# Heuristic Network Security Risk Assessment Based on Attack Graph

Wei Sun[1], Qianmu Li[1,2(✉)], Pengchuan Wang[1], and Jun Hou[3]

[1] Nanjing University of Science and Technology, Nanjing, China
{sw24816,qianmu,wangpc}@njust.edu.cn
[2] Intelligent Manufacturing Department, Wuyi University, Nanping, China
[3] School of Social Science, Nanjing Vocational University of Industry Technology, Nanjing, China

**Abstract.** With the development of attack technology, attackers prefer to exploit multiple vulnerabilities with a combination of several attacks instead of simply using violent cracking and botnets. In addition, enterprises tend to adopt microservices architectures and multi-cloud environments to obtain high efficiency, high reliability and high scalability. It makes modeling attack scenarios and mapping the actions of potential adversaries an urgent and difficult task. There have been many improvements that can automatically generate attack graphs for complex networks. However, extracting enough effective information from such complex attack graphs is still a problem to be solved. Traditional algorithms can't always accomplish this task because of variable and complex attack graph inputs. In contrast, heuristic algorithms have the advantages of adaptability, self-learning ability, robustness and high efficiency. In this paper, we present heuristic algorithms to complete the analysis of attack graphs, including fusion algorithm of particle swarm optimization (PSO) algorithm and grey wolf optimization (GWO) algorithm for finding the spanning arborescence of maximum weight and improved genetic simulated annealing (GA-SA) algorithm for finding attack path with the biggest risk. Also, we present a method for node importance evaluation based on the interpretive structural modeling (ISM) method. We test our methods on a multi-cloud enterprise network, and the result shows that our methods perform well.

**Keywords:** Attack graph · Attack paths · Heuristic algorithm · CVE · Cyber security

## 1 Introduction

The report on major global cyber attacks and data breaches in the first half of 2021 pointed out that cyber attacks, hacking organizations, and data breaches have always existed in the cyber world. For example, 30 TB data in a Brazilian database was destroyed in January, 220 million people were affected. In February,


M. R. Khosravi et al. (Eds.): CloudComp 2021, LNICST 430, pp. 181–194, 2022.
https://doi.org/10.1007/978-3-030-99191-3_14

the user data of Clubhouse was stolen by malicious hackers or spies and published on the third-party website. In March, 8 million COVID-19 nucleic acid test results leaked in India, and the information contained sensitive personal information such as name, age, marital status, test time, residential address, etc. Therefore, grasping the current network security situation in real-time in a complex and changeable network environment, providing early warning and protection against the security situation, and reducing the harm of network attacks is the primary task of network security work.

Attack graph technology displays possible attack paths in the network in a graphical form, helping defenders to understand the relationship between vulnerabilities in the target network intuitively so that the defenders can take corresponding defensive measures. Phillips and Swiler [16] proposed the concept of attack graph and applied it to network vulnerability analysis. With the development of attack graph technology, it is used in alert information correlation analysis, network security risk assessment, network intrusion intent identification, etc. Unlike traditional passive defense technology, attack graph models attack scenarios from attackers' perspective, showing the relationship between the exploitation of vulnerabilities. By analyzing the attack graph, defenders can prepare for possible cyber attacks in advance and reduce security risks.

In recent decades, many heuristic algorithms have been proposed to solve complex optimization problems in engineering technology. Swarm intelligence algorithm is a new bionic evolutionary algorithm, mainly including ACO and PSO algorithms. Swarm intelligence algorithm has strong robustness and is easy to implement. Distributed computer system makes it effective enough to be applied to many fields, such as function optimization, multi-objective optimization, solving integer constraints and mixed integer constraint optimization, neural network training, signal processing, routing algorithms, etc. The practical results have proved the feasibility and efficiency of these algorithms [7]. Genetic algorithm is also a classic heuristic algorithm used in many fields. However, it is easy to fall into local optimization and precocity. In advantages of the simulated annealing algorithm complement the shortcomings of the genetic algorithm [9]. Different heuristic algorithms perform great differently when facing the same problem, so it is necessary to choose the proper algorithm and improve it according to the actual situation.

## 2 Related Work

After Phillips and Swiler [16] proposed the concept of attack graph, attack graph generation technology has always been a hot research topic. Sheyner et al. [13] created state enumeration attack graphs. Each node of the state enumeration attack graph represents one state of the entire system, and edges mean attacks lead to state transitions. They used finite automata to get the attack path in the graph, but it didn't play well when the graph size became bigger. Ammann et al. [1] made each node a system condition instead of the entire system state and made edges the relationships between nodes. This type of attack graph can

reflect the dependencies between system conditions. In their analysis method, attackers will always hold the permissions that have been obtained, which is closer to the real situation.

