Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 404681, 8 pages
http://dx.doi.org/10.1155/2014/404681

Research Article

A Strategy Optimization Approach for Mission Deployment in Distributed Systems

Liu Xiaojian,[1] Yuan Yuyu,[2] Fang Binxing,[1] and Gao Yi[2]

[1] *College of Computer Science and Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2] *College of Software Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China*

Correspondence should be addressed to Liu Xiaojian; chn1989lxj@163.com

Received 16 October 2014; Accepted 5 November 2014; Published 23 November 2014

Academic Editor: Hui Zhang

In order to increase operational efficiency, reduce delays, and/or maximize profit, almost all the organizations have split their mission into several tasks which are deployed in distributed system. However, due to distributivity, the mission is prone to be vulnerable to kinds of cyberattacks. In this paper, we propose a mission deployment scheme to optimize mission payoff in the face of different attack strategies. Using this scheme, defenders can achieve "appropriate security" and force attackers to jointly safeguard the mission situation.

1. Introduction

Modern organizations embed information and communication technologies (ICT) into their core processes as means to facilitate the collection, storage, processing, and exchange of data to increase operational efficiency, improve decision quality, and reduce costs [1]. In this way, organizations tend to split their mission into smaller tasks which can be executed in distributed system.

Despite the significant benefits of distributed system, the system also places the mission at the risk due to "distributed vulnerability." Traditional approaches to improve security generally consider only system vulnerabilities and attempt to defend all the attacks through system upgrading. No matter whether the assumed attacks come, the defending resources have to be inputted. In distributed system, these keeping upgrading approaches will result in a huge waste of defending resource. Regarding this, the concept of "appropriate security" is proposed to pursue a tradeoff between security risk and defending resource. The nature of attack and defense can be abstracted to the game between attack strategy and defense strategy. Whether the defense strategy is effective or not depends not only on self-condition but also on attack and defense strategy. Therefore, when we deploy mission tasks, it is critically important to take both system vulnerabilities and attack strategy in consideration.

In this paper, we improve mission assurance through suitable mission deployment and propose a novel defense scheme that enables deployment of mission tasks such that the payoff of mission is maximized considering all the possible attack strategies and vulnerabilities exposed. In other words, we figure out a trade-off between attack strategy and defense strategy, which neither attacker nor defender has enough reasons to break it. That means every strategy change of attacker and defender is sure to reduce self-payoff.

The contribution of this paper is as follows: first, we formulate the mission deployment problem about attacking and defending using game model; second, we improve game theory formula, and, based on that, we design particle swarm optimization algorithm suitable to find optimal mission deployment strategy; finally, we prove that, using proposed method, the defender can force rational attackers to jointly safeguard the mission situation.

The remainder of this paper is structured as follows. In Section 2, we discuss related work. In Section 3, we describe the problem to be solved and present some preliminary definitions. In Section 4, we model the mission deployment problem using game theory. In Section 5, we design particle

swarm optimization (PSO) algorithm to calculate the Nash equilibrium. In Section 6, we report our experimental results and comparison of other works. Finally, in Section 7, we present our conclusions and make recommendations for future works.

2. Related Work

The problem of mission assurance by deployment has not been sufficiently studied in the literature. Reference [2] presents a solution for deploying mission tasks in a distributed environment in a way that minimizes a mission's exposure to vulnerabilities by taking into account available information about vulnerabilities and dependencies. In this paper, we propose a novel model and method to discuss this problem of mission assurance by deployment using game theory.

The main related works can be summarized into two categories.

2.1. Task Allocation. The problem of task allocation in distributed system has been widely studied. Some researchers study task allocation as NP-hardness problem. Most solutions are based on heuristics. Several heuristics algorithms were studied to pursue the optimal finishing time in [3, 4]. Reference [5] studies the problem of task assignment based on the graph theoretic approach, taking two incompatible factors, interference cost and communication costs, into account. Some other works integrate the security requirements within task assignment process. Reference [6] describes a task allocating method that introduces safety into distributed system and attempts to maximize the reliability. Reference [7] proposed a scheduling strategy, named SAREC, to satisfy security requirements while meeting timing constraints imposed by applications. Reference [8] extended the research of [7] by solving the problem of greedy and proposed a new security-aware real-time scheduling algorithm based on genetic algorithm (SAREC-GA). Although the problems of security in task assignment have already been paid attention to for several years, studying it in game theory field is still a new issue. A research focuses on game theoretic resources allocation algorithm that considers the fairness among users and the resources utilization. Moreover, the allocation problem use in other areas has been studied [9–12], especially in the area of nonlinear systems [13] and fuzzy systems [14].

2.2. Game. Game theory is a study of mathematical models of conflict and cooperation between intelligent rational decision makers [15]. In 1928, Von Neumann proved the basic principle of game theory, which formally declared the birth of game theory. Due to the superiority of understanding and modeling conflict, game theory has been studied in many fields. Hafezalkotob and Makui proposed a Nash game approach for supply chains competition considering two main sources of uncertainty: customers purchasing behaviors and rival chains strategies. Sekine et al. develop a scheme dealing with fairly allocated problems in the multicriteria environment based on data envelopment analysis (DEA) and game theory. Jia et al. proposed a distributed localization method based on game theory for road sensor networks. Game theory has recently been used in the field of computer network security. Reference [16] proposes a model to reason the friendly and hostile nodes in secure distributed computation using game theoretic framework. Reference [17] presents an incentive-based method to model the interactions between a DDoS attacker and the network administrator, and a game theoretic approach to inferring intent, objectives, and strategies (AIOS). References [18, 19] also focused on DDos attack and defense mechanisms using game theory. Reference [20] models the interactions between an attacker and the administrator as a two-player stochastic game and computes Nash equilibria using a nonlinear program. For more related works about applying game theory in network security, the reader can be referred to [21].

