Computer Systems Science & Engineering
DOI:10.32604/csse.2021.014680
Article

A Network Security Risk Assessment Method Based on a B_NAG Model

Hui Wang[1], Chuanhan Zhu[1], Zihao Shen[1,*], Dengwei Lin[2], Kun Liu[1] and MengYao Zhao[3]

[1]School of Computer Science & Technology, Henan Polytechnic University, Jiaozuo, 454000, China
[2]Office of Educational Administration, Jiaozuo University, Jiaozuo, 454000, China
[3]Department of Computer Science, University College London, London, United Kingdom
[*]Corresponding Author: Zihao Shen. Email: szh@hpu.edu.cn
Received: 08 October 2020; Accepted: 09 January 2021

Abstract: Computer networks face a variety of cyberattacks. Most network attacks are contagious and destructive, and these types of attacks can be harmful to society and computer network security. Security evaluation is an effective method to solve network security problems. For accurate assessment of the vulnerabilities of computer networks, this paper proposes a network security risk assessment method based on a Bayesian network attack graph (B_NAG) model. First, a new resource attack graph (RAG) and the algorithm E-Loop, which is applied to eliminate loops in the B_NAG, are proposed. Second, to distinguish the confusing relationships between nodes of the attack graph in the conversion process, a related algorithm is proposed to generate the B_NAG model. Finally, to analyze the reachability of paths in B_NAG, the measuring indexs such as node attack complexity and node state transition are defined, and an iterative algorithm for obtaining the probability of reaching the target node is presented. On this basis, the posterior probability of related nodes can be calculated. A simulation environment is set up to evaluate the effectiveness of the B_NAG model. The experimental results indicate that the B_NAG model is realistic and effective in evaluating vulnerabilities of computer networks and can accurately highlight the degree of vulnerability in a chaotic relationship.

Keywords: Network attack graph; Bayesian network; state transition; reachability; risk assessment

1 Introduction

Computer networks play an indispensable role in people's productivity and daily life. However, these networks face a variety of cyberattacks, most of which are highly contagious and destructive. These attacks threaten the network security of devices, affecting the popularization of networks and even severely damaging information security [1–3].

According to the "Development Status of China Internet Sites and Security Report in 2018" [4], the National Computer Network Emergency Response Technical Team of China (CNCERT) discovered that over 1.254 million Internet of Things smart devices was attacked successfully and therefore had a great threat to the security of networks. Moreover, in 2018, CNCERT discovered over 2.05 million cyberattacks continuing a trend of high growth over the previous six years. A survey of these attacks announced that the number of applications had quickly increased and was nearly three times higher than the percentage in 2016.

In recent years, researchers have introduced methods based on Bayesian probabilities in evaluating vulnerabilities of attack graphs [5–7]. Bayesian networks are capable of representing nondeterministic relationships and can be used to quantify the correspondences within attack graphs. Therefore, methods of effectively combining a Bayesian network with an attack graph for network vulnerability assessment have become an important focus of research.

2 Related Research

Recently, lots of scholars analyzed vulnerabilities of networks by using attack graph. Because of the asymmetric information between attackers and defenders, the detection of Zero Day attacks is still challenging. Revealing Zero Day attacks based on attack paths is a better strategy than targeting them individually.

Sun et al. [8] implemented the system ZePro to identify Zero Day attack paths by adopting the probabilistic approach. With evidence of intrusion as input, the Bayesian network used in this system can calculate the infection probabilities concerning object instances.

The dynamic defense framework was presented to select best countermeasures against diverse attack damage costs [9]. To calculate these costs, a new defense-centric model was designed on the basis of service dependency graphs. The current approaches suffer from some limitations. For example, only static countermeasure effectivity and static countermeasure deployment costs are considered, but the negative impacts of the possible countermeasures on service quality are neglected [10]. These above-mentioned restrictions may lead an industrial control system (ICS) to choose improper countermeasures and deployment locations. And then they can degrade the network performance and frustrate legitimate users.

The construction and analysis on inference rules of attack graph was presented by Garg et al. [11]. They developed a methodology for prioritizing individual vulnerabilities and attack paths using a PageRank model. The results were verified by using a Markov model, and showed that the methodology outperformed lots of current technologies [12] about risk analysis. However, the relevant experiment was lack of specific indicators, and the results were not convincing.