Ou et al. [11] groundbreakingly proposed the MulVal framework. MulVal is an end-to-end framework and reasoning system that can perform multi-host, multi-stage vulnerability analysis on the network. MulVal has two different nodes: derivation nodes and fact nodes. Ingols et al. [6] developed a system called NetSPA to build multiple-prerequisite graphs from massive source data, and this kind of attack graph has greater expressive power. Ibrahim et al. [5] presented a method to generate attack graphs for microservice architecture by relating microservices to network nodes. Liu et al. [8] combined the attack graph and evidence graph. This combination of refined attack graphs and evidence graphs can help defenders compute or refine potential attack success probabilities.

As important as generating attack graphs, many scholars are also concerned about how to analyze attack graphs. Dai et al. [2] used the fuzzy comprehensive evaluation method to quantify the number and the length of attack paths, combined with attack paths to discover threats to the network and analyze the riskiest attack paths in the network system to predict the risks of potential attacks to network security. Abraham et al. [14] judged the value of the network security situation by analyzing the variation of the vulnerability life cycle with the release event and the relationship between the attack path and the vulnerability life cycle. Musa et al. [10] presented an effective model of depicting the devices and the data flow that efficiently identifies the weakest nodes along with the concerned vulnerability's origin, making attack graphs easier for the users to interpret and reducing the time taken to identify the attack paths. Stergiopoulos et al. [15] split the analysis of complex attack graphs into multiple steps. The core idea was to use Edmonds' algorithm, graph centrality metrics and clustering on attack graphs weighted with risk assessment calculations. It provided automated prioritization of systems and detected vulnerabilities.

## 3 Heuristic Network Security Risk Assessment Based on Attack Graph

### 3.1 Attack Graph

Common Vulnerabilities and Exposures (CVE) is like a dictionary table, giving a common name for widely recognized vulnerabilities or vulnerabilities that have been exposed. Using a common name can help users share data in various independent vulnerability databases and vulnerability assessment tools. Suppose a vulnerability is specified in a vulnerability report and has a CVE name. In that case, you can quickly find the corresponding patch information in any other CVE-compatible database to solve the security problem. Common Vulnerability Scoring System (CVSS) is an industry open standard designed to measure the severity of vulnerabilities and help determine the urgency and importance of

the required response. CVE and CVSS are both published and updated by the U.S. government repository of standards based vulnerability management data (NVD).

This paper utilizes tools and methods from [5,8] to generate attack graphs. This kind of attack graph is similar to the attack graph introduced in [6], whose nodes correspond to states and edges to vulnerability instances. CVSS provides a way to score vulnerability instances [12] so that we can determine the weight of each edge.

Between two nodes in our attack graph, there could be several edges that indicate different vulnerabilities. To simplify the initial attack graph, We only keep one edge with the most prominent weight between two nodes, in which case we can get the worst scenario of the target network. When conducting network security risk assessments, it is critical to consider the worst scenario.

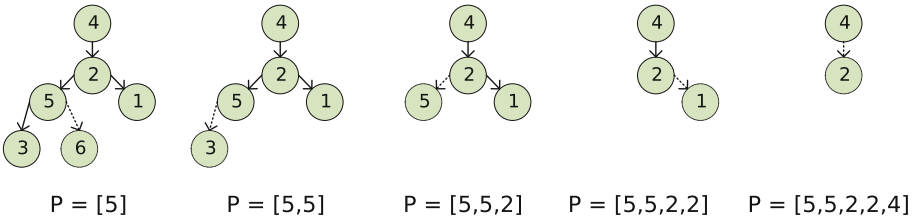


**Fig. 1.** An example of the encoding process of a directed tree.

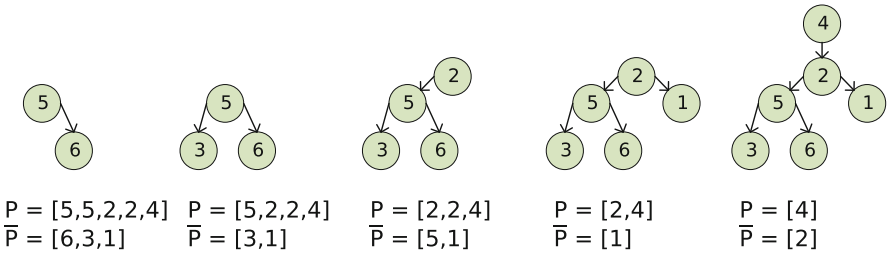


**Fig. 2.** An example of the encoding process of the code string P.

### 3.2 Heuristic Graph Arborescences of Maximum Weight Generation Algorithm

Attack graphs are directed acyclic graphs, and it is applicable to utilize Edmonds' algorithm to find spanning arborescence of minimum weight. After a simple adjustment, it can be used to find spanning arborescence of maximum weight. Edmonds' algorithm is based on the idea of greed and shrinkage, and it can deal with directed graphs with an actual minimum spanning tree, which means

Edmonds' algorithm requires input that meets certain conditions. The problem of finding minimum weight arborescence is similar to the degree-constrained minimum spanning tree problem for undirected graphs. Usually, we employ algorithms to find approximate solutions. Heuristic algorithms have an excellent performance in solving such issues [3].