3. Problem Formalization

In this section, we present some preliminary concepts and our assumptions on computing infrastructure and missions.

A distributed system consists of a collection of autonomous physical hosts, connected through a network and distribution middleware, which enables these hosts to coordinate their activities and to share the computing resources of the system. Modern organizations tend to use distributed system as means to facilitate the collection, storage, processing, and exchange of data to increase operational efficiency, improve decision quality, and reduce costs. In other words, organizations improve the quality of the completion of mission taking the advantage of the distributed system. About the notion of mission, several literatures have given the definition based on their scenes. Reference [22] views mission as the task, together with the purpose, that clearly indicates the action to be taken and the reason therefore. Donley [23] analyzed the use of the term mission across multiple DoD documents and identified that, more commonly, a mission is "considered generally as integrating many activities around a common theme or purpose." To simplify our model, we view the mission as a set of tasks, known as mission's subtask chain, which have logical relationship and make a contribution to a common objective. Noting that mission is a long-term objective, we assume that mission's subtask chain will be executed many times.

We simply define a mission M as a set of tasks $M = \{T1, T2, T3, \ldots, Tm\}$. Each task produces payoff to the mission. A mission is successful if all its tasks are correctly executed. In many cases, these tasks may be executed repeatedly. For each task $Ti \in M$, a set of procedure replicas can fulfill the task, $Ti = \{Pi1, Pi2, Pi3, \ldots, Pin\}$. For example, both ftp and NFS can achieve the task of remote file accessing. In our framework, we define the set P of procedures to be deployed $P = UTi$. The physical hosts in distributed system provide different kinds of procedures so that we can deploy mission's subtasks into procedures and execute the tasks separately. Assumption that there are adequate hosts provides $Pj \in P$.

The deployment is a function d: $M \rightarrow P$ which maps each task $Ti \in M$ to a procedure $Pj \in P$. The binary variable dij denotes the truth value of $d(Ti) = Pj$; that is, $dij = 1$ if $d(Ti) = Pj$, 0 otherwise. Also, each procedure and its executing environment can introduce vulnerability and failures. At the same time the executed tasks produce positive payoff to the mission, the attacks to these procedures and their executing environment produce negative payoff to the mission. The purpose of this paper is to find a deployment strategy to optimize mission payoff in the face of attacking. Let $A = \{A1, A2, A3, \ldots, Ak\}$ be the set of attacks. The attack influence to the mission will be described in the next section.

4. Game Model

Rational attackers will consider both attack cost and benefit comprehensively before implementing attacking and choose the attack which costs less and wins more, meaning higher payoff. However, irrational attacker will aggressively seek higher benefit without regard to attack cost. This paper focuses on the rational attackers and discusses the best defense strategies.

In the gaming process, both attackers and defenders attempt to gain maximized payoff through optimal strategies. Based on the assumption that they are rational, we formalize their conflict relations as game model and work out attack intention and best defense strategies.

4.1. Model Definition. *Game*: A description of the strategic interaction between opposing or cooperating interests where the constraints and payoff for actions are taken into consideration [21]. There are three basic elements in game model Game = (E, S, U).

(1) $E = (E1, E2, \ldots, En)$ means a set of basic entities in a game which join in the game and make choices of strategies. An entity can represent a person, machine, or group of persons within a game. In this paper, there are two entities, attacker and defender.

(2) $S = (S1, S2, \ldots, Sn)$ means strategies set of entities within the game which a given entity can take during game play. Si is the set of strategies for entity Ei and, for each, entity the number of strategies is no less than 2; for example, $Si = (s_1^i, s_2^i, \ldots, s_{mi}^i)$ and mi is the number of strategies for entity Ei. In this paper, defender's strategies are different deployment strategies with different vulnerabilities exposed.

(3) $U = (U1, U2, \ldots, Un)$ means utility function set of entities. Utility function is the function of all the entities' strategies which presents the payoffs entities can gain from game. We will make quantification example of utility function in next section.

4.2. Utility Function. The quantification of utility function is mainly based on one security evaluation and risk analysis. Many literatures have studied in this area; for example, [24] proposes intrusion detection systems (IDSs) cost model based on cost quantification, cost classification, and attack classification. Since this part of content is not the focus of this paper, here, we just give the simple quantification based on the factors of benefit, cost, and successful probability for attack and defense separately.

Attack benefit (AB) presents the reward attackers can obtain by launching a successful attack. AB is related to attack strategy.

Attack cost *(AC)* presents resources, professional knowledge, or response to legal sanction spent by attacker to launch an attack. Noting that no matter whether the attack is successful or not, the same cost has to be paid. To simplify the analysis, we can regard expected cost of statistical result as AC. AC is related to attack strategy.