As Zhang et al. [13] said, dynamic risk analysis is an important component of protecting network security. However, risk assessment methods used in network systems are not very appropriate for ICSs due to their unique characteristics. That paper proposed a multilevel network model including attack functions and incidents based on Bayesian. On this basis, it proposed a new risk incident prediction method, and designed a dynamic security risk assessment method which can assess the risk caused by unknown attacks [14]. Moreover, a quantification method was presented to further calibrate the accuracy of assessment. Finally, to test and verify the method, the simplified control system was simulated in MATLAB.

On the basis of previous researches, the paper presents a Bayesian network attack graph (B_NAG) model and an algorithm to assess network vulnerabilities. In this paper, probability theory is introduced into the resource attack graph (RAG) model and converted into the corresponding B_NAG model. The reachability probability of nodes can be calculated, and the final reachability probability of attack paths can be calculated. Finally, the related posterior probability can be calculated, and enable network security administrators to assess network security more accurately and effectively.

3 The RAG Model

Attack graph is a method to analyze all sequences of vulnerabilities exploited by attackers. Attacks can be occurred against all available node status and vulnerability, and all sequences can be constructed into a directed graph. The purpose of the RAG model is to characterize an attack sequence launched against the attacker's intentions according to Bayesian probability calculations to help network administrators properly understand the security status of their networks. The RAG model is constructed as described below.

Definition 1 The graph $RAG = (S, S_0, A, E, \Gamma, L, O)$ is a directed graph, where the relevant notations are defined as follows:

- $S = \{s_i | i = 1, \ldots, N\}$ denotes a resource state nodes set.
- $S_0 \in S$ denotes the initial resource state nodes which are occupied by the attacker.
- $A = \{a_i | i = 1, \ldots, N\}$ represents a set of attack behavior nodes.
- $E = \{E_1 \cup E_2\}$ denotes a set of directed edges connecting all related nodes. $E_1 \subseteq S \times A$ means that the attack will be occurred only if one attacker occupies some resources; $E_2 \subseteq A \times S$ means that the attack can make this attacker occupy some resources. Its parent nodes set m is denoted as $Pre(m)$, and the child nodes set m is denoted as $Nex(m)$.
- Γ is the node state discriminant function. $\Gamma(x)$ denotes the current status of the node x and $\Gamma(x) \in \{1, 0\}$, where $\Gamma(s_i)$ means the current status of s_i. $\Gamma(s_i) = 1$ indicates that the attacker has occupied the resource s_i. Conversely, 0 indicates that the attacker has not occupied the resource.
- L is the logical relationships set between nodes, and $L = \{and, or, ble\}$. There is an *and* relationship between Pre(a_i) only if all preconditions for the corresponding attack node a_i are met. And a successful attack will enable $\Gamma(s_i) = 1$ only if the attacker has occupied the resource s_i. There is an *or* relationship between attack nodes when resource state nodes are child nodes. Finally, *ble* denotes a kind of chaotic logical relationship which exists between parent nodes.
- $O = \{o_i | i = 1, 2, 3, \ldots, N\}$ represents the set of resource state nodes associated with those successful attacks which have been detected. For $\forall o_i \in S$, o_i represents the resource state nodes associated with the successful attacks are detected by IDS.

Definition 2 Attack path: In the RAG, if there exists a status sequence $s_0, a_0, s_1, a_1, \ldots, a_{n-1}, s_n$, where s_0 represents the initial node of resource state and s_n represents the target node. So the $Path_k = < s_0 \to a_0 \to s_1 \to a_1 \to \ldots \to a_{n-1} \to s_n >$ can be defined, where $\forall s_i \in S, \forall a_j \in A$ $(0 \le i \le n, 0 \le j \le n-1)$; The $Path_k$ denotes the attack path k_{th}.

Definition 3 Attack behavior: One attack behavior can be denoted by a four-tuple of the form $(Src\_id, Dst\_id, Att\_code, Res)$, where *Src_id* denotes the host *id* launching an attack, *Dst_id* denotes the host *id* which has been attacked, *Att_code* is the number which can identify attack behaviors, and *Res* is the result of this attack.