**Coding Design of Directed Graph Spanning Tree.** As the root node in our attack graph indicates the end goal of attacks, the spanning arborescence $T$ of maximum weight can reveal the potential attack surfaces. $T$ has several characteristics: 1) The in-degree of all nodes in $T$ except the root is 1. 2) There is only one directed path from the root to any other node. According to those characteristics, we learn from the Prüfer coding. There are $n^{n-2}$ different spanning trees in an undirected graph complete graph with $n$ nodes. Prüfer coding can uniquely express every tree by using the arrangement of $n-2$ numbers between 1 and $n$. Learned from Prüfer, we encode the directed graph spanning tree. Suppose that $T$ is a directed tree that has $n$ nodes, and $u$ is its root. We need $n-1$ numbers between 1 and $n$ to express it.

There are four steps when we encode a directed tree:

Step1: Let $j$ be the biggest label among leaves' labels, if $i$ is the starting node of the directed edge $<i, j>$ adjacent to $j$, add $i$ to the rightmost side of the code string $P$.
Step2: Delete $j$ and edge $<i, j>$.
Step3: When the number of nodes left in the tree $T$ is greater than 1, go to the first step and repeat the above steps; when there is only one node left in the tree $T$, go to step 4.
Step4: Output the code string $P$ with $n-1$ numbers.

When decoding code string $P$, we need three steps:

Step1: Let $\overline{P}$ contain node labels do not appear in $P$, and arrange the labels in $\overline{P}$ in descending order from left to right.
Step2: Suppose that $i$ is the leftmost label in $P$ and $j$ is the leftmost label in $\overline{P}$, add edge $<i, j>$ to the tree $T$ and then delete the chosen labels from $P$ and $\overline{P}$. If $i$ no longer appears in the remaining part of $p$, then add $i$ to $\overline{P}$ (keep in descending order).
Step3: Repeat the above steps until there are no labels in $P$ and $\overline{P}$.

Figure 1 shows an example of the encoding process of a directed tree, and Fig. 2 shows an example of the decoding process of the code string obtained in Fig. 1.

**Fusion Algorithm of PSO Algorithm and GWO Algorithm.** Suppose we have a processed attack graph $G = \{V, E\}$, $V = \{v_1, v_2, ..., v_n\}$ is a set of nodes, $E = \{w_{1,2}, ..., w_{i,j}, ..., w_{n-1,n}\}$ is a set of edges. If it exists a directed edge from $i$ to $j$, $w_{i,j}$ represents the weight of the edge. Otherwise, we make $w_{i,j} = -1$. When finding the spanning arborescence $T$ of maximum weight, we select $n-1$

edges each time. In this case, the encoding and decoding algorithms are used to verify whether the selected edges can form a tree. If the selected edges meet the requirements, the encoding algorithm can output a code string with $n-1$ numbers, and the decoding algorithm uses the code string to rebuild the tree while the tree may not exist in $G$ at the very beginning. With the iteration of the fusion algorithm, the tree will meet the requirements more and more, and the algorithm will do its best to find the spanning arborescence $T$ of maximum weight. Suppose that $X = \{x_{1,2}, x_{1,3}, ..., x_{i,j}, ..., x_{n-1,n}\}$ indicates the tree rebuilt by decoding algorithm, $x_{i,j} = 1$ means edge $w_{i,j}$ is selected, otherwise $x_{i,j} = 0$. The mathematical model of the maximum spanning tree problem for weighted directed graphs is as follows, and it can be the fitness function of our fusion algorithm:

$$\begin{aligned} \max \quad & f(x) = \sum_{i=1}^{n} \sum_{\substack{j=1, \\ j \neq i}}^{n} w_{i,j} x_{i,j} \\ \text{s.t.} \quad & \begin{cases} 0 \leq \sum_{i=1}^{n} x_{i,j} \leq 1, \quad j = 1, 2, \ldots, n, \\ x \in X, \end{cases} \end{aligned} \tag{1}$$