Attack success probability (ASP) presents the possibility of whether the attack can be successful. This factor is impacted by attack strategy and defense strategy. If attacker attempts to launch attack A when defender happens to implement corresponding defense strategy and the vulnerability exploited by attack A is not exposed, the ASP must below, even to 0. Otherwise, if there happens to be no defense strategy aiming at attack A, the ASP may be high. ASP is related to both attack strategy and defense strategy. So ASP is a matrix, as in Table 1.

Table 1: ASP matrix.

	Defense 1	Defense 2	⋯	Defense n
Attack 1	ASP [1][1]	ASP [1][2]	⋯	ASP [1][n]
Attack 2	ASP [2][1]	ASP [2][2]	⋯	ASP [2][n]
⋮	⋮	⋮	⋮	⋮
Attack n	ASP [n][1]	ASP [n][2]	⋯	ASP [n][n]

Payoff of attacker (PoA) presents the payoff an attacker can gain from game play. The utility function is as follows:

$$\text{PoA} = \text{AB} * \text{ASP} - \text{AC}. \quad (1)$$

Defense benefit (DB) presents the reward defenders can obtain by launching an effective defense against an attack. In the mission environment, this reward can be contributions to mission made by the mission subtasks' normal execution. DB is related to defense strategy, meaning deployment strategy.

Defense cost (DC) presents resources and operational time spent by defenders to launch a defense. Noting that no matter whether the defense is effective or not, the same cost has to be paid. To simplify the analysis, we can regard expected cost of statistical result as DC. DC is related to defense strategy.

Defense effective probability (DEP) presents the possibility of whether the defense is effective against attack. The defense is effective means that the exposed vulnerabilities cannot be exploited or are hard to be exploited. This factor is impacted by attack strategy and defense strategy. The meaning of DEP is similar to ASP. DEP is also a matrix, as in Table 2.

Payoff of defender (PoD) presents the payoff a defender can gain from game play. The utility function is as follows:

$$\text{PoD} = \text{DB} * \text{DEP} - \text{DC}. \quad (2)$$

Table 2: DEP matrix.

	Attack 1	Attack 2	⋯	Attack n
Defense 1	DEP [1][1]	DEP [1][2]	⋯	DEP [1][n]
Defense 2	DEP [2][1]	DEP [2][2]	⋯	DEP [2][n]
⋮	⋮	⋮	⋮	⋮
Defense n	DEP [n][1]	DEP [n][2]	⋯	DEP [n][n]

Based on utility functions, we can work out payoff matrix. For convenience of discussion, Game scene Game = (E, S, U) will be given as follows.

$E = (Ea, Ed)$ means that, in the game, there are two entities: attacker and defender. Attacker chooses attack strategy to exploit mission tasks vulnerabilities and defender chooses task deployment strategy to decrease threat caused by vulnerabilities exposed.

$S = (Sa, Sd)$ means each entity, attacker or defender, in the game has a strategy set separately. $Sa = (A1, A2, A3)$ means attacker has three strategies to exploit mission tasks. For defender, given $M = \{T1, T2\}$ and $T1 = \{P11, P12\}$, $T2 = \{P21, P22, P23\}$ which means that the mission has two tasks and task one has two procedure replicas and task two has three procedure replicas, so defender has six deployment strategies $Sd = ((P12, P21), (P12, P22), (P12, P23), (P11, P21), (P11, P22)(P11, P23))$.

$U = (Ua, Ud)$ means each entity, attacker or defender, in the game has a utility function set separately. Quantification of utility function has been discussed above in this section.

4.3. Nash Equilibrium

Definition 1 (Nash equilibrium). In the Game = $((Ea, Ed), (Sa, Sd), (Ua, Ud))$, a pair of strategies (s^{a*}, s^{d*}) is a Nash equilibrium, if and only if for each entity e in the game, s^{e*} is the best response for the other entity: $\forall s^a \in Sa, Ua(s^{a*}, s^{d*}) \geqslant Ua(s^a, s^{d*})$; $\forall s^d \in Sd, Ud(s^{a*}, s^{d*}) \geqslant Ud(s^{a*}, s^d)$.

In the Nash equilibrium, the entities choose strategies that are best responses to each other; then, no entity has an incentive to deviate to an alternative strategy. So the system is in a kind of equilibrium state, with no force pushing it toward a different outcome. Nash equilibrium can be thought of as equilibrium in beliefs. If each entity believes that the other entity will actually play a strategy that is part of Nash equilibrium, then they are willing to play their part of the Nash equilibrium. Noting mission's subtask chain will be executed many times; we should consider attack and defend mixed strategies game.

Definition 2 (mixed strategy). In the Game = $((Ea, Ed), (Sa, Sd), (Ua, Ud))$, the mixed strategies of attacker and defender are the probabilities of $Sa = (s_1^a, s_2^a, s_3^a)$ and $Sd = (s_1^d, s_2^d, s_3^d, s_4^d, s_5^d, s_6^d)$, $Pa = (p_{a1}, p_{a2}, p_{a3})$ and $Pd = (p_{d1}, p_{d2}, p_{d3}, p_{d4}, p_{d5}, p_{d6})$. For Pa and Pd, $0 \leqq p_{ai} \leqq 1$, $0 \leqq p_{dj} \leqq 1$, $\sum_i p_{ai} = 1$,and $\sum_j p_{dj} = 1$.

