targeted an effective defense against attacks with certain characteristics. For instance, recent developments in the field of artificial intelligence have contributed to a new round of innovation in passive defense technology. Esmaeilpour et al. [29] evaluated the resilience of Support Vector Machines (SVM) in audio classification against adversarial attacks and proposed a defense strategy based on a preprocessing module. Al-Haija et al. [30] developed an intelligent classification model based on machine learning methods to assist in identifying malicious packets in firewall systems. Compared with ordinary deep learning, Lian et al. [31] incorporated multimodal features into deep learning, which contributes to multiple features and adaptive learning. Wang et al. [32] introduced a novel learning framework for capturing the structural features of programs in accordance with Graph Neural Networks (GNN), which combines probabilistic learning and statistical evaluation to detect code vulnerabilities. Xu et al. [7] designed a Deep Neural Network (DNN) based on a meta-learning framework for few-shot network intrusion detection.

Recent evidence suggests that the widespread application of machine learning technology in security has spawned a considerable number of attacks targeting machine learning. Within the domain of artificial intelligence, literature has emerged that offers contradictory findings about the security of learning algorithms. Chen et al. [2] proposed an adversarial attack against machine learning that can successfully deceive detectors. Jagielski et al. systematically studied poisoning attacks against linear regression models and proposed a considerably robust TRIM (a defense algorithm against poisoning attacks in [33]). Wen et al. [34] presented a novel defense algorithm termed Proda by introducing the probability estimation of clean data points into the algorithm for the first time. Compared with TRIM, Proda dominates both effectiveness and time complexity. Weerasinghe et al. [35] introduced a new approximation of Local Intrinsic Dimensionality (LID) for characteristic discrimination, which effectively protects support vector machines from poisoning attacks and label flips. Flowers et al. [36] evaluated adversarial machine learning attacks against deep learning in a wireless communication environment.

However, questions have been raised about the application of artificial intelligence technology in security. It is unrealistic for defenders to be certain that the defensive modules they impose are absolutely secure. Alternatively, even the patch has some kind of vulnerability, which just hasn't been discovered and exploited by attackers yet. To break the security chain of suspicion once and for all, the defensive mindset needs to be shifted to active.

2.2 Active Defense Concept

Different from the perspective of passive defense, active defense pays little attention to the characteristics of the attacker. It is more concerned with strengthening the system itself rather than dealing with the attacker. Current research on active defense focuses on honeypot technology, moving target defense, and mimetic defense.

A honeypot is a closely monitored computing resource. Deployed as a decoy to examine attackers and their attack strategies, it offers some important insights into the improvement of cybersecurity infrastructure. Agrawal et al. [12] deployed low and medium-interaction honeypots in a public cloud environment and analyzed the propensity of honeypots with different interaction attributes to be attacked. Shan et al. [11] explored a new honeypot approach trapdoors to protect deep neural network models, which can effectively influence the generation of adversarial examples. Naik et al. [37] introduced Dynamic fuzzy rule interpolation (D-FRI) for the detection and prediction of fingerprinting attacks on honeypots. Learning from current network characteristics, it supports more precise detection based on a dynamically enriched rule. Nevertheless, the effectiveness of a honeypot to get attackers hooked requires a high level of prior knowledge. The development of moving target defense (MTD) has gained fresh prominence with many turning their attention to this dynamic strategy.

By changing its attack surface dynamically, MTD

ensures the security of its own structure. The most significant current discussions in MTD focus on two aspects: modeling applications and performance analysis. On modeling applications, Zhang et al. [15] deployed MTD against False Data Injection (FDI) attacks in a power system and analyzed the strength and running cost of MTD with different numbers of branches. Higgins et al. [14] proposed an unsupervised smart FDI attack, which is shown to be able to maintain stealth in the presence of traditional MTD strategies. Xu et al. [16] systematically studied the design difficulty of MTD in noisy environments and proposed a robust MTD to guarantee the worst-case detection rate for all unknown attacks. Heydari et al. [17] designed an MTD-based anti-censorship framework for the Internet, which significantly reduces the performance overhead. On performance analysis, Jin et al. [18] developed a multi-dimensional attack graphs model to formalize complex attack scenarios and proposed an MTD deployment strategy that can be adaptively evaluated and optimized. Zhang et al. [38] introduced diversity metrics to assess the resilience of MTD against zero-day attacks, which inspired new ideas for the evaluation of MTD. With the profound study on MTD, the inherent problems of this mechanism are exposed as follows. Such approaches, however, have failed to recognize attacks that have already occurred. In addition, the dynamic nature of the MTD imposed on the system leads to its operational inefficiency [19]. To this end, academics hope to find an innovative theory of active defense.

Inspired by the "natural mimicry" phenomenon of animals, Wu et al. [20] proposed Cyberspace Mimic Defense (CMD) theory to deal with unknown vulnerabilities in cyberspace. DHR is the core concept of CMD [39]. DHR constructs a secure structure through the combination of insecure individuals. CMD has promising applications. Yu et al. [21] introduced the DHR mechanism in distributed object storage architecture, which significantly improved data security. Sang et al. [22] established a mimic defense model of the Edge-Computing terminal to prevent IoT nodes from being maliciously forged. Based on CMD, Zhou et al. [23] proposed a negative feedback dynamic scheduling algorithm that implements the real-time monitoring of virtual machines in a cloud environment. To ensure the security of the NFV network, Zhang et al. [40] presented a heterogeneous entity pool construction method by genetic algorithm. Dai et al. [41] introduced the application of mimic defense in industrial control systems (ICS). Li et al. [42] proposed an intelligent flow-forwarding scheme with endogenous security in a software-defined network (SDN) architecture.

The literature on mimic defense technology has highlighted security performance in terms of heterogeneous metrics, scheduling strategy, and adjudication feedback [43, 44]. Zhang et al. [24] combined heterogeneity and confidence to characterize the executors and introduced the Technique for Order Preference by Similarity to an Ideal Solution (TOPSIS) to optimize the scheduling strategy. Li et al. [25] proposed the concept of time and task threshold in scheduling, which is used to design a multi-level queue dynamic scheduling strategy. Wu et al. [26] introduced random seeds to the scheduling, making the replacement of service executors set completely random. Shao et al. [27] summarized the false-positive problem of mimic systems, while the best mimic component set theory is proposed to solve the problem. Essentially, scheduling strategy is the key to DHR, but a dynamic scheduling strategy cannot be separated from the extraction of heterogeneous and redundant characteristics. How to improve the heterogeneous metrics to enhance the dynamics of the scheduling process is a current research hotspot. To solve the problems of large granularity of heterogeneous metrics, inadequate utilization of historical information, and over-reliance on relatively-correct axiom for adjudication mechanism, this paper designs a novel mimic defense system with a new definition of functional mapping.

III. PRELIMINARY

In this section, we first list the notations and abbreviations used in this paper in Table 1 for ease of reading. Furthermore, we introduce the basic structure of DHR and then take a deep dive into our new definition from a functional implementation perspective.

3.1 Basic Structure of DHR

"Dynamic, Heterogeneous, Redundancy" (DHR) is the key structure of mimic defense. Combining functionally equivalent executors with implementations that are structurally heterogeneous in hardware or soft-

Table 1. *Main notations and abbreviations.*

Notation	
n	redundancy of the service set
P_i	the ith executor
$\omega_1^{(i)}$	first type of error of P_i (Def. 8)
$\omega_2^{(i)}$	second type of error of P_i (Def. 9)
$\omega_3^{(i)}$	function mapping of P_i (Def. 10)
$\omega_4^{(i)}$	unknown function mapping of P_i (Def. 11)
$V_k(P)$	all k-order symbiotic vulnerabilities in $\{P\}$ (Eq. (2))
$A_k(P)$	all k-order symbiotic areas in $\{P\}$ (Eq. (4))
$Het(P)$	heterogeneity of $\{P\}$ (Def. 14)
N	current adjudication order
M	the number of valid adjudications ($M = N - 1$)
M_i	task threshold of P_i
$con_i(M)$	confidence factor of P_i after the Mth adjudication (Def. 15)
$\gamma_i(M)$	decay factor of P_i after the Mth adjudication (Def. 16)
D_N	the output information in the Nth adjudication
y_N	the set of output results in the Nth adjudication
$\mathcal{F}_i$	$y_i \in y_N$ is the correct output, $\forall i \in [n^{'}]$
$\mathcal{F}_0$	the correct output is not observed by y_N
$E_m(N)_i$	historical adjudication information (Def. 18)
T_i^+	P_i makes a correct judgment
T_i^-	P_i makes an incorrect judgment
Abbreviation	
FBAMD	Function-aware, Bayesian adjudication, and Adaptive updating Mimic Defense theory
MTD	Moving Target Defense
CMD	Cyber Mimic Defense
DHR	Dynamic, Heterogeneous, Redundancy
StatP	Statistical-based Poisoning Attack [33]
TRIM	a defense algorithm against poisoning attack using a trimmed loss function [33]
alfa	Adversarial Label Flip Attack [35]
K-LID-SVM	a defense algorithm using Kernel distance in the Local Intrinsic Dimensionality calculation [35]
Logi	a poisoning attack against Logistic Regression [45]

ware, a service set is built to achieve the required functions. By adding multi-dimensional dynamic uncertainties to the system, including dynamic scheduling, reconfiguration, and virtualization, DHR makes it exponentially more difficult for attackers to obtain valid information about the system and carry out a targeted attack at the current moment, thus achieving an effective defense against attacks from unknown sources.

The mimic defense system mainly consists of the following components: input agent, service set, waiting set, arbiter, and scheduler, which are shown in Figure 1. During operation, the input agent collects input data and assigns it to heterogeneous executors in the service set. Each executor processes the data independently and maps the input data to the output space, and then the arbiter decides the final output of the mimic defense system from the output space according to a certain strategy. In addition, the information from each adjudication is fed back to the scheduler, which analyzes the feedback and uses an appropriate strategy to select executors from the waiting set to update the service set.