The fusion algorithm has the advantages of both PSO algorithm and GWO algorithm. GWO algorithm simulates grey wolf social class and grey wolf hunting process. The leadership levels of wolves are divided into four categories: $\alpha$, $\beta$, $\delta$, $\omega$ wolves, among which $\alpha$, $\beta$ wolves are responsible for leading the entire wolf pack. $\alpha$ wolf has the largest fitness value, wolves have the second and third fitness values are marked as $\beta$ and $\delta$. The optimization process of GWO algorithm is mainly guided by the $\alpha$, $\beta$, $\delta$. Assuming that the number of wolves is $N$ and the search area is $d$-dimensional, the position of the $i$-th wolf can be represented as $X_i = \{x_{i1}, x_{i2}, ..., x_{id}\}$. When the grey wolves search for prey, they will gradually approach the prey and surround it. The mathematical model of this behavior is as follows:

$$\begin{gathered} D = C \circ X_p(t) - X(t) \\ X(t+1) = X_p(t) - A \circ D \\ A = 2a \circ r_1 - a \\ C = 2r_2 \\ a = 2\left(1 - \frac{t}{T_{\max}}\right) \end{gathered} \tag{2}$$

Where $t$ represents the current iteration number, $\circ$ is hadamard product, $A$ and $C$ are the vector of synergy coefficients, $X_p$ indicates the position of prey, $X_t$ is the current position of the wolf. $a$ is the convergence factor, which decreases linearly from 2 to 0 as $t$ increases. $r_1$ and $r_2$ are two random numbers in $[0, 1)$.

Grey wolves have the ability to identify the position of potential prey (optimal solution). The search process is mainly completed by the guidance of $\alpha$, $\beta$, and $\delta$. In each iteration, keep the positions of $\alpha$, $\beta$ and $\delta$, and then update the positions of other search agent wolves (including $\omega$) based on their position information. The mathematical model of this process is as follows:

$$\begin{gathered} D_\alpha = |C_1 \circ X_\alpha - X|, D_\beta = |C_2 \circ X_\beta - X|, D_\delta = |C_3 \circ X_\delta - X| \\ X_1 = X_\alpha - A_1 \circ D_\alpha, X_2 = X_\beta - A_2 \circ D_\beta, X_3 = X_\delta - A_3 \circ D_\delta \\ X(t+1) = \frac{X_1 + X_2 + X_3}{3} \end{gathered} \tag{3}$$

Where $X_\alpha$, $X_\beta$, $X_\delta$ represent the positions of $\alpha$, $\beta$, and $\delta$ grey wolves, $D_\alpha$, $D_\beta$, $D_\delta$ are the distances from the position of the agent wolf $X_t$ to $\alpha$, $\beta$ and $\delta$.

However, when finding the spanning arborescence $T$ of maximum weight, GWO algorithm doesn't play well because the GWO algorithm only considers the position information of $\alpha$, $\beta$ and $\delta$, and ignores the information exchange between grey wolf individuals and their own experience. So the idea of PSO algorithm is introduced to improve the position update process. The new mathematical model is as follows:

$$\begin{aligned} X_i(t+1) &= c_1 r_3 \left(w_1 X_1(t) + w_2 X_2(t) + w_3 X_3(t)\right) \\ &\quad + c_2 r_4 \left(X_{ibest} - X_i(t)\right) \\ w_i &= \frac{|X_i|}{|X_1 + X_2 + X_3|}, i = 1, 2, 3 \end{aligned} \tag{4}$$

Where $c_1$ is the social learning factor that controls the influence of the optimal value of population, $c_2$ is the cognitive learning factor that controls the influence of the optimal value of the individual. $w_1$, $w_2$ and $w_3$ are inertia weights that affect the strength of global optimization capability. $r_3$ and $r_4$ are two random numbers in $[0, 1)$.

The process of the fusion algorithm is divided into six steps:

Step1: Set the size of the population $N$, the dimension $d$, which equals the number of edges, and initialize the values of $A$, $C$, and $a$.
Step2: Randomly generate population individuals $\{X_i, i = 1, 2, 3 \ldots N\}$.
Step3: Use Eq. 1 to calculate the fitness of each individual, and then choose top-3 individuals as $\alpha$, $\beta$ and $\delta$. Use Eq. 3 to get $X_\alpha$, $X_\beta$ and $X_\delta$.
Step4: Use Eq. 2 to calculate the parameter $a$, and then update the values of $A$ and $C$.
Step5: Use Eq. 4 to update the position of each individual, and then calculate the fitness again to update $\alpha$, $\beta$ and $\delta$.
Step6: If the number of iterations reaches the maximum, output $\alpha$; else go back to step3.