In the mixed strategy game, the payoffs expectation of attacker and defender are computed as follows:

$$\begin{aligned}\overline{U}_a(P_a, P_d) &= \sum_i p_{ai}\left[\sum_j p_{dj} U_a\left(s_i^a, s_j^d\right)\right] \\ &= \sum_i \sum_j p_{ai} \cdot p_{dj} \cdot U_a\left(s_i^a, s_j^d\right) \\ \overline{U}_d(P_a, P_d) &= \sum_j p_{dj}\left[\sum_i p_{ai} U_d\left(s_i^a, s_j^d\right)\right] \\ &= \sum_j \sum_i p_{ai} \cdot p_{dj} \cdot U_d\left(s_i^a, s_j^d\right).\end{aligned} \tag{3}$$

Theorem 3. *If the mixed strategies situation (Pa, Pd) is a Nash equilibrium, then, for each entity $\mathbf{e}$ in the game, each pure strategy $s_{\mathbf{m_e}}^{\mathbf{e}}$ ($\mathbf{m_e}$ is the number of pure strategies for entity $\mathbf{e}$) satisfies:*

$$\begin{aligned}\overline{\mathbf{U}}_{\mathbf{e}}\left(\left(P_a, P_d\right) \| \mathbf{s}_{\mathbf{i}}^{\mathbf{e}}\right) &\leq \overline{\mathbf{U}}_{\mathbf{e}}\left(P_a, P_d\right) \\ &\left(\mathbf{i} = \mathbf{1}, \mathbf{2}, \ldots, \mathbf{m_e}\right).\end{aligned} \tag{4}$$

$(P_a, P_d)\|\mathbf{s}_{\mathbf{i}}^{\mathbf{e}}$ means only entity e replaces strategy in mixed strategies by pure strategy $\mathbf{s}_{\mathbf{i}}^{\mathbf{e}}$, and other entities' strategies do not change.

Based on Theorem 3, if the mixed strategies situation (Pa, Pd) is not a Nash equilibrium, then there is at least one $\mathbf{s}_{\mathbf{i}}^{\mathbf{e}}$ satisfying:

$$\begin{aligned}\overline{\mathbf{U}}_{\mathbf{e}}\left(\left(P_a, P_d\right) \| \mathbf{s}_{\mathbf{i}}^{\mathbf{e}}\right) - \overline{\mathbf{U}}_{\mathbf{e}}\left(P_a, P_d\right) &< \mathbf{0} \\ &\left(\mathbf{i} = \mathbf{1}, \mathbf{2}, \ldots, \mathbf{m_e}\right).\end{aligned} \tag{5}$$

So, we can figure out the following:

$$\begin{aligned}\mathbf{min}\left\{\overline{\mathbf{U}}_{\mathbf{e}}\left(\left(P_a, P_d\right) \| \mathbf{s}_{\mathbf{i}}^{\mathbf{e}}\right) - \overline{\mathbf{U}}_{\mathbf{e}}\left(P_a, P_d\right)\right\} &< \mathbf{0} \\ &\left(\mathbf{i} = \mathbf{1}, \mathbf{2}, \ldots, \mathbf{m_e}\right).\end{aligned} \tag{6}$$

On the other hand, according to the implication of Nash equilibrium, in Nash equilibrium situation, for given other entities' strategies, the pure strategies of each entity should gain the same payoffs. As a result, in Nash equilibrium situation, for each entity, it satisfies:

$$\begin{aligned}&\overline{\mathbf{U}}_{\mathbf{e}}\left(\left(P_a, P_d\right) \| \mathbf{s}_{\mathbf{i}}^{\mathbf{e}}\right) - \overline{\mathbf{U}}_{\mathbf{e}}\left(P_a, P_d\right) = \mathbf{0} \\ &\qquad \left(\mathbf{i} = \mathbf{1}, \mathbf{2}, \ldots, \mathbf{m_e}\right) \\ &\lim_{\substack{(P_a, P_d) \to \mathbf{Nash} \\ \mathbf{equilibrum}}} \mathbf{min}\left\{\overline{\mathbf{U}}_{\mathbf{e}}\left(\left(P_a, P_d\right) \| \mathbf{s}_{\mathbf{i}}^{\mathbf{e}}\right) - \overline{\mathbf{U}}_{\mathbf{e}}\left(P_a, P_d\right)\right\} = 0^- \\ &\qquad \left(\mathbf{i} = \mathbf{1}, \mathbf{2}, \ldots, \mathbf{m_e}\right).\end{aligned} \tag{7}$$

By integrating formulas (5), (6), and (7), we figure out a function based on which PSO fitness function can be designed in Section 5.2:

$$\begin{aligned}F = \sum_e \left|\mathbf{min}\left\{\overline{\mathbf{U}}_{\mathbf{e}}\left(\left(P_a, P_d\right) \| \mathbf{s}_{\mathbf{i}}^{\mathbf{e}}\right) - \overline{\mathbf{U}}_{\mathbf{e}}\left(P_a, P_d\right)\right\}\right| \\ \left(\mathbf{i} = \mathbf{1}, \mathbf{2}, \ldots, \mathbf{m_e}\right).\end{aligned} \tag{8}$$

F satisfies

$$F = \begin{cases} 0, & \text{if } (P_a, P_d) \text{ is Nash equilibrium;} \\ R^+, & \text{if } (P_a, P_d) \text{ isn't Nash equilibrium,} \end{cases} \qquad \lim_{(P_a,P_d) \to \text{Nash equilibrum}} F = 0^+. \tag{9}$$

5. Particle Swarm Optimization

5.1. PSO Framework. Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995 [25], inspired by social behavior of bird flocking or fish schooling.