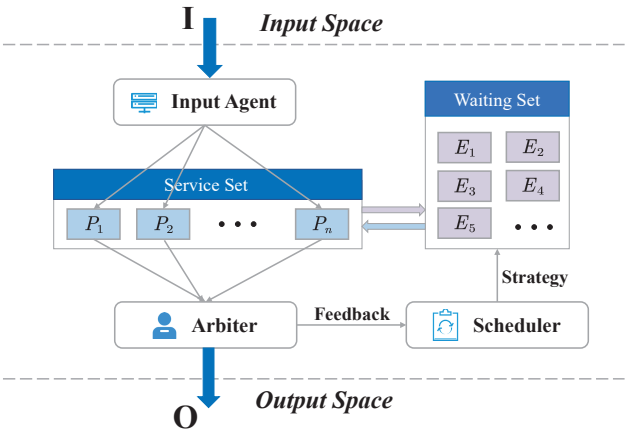

Figure 1. *DHR framework.*

Composed of differentiated hardware and software structures, the executor is the individual that implements the function in DHR. For the implementation of certain functions, the executor is designed and given a specific structure. These functions that determine the structure of an executor are defined as desired functions. However, under some specific input or environmental stimulation, the executor may also implement other functions beyond the desired functions, which are partly explicit side effects and the rest completely unknown. These functions that we do not want to implement are uniformly defined as dark functions. It should be particularly emphasized that even if two executors have the same structure, their actual hardware or software implementations cannot be completely identical. In other words, executors that are expected to have identical functions do not have the same dark functions, which provides the necessary conditions for the heterogeneity and redundancy of executors.

3.2 Functional Implementation

Functional implementation is the theoretical foundation of our FBAMD framework. There is no existing work on defining the mathematical theory of symbiotic vulnerabilities by analyzing the process of functional implementation. We present this essential analytical idea for the first time. The rest of the narrative is developed on this basis.

First, the concept of the Turing machine model is introduced. Turing machine is a mathematical computational model, which defines an abstract machine that

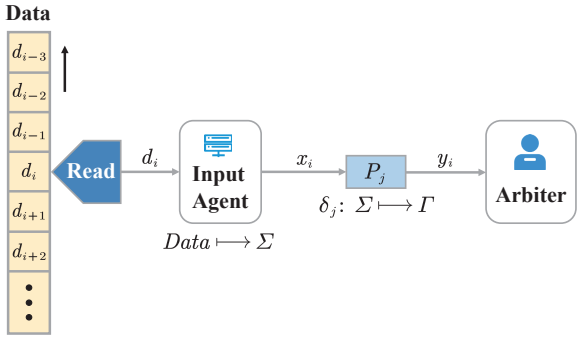

Figure 2. *Functional implementation.*

can read and write data on paper tape according to a specified table of rules. Turing believed that such a machine would be able to simulate any computational process that humans could perform [46]. Inspired by the Turing machine model, the functional implementation of an executor is essentially a replacement for humans to perform tedious computations, and the executor itself does not have the function to generate functions. If the process of functional implementation is analogous to the Turing machine model, the implementation of different functions of an executor can be expressed as different transfer functions obeyed by the machine to perform read and write operations on the data stream. For a finite input space, the output space mapped by the transfer function is also finite, so the mapping rule table of the machine can be extracted from it. In this case, the executor corresponds to its mapping rule table one-to-one. For a better explanation of the above process, we define the important concepts as follows.

Definition 1. *Transfer Function $\delta_i(\cdot)$. The way input data is processed within an executor is abstracted as using a transfer function to map it. We define the transfer function of the ith executor P_i as $\delta_i(\cdot)$. Assuming that P_i reads in data as x and the output of the executor is y, then y satisfies $y = \delta_i(x)$.*

Definition 2. *Input space Σ and output space Γ. The sets Σ, Γ contain all possible occurrences of x, and y, respectively. Provided that the input space Σ and the transfer function $\delta_i(\cdot)$ are known, the output space Γ is uniquely determined.*

Definition 3. *Mapping pair (x, y). We define the binary-ordered group as a mapping pair when the input data x is mapped by a certain transfer function to y.*

Definition 4. *Mapping rule table Ω_i. Provided that the input space Σ and the transfer function $\delta_i(\cdot)$ are known, We define all mapping pairs generated by $\delta_i(\cdot)$ as the mapping rule table for the executor P_i, denoted as Ω_i.*

Definition 5. *Functional implementation. One functional implementation means that the executor reads one unit of data distributed by the input agent and gets the output after mapping by the transfer function. As shown in Figure 2, the input agent can convert the information input to the system into elements in the input space and assign them to the executor, enabling the executor to complete the function mapping.*

Definition 6. *Desired function set $\{(a_o, A_o)\}_{o\in\Lambda_i}$. As users, we want the executor P_i to successfully output A_o on input a_o, then the mapping pair (a_o, A_o) is an element in the desired function set $\{(a_o, A_o)\}_{o\in\Lambda_i}$. Since the desired function is the existing condition for executor P_i, indicator set Λ_i is known.*

Definition 7. *Dark function set. Theoretically, the implementation of any desired function derives explicit side effects or implicit unknown functions, and these non-desired functional parts are defined as the dark function set. The dark function set can be expressed as the mapping rule table removing the desired function set.*

Many researchers have simply utilized symbols to measure function sets, which lacks analysis of the function from the essence. From the perspective of functional implementation, a binary-ordered set with the indicator set known is a refined way of measuring functionality.

The emergence of dark function is the inevitable result of the multifaceted nature of things. The philosophical essence of the endogenous security problem is the structural contradiction of the target object [9]. According to our definition from the functional implementation perspective above, the common features of endogenous security problems can be normalized and expressed as follows: In the service set, functions with explicit side effects within a common subset of dark functions are implemented.

To distinguish the dark function set with explicit side effects, we next divide the mapping rule table in

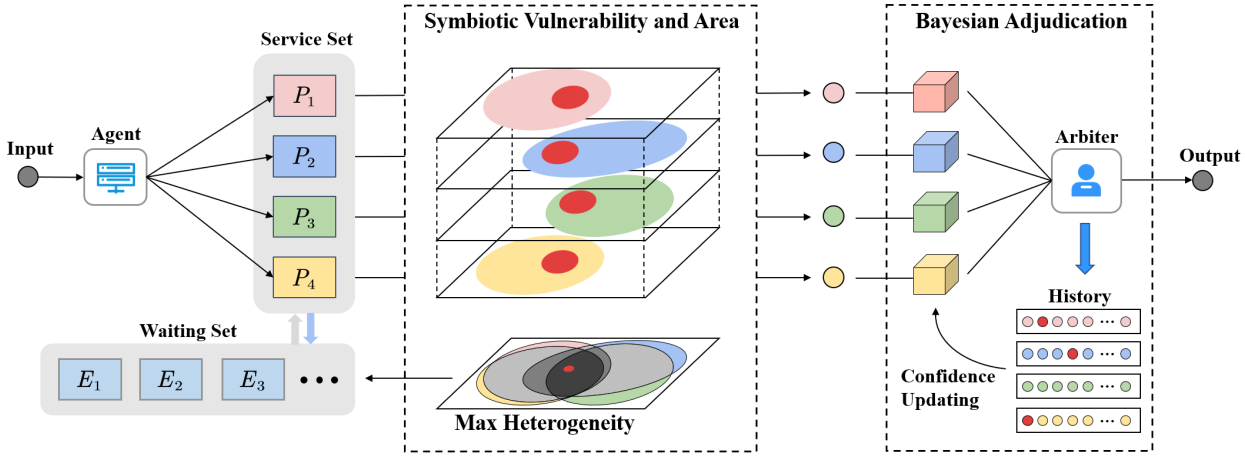

Figure 3. *FBAMD framework.*

more detail. Assume that the desired function set is $\{(a_o, A_o)\}_{o\in\Lambda_i}$.

Definition 8. *First type of error $\omega_1^{(i)}$. We define all mappings similar to (a_o, Y) in the mapping rule table Ω_i as the first type of error $\omega_1^{(i)}$, where $Y \in \Gamma$, $Y \neq A_o$, $\forall o \in \Lambda_i$.*

Definition 9. *Second type of error $\omega_2^{(i)}$. We define all mappings similar to (X, A_o) in the mapping rule table Ω_i as the second type of error $\omega_2^{(i)}$, where $X \in \Sigma$, $X \neq a_o$, $\forall o \in \Lambda_i$.*

Definition 10. *Function mapping $\omega_3^{(i)}$. We define all mappings similar to (a_o, A_o) in the mapping rule table Ω_i as the function mapping $\omega_3^{(i)}$, $\forall o \in \Lambda_i$. Function mapping is essentially part of the mapping rule table that belongs to the desired function set.*

Definition 11. *Unknown function mapping $\omega_4^{(i)}$. We define all mappings similar to (X, Y) in the mapping rule table Ω_i as the unknown function mapping $\omega_4^{(i)}$, where $X \in \Sigma$, $Y \in \Gamma$, $X \notin \{a_o\}$, $Y \notin \{A_o\}$.*

Obviously, $\Omega_i = \omega_1^{(i)} \cup \omega_2^{(i)} \cup \omega_3^{(i)} \cup \omega_4^{(i)}$ and $\omega_1^{(i)}$, $\omega_2^{(i)}$, $\omega_3^{(i)}$, $\omega_4^{(i)}$ mutually disjoint.

According to the above definition, $\omega_1^{(i)}$ and $\omega_2^{(i)}$ are the dark function set with explicit side effects, $\omega_3^{(i)}$ is the desired function set, and $\omega_4^{(i)}$ is the dark function set with implicit functions. In a real-world scenario, the implementation of $\omega_1^{(i)}$ will result in the absence of the desired function, while the implementation of $\omega_2^{(i)}$ will lead to an unexpected implementation of the desired function.