**Table 1.** Node ID and description of the corresponding system state

| ID | Discription of state | ID | Discription of state |
|---|---|---|---|
| S1 | Start | S10 | VMGroupsATL (root access privilege) |
| S2 | Admin (root access privilege) | S11 | VMGroupsC lib (root access privilege) |
| S3 | DBServer (execCode [user]) | S12 | VMGroupsLICQ (user access privilege) |
| S4 | DBServer (netAccess [tcp, 1434]) | S13 | WebServer (execCode) |
| S5 | MailServerACLs (root access privilege) | S14 | WebServer (netaccess [tcp, 80]) |
| S6 | MailServerSMTP (root access privilege) | S15 | WebServer (user access privilege) |
| S7 | NatServerOpenSSH1 (user access privilege) | S16 | WorkStation (execCode) |
| S8 | NatServerOpenSSH2 (root access privilege) | S17 | WorkStation (access Malicious input [secretary, 'IE']) |
| S9 | Root access to VMs | S18 | End |

### 3.3 Heuristic Attack Path Finding Algorithm for Maximum Risk

**Coding Design of Path in Directed Graph.** The most significant improvement of our method is the choice of coding method. When looking for paths between two nodes, the conventional coding method may use a binary code string of length $N$, and the value of 1 indicates that the node is selected. Set the value at the position of the start node and the end node as 1, then take other points at random. However, there are too many possibilities for the method, and most of them are useless because the attack graph is directed. Under these circumstances, we designed a new coding method that codes nodes with priority. To obtain a feasible path, the adjacent nodes connected by directed edges need to be put into the alternative list $L$ from the start node to the end node. Each time along the path, there may be multiple nodes as the next choice. The priority of nodes determines which next node to choose to put into the alternative list $L$. The steps of the method are as follows:

Step1: Generate sets of adjacent nodes for each node.
Step2: Select the next node from the set of adjacent nodes of the current node as the next current node.
Step3: Until the selected current node is the end node, otherwise repeat Step2.

**GA-SA Algorithm.** Genetic algorithm is an iterative process. Its global search capability is better than local search capability, while simulated annealing algorithm's local search capability is better than global search capability. GA-SA algorithm gets the advantages of both GA and SA, and becomes a comprehensive algorithm. In this algorithm, the fitness function is set as follows:

$$\max \quad f(x) = \sum w_{node_i, node_{i+1}} \qquad (node_i, node_{i+1}) \in L \tag{5}$$

To ensure the diversity of the population, the crossover probability $P_c$ is set to a relatively large value in the early evolution and a relatively small value in the later evolution. It makes the algorithm converge to the optimal result faster. The mathematical model is as follows:

$$P_c = \begin{cases} \mathrm{e}^{P_{c1} \times (1 - t/T)}, & P_{c1} \times (1 - t/T) > P_{c2} \\ e^{P_{c2}}, & P_{c1} \times (1 - t/T) < P_{c2} \end{cases} \tag{6}$$

Where $t$ represents the current iteration number, $T$ is the max iteration number. $P_{c1}$ and $P_{c2}$ are two numbers in $(0, 1)$.

In the mutation operation, in order to ensure the optimization of alternative list $L$, the principle adopted is: robust individuals try to reduce the mutation rate, and inferior individuals increase the mutation rate, which can be reflected in the following equation:

$$P_m = \begin{cases} \frac{k_1 (F_{\max} - F)}{F_{\max} - F_{\text{avg}}}, & F > F_{\text{avg}} \\ k_2, & F < F_{\text{avg}} \end{cases}. \tag{7}$$

Where $F_{\max}$ indicates the maximum fitness of the population, $F_{avg}$ indicates the average fitness of the population. $k_1$ and $k_2$ are adjustment constants and $k_1 < k_2$.

The process of the GA-SA algorithm is divided into six steps:

Step1: Initialize the population. Set the simulated annealing temperature as $t_0$.
Step2: Calculate fitness function $f(x)$.
Step3: Use the roulette wheel selection algorithm for selection operation.
Step4: Select the crossover method, and perform crossover operation on the selected two individuals according to the probability $P_c$.
Step5: Evaluate each individual after the crossover operation and perform mutation operation with probability $P_m$. In the mutation operation, the probability of child generation varies according to the temperature $T_R$.
Step6: If the number of iterations reaches the maximum, output the optimal result; else go back to step2.

**Table 2.** Edge ID and CVE reference

| ID | CVE reference | Risk | ID | CVE reference | Risk |
|---|---|---|---|---|---|
| E1 | Browse malicious websites | 8 | E14 | CVE-2018-7841 | 6 |
| E2 | – | 0 | E15 | CVE-2004-0840 | 10 |
| E3 | CVE-2010-3847 | 7 | E16 | CVE-2009-1918 | 8 |
| E4 | CVE-2010-3847 | 7 | E17 | CVE-2008-5416 | 4.8 |
| E5 | CVE-2003-0693 | 10 | E18 | CVE-2018-7841 | 6 |
| E6 | CVE-2003-0693 | 10 | E19 | CVE-2008-5416 | 6 |
| E7 | CVE-2007-4752 | 6 | E20 | CVE-2010-3847 | 7 |
| E8 | CVE-2001-0439 | 8 | E21 | CVE-2008-0015 | 9 |
| E9 | CVE-2008-4050 | 9 | E22 | CVE-2008-0015 | 9 |
| E10 | CVE-2008-4050 | 9 | E23 | CVE-2008-0015 | 9 |
| E11 | CVE-2008-0015 | 9 | E24 | CVE-2001-1030 | 8 |
| E12 | CVE-2008-0015 | 9 | E25 | CVE-2009-1535 | 8 |
| E13 | CVE-2009-1918 | 8 | | | |