In PSO, a swarm of particles is represented as potential solution, and firstly a swarm of random particles is initialized. Each particle is associated with two vectors, velocity and position; that is, the velocity vector $V_i = [V_i^1, V_i^2, \ldots, V_i^D]$ and the position vector $X_i = [X_i^1, X_i^2, \ldots, X_i^D]$, where D stands for the dimensions of the solution space and i stands for the ith particle. Each particle is also associated with a fitness value decided by a unified fitness function. During the evolutionary process, the velocity and position of particle are updated followed by two current best particle positions. The first "best" particle position is called local best which is the historical best position of the particle itself. The second "best" particle value is called global best which is the best position in the neighborhood. The velocity and position of particle are updated as follows:

$$\begin{aligned} V_i^d &= \omega V_i^d + c_1 \cdot r_1 \cdot \left(l\text{Best}_i^d - X_i^d\right) \\ &\quad + c_2 \cdot r_2 \cdot \left(g\text{Best}_i^d - X_i^d\right) \\ X_i^d &= X_i^d + V_i^d, \end{aligned} \tag{10}$$

where ω is the inertia weight, c_1 and c_2 are the acceleration coefficients, and r_1 and r_2 are two uniformly random numbers independently generated within $[0, 1]$. lBest is local best position and gBest is global best position.

5.2. PSO Design for Game. In this algorithm, we regard a game strategies situation as a particle position. In the game scene described in Section 4.2, particle position is like $X = (Pa, Pd) = ((p_{a1}, p_{a2}, p_{a3}), (p_{d1}, p_{d2}, p_{d3}, p_{d4}, p_{d5}, p_{d6}))$, and first vector is attack mixed strategies and second vector is defend mixed strategies. We use x_j^i for the probability of the jth strategy for entity i, noting that the position in this algorithm is multidimensional vector group which is different from the description in Section 5.1. In Nash equilibrium, every entity's strategy is best response for others, so the particle which represents Nash equilibrium situation has the best fitness value. During the evolutionary process, particles search the best position within game space and update position following by local best solution and global best solution and move gradually to Nash equilibrium situation.

In order to improve the convergence speed, as proposed by Shi and Eberhart [26], in this algorithm, ω is linearly decreasing with the iterative generations as follows:

$$\omega = \omega_{\max} - (\omega_{\max} - \omega_{\min}) \frac{g}{G}, \tag{11}$$

where g is the generation index representing the current number of evolutionary generations and G is a predefined maximum number of generations.

Based on the Nash equilibrium definition and formula (8), we design the fitness function as follows:

$$f(X) = \sum_{i=1}^{n} \left| \min \left(\overline{U}_i(X) - \overline{U}_i\left(X \| s_j^i\right) \right) \right| \qquad j = 1, 2, \ldots, m_i. \tag{12}$$

Obviously, if and only if X is Nash equilibrium, the fitness function is equal to the minimum value 0. That means, in the situation of Nash equilibrium, for every entity in the game, no matter which strategy they choose, the payoff is the same. If X is not Nash equilibrium, then $f(X) > 0$.

In addition, in order to keep particles staying in game space during the evolutionary process, we should control the particle position space. Since the vectors of particle position are representation of mixed strategies, so the sum of vector elements must be 1 and the elements must be positive numbers; namely, $\sum_{j=1}^{m_i} x_j^i = 1$, $x_j^i \geq 0$, $i = 1, 2, \ldots, n$, m_i is the number of entity i's pure strategies. We also ensure that the initial particle velocity $\sum_{j=1}^{m_i} v_j^i = 0$, $i = 1, 2, \ldots, n$, so that, in every generation, the velocity is zero.

Another factor, particle update steps, is needed to be control to keep particles staying in game space. The particles update positions are followed by the function $X_i^d = X_i^d + V_i^d$. To keep $x_j^i \geq 0$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m_i$, we add the step control factor in the function:

$$X_i^d = X_i^d + \alpha^i V_i^d, \tag{13}$$

$$\forall v_j^i < 0, \quad \alpha_j^i = \frac{-x_j^i}{v_j^i}, \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m_i, \tag{14}$$

$$\forall v_j^i \geq 0, \quad \alpha_j^i = \frac{1 - x_j^i}{v_j^i}, \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m_i, \tag{15}$$

$$\alpha^i = \min\left(\alpha_j^i\right), \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m_i. \tag{16}$$

Based on the above-mentioned design, it can be ensured that every particle in every generation stays in mixed strategy game space. In this way, many invalid particles are avoided and algorithm performance is improved.

5.3. Algorithm Description

Step 1. Define the particle swarm size Np, maximum number of generations G, maximum and minimum inertia weight

$\omega_{\max}, \omega_{\min}$, and the acceleration coefficients c_1 and c_2. Randomly initialize particles' positions and velocities satisfying $\sum_{j=1}^{m_i} x_j^i = 1$, $x_j^i \geq 0$, $\sum_{j=1}^{m_i} v_j^i = 0$, $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m_i$.