To analyze the safety of the mimic system, a certain degree of simplification is required, which is given in the form of assumptions.

Assumption 1: From the perspective of functional implementation, all mapping pairs have the same status in the mapping rule table.

There are differences in the difficulty of implementation depending on the type of functions. However, mapping pairs are a refined decomposition of functions. The different objectives of the functions are reflected in the different types of mapping pairs, while the different difficulties in implementing the functions are embodied in the different numbers of mapping pairs. Essentially, mapping pairs are the most basic units of function, and there is no metric difference between different kinds of mapping pairs.

Assumption 2: The input space and output space of the executor are both limited.

The Turing machine model restricts its input alphabet to be limited. Extending to any one executor, the data types that can be recognized by its underlying processor are also limited. Consequently, after the mapping of the transfer function, its output space is also limited.

Assumption 3: Mimic system components that can be implemented by hardware logic are theoretically secure.

We assume that the input agent, arbiter, and scheduler as in Figure 1 are secure. The input agent undertakes data read-in and distribution, the arbiter is responsible for data comparison and human-computer interaction, and the scheduler controls component re-

placement of the service set. With simple hardware logic implementations, their security can be proven by formal analysis.

Assumption 4: The attacker will tend to successfully attack more executors.

For a greater influence on the system output results, the attacker will attack as many executors within the service set as possible to make it wrong. In other words, attackers treat the executors equally. Without enough prior knowledge, an attacker does not appear to have the propensity to attack a particular executor.

3.3 FBAMD Framework

As shown in Figure 3, we design three major components in FBAMD: heterogeneity calculation (in service set), confidence and its updating (in executor), and Bayesian adjudication (in arbiter).

(1) **Heterogeneity Calculation (service set):** The heterogeneity of the service set is calculated based on the symbiotic vulnerability (red region in Figure 3) and symbiotic area (grey region in Figure 3) of the executors within the current service set. The heterogeneity is used to determine the strategy for replacing executors. When a replacement occurs, the executor in the waiting set with the highest heterogeneity to the executors in the current service set will be swapped in.

(2) **Bayesian Adjudication (arbiter):** After obtaining the output results for each executor, the arbiter makes a decision based on the confidence of the executor using Bayesian theory. The results of each adjudication are saved in the historical adjudication information.

(3) **Confidence and its Updating (executor):** The confidence of an executor consists of two components, correctness and the number of tasks implemented. After several adjudications, the confidence is regularly updated with historical adjudication information (red dots for errors in Figure 3) by Bayesian theory and executors that drop below the threshold will be replaced.

IV. DESIGN OF EVALUATION METRICS

Most existing work of mimic defense listed in Section II has only designed a heterogeneity metric to capture the characteristics of the service set, without taking into account the specificity of each executor. However, it is necessary to establish a separate metric for each executor to achieve a fine-grained design of the mimic system.

In this section, we introduce and present the calculation methods of heterogeneity and confidence metrics, which are built on the service set and executor respectively.

4.1 Heterogeneity

With the desired function set identical, the DHR structure requires the least intersections of the dark function set between heterogeneous executors in the service set. Consequently, a metric is demanded to evaluate the variability of dark function sets within an executor set, namely, heterogeneity.

Definition 12. *k-order symbiotic vulnerability. In an executor set, we define (x, y) as a k-order symbiotic vulnerability when (x, y) occurs only in k executors' ω_1 or ω_2, $k \geqslant 2$.*

Compared with previous work that used the intersection of the entire dark function set to calculate symbiotic vulnerability [24–26], only ω_1 and ω_2 in the dark function set can lead to vulnerability from the perspective of functional implementation. Next, the intersection part of ω_1 and ω_2 is considered to introduce the calculated approach of the k-order symbiotic vulnerability between an executor set.

For an executor set $\{P\} = \{P_1, P_2, \cdots, P_n\}$ with n redundancy, the set of all subsets of $\{P\}$ with k elements is denoted by $\{\Pi_k\}$. $\{P_k\}_j$ is the element in $\{\Pi_k\}$, where $\left|\{P_k\}_j\right| = k$, $|\{\Pi_k\}| = C_n^k$, $j \in [C_n^k]$. All k-order symbiotic vulnerabilities in $\{P_k\}_j$ can be expressed as,

$$V_k(\{P_k\}_j) = \bigcap_{P_i \in \{P_k\}_j} \omega_1^{(i)} \cup \omega_2^{(i)}, \tag{1}$$

where $i \in [k]$.

To ensure that a $k+1-order$ symbiotic vulnerability is not repeatedly computed as k k-order symbiotic vulnerabilities, it is necessary to remove all elements that recur between $V_k(\{P_k\}_j)$. All k-order symbiotic

vulnerabilities of $\{P\}$ can be expressed as,

$$V_k(P) = \bigcup_{j\in[C_n^k]} V_k(\{P_k\}_j) \setminus \bigcup_{j_1\neq j_2} V_k(\{P_k\}_{j_1}) \cap V_k(\{P_k\}_{j_2}), \tag{2}$$

where $j_1, j_2 \in [C_n^k]$.

Definition 13. *k-order symbiotic area. In an executor set, we define (x, y) as a k-order symbiotic area when (x, y) occurs only in k executors' mapping rule table Ω, $k \geqslant 2$.*

Similarly, all k-order symbiotic areas in $\{P_k\}_j$ can be expressed as,

$$A_k(\{P_k\}_j) = \bigcap_{P_i\in\{P_k\}_j} \Omega_i, \tag{3}$$

where $i \in [k]$.

All k-order symbiotic areas of $\{P\}$ can be expressed as,

$$A_k(P) = \bigcup_{j\in[C_n^k]} A_k(\{P_k\}_j) \setminus \bigcup_{j_1\neq j_2} A_k(\{P_k\}_{j_1}) \cap A_k(\{P_k\}_{j_2}), \tag{4}$$

where $j_1, j_2 \in [C_n^k]$.

If an attacker can perform a coordinated attack, after acquiring the characteristics of the symbiotic vulnerability, a simultaneous attack on the executors within an executor set may successfully cause the arbiter to output an incorrect result. Consequently, to make the probability of symbiotic vulnerabilities between actuators as small as possible, the criteria for selecting an executor set are as follows: (1). Minimize the number of symbiotic vulnerabilities within the executor set. (2). Maximize the number of symbiotic areas within the executor set. The following definition is intended to measure the above conclusions.

Definition 14. *Heterogeneity $Het(P)$. Heterogeneity is a comprehensive static metric for evaluating the variability of dark function sets within an executor set $\{P\}$. The smaller the heterogeneity, the more difficult for attackers to identify the symbiotic vulnerabilities and conduct an effective attack.*

For the calculation of the heterogeneity, we should not only consider the symbiotic vulnerabilities of a certain order but also combine orders from 2 to n comprehensively. Higher-order symbiotic vulnerabilities pose a greater threat to the system, resulting in considerable weight. From the perspective of vulnerability construction, a k-order symbiotic vulnerability can be decomposed into k $k-1$-order symbiotic vulnerabilities, and each $k-1$-order symbiotic vulnerability can be decomposed into $k-1$ $k-2$-order symbiotic vulnerabilities, and so on. Mathematically, a K-order symbiotic vulnerability should be A_K^{K-L} times more threatening than an L-order symbiotic vulnerability $(L < K)$.

For an executor set $\{P\}$ with n redundancy, heterogeneity can be calculated as,

$$Het(P) = 1 - \frac{1}{A_N^N C} \frac{\sum_{k=2}^{N} C_k A_N^k |V_k(P)|}{\sum_{k=2}^{N} |A_k(P)|}, \tag{5}$$

where C_k is the risk indicator for the k-order symbiotic vulnerability. We can obtain C_k through CVSS [47]. $C = 10$ is a normalized constant.

For any $\{P\}$, $Het(P)$ is a real number between $(0, 1)$. With the number of symbiotic vulnerabilities constant, heterogeneity decreases as the proportion of high-order vulnerabilities increases. In contrast, heterogeneity increases monotonously with the expansion of the symbiotic areas owing to the reduction of symbiotic vulnerabilities' proportion. When replacement is required for a mimic defense system, an executor is specially chosen from the waiting set because it can maximize the heterogeneity of the service set.

4.2 Confidence

Different from heterogeneity, which is a security measurement of an executor set, confidence considers the trustworthiness of a particular executor. According to the evaluation criterion of heterogeneity, an executor set with the maximum heterogeneity ensures the best security of the mimic defense system. Nevertheless, if attacks selectively attack certain symbiotic vulnerabilities with partial prior knowledge, the security of the mimic defense system is still considered threatened. To dynamically assess attackers' knowledge of

the mimic system, we propose a confidence metric based on the confidence factor and decay factor.

Definition 15. *Confidence factor $con_i(M)$. Confidence factor is a measure of the trustworthiness of executors from their own perspective. Since $con_i(M)$ is related to the number of valid adjudications M, it also has the ability to change dynamically.*

It's necessary to mention that this section only describes how the initial confidence factor $con_i(0)$ is set. The confidence factor posterior updating is introduced in Section V.

Before entering the service set, the initial confidence factor $con_i(0)$ can be derived from the executor's information or historical experience. Compared with previous work that defaulted to the principle of indifference and set the initial confidence factor $con_i(0)$ to the same parameter, we analyze the mapping rule table to obtain reasonable information about the executor.

Inspired by the confusion matrix, we set the input data as the true category and the output data as the predicted category of the system. When $\{(a_o, A_o)\}_{o\in\Lambda_i}$ is the desired function set of P_i, inputting $x \in \{a_o\}$ indicates that x belongs to the positive class, otherwise x belongs to the negative class. Outputting $y \in \{A_o\}$ indicates that P_i predicts y to be in the positive class, otherwise P_i predicts y to be in the negative class. According to the four divisions of the mapping rule table, $\omega_1^{(i)}$, $\omega_2^{(i)}$, $\omega_3^{(i)}$, $\omega_4^{(i)}$ are also known as true positive, true negative, false positive, and false negative, respectively.