### 3.4 Node Importance Evaluation Based on ISM

The importance of state nodes is one of the core indicators we can extract from attack graphs. Before calculating the importance, we should reverse the direction of each edge because the root node is the target of attackers, and ISM needs to get the reachability of the graph. The simple method of node importance evaluation is to use the degree value of the node to express the importance of the node. This method is easy to operate, but it is one-sided. The interpretive structure model (ISM) is widely used in various fields. It is used for attribute

recognition, analysis, and investigation in the software development process. We can use ISM methods to identify the superior-subordinate relationship between attributes and establish a structural hierarchy model [4].

Using ISM can divide the attack graph into layers. Nodes at a higher level are more important in the attack graph. Also, the results of ISM and graph arborescences of maximum weight generation can confirm each other. Then, we assign weight information to each layer. The formula to define the weight is as follows:

$$Q_i = \frac{1/L_i}{\sum_1^N (1/L_i)} \tag{8}$$

In the process of analyzing the importance of attack graph nodes, it is necessary to consider not only the level of the node to be evaluated but also the level of other nodes connected to the node. Suppose the coefficient of the in-degree node is $I$, and the coefficient of the out-degree node is $O$. $I$ is always greater than $O$. Using an example to illustrate the reason, the importance of the paper is reflected in the number of citations by others. Finally, the formula for node importance evaluation is as follows:

$$T_i = Q_i \left( I \sum_k Q_{k \to i} W_{k \to i} + O \sum_j Q_{i \to j} W_{i \to j} \right) \tag{9}$$

Where $Q_i$ represents the layer weight of node $i$, $Q_{k \to i}$ represents the layer weight of node $k$ which points to node $i$, $Q_{i \to j}$ represents the layer weight of

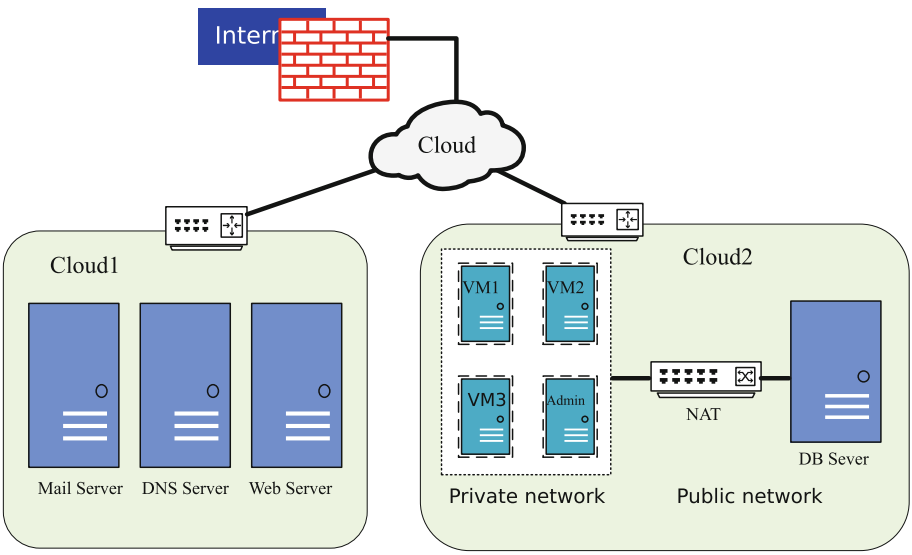


**Fig. 3.** Web application framework.

node $j$ which is pointed by node $i$. $W_{k\to i}$ is the weight of $<k,i>$ and $W_{i\to j}$ is the weight of $<i,j>$. We set $I+O=1$, and $I=0.75$, $O=0.25$ (Fig. 3).

## 4 Experimental Settings and Results

### 4.1 Experimental Environment

To verify our method, we built a client-server web application that contains two cloud infrastructures connected to the Internet through an external firewall [15]. The framework is shown in Fig. 2. After scoring the edges and making edge reduction, we got the attack graph given in Fig. 4(a). It reveals attackers' attack ways aimed at VMs in the private network and the DB server in the public network. The details of the attack graph are contained in Table 1 and Table 2.