Definition 4 State transition: One state transition is denoted by a three-tuple of the form (sid, vid, r), where *sid* is the number which can identify state transitions, *vid* is the number which identify vulnerabilities used by attackers, and *r* is the resulting state transition which is caused by one attack using vulnerabilities.

4 The Algorithm E-Loop

4.1 The Method of Metrics

To remove loops in an attack graph, an attack difficulty metric is introduced. In the Common Vulnerability Scoring System (CVSS), three basic indexes are used to characterize vulnerabilities: the access vector index, the access complexity index, and the authentication index, which are denoted by Acc_com, Acc_vec and Auth respectively. The values of these indexes associated with different levels of severity of a vulnerability are shown in Tab. 1.

Table 1: Index levels

Index	Acc_com	Acc_vec	Auth
Low	0.35	0.359	0.45
Mid	0.61	0.646	0.56
High	0.71	1.0	0.704

Based on these indexes, the availability score of a vulnerability used in the CVSS is defined as

$$Exp = 20 \times Acc\_vec \times Acc\_com \times Auth \quad (0 \le Exp \le 10) \tag{1}$$

An attack becomes more difficult to perform successfully as the value of *Exp* gets smaller. Thus, the attack difficulty is inversely proportional to the availability of a vulnerability. Accordingly, an attack difficulty metric *Aga_Dif* may be defined based on the above three indexes as shown in Eq. (2). The larger the value of *Aga_Dif* is for a particular node, the more difficult the node is to attack.

$$Aga\_Dif = \frac{1}{2Acc\_vec \ \times Acc\_com \ \times \ Auth} \ (Aga\_Dif \ge 1) \tag{2}$$

4.2 The Algorithm E-Loop

In the generation of the RAG, a loop may arise that leads to repeated traversals over a given node. It has a great influence on Bayesian probability calculation in network security assessment. In order to overcome the problem, the algorithm E-Loop is proposed to eliminate loops in the RAG. The specific steps are as follows:

Algorithm 1: $E - Loop(RAG)$

```
Input: $RAG$
Output: Acyclic RAG ( $Ac\_RAG$ )
Step 1 Start nodes are added to the queue of the root node.
Step 2 All loops found are stored in the initialization stack: $Init()$.
Step 3 Carry out a depth-first traversal from the begining node and then traverse every node:
       $root = GetRoot()$.
Step 4 Push every visited node into the stack: $PushStack(root)$. Until all nodes are traversed or the
       currently traversed node has been traversed. Finally, a loop has been stored in the stack.
Step 5 In the loop, $Aga\_Dif_i$ of every node must be calculated, and the node attacked most difficultly can
       be found: $S_m = Max(Aga\_Dif_i)$.
Step 6 Delete $S_m$ to eliminate the loop: $Delete(S_m)$.
Step 7 Loop through Step3- Step6 until there is no loop in the RAG.
Step 8 Output $Ac\_RAG$.
```

Fig. 1 shows an RAG built as described above. There are two loops, $Path_1 = < s_2 \rightarrow a_3 \rightarrow s_5 \rightarrow a_5 \rightarrow s_2 >$ and $Path_2 = < a_9 \rightarrow s_{11} \rightarrow a_{12} \rightarrow s_{12} \rightarrow a_9 >$. For $Path_1$, the node a_5 will be eliminated by the algorithm $E - Loop$ to remove the loop; For $Path_2$, node a_9 can never be reached because of $Aga\_Dif(a_9) \rightarrow \infty$, so this loop can be removed by eliminating this node and all subsequent nodes. Fig. 2 shows the acyclic RAG (Ac_RAG) obtained after the loops are eliminated by the E-Loop algorithm.