Step 2. Compute the fitness values of particles using function (12) and find out local best lBest and global best gBest.

Step 3. Compute ω using function (11) and r_1 and r_2 using random function. Compute V_i^d.

Step 4. Compute step control factor α for every entity and every particle, and update particles' position using function (13).

Step 5. Renew lBest and gBest. If gBest = 0, then stop; otherwise, turn to Step 3.

5.4. Performance Index. We evaluate the convergence property of algorithm using off-line performance, referred from the method proposed by De Jong [27].

Definition 4 (off-line performance). In generation T, we define $E(T)$ as follows:

$$E(T) = \frac{1}{T} \cdot \sum_{t=1}^{T} f^*(t), \tag{17}$$

where $f^*(t)$ is the best fitness in generation t.

6. Experimental Results

6.1. Convergence Property. In this section, we use Dekel-Scotchmer game [28] as an example. The payoff matrix is as follows:

$$P1 = \left(\begin{pmatrix} 1 \\ 235 \\ 0 \\ 0.1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 235 \\ 0.1 \end{pmatrix} \begin{pmatrix} 235 \\ 0 \\ 1 \\ 0.1 \end{pmatrix} \begin{pmatrix} 1.1 \\ 1.1 \\ 1.1 \\ 0 \end{pmatrix} \right)$$
$$P2 = \left(\begin{pmatrix} 1 \\ 0 \\ 235 \\ 1.1 \end{pmatrix} \begin{pmatrix} 235 \\ 1 \\ 0 \\ 1.1 \end{pmatrix} \begin{pmatrix} 0 \\ 235 \\ 1 \\ 1.1 \end{pmatrix} \begin{pmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0 \end{pmatrix} \right). \tag{18}$$

In the algorithm, the scale of particles is set as 30. The maximum iteration generation is set as 1000. $\omega_{\max} = 0.9$ and $\omega_{\min} = 0.4$. Accuracy is set as $1E-5$. The experiment is performed on matlab R2013a. We perform the experiment for 5 times and achieve Nash equilibrium at $(1/3, 1/3, 1/3, 0; 1/3, 1/3, 1/3, 0)$ after the average iteration generation of 268. The results are as in Table 3.

We choose off-line performance of the last experiment to plot Figure 1 in blue line.

Reference [29] presented a game genetic algorithm (GGA) which realized the choice of strategies using genetic algorithm (GA). We realize the genetic algorithm on matlab R2013a within the same environment as PSO. The comparison results of PSO method proposed in our paper and GA are shown in Figure 1. Obviously, our method can achieve a higher convergence speed. That is because, relative to GA in which chromosomes share information with each other and the entire population of group moves to the optimal area in a uniform way, in PSO only gBest and lBest particles deliver information and the entire population of group moves in a more targeted way, which means a higher convergence speed.

Table 3: Results of convergence experiment.

Times	Generation	Best fitness
1	336	$9.1378e-06$
2	246	$8.5779e-06$
3	245	$9.3169e-06$
4	217	$8.7814e-06$
5	296	$9.6812e-06$

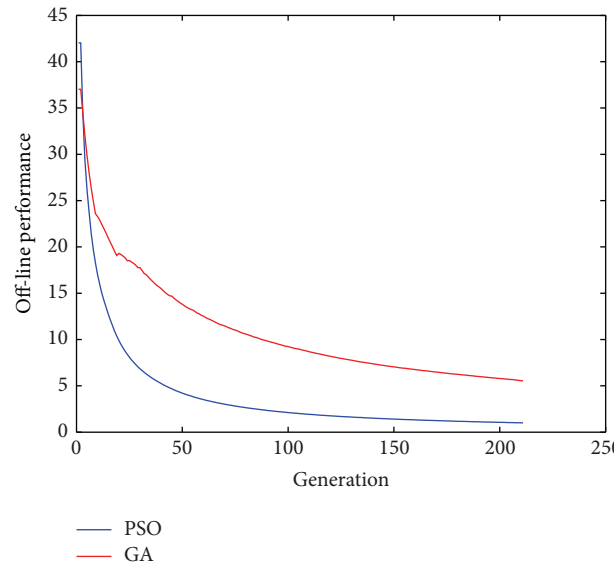

Figure 1: Off-line performance of convergence experiment.

6.2. Effectiveness. In this section, we evaluate the influence of defending strategy on the payoff of defender. As mission deployment must be performed many times as assumption, both defender and attacker are nearly impossible to choose single strategy in practice. As a result, pure strategy is not considered in our discussion.

In our discussion, both attacker and defender have two ways to choose strategy: random way and Nash way. Random way means that entity, attacker or defender, randomly chooses the strategy from strategy set every time. On the other hand, Nash way means that entity chooses the strategy in the same probability as Nash equilibrium, which is more rational than random way. Therefore, there are four situations in our discussion: attack in random way versus defend in random way; attack in random way versus defend in Nash way; attack in Nash way versus defend in random way; attack in Nash way versus defend in Nash way.