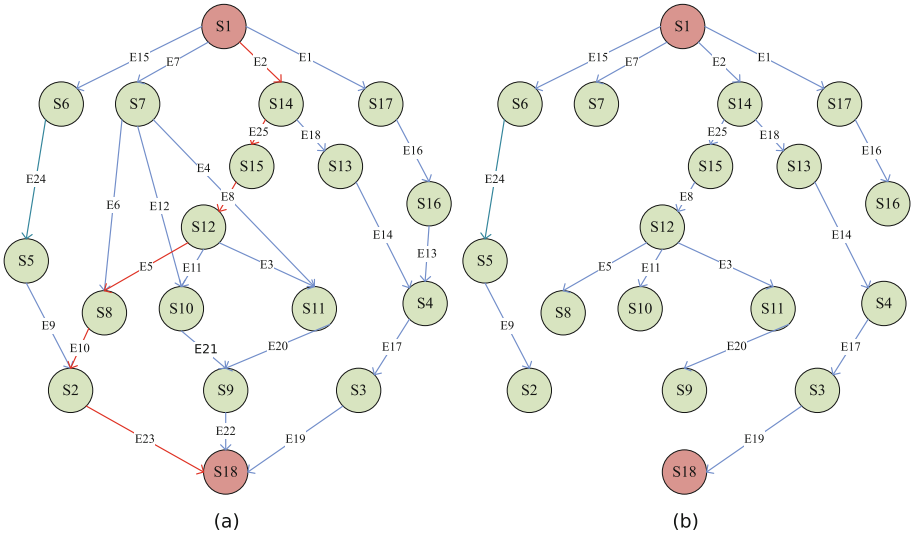


**Fig. 4.** (a) Attack graph of the framework. (b) The spanning arborescence of maximum weight. (Color figure online)

### 4.2 Experimental Results

Firstly, for the purpose of assessing the overall risk of the network, we use the fusion algorithm of PSO and GWO to find the spanning arborescence $T$ of maximum weight. We set the size of the population as 30 and set the maximum number of iterations as 300. In the limited number of iterations, it can always find the target spanning arborescence $T$, which is shown in Fig. 4(b). In addition, the change of its fitness function value can be seen in Fig. 5(a), which shows the process of the algorithm to obtain the optimal result.

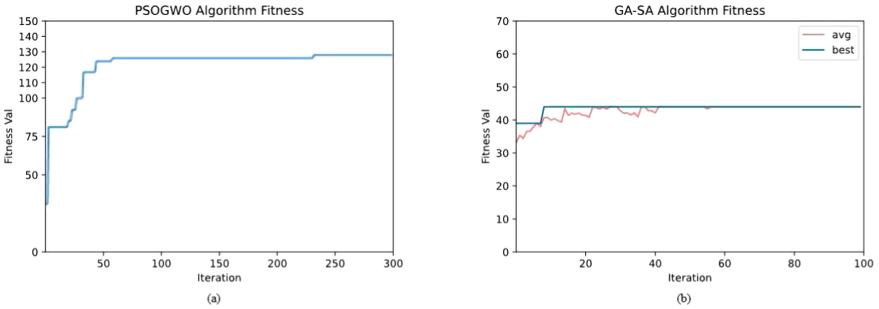


**Fig. 5.** (a) Fitness for PSOGWP. (b) Fitness for GA-SA.

Then, to find the path with maximum risk, we set the size of the population as 15 and set the max number of iterations as 100. The path found in the attack graph is marked red in Fig. 4(a). Its best and average fitness function values are shown in Fig. 5(b).

Finally, measuring the importance of each state node in the attack graph is also a significant job. The result of ISM shows that the graph is divided into four layers. From the first layer to the fourth layer are $\{S1, S6, S7, S14, S15, S17\}$, $\{S5, S12, S13, S16\}$, $\{S4, S8, S10, S11\}$, $\{S2, S3, S9, S18\}$. This result can be mutually confirmed by the spanning arborescence $T$. Table 3 shows the ranking list of node importance.

**Table 3.** Statistical table of importance degree of each node

| ID | Q | Importance | ID | Q | Importance |
|---|---|---|---|---|---|
| S1 | 0.48 | 4.320 | S8 | 0.16 | 0.418 |
| S14 | 0.48 | 1.958 | S10 | 0.16 | 0.389 |
| S7 | 0.48 | 1.843 | S13 | 0.24 | 0.346 |
| S6 | 0.48 | 1.267 | S11 | 0.16 | 0.302 |
| S15 | 0.48 | 1.520 | S2 | 0.12 | 0.205 |
| S17 | 0.48 | 1.520 | S4 | 0.16 | 0.204 |
| S12 | 0.24 | 0.979 | S9 | 0.12 | 0.174 |
| S16 | 0.24 | 0.461 | S3 | 0.12 | 0.088 |
| S5 | 0.24 | 0.425 | S18 | 0.12 | 0.086 |

## 5 Conclusion

This paper proposes two heuristic algorithms and a node importance evaluation method for extracting messages from attack graphs, and they get excellent performance in the test environment. Compared to traditional algorithms, their

advantage is not obvious when the input scale is small, and the input is specific. However, the attack graph scale is growing with the development of network technology, and the attack graph situation becomes variable. Traditional algorithms are no longer applicable. Heuristic algorithms have natural advantages in dealing with such complex problems. No matter what the input attack graph looks like, heuristic algorithms can always find the optimal result within a fixed time to complete the iteration.