In this setting, precision and recall of the executor P_i can be computed by using Eq. (6) and Eq. (7). Precision denotes the percentage of the desired functional input that completes the desired functional output. Recall measures the percentage of the desired functional output that comes from the desired functional input.

$$pre_i = \left| \frac{\omega_3^{(i)}}{\omega_1^{(i)} + \omega_3^{(i)}} \right|, \tag{6}$$

$$rec_i = \left| \frac{\omega_3^{(i)}}{\omega_2^{(i)} + \omega_3^{(i)}} \right|. \tag{7}$$

If an executor has a larger precision, it prefers to complete functional implementation as possible, which may lead to the second type of error instead. Similarly, a greater recall reduces unexpected functional implementation, which results in the first type of error.

Considering the different focuses of precision and recall, we choose the initial confidence factor $con_i(0)$ as a reconciliation averaging metric to avoid the extreme number of $\omega_1^{(i)}$ or $\omega_2^{(i)}$, which is also known as the F1-score.

$$con_i(0) = \frac{2pre_i \cdot rec_i}{pre_i + rec_i}. \tag{8}$$

Definition 16. *Decay factor $\gamma_i(M)$. Decay factor takes into account the trend of the trustworthiness of the overall service phase, which provides a decreasing trend for the confidence factor as the number of valid adjudications M increases. It enables the executor to be swapped out after achieving a specific number of functional implementations, preventing attackers from acquiring substantial features due to long-time work. $\gamma_i(M)$ can be calculated by:*

$$\gamma_i(M) = \frac{M_i}{M_i + M}, \tag{9}$$

where M is the number of functional implementations completed by the executor P_i at the current moment. $\gamma_i(M)$ is a concave function monotonically decreasing with respect to M, satisfying $\lim_{M\to\infty} \gamma_i(M) = 0$. M_i is the task threshold of P_i that can control the decay rate of $\gamma_i(M)$. For the dynamics and controllability of the scheduling by the mimic system, the task thresholds of executors should satisfy a certain distribution. At the same time, by assigning the parameters of the distribution function artificially, it is possible to control the range of threshold values. A task threshold can be determined using sampling techniques.

The long-term practice of academia and industry shows that normal distribution satisfies our requirement for threshold selection. Mathematically, the central limit theorem states the conditions that a random variable under the influence of multiple factors approximately follows a normal distribution. Consequently, artificially given parameters μ, σ, $M_i \sim N(\mu, \sigma)$. Based on the probability density function, the Ziggurat algorithm is employed to select the random threshold [25].

In practice, it is indispensable to limit the selection range of the threshold. A too-large threshold may lead

to the continuous work of an executor, while a threshold that is too small or even less than 0 means invalid. According to the 3σ principle [48], the probability of sampling results beyond $(\mu - 3\sigma, \mu + 3\sigma)$ is less than 0.3%. Excluding this almost impossible threshold interval can make the scheduling smoother. Certainly, it is important to note that given μ, σ should satisfy $\mu - 3\sigma > 0$.

Definition 17. *Confidence $Con_i(M)$. Confidence is a dynamic and comprehensive metric to evaluate the trustworthiness of P_i after the Nth adjudication. Higher confidence indicates that the executor is relatively more difficult to be controlled by an attacker.*

Combining the above two factors, the confidence of P_i can be described by,

$$Con_i(M) = con_i(M) \cdot \gamma_i(M). \tag{10}$$

Algorithm 1. *Initialization*

Input: Redundancy of the service set n, CVSS score of k-order symbiotic vulnerability C_k, normal distribution of task threshold $N(\mu, \sigma)$.

1: Store all subsets of the waiting set with n elements in $\{\Pi_n\}$;
2: **for** $P_s \in \{\Pi_n\}$ **do**
3: Calculate $Het(P_s)$ according to Eq. (5);
4: **end for**
5: Get initial service set $P \leftarrow \arg\max_{P_s} Het(P_s)$;
6: **for** $i = 1 : n$ **do**
7: Sample from $N(\mu, \sigma)$ to get N_i;
8: Calculate $Con_i(0)$ according to Eq. (10) for $P_i \in P$;
9: **end for**

Output: Initial service set P.

The steps for initializing the mimic system are shown in Algorithm 1. First of all, all possible service sets from the waiting set are stored into $\{\Pi_n\}$ (line (1)). Secondly, the heterogeneity of each service set scheme can be calculated according to Eq. (5). The scheme with the largest heterogeneity is selected as the initial service set P (line (2)-(5)). Finally, for each executor in P, a task threshold is sampled from the normal distribution $N(\mu, \sigma)$ and the initial confidence $Con_i(0)$ is calculated according to Eq. (10) (line (6)-(9)). At this point, the entire initialization process from selecting the set to calculating the initial confidence is completed, and the subsequent working process is detailed in Algorithm 2.

V. BAYESIAN FRAMEWORK

To address the current problems of existing work including over-reliance on relatively-correct axiom and poor robustness, a novel adjudication approach under a probabilistic framework that combines heterogeneity and confidence metrics is proposed.

This section presents two applications of Bayesian theory to the mimic system we have designed: Bayesian adjudication and confidence factor posterior updating.

5.1 Bayesian Adjudication

Mimic adjudication plays a role in the arbiter, in essence: all outputs within the service set are given to the arbiter, from which the arbiter chooses the one that most resembles the correct output.

Bayesian adjudication is the basic method of implementing decisions under the framework of probability. Bayesian adjudication considers all executors in the service set to be trustworthy, which is measured by their confidence metric. As an independent group member, each executor gives the correct output they think. As the group leader, combining the confidence of each member, the arbiter obtains the result with the highest posterior probability as the final output based on Bayesian theory. Traditional voting strategies based on relatively-correct axiom only exploit the quantitative characteristics of the outputs, ignoring the information reflected by individual cases. By contrast, Bayesian adjudication combines the current confidence and all outputs to make a judgment, which is more convincing.

Assuming that $N-1$ adjudications have been completed ($M = N - 1$), in the Nth adjudication, the output information of a service set with n redundancy is denoted as D_N. The set of output results is denoted as $y_N = \{y_1, y_2, \cdots, y_n\}$, where n is the number of elements in y_N. Define events set $\mathcal{A} = \{\mathcal{F}_0, \mathcal{F}_1, \cdots, \mathcal{F}_{n'}\}$, where n' is the number of different kinds of elements in y_N, event $\mathcal{F}_i$ means that $y_i \in y_N$ is the correct output, $\forall i \in [n']$. For instance, there is $n = |\{B, B, A, B\}| = 4$ and $n' = |\{B, A\}| = 2$

Table 2. *Confidences and probabilities for case (a) and case (b),* $M = N - 1$.

Case	Confidence				Prior probability		Posterior probability		
	$Con_1(M)$	$Con_2(M)$	$Con_3(M)$	$Con_4(M)$	$P(\mathcal{F}_1)$	$P(\mathcal{F}_2)$	$P(\mathcal{F}_0\|D_N)$	$P(\mathcal{F}_1\|D_N)$	$P(\mathcal{F}_2\|D_N)$
(a)	0.2	0.2	0.8	0.2	0.25	0.75	0.20	0.99	0.01
(b)	0.4	0.8	0.6	0.5	0.50	0.50	0.19	0.36	0.64

in Figure 4(a). Considering that the correct output can be any value in the output space Γ, it is possible that the correct output is not observed by y_N, which is regarded as $\mathcal{F}_0$. Symbolically, event $\mathcal{F}_0$ means: $\hat{y}$ is the correct output, $\hat{y} \in \Gamma$, $\hat{y} \notin y_N$.

The posterior probability $P(\mathcal{F}_i \mid D_N)$ of the arbiter taking y_i as the correct output result can be expressed as,

$$P(\mathcal{F}_i \mid D_N) = \frac{P(\mathcal{F}_i)P(D_N \mid \mathcal{F}_i)}{P(D_N)}, i \geqslant 1, \quad (11)$$

where

$P(\mathcal{F}_i)$ is the prior probability. According to the idea of frequency approaching probability, in this scenario, $P(\mathcal{F}_i)$ is set as the proportion of y_i in y_N;

$P(D_N \mid \mathcal{F}_i)$ is the likelihood. Under the condition that y_i is the correct output, $P(D_N \mid \mathcal{F}_i)$ represents the probability that the service set gets the output information D_N. $Con_j(N-1)$ denotes the probability that P_j gets y_i. Since the executors in the service set are independent of each other, $P(D_N \mid \mathcal{F}_i)$ is the product of the probability of getting each executor's output;

$P(D_N)$ can be calculated using the full probability formula,

$$P(D_N) = \sum_{i=1}^{n'} P(\mathcal{F}_i)P(D_N \mid \mathcal{F}_i). \quad (12)$$

Especially, if $\mathcal{F}_0$ occurs, it indicates that there is no valid prior information in D_N. According to the principle of indifference, for any $\mathcal{F}_i$ in $\mathcal{A}$, $P(\mathcal{F}_i) = \frac{1}{n'+1}$. $P(\mathcal{F}_0 \mid D_N)$ can be simplified as,

$$P(\mathcal{F}_0 \mid D_N) = \frac{P(D_N \mid \mathcal{F}_0)}{\sum_{i=0}^{n'} P(D_N \mid \mathcal{F}_i)}. \quad (13)$$

Compared with other $\mathcal{F}_i$, the posterior probability $P(\mathcal{F}_0 \mid D_N)$ needs to be calculated separately.