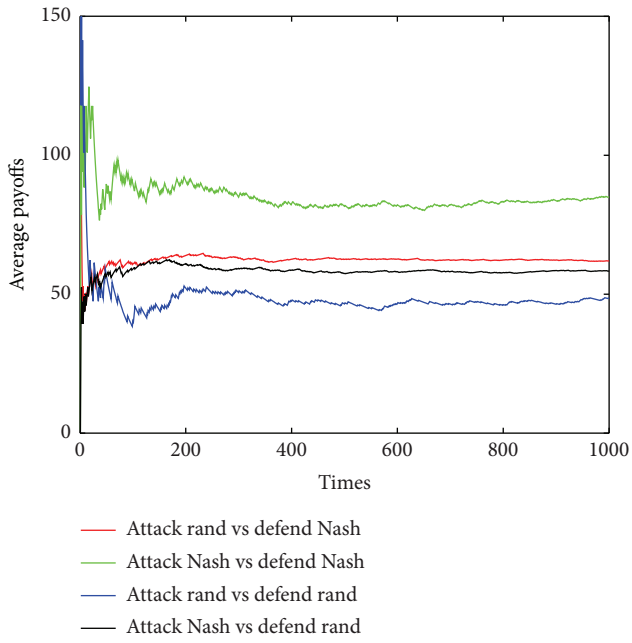

Figure 2: Average payoffs of strategy choosing ways.

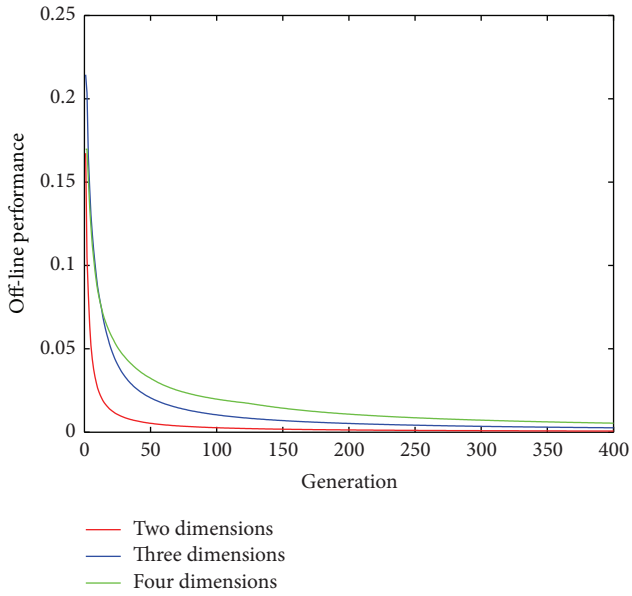

Figure 3: Comparisons of strategy dimensions.

We use game scene and Nash equilibrium in Section 6.1 for reference and the time of mission deployment and attacking is set as 1000. The average payoff of defender is used as evaluation indicator. The experiment is performed on matlab R2013a. The result of experiment is shown in Figure 2.

As shown in Figure 3, defender will achieve highest average payoff, about 78, after 400 times, if both attacker and defender choose strategy in Nash way. If attacker is rational and chooses strategy in Nash way and defender randomly chooses strategy, the average payoff of defender is the lowest, about 50. If one of the entities in the game chooses strategy in random way and the other one choose it in Nash way, defender will achieve medium average payoff, and, in this situation, defenders will achieve a little higher average payoff if they are rational and choose strategy in Nash way. We can see in the Figure 3, in the earlier game stage, the indicator of average payoffs waves seriously that the low average line can be higher than high average line. That is because the average payoffs are still unstable as average values. It is easy to come to the conclusion that defender will achieve higher payoff in Nash way no matter which way attacker will choose.

On the other hand, similarly, attacker will achieve higher payoff when choosing strategy in Nash way.

So we can make the conclusion that both rational attacker and defender will choose strategy in Nash way and jointly safeguard the Nash equilibrium.

6.3. Comparisons of Dimensions. In this section, we evaluate the influence of number of strategies on algorithm performance. In the algorithm, the scale of particles is set as 30. The maximum iteration generation is set as 1000. $\omega_{\max} = 0.9$, $\omega_{\min} = 0.4$. Accuracy is set as $1E-5$. The experiment is performed on matlab R2013a. We calculate the off-line performance for two dimensions, three dimensions, and four dimensions. "Dimension" is the number of strategies in strategy set of the game. The comparison result is shown in Figure 3.

We can see that the average convergence speed is reducing gradually with the increase of the strategy dimensions. In the comparison experiment, we fix PSO parameters and, as the result, the major factor influencing the convergence speed is fitness function. The more dimensions means the greater scale of fitness function (as function (12)) and corresponding higher computing cost.

7. Conclusions

In this paper, we have highlighted the fact that the mission is prone to be vulnerable to kinds of cyberattacks. As a result, we propose a mission deployment strategy using game theory to optimize mission payoff in the face of different attack strategy and design particle swarm optimization algorithm to calculate Nash equilibrium. Experiments show that both rational attacker and defender will choose strategy in Nash way and jointly safeguard the Nash equilibrium. However, we can see in Nash equilibrium, the payoff of defender is just the better one, not the best one globally. So our future work will be driven towards the way to achieve the integration of Nash equilibrium and "social best" which means in Nash equilibrium the payoff of defender is the social best and the payoff of attacker is the worst.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] M. R. Grimaila, R. F. Mills, M. Haas, and D. Kelly, "Mission assurance: issues and challenges," Report Documentation, Air Force Institute of Technology, Wright-Patterson, AFB Ohio Center for Cyberspace Research, 2010.