This paper analyzes the attack graph from three levels: entire attack graph, single attack path, single state node and extracts as much information as possible that may be contained in the attack graph. The results generated by algorithms are related to each other, and they can be used to complete a comprehensive network risk assessment.

**Acknowledgement.** This work is supported by the National Key R&D Program of China (Funding No. 2020YFB1805503). The 2020 Industrial Internet Innovation and Development Project from Ministry of Industry and Information Technology of China, the Fundamental Research Fund for the Central Universities (30918012204, 30920041112), Jiangsu Province Modern Education Technology Research Project (84365); National Vocational Education Teacher Enterprise Practice Base "Integration of Industry and Education" Special Project (Study on Evaluation Standard of Artificial Intelligence Vocational Skilled Level); Scientific research project of Nanjing Vocational University of Industry Technology (2020SKYJ03).

## References

1. Ammann, P., Wijesekera, D., Kaushik, S.: Scalable, graph-based network vulnerability analysis. In: Proceedings of the ACM Conference on Computer and Communications Security, pp. 217–224 (2002). https://doi.org/10.1145/586110.586140
2. Dai, F., Hu, Y., Zheng, K., Wu, B.: Exploring risk flow attack graph for security risk assessment. IET Inf. Secur. **9**(6), 344–353 (2015)
3. Ghoshal, S., Sundar, S.: Two approaches for the min-degree constrained minimum spanning tree problem. Applied Soft Computing **111**, 107715 (2021). https://doi.org/10.1016/j.asoc.2021.107715
4. Hasteer, N., Bansal, A., Murthy, B.K.: Assessment of cloud application development attributes through interpretive structural modeling. Int. J. Syst. Assur. Eng. Manag. **8**, 1069–1078 (2017). https://doi.org/10.1007/s13198-017-0571-2
5. Ibrahim, A., Bozhinoski, S., Pretschner, A.: Attack graph generation for microservice architecture. In: Proceedings of the ACM Symposium on Applied Computing, vol. Part F147772, pp. 1235–1242 (2019). https://doi.org/10.1145/3297280.3297401
6. Ingols, K., Lippmann, R., Piwowarski, K.: Practical attack graph generation for network defense. In: Proceedings - Annual Computer Security Applications Conference, ACSAC, pp. 121–130 (2006). https://doi.org/10.1109/ACSAC.2006.39
7. Kar, A.K.: Bio inspired computing - a review of algorithms and scope of applications. Expert Syst. Appl. **59**, 20–32 (2016). https://doi.org/10.1016/j.eswa.2016.04.018
8. Liu, C., Singhal, A., Wijesekera, D.: Mapping evidence graphs to attack graphs. In: WIFS 2012 - Proceedings of the 2012 IEEE International Workshop on Information

Forensics and Security, pp. 121–126 (2012). https://doi.org/10.1109/WIFS.2012.6412636
9. Mann, M., Sangwan, O.P., Tomar, P., Singh, S.: Automatic goal-oriented test data generation using a genetic algorithm and simulated annealing. In: Proceedings of the 2016 6th International Conference - Cloud System and Big Data Engineering, Confluence 2016, pp. 83–87 (2016). https://doi.org/10.1109/CONFLUENCE.2016.7508052
10. Musa, T., et al.: Analysis of complex networks for security issues using attack graph. In: 2019 International Conference on Computer Communication and Informatics, ICCCI 2019 (2019). https://doi.org/10.1109/ICCCI.2019.8822179
11. Ou, X., Govindavajhala, S., Appel, A.W.: MulVAL: a logic-based network security analyzer. In: 14th USENIX Security Symposium, pp. 113–128 (2005)
12. Blank, R.M., Gallagher, P.D.: NIST Special Publication 800-30 Revision 1 - Guide for Conducting Risk Assessments, p. 95. NIST Special Publication, September 2012
13. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.: Automated generation and analysis of attack graphs. In: IEEE Symposium on Security and Privacy, p. 273, May 2002
14. Sing, A.N.U., Raphs, A.T.G.: A predictive framework for cyber security analytics using attack graphs. Int. J. Comput. Netw. Commun. **7**(1), 1–17 (2015)
15. Stergiopoulos, G., Dedousis, P., Gritzalis, D.: Automatic analysis of attack graphs for risk mitigation and prioritization on large-scale and complex networks in Industry 4.0. Int. J. Inf. Secur. **21**, 37–59 (2021). https://doi.org/10.1007/s10207-020-00533-4
16. Swiler, L.P., Phillips, C.: A graph-based system for network-vulnerability analysis. In: The 1998 Workshop (1998)