Consequently, the expected result of the Bayesian adjudication is,

$$\hat{y} = \arg\max_i P(\mathcal{F}_i \mid D_N). \quad (14)$$

In Bayesian adjudication, $P(\mathcal{F}_0 \mid D_N)$ can be considered as a baseline for all posterior probabilities. If $\forall i, P(\mathcal{F}_i \mid D_N) \geqslant P(\mathcal{F}_0 \mid D_N)$, it means that the probability of any $y_i \in y_N$ being the correct output is less than the probability of all of the executors being wrong. In this case, the arbiter is unable to get the correct output. The system requires initialization of the service set or executor updating. However, $\mathcal{F}_0$ occurs only when each executor's confidence is less than 0.5. In general, with the confidence factor posterior updating described next, we will not tolerate the occurrence of $\mathcal{F}_0$.

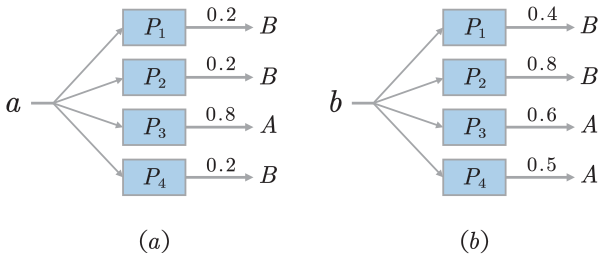

Figure 4. *Example of a service set of four redundancy in Nth adjudication.*

To illustrate the advantages of Bayesian adjudication over voting strategy, in Figure 4 we show a simple example in a service pool of 4 redundancy. We consider that $(a, A) \in \omega_3$, $(b, B) \in \omega_4$, $\mathcal{A} = \{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2\}$, where $\mathcal{F}_0$: Both A, B are wrong, $\mathcal{F}_1$: A is the correct output, $\mathcal{F}_2$: B is the correct output. In Figure 4(a), we input a: Only the executor P_3 outputs A, whereas in Figure 4(b), we input b: Both P_1 and P_2 output B. In Table 2, we detail the confidences and probabilities for case (a) and case (b) respectively. We can observe that Bayesian adjudication always outputs the correct result.

In Figure 4(a), we show a relatively extreme case that the confidences of P_1, P_2, P_3, P_4 are 0.2, 0.2, 0.8,

and 0.2, respectively. From the outputs, it can be observed that P_1, P_2, and P_4 all make the first type of error. This can be due to the interference of attackers. However, in Table 1 we appreciate that the probability that A is the correct output is much higher than that of B and the baseline value (the probability that both A and B are wrong), which reflects the robustness of Bayesian adjudication in certain attack scenarios. According to Assumption 4, when the attacker has no knowledge of the confidences of executors, P_1, P_2, and P_4 will make an error earlier than P_3 theoretically under a uniform attack. In the worst case as (a), the Bayesian adjudication is still able to output the correct result A, while the voting strategy based on relatively-correct axiom will output the wrong result B.

In Figure 4(b), We analyze a more general case that the confidences of P_1, P_2, P_3, P_4 are 0.4, 0.8, 0.6, 0.5, respectively. Although the low confidence of P_1 brings negative effects to the judgment of the correct result B, decision power is still dominated by P_2. Incidentally, the voting strategy based on relatively-correct axiom will fail when the output is equally divided as in case (b), whereas the Bayesian adjudication demonstrates a higher universality.

5.2 Confidence Factor Posterior Updating

Confidence factor posterior updating reflects the behavior of the mimic system in changing its internal characteristics based on real-time attack information. During the operation of the mimic system, there is quite a bit of information available in addition to the output results, for example, whether each actuator correctly outputs what the arbiter considers to be the correct result. According to the introduction section IV, the value of confidence is the key to whether the Bayesian adjudication can produce the correct output results. To improve the accuracy of the arbiter and enhance the dynamics of the system, we propose a posterior updating strategy for confidence based on historical adjudication information.

Definition 18. *Historical adjudication information $E_m(N)_i$. $E_m(N)_i$ is defined as the number of errors made by P_i in the Nth to $N+m$th adjudications. This information can be easily extracted from the mimic system and recorded, and it reflects the performance of the executor's adjudications recently. The ratio of $E_m(N)_i$ to m will increase significantly if the executor P_i is attacked intensively by attackers.*

Algorithm 2. *Mimic System Working Process*

Input: Current adjudication order N, input data sequences $\{x_k\}$, updating step m, confidence threshold Con_{th}, size of waiting set W.

Initialize $E_m(N)_i \leftarrow 0$;
for $k = N : N+m$ **do**
  Input x_k into the system;
  Collect the output information D_k;
  Calculate $P(\mathcal{F}_i \mid D_k)$ according to Eq. (11) and Eq. (13);
  Arbiter output $\hat{y}_k \leftarrow \arg\max_{y_i} P(\mathcal{F}_i|D_k)$;
  for $i = 1 : n$ **do**
    if $\hat{y}_k \neq y_i$ **then**
      $E_m(N)_i \leftarrow E_m(N)_i + 1$;
    end if
  end for
end for
for $i = 1 : n$ **do**
  Calculate $Con_i(N+m)$ according to Eq. (16);
  if $Con_i(N+m) < Con_{th}$ **then**
    $\{P\} \leftarrow \{P\} \setminus P_i$;
    for $j = 1 : W$ **do**
      $\{P_j\} \leftarrow \{P\} \cup P_j$;
      Calculate $Het(P_j)$;
    end for
    Service set $\hat{P} \leftarrow \arg\max_{\{P_j\}} Het(P_j)$;
    Initialize N_j, $Con_j(0)$ of the executor P_j;
  else
    $Con_i(N) \leftarrow Con_i(N+m)$;
  end if
end for

Output: Result sequences $\{\hat{y}_k\}$.

Assume that the result determined by the arbiter is "correct". We use T_i^+ to indicate that P_i made a correct judgment and T_i^- to indicate that P_i made an incorrect judgment. $\pi_{N+m}(T_i^+)$ denotes the prior probability that P_i can make a correct judgment after the $N+m$th adjudication, $\pi_{N+m}(T_i^+) = con_i(N)$. Similarly, $\pi_{N+m}(T_i^-) = 1 - con_i(N)$.

The posterior probability that P_i can make a correct

judgment after the $N+m$th adjudication is,

$$P(T_i^+ \mid E_m(N)_i) = \frac{\pi_{N+m}(T_i^+)P(E_m(N)_i \mid T_i^+)}{P(E_m(N)_i)}, \tag{15}$$

where

$$P(E_m(N)_i \mid T_i^+) = \sum_{\alpha \in \{+,-\}} P(T_i^\alpha)P(E_m(N)_i \mid T_i^\alpha).$$

$P(T_i^+ \mid E_m(N)_i)$ is the latest confidence factor of the executor, denoted as $con_i(N+m)$.

The above process is referred to an m-order confidence factor posterior updating, where m is the step size. At present, the confidence of P_i can be calculated from Eq. (10),

$$Con_i(N+m) = con_i(N+m) \cdot \gamma_i(N+m). \tag{16}$$

The steps for adjudication and confidence updating are shown in Algorithm 2. The first step in this process is taking every m adjudications as a cycle to collect the historical adjudication information $E_m(N)_i$. In each adjudication, the posterior probabilities of $\mathcal{F}_i$ and $\mathcal{F}_0$ are calculated by Eq. (11) and Eq. (13), respectively. The result with the largest posterior probability is regarded as the correct output $\hat{y}$. The total number of errors made by P_i in these m adjudications is recorded in $E_m(N)_i$ (line (1)-(12)).

Then, according to Eq. (16), $Con_i(N+m)$ can be calculated in combination with historical adjudication information $E_m(N)_i$. Meanwhile, for each executor, the relationship between $Con_i(N+m)$ and Con_{th} is determined. If $Con_i(N+m) < Con_{th}$, P_i is removed from the service set $\{P\}$. The executor that minimizes the current heterogeneity is selected from the waiting set to compose $\left\{\hat{P}\right\}$. $Con_i(0)$ is initialized by Eq. (10). If $Con_i(N+m) \geqslant Con_{th}$, $Con_i(N+m)$ is taken as the updated confidence. At present, the current round of adjudication and confidence updating is done.

VI. EXPERIMENTAL ANALYSIS

We implemented our FBAMD framework in Python and verified its security, stability, and efficiency on the real dataset. The FBAMD framework is based on the DHR structure, which has the advantage of high defense performance against different attacks. We deployed multiple machine learning sub-models and applied individual and hybrid attacks on each model. We implemented Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Decision Tree as four sub-models, using the numpy, sklearn, and secml packages. We also constructed two different DHR models using these sub-models. Jia et al. [49] demonstrated the effectiveness and robustness of Bagging against poisoning attacks, and we built our model according to this strategy. On the attack dimension, we chose the Statistical-based Poisoning Attack (StatP) [33] against Linear Regression, the Adversarial Label Flip Attack (alfa) [35] against SVM, and the poisoning attack [45] against Logistic Regression (abbreviated as Logi). The poisoning rate for each poisoning attack is set to a fixed value $\alpha = 0.2$, namely, there are 20% of the poisoning samples in the training set for model training. In addition to the three individual attacks mentioned above, we balanced the three poisoned samples to deploy a Mixed attack with the same total poisoning rate of 20%. On the attack dimension, except for our FBAMD, we considered the defense strategies including TRIM against linear regression [33], a defense algorithm using Kernel distance in the Local Intrinsic Dimensionality calculation (K-LID-SVM) against SVM [35], and the voting-based DHR [43] (referred to as DHR). We use three main metrics for evaluating our algorithms: classification accuracy for all models, confidence fluctuations for sub-models, and the updating frequency of executors in FBAMD.