[2] M. Albanese, S. Jajodia, R. Jhawar, and V. Piuri, "Reliable mission deployment in vulnerable distributed systems," in *Proceedings of the 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W '13)*, pp. 1–8, IEEE, Budapest, Hungary, June 2013.

[3] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors," *Journal of the Association for Computing Machinery*, vol. 24, no. 2, pp. 280–289, 1977.

[4] R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions," in *Proceedings of the 7th Heterogeneous Computing Workshop (HCW '98)*, pp. 79–87, Orlando, Fla, USA, March 1998.

[5] V. M. Lo, "Heuristic algorithms for task assignment in distributed systems," *IEEE Transactions on Computers*, vol. 37, no. 11, pp. 1384–1397, 1988.

[6] S. Srinivasan and N. K. Jha, "Safety and reliability driven task allocation in distributed systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 3, pp. 238–251, 1999.

[7] T. Xie, X. Qin, and A. Sung, "SAREC: a security-aware scheduling strategy for real-time applications on clusters," in *Proceedings of the International Conference on Parallel Processing*, pp. 5–12, Oslo, Norway, June 2005.

[8] S.-W. Xiong, Y.-X. Zhao, and N. Xu, "SAREC-GA: a security-aware real-time scheduling algorithm with genetic algorithm," in *Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC '07)*, vol. 6, pp. 3122–3127, Hong Kong, August 2007.

[9] Z. Shuai, H. Zhang, J. Wang, J. Li, and M. Ouyang, "Lateral motion control for four-wheel-independent-drive electric vehicles using optimal torque allocation and dynamic message priority scheduling," *Control Engineering Practice*, vol. 24, no. 1, pp. 55–66, 2014.

[10] H. Zhang, Y. Shi, and A. Saadat Mehr, "Robust static output feedback control and remote PID design for networked motor systems," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 12, pp. 5396–5405, 2011.

[11] H. Zhang, Y. Shi, and M. Liu, "H_{∞} step tracking control for networked discrete-time nonlinear systems with integral and predictive actions," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 337–345, 2013.

[12] Z. Shuai, H. Zhang, J. Wang, J. Li, and M. Ouyang, "Combined AFS and DYC control of four-wheel-independent-drive electric vehicles over CAN Network with time-varying delays," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 2, pp. 591–602, 2014.

[13] H. Zhang, Y. Shi, and J. Wang, "On energy-to-peak filtering for nonuniformly sampled nonlinear systems: a markovian jump system approach," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 1, pp. 212–222, 2014.

[14] H. Zhang and J. Wang, "State estimation of discrete-time Takagi-Sugeno fuzzy systems in a network environment," *IEEE Transactions on Cybernetics*, 2014.

[15] R. B. Myerson, *Game Theory: Analysis of Conflict*, Harvard University Press, 1991.

[16] P. F. Syverson, "A different look at secure distributed computation," in *Proceedings of the 10th IEEE Computr Security Foundations Workshop (CSFW '97)*, pp. 109–115, June 1997.

[17] P. Liu, W. Zang, and M. Yu, "Incentive-based modeling and inference of attacker intent, objectives, and strategies," *ACM Transactions on Information and System Security*, vol. 8, no. 1, pp. 78–118, 2005.

[18] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *Computer Communication Review*, vol. 34, no. 2, 2004.

[19] J. Xu and W. Lee, "Sustaining availability of web services under distributed denial of service attacks," *IEEE Transactions on Computers*, vol. 52, no. 2, pp. 195–208, 2003.

[20] K.-W. Lye and J. M. Wing, "Game strategies in network security," *International Journal of Information Security*, vol. 4, no. 1-2, pp. 71–86, 2005.

[21] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu, "A survey of game theory as applied to network security," in *Proceedings of the 43rd Annual Hawaii International Conference on System Sciences (HICSS '43)*, pp. 1–10, January 2010.

[22] JP1-02, "Joint Publication 1-02, Department of Defense Dictionary of Military and Associated Terms," October 2009.

[23] M. B. Donley, "Problems of defense organization and management," *Joint Forces Quarterly*, vol. 8, pp. 86–94, 1995.

[24] W. Lee, W. Fan, M. Miller, S. J. Stolfo, and E. Zadok, "Toward cost-sensitive modeling for intrusion detection and response," *Journal of Computer Security*, vol. 10, no. 1-2, pp. 5–22, 2002.

[25] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[26] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE World Congress on Computational Intelligence*, Evolutionary Computation Proceedings, pp. 69–73, 1998.

[27] K. A. De Jong, *Analysis of the behavior of a class of genetic adaptive systems [Doctoral dissertation]*, 1975.

[28] E. Dekel and S. Scotchmer, "On the evolution of optimizing behavior," *Journal of Economic Theory*, vol. 57, no. 2, pp. 392–406, 1992.

[29] H. Xu, Y. Ding, and Z. Hu, "Evolutionary design of combinational circuit based on game genetic algorithm," *Journal of Computer Applications*, vol. 3, p. 079, 2009.