Dataset. We used the public darknet traffic classification dataset CIC-Darknet2020 [28] as an example in our experimental evaluation. The Darknet dataset consists of 158,659 records in total. There are 134,348 benign samples and 24,311 darknet samples. This dataset is created to cover Tor and VPN traffic respectively by amalgamating two public datasets, namely, ISCXTor2016 and ISCXVPN2016, with the goal of effectively classifying benign and darknet traffic. We chose to utilize all 76 statistical features in the data and sampled 40,000 data in a category-balanced manner to construct the data in stream form. In each independent experiment investigating the comparison of different attacks and defenses, we randomly selected 2000 data as the training set and 1500 data as the test set, tested independently 5 times, and reported results as aver-

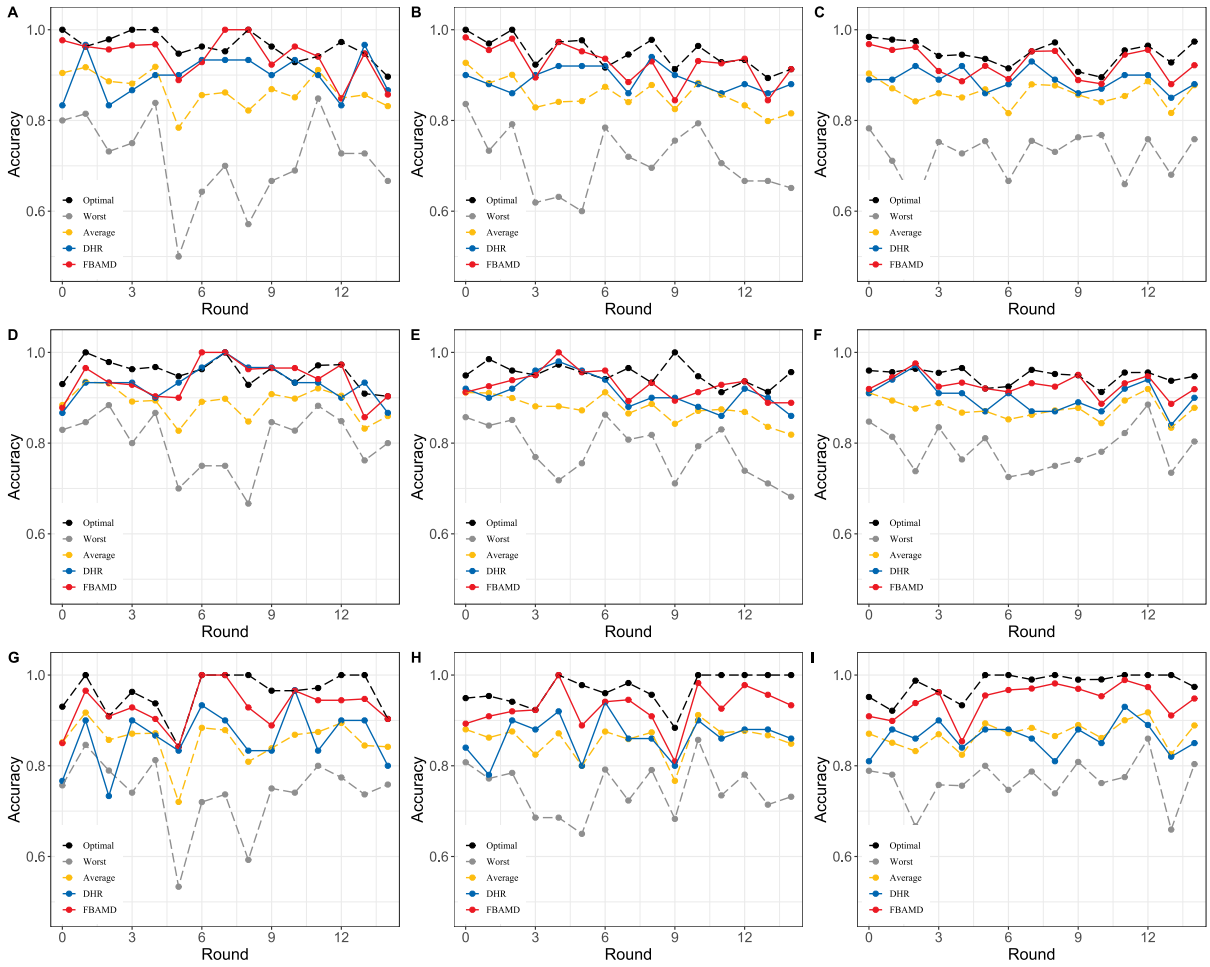

Figure 5. *The optimal performance, the worst performance, the average performance of four sub-models and the performance of DHR and FBAMD when under Logi, StatP, and alfa attack at step size 30, 50, and 100, respectively.*

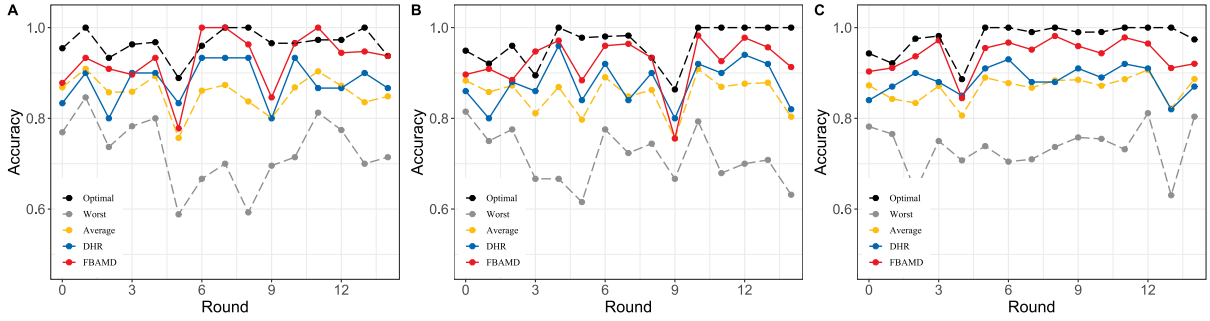

Figure 6. *The optimal performance, the worst performance, the average performance of four sub-models and the performance of DHR and FBAMD when under Mixed attack at step sizes 30, 50, and 100, respectively.*

Table 3. *Classification accuracy of TRIM, K-LID-SVM, DHR, and FBAMD under different attack scenarios at a poisoning rate of 0.2. FBAMD has the highest accuracy against all kinds of attacks.*

Defense \ Attack	None	Logi [45]	alfa [35]	StatP [33]	Mixed
Logistic	83.20%	72.97%	78.86%	79.13%	75.27%
Logistic with TRIM [33]	82.36%	82.45%	83.80%	83.74%	83.80%
SVM	87.97%	84.47%	79.75%	82.62%	83.51%
SVM with K-LID-SVM [35]	89.92%	87.40%	86.52%	85.29%	86.53%
DHR[43]	90.37%	88.75%	87.31%	90.18%	87.93%
FBAMD	**92.46%**	**94.60%**	**92.13%**	**92.60%**	**91.80%**

ages over 5 runs. For models that require a validation set, the test set is divided equally into a test set and a validation set.

Security. We performed experiments on the darknet dataset to evaluate the classification accuracy of different models when under attack. We chose Logi, StatP, alfa, and Mixed poisoning attacks to poison the training set and set different confidence updating step sizes of 30,50,100, respectively, with the rest of the settings as described previously. We recorded the optimal performance, the worst performance, and the average performance of four sub-models in each round and compared them with the performances of two DHR models. Figure 5 shows the classification accuracy of three individual attacks Logi, StatP, and alfa at different step sizes, while Figure 6 is under Mixed attack. We highlight several interesting observations and remarks for Figure 5 and 6.

First, we point out the following observations.

(1) DHR's performance is comparable to the average performance of the four sub-models and slightly above the average performance. For instance, the performance of DHR is similar to the four sub-models' average performance and even identical in some cases. From the perspective of statistics, this result is consistent with the voting-based adjudication, essentially averaging the results of each sub-model.

(2) The overall performance of our FBAMD is higher than DHR and can approximate the optimal performance of sub-models. Under individual attacks, FBAMD can achieve the optimal performance of sub-models, which indicates that FBAMD is not affected by targeted attacks. Under hybrid attacks, FBAMD performs better than DHR, which demonstrates that systems deployed with FBAMD in real-world scenarios can maintain an optimal defense when encountering hybrid attacks in cyberspace. Even if some executors are breached by attackers and show the worst performance, FBAMD's performance is still on par with the optimal performance (Theoretically, if an attacker can breach all sub-models at the same time, then FBAMD's performance will also be significantly degraded). For instance, the worst model has an error rate of 30% in Figure 6C, while FBAMD shows less than 10% and is only 5% more than the model with optimal performance.

(3) We observe an occasional jitter in the classification accuracy at different step sizes. For instance, in Figure 6B, there is a clear trend of decreasing after the 8th run, and it returns to normal in the 10th run. With the reduction of confidence, this phenomenon occurs due to a significantly large number of errors made by sub-models. Before the 10th run, the sub-model with the current confidence dropped below the threshold was replaced, ensuring the stability of the subsequent run. A closer inspection of the figures shows that the interval between jitters increases as the step size increases, which is quite revealing. These results suggest that jitter can be avoided to some extent by choosing the right step size. We will discuss the jitter phenomenon next.

Then, we make a few remarks about the sub-models as follows.

(1) In this paper, sub-models are regarded as executors, and the selection of sub-models should obey the principle of "Dynamic, Heterogeneous, Redundancy". We picked these four sub-models as they have the same classification function with different fundamen-

Table 4. *Confidence of the four sub-models Logistic, SVM, Tree, KNN when FBAMD is subjected to different attacks at step size 50. The bolded position is where the sub-model replacement is performed.*

Attack	Model	0	1	2	3	4	5	6	7	8	9	10
Mixed	Logistic	0.664	0.516	0.502	**0.349**	0.534	**0.411**	0.602	**0.459**	0.587	**0.434**	0.634
	SVM	0.700	0.570	**0.445**	0.647	0.574	**0.415**	0.667	0.540	0.503	**0.343**	0.712
	Tree	0.751	0.603	0.536	**0.436**	0.751	0.645	0.541	**0.492**	0.744	0.572	0.566
	KNN	0.708	0.593	0.506	**0.422**	0.795	0.570	0.553	**0.422**	0.682	0.511	0.528
Logi	Logistic	0.664	**0.483**	0.620	**0.399**	**0.499**	**0.476**	0.625	**0.477**	0.557	0.509	**0.454**
	SVM	0.707	0.576	**0.481**	0.709	0.568	**0.484**	0.716	0.541	0.510	**0.370**	0.693
	Tree	0.803	0.651	0.576	**0.466**	0.787	0.664	0.537	0.506	**0.427**	0.737	0.652
	KNN	0.781	0.629	0.540	**0.431**	0.718	0.623	0.514	**0.440**	0.740	0.574	0.508
StatP	Logistic	0.688	0.562	**0.489**	0.610	**0.471**	0.606	0.578	**0.464**	0.651	**0.469**	0.641
	SVM	0.722	0.597	0.508	**0.467**	0.707	0.604	0.512	**0.421**	0.736	0.537	**0.486**
	Tree	0.743	0.655	0.546	**0.461**	0.779	0.653	0.529	**0.484**	0.723	0.666	0.540
	KNN	0.737	0.579	**0.475**	0.709	0.639	0.503	**0.480**	0.665	0.587	**0.470**	0.677
alfa	Logistic	0.654	0.525	**0.499**	0.562	**0.476**	0.519	0.526	**0.412**	0.631	**0.477**	0.686
	SVM	0.674	0.532	**0.437**	0.637	0.541	**0.398**	0.667	0.561	0.509	**0.345**	0.686
	Tree	0.757	0.633	0.534	**0.458**	0.801	0.637	0.539	0.501	**0.426**	0.725	0.663
	KNN	0.733	0.610	0.534	**0.454**	0.806	0.600	0.558	**0.455**	0.683	0.512	0.520

tals, reflecting heterogeneity and redundancy. When the threshold drops below 0.5, we cleaned and replaced the sub-model, which demonstrates dynamic.

(2) The essential difference between the four sub-models is that the fundamentals used to implement the classification function are different, so an attacker can't design an algorithm to target all four sub-models simultaneously. For example, the Adversarial Label Flip Attack (alfa) [35] is a targeted attack method against the structure of Support Vector Machine, so its utility against Decision Tree will be greatly reduced. The greater the difference in structure between the sub-models (heterogeneity), the less effective the attack will be.

(3) The sub-models are independently parallel in DHR. Consequently, the overall runtime complexity of FBAMD is the worst of the sub-models plus a very small constant complexity. The sub-models in DHR are base models without any defense strategies imposed, which are less difficult to implement. More classification sub-models can be deployed in practical scenarios.

(4) For experimental convenience, we only introduced four kinds of sub-models. Different kinds of sub-models reflect heterogeneity in hardware (complexity), and different parameters of similar sub-models reflect heterogeneity in software (diversity) [50].

(5) In our experiment, the first step for new sub-model generation is to get samples from the poisoning dataset, and the samples are split the same as in the previous experiment. Then, the model is trained and the F1-score is calculated as the initial confidence $Con_i(0)$. Finally, the task threshold N_i is obtained by sampling from the normal distribution.

In Table 3, we detail the classification accuracies of TRIM, K-LID-SVM, DHR, and FBAMD under different attack scenarios. With the total poisoning rate α = 0.2, we record the mean classification accuracy value from five testing rounds in the table. Our results confirm that the security of FBAMD is the highest in various attack scenarios. Compared to the two defense strategies TRIM and K-LID-SVM, FBAMD's accuracy is on average 10% higher; compared to DHR, FBAMD dominates and accuracy can be above 90% in all cases. The single most striking observation to emerge from the data comparison is that TRIM and K-LID-SVM are only effective in defending against Logi

and alfa attacks, respectively, and perform poorly in defending against other attacks. This demonstrates the shortcoming of passive defense. Only when the type of attack is known can the appropriate defense strategy be used for effective defense. On the other hand, as active defense strategies, DHR and FBAMD maintain high classification accuracy in the face of various attacks. Simultaneously, FBAMD is more dynamic than DHR, takes more information into account, and has better defense results.

Stability. In the experiments to assess the stability of FBAMD, we keep the model, dataset, poisoning rate, attack type, etc. the same as in the above experiments. Confidence fluctuations between sub-models are used as a measure to evaluate the stability of the model. When under attack, if the confidence fluctuations of each sub-model are trendless, then the adjudication of FBAMD will strictly base on the current confidence according to Bayesian theory. The high-confidence output is given a relatively large weight. Conversely, some particular fluctuations of the sub-model will lead to the jitter phenomenon described above in the system.

The jitter phenomenon is expressed as a simultaneous decrease in the accuracy of each sub-model, DHR, and FBAMD, whose occurrence is caused by multiple reasons. Since each sub-model works independently, its accuracy degradation is only related to its current working state. Furthermore, there is a strong correlation between the results of DHR and sub-models. The decrease in FBAMD's accuracy is on the one hand due to the output of wrong results by high-confidence executors. On the other hand, the uniformly distributed confidence level results in no sub-model that can correctly lead Bayesian adjudication, with an increasing weight of the randomness factor.

Table 4 provides the numerical variation of the four sub-models' confidence at a step size of 50 under different attacks in detail. We have bolded the location where the sub-model replacement was performed ($Con_i(N) < 0.5$). It is apparent from this table that the locations where the jitter phenomenon appears are accompanied by a large number of replacements of sub-models simultaneously. The result suggests that there is an association between jitter and the replacement of sub-models. Consequently, when performing FBAMD, we can avoid the jitter phenomenon by selecting submodels with large heterogeneity. In addition, an appropriately higher step size can make FBAMD gain high accuracy and stability.

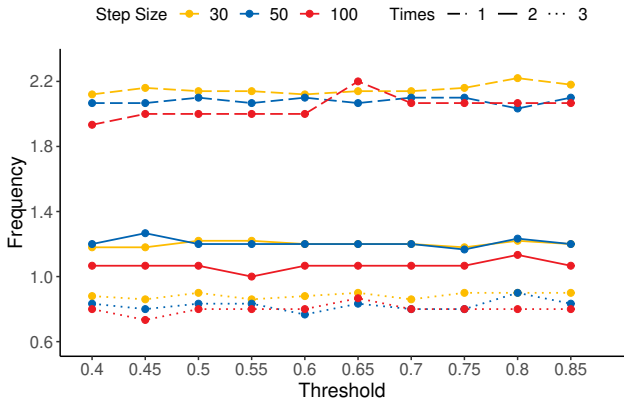

Figure 7. *Frequency of sub-model replacement when the confidence update step size is 30, 50, 100 and the task threshold mean μ is 1x, 2x, 3x step size.*

Efficiency. We measure the operational efficiency of FBAMD using the updating frequency of the sub-models. In practice, we need to balance the security gains from dynamic replacement with the service disruptions caused by replacement. Ideally, we hope model replacement occurs with each confidence update, but do not want to replace too many at once. In FBAMD framework, the confidence is not only decreased by the posterior update of the confidence factor $con_i(N)$, but also by the natural decrease of the decay factor γ_i. The selection of the task threshold N_i in the decay factor γ_i will directly affect the decline rate of confidence. As shown in Figure 7, we present the intercorrelations between the number of replacements (Frequency) and the threshold at different step sizes. We set the mean of the normal distribution μ of the task threshold N_i in the decay factor γ_i to be 1,2,3 times the step size and the variance σ to be 1/10 of the corresponding mean value μ.

It is apparent from this figure that replacement frequency shows a decreasing trend with increasing step size (independent of the task threshold). When the mean of the normal distribution μ is 1-time step size, frequency is around 2, which may lead to the jitter phenomenon. When μ is 3 times the step size, frequency is less than 1. The long operation of a certain executor may allow an attacker to obtain more information and increase the risk of system exposure. In summary, we suggest choosing 2 times the step size as the mean value of the task threshold and setting the threshold

value to 0.55 and the step size to 100.

VII. CONCLUSION

In this paper, a comprehensive theory of mimic defense from the perspective of functional implementation named FBAMD is proposed. We first introduce the extraction of common features of executors' vulnerabilities and the division of the mapping rule table (i.e., first and second error types). Then, a heterogeneity metric is defined based on different orders of symbiotic vulnerability and their CVSS scores. A confidence metric dynamically updated with task flow is designed, which can ensure certain dynamics and controllability consisting of a confidence factor and a decay factor. Additionally, to solve the problem of the current adjudication mechanisms' over-reliance on the relatively-correct axiom, we propose a new adjudication mechanism under a probabilistic framework. According to Bayesian theory, this mechanism can update the confidence using the adjudication result information, which provides the mimic system with adaptive capability. Experimental results under a variety of settings have shown the outstanding security performance and stability of FBAMD against hybrid attacks.

ACKNOWLEDGEMENT

This work is supported by the National Key Research and Development Program of China (Grant No.2020YFB1804604).

References

[1] BRINGHENTI D, MARCHETTO G, SISTO R, et al. Automated firewall configuration in virtual networks[J]. IEEE Transactions on Dependable and Secure Computing, 2022.

[2] CHEN X, LI C, WANG D, et al. Android hiv: A study of repackaging malware for evading machine-learning detection[J]. IEEE Transactions on Information Forensics and Security, 2020, 15: 987-1001.

[3] ARORA A, PEDDOJU S K, CONTI M. Permpair: Android malware detection using permission pairs[J]. IEEE Transactions on Information Forensics and Security, 2020, 15: 1968-1982.

[4] LI D, LI Q. Adversarial deep ensemble: Evasion attacks and defenses for malware detection [J]. IEEE Transactions on Information Forensics and Security, 2020, 15: 3886-3900.

[5] WANG N, CHEN Y, HU Y, et al. Manda: On adversarial example detection for network intrusion detection system[J]. IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, 2021: 1-10.

[6] SHARON Y, BEREND D, LIU Y, et al. Tantra: Timing-based adversarial network traffic reshaping attack[J]. IEEE Transactions on Information Forensics and Security, 2022, 17: 3225-3237.

[7] XU C, ZHONG SHEN J, DU X. A method of few-shot network intrusion detection based on meta-learning framework[J]. IEEE Transactions on Information Forensics and Security, 2020, 15: 3540-3552.

[8] SOMMER R, PAXSON V. Outside the closed world: On using machine learning for network intrusion detection[J]. 2010 IEEE Symposium on Security and Privacy, 2010: 305-316.

[9] WU J. Development paradigms of cyberspace endogenous safety and security[J]. SCIENTIA SINICA Informationis, 2021.

[10] PROVOS N. A virtual honeypot framework[C]// USENIX Security Symposium. 2004.

[11] SHAN S, WENGER E, WANG B, et al. Gotta catch'em all: Using honeypots to catch adversarial attacks on neural networks[J]. Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, 2020.

[12] AGRAWAL R, STOKES J W, RIST L, et al. Long-term study of honeypots in a public cloud [J]. 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S), 2022: 1-4.

[13] ZHENG J, NAMIN A S. A survey on the moving target defense strategies: An architectural perspective[J]. Journal of Computer Science and Technology, 2019, 34: 207-233.

[14] HIGGINS M, TENG F, PARISINI T. Stealthy mtd against unsupervised learning-based blind fdi attacks in power systems[J]. IEEE Transactions on Information Forensics and Security, 2021, 16: 1275-1287.

[15] ZHANG Z, DENG R, YAU D K Y, et al. Analysis

of moving target defense against false data injection attacks on power grid[J]. IEEE Transactions on Information Forensics and Security, 2020, 15: 2320-2335.

[16] XU W, JAIMOUKH I M, TENG F. Robust moving target defence against false data injection attacks in power grids[J]. IEEE Transactions on Information Forensics and Security, 2022: 1-1.

[17] HEYDARI V, IL KIM S, YOO S M. Scalable anti-censorship framework using moving target defense for web servers[J]. IEEE Transactions on Information Forensics and Security, 2017, 12: 1113-1124.

[18] JIN H, LI Z, ZOU D, et al. Dseom: A framework for dynamic security evaluation and optimization of mtd in container-based cloud[J]. IEEE Transactions on Dependable and Secure Computing, 2021, 18: 1125-1136.

[19] HOBSON T, OKHRAVI H, BIGELOW D, et al. On the challenges of effective movement[C]// MTD '14. 2014.

[20] WU J. Research on cyber mimic defense[J]. Journal of Cyber Security, 2016(4): 10.

[21] YU H, LI H, YANG X, et al. On distributed object storage architecture based on mimic defense [J]. China Communications, 2021, 18: 109-120.

[22] SANG X, LI Q. Mimic defense techniques of edge-computing terminal[J]. 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService), 2019: 247-251.

[23] MENGLI Z, FUCAI C, WENYAN L, et al. Negative feedback dynamic scheduling algorithm based on mimic defense in cloud environment [J]. 2020 IEEE 6th International Conference on Computer and Communications (ICCC), 2020: 2265-2270.

[24] ZHANG W, WEI S, TIAN L, et al. Scheduling algorithm based on heterogeneity and confidence for mimic defense "in prepress"[J]. J. Web Eng., 2020, 19: 54-78.

[25] LI Q, MENG S, SANG X, et al. Dynamic scheduling algorithm in cyber mimic defense architecture of volunteer computing[J]. ACM Trans. Internet Techn., 2021, 21: 75:1-75:33.

[26] QI WU Z, WEI J. Heterogeneous executors scheduling algorithm for mimic defense systems [J]. 2019 IEEE 2nd International Conference on Computer and Communication Engineering Technology (CCET), 2019: 279-284.

[27] SHAO Y, ZHANG Z, WANG X, et al. Solve the false-positive problem of the mimic system based on the best mimic component set[J]. China Communications, 2022, 19: 253-266.

[28] LASHKARI A H, KAUR G, RAHALI A. Didarknet: A contemporary approach to detect and characterize the darknet traffic using deep image learning[J]. 2020 the 10th International Conference on Communication and Network Security, 2020.

[29] ESMAEILPOUR M, CARDINAL P, KOERICH A L. A robust approach for securing audio classification against adversarial attacks[J]. IEEE Transactions on Information Forensics and Security, 2020, 15: 2147-2159.

[30] AL-HAIJA Q A, ISHTAIWI A. Machine learning based model to identify firewall decisions to improve cyber-defense[J]. International Journal on Advanced Science, Engineering and Information Technology, 2021.

[31] FEI LIAN W, NIE G, KANG Y, et al. Cryptomining malware detection based on edge computing-oriented multi-modal features deep learning[J]. China Communications, 2022, 19: 174-185.

[32] WANG H, YE G, TANG Z, et al. Combining graph-based learning with automated data collection for code vulnerability detection[J]. IEEE Transactions on Information Forensics and Security, 2021, 16: 1943-1958.

[33] JAGIELSKI M, OPREA A, BIGGIO B, et al. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning[J]. 2018 IEEE Symposium on Security and Privacy (SP), 2018: 19-35.

[34] WEN J, ZHAO B Z H, XUE M, et al. With great dispersion comes greater resilience: Efficient poisoning attacks and defenses for linear regression models[J]. IEEE Transactions on Information Forensics and Security, 2021, 16: 3709-3723.

[35] WEERASINGHE S, ALPCAN T, ERFANI S M, et al. Defending support vector machines against data poisoning attacks[J]. IEEE Transactions on Information Forensics and Security, 2021, 16: 2566-2578.

[36] FLOWERS B, BUEHRER R M, HEADLEY

W C. Evaluating adversarial evasion attacks in the context of wireless communications[J]. IEEE Transactions on Information Forensics and Security, 2020, 15: 1102-1113.
[37] NAIK N, SHANG C, JENKINS P, et al. D-frihoneypot: A secure sting operation for hacking the hackers using dynamic fuzzy rule interpolation[J]. IEEE Transactions on Emerging Topics in Computational Intelligence, 2021, 5: 893-907.
[38] ZHANG M, WANG L, JAJODIA S, et al. Network diversity: A security metric for evaluating the resilience of networks against zero-day attacks[J]. IEEE Transactions on Information Forensics and Security, 2016, 11: 1071-1086.
[39] WU J. Development paradigms of cyberspace endogenous safety and security[J]. Science China Information Sciences, 2022, 65: 1-3.
[40] ZHANG Q, TANG H, YOU W, et al. A method for constructing heterogeneous entities pool in nfv security architecture based on mimic defense [J]. 2020 IEEE 6th International Conference on Computer and Communications (ICCC), 2020: 1029-1033.
[41] DAI W, LI S, LU L, et al. Research on application of mimic defense in industrial control system security[J]. 2021 IEEE 2nd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA), 2021, 2: 573-577.
[42] LI Z, HU Y, ZHU D, et al. Esmd-flow: An intelligent flow forwarding scheme with endogenous security based on mimic defense in space-air-ground integrated network[J]. China Communications, 2022, 19: 40-51.
[43] ZHENG Y, LI Z, XU X, et al. Dynamic defenses in cyber security: Techniques, methods and challenges[J]. Digit. Commun. Networks, 2022, 8: 422-435.
[44] TONG Q, GUO Y. A comprehensive evaluation of diversity systems based on mimic defense[J]. Sci. China Inf. Sci., 2021, 64.
[45] DEMONTIS A, MELIS M, PINTOR M, et al. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks [C]//USENIX Security Symposium. 2019.
[46] TURING A. On computable numbers, with an application to the entscheidungsproblem[J]. Proc. London Math. Soc., 1937, s2-42: 230-265.
[47] MELL P, SCARFONE K, ROMANOSKY S. The common vulnerability scoring system (cvss) and its applicability to federal agency systems [C]//2007.
[48] WANG S, CAO L. Inferring implicit rules by learning explicit and hidden item dependency[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2020, 50: 935-946.
[49] JIA J, CAO X, GONG N Z. Intrinsic certified robustness of bagging against data poisoning attacks[C]//AAAI Conference on Artificial Intelligence. 2021.
[50] BREIMAN L. Bagging predictors[J]. Machine Learning, 2004, 24: 123-140.

Biographies

He Jiajun was born in Shaanxi, China, in 2003. He is currently pursuing the B.S. degree with the School of Mathematics, Southeast University. His research interests include endogenous security, causal inference and statistical learning theory.

Yuan Yali received her Ph.D. degree from Göttingen University, Göttingen, Germany, in 2018. Dr. Yuan joined the school of Cyber Science and Engineering, Southeast University, Nanjing, China, as an assistant professor in 2021. Her current research interests network traffic anomaly detection and endogenous security.

Liang Sichu was born in Jiangsu, China, in 2002. He is currently pursuing the B.Eng. degree with the School of Artificial Intelligence, Southeast University. His research interests include machine learning and pattern recognition, specifically graph representation, computer vision and mimic defense.

Fu Jiale was born in Shaanxi, China, in 2002. He is currently pursuing the B.S. degree with the School of Mathematics, Southeast University. His research interests include endogenous security, machine learning and complex network.

Zhu Hongyu was born in Jiangsu, China, in 2002. He is currently pursuing the B.Eng. degree with the School of Cyber Science and Engineering, Southeast University. His research interests include artificial intelligence and security, specifically exploring vulnerabilities in machine learning techniques, intrusion detection systems and mimic defense.

Cheng Guang received the B.S. degree in Traffic Engineering from Southeast University in 1994, the M.S. degree in Computer Application from Hefei University of Technology in 2000, and the Ph.D. degree in Computer Network from Southeast University in 2003. He is a Full Professor in the School of Cyber Science and Engineering, Southeast University, Nanjing, China. He has authored or coauthored seven monographs and more than 100 technical papers, including top journals and top conferences. His research interests include network security, network measurement, and traffic behavior analysis. He is a Member of IEEE and a Senior Member of CCF.