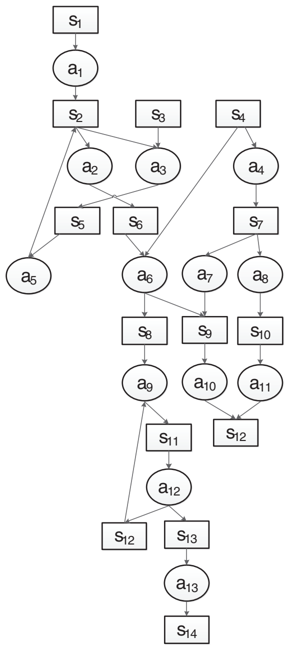


Figure 1: RAG

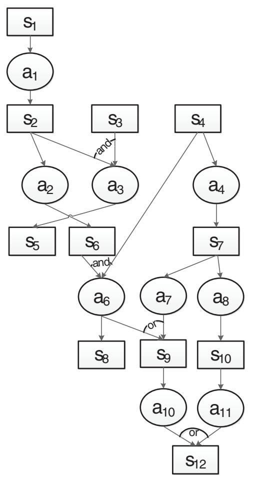


Figure 2: Acyclic RAG

5 Probability Calculation in the B_NAG Model

In the B_NAG, the probability of each node is only constrained by its parent nodes, and the node remains conditionally independent of the others. In the RAG, the transition of node state is only correlated to whether the relevant resource has been occupied or not. A child node can occur a state transition only if its parent nodes are occupied. Thus, the state transition needs be associated with conditional independence in the B_NAG.

Tab. 2 presents the corresponding relationship between an Ac_RAG and a B_NAG. Although these graphs have corresponding structures, differences exist in their certain nodes. The detailed implementation described below is based on a B_NAG.

Table 2: Corresponding relationship

Acyclic resource attack graph (Ac_RAG)	B_NAG
Directed acyclic graph	Directed acyclic graph
Causality between the vulnerability and the state transition	Causality network
Network state nodes	Network nodes
Attack behavior processes occupying node resources	Directed edges between nodes
Probability of state transition	Conditional probability between nodes
Negligible influences of node state transitions and other nonparent nodes	Conditional independence of nodes

5.1 Implementation of the B_NAG Model

Definition 5 The resulting resource state node and the conditional resource state node: The resource state node where the attack has been occurred successfully is called the resulting resource state node; When the attack condition is satisfied, the required resource state node is called the conditional resource state node.

Definition 6 $W = \{w_{ij} | i,j = 1, 2, \ldots, N\}$, the set of weights between the resource state nodes: W is represented in the form of two-tuples $(depcoef, \cos t))$, where *depcoef* denotes the correlation coefficient between resource state nodes and $\cos t$ denotes the cost required to attack another resource state node from the current node. w_{ij} is the weight value between the node s_i and the node s_j.

As illustrated in the example shown in Fig. 2, an RAG consists of four structures: a series structure, a parallel *or* structure, a parallel *and* structure, and a mixed structure. While converting such a graph into a B_NAG, each of these structures can be transformed as follows:

(1) Series structure: By deleting the attack behavior node a_1, the attack behavior can be represented by the directed edge from s_1 to s_2:

$$(s_1 \rightarrow a_1 \rightarrow s_2) \Rightarrow (s_1 \rightarrow s_2)$$

(2) Parallel *or* structure: The nodes a_{10} and a_{11} exist an *or* relationship, meaning that the attack is able to occur when the resource state condition corresponding to either of the parent nodes s_9 or s_{10} can be satisfied. The related attack behavior nodes are removed. And the resulting resource state node and the conditional resource state node can be linked by one directed edge. In the B_NAG, the resource state nodes have an *or* relationship:

$$(s_9 \rightarrow a_{10}, s_{10} \rightarrow a_{11}, a_{10} \vee a_{11} \rightarrow s_{12}) \Rightarrow (s_9 \vee s_{10} \rightarrow s_{12})$$

(3) Parallel *and* structure: The parent nodes s_2 and s_3 of a_3 have an *and* relationship, meaning that the attack behavior may occur only if all resource state conditions are satisfied. After the attack node is removed, and the resulting resource state node and the conditional resource state nodes can be linked by one directed edge, which represents the attack behavior. In the transformed B_NAG, the resource state nodes still have an *and* relationship:

$$(s_2 \wedge s_3 \rightarrow a_3 \rightarrow s_5) \Rightarrow (s_2 \wedge s_3 \rightarrow s_5)$$

(4) Mixed structure: The parent nodes s_6 and s_4 of the node a_6 have an *and* relationship, and the two nodes a_6 and a_7 that can get to the resulting resource state node have an *or* relationship. If the node a_6 is directly removed, the structure of the RAG will become confusing, causing inconvenience in the

conditional probability calculation. In order to solve the problem, this paper defines a temporary mixed node*blend*; namely, the node a_6 is denoted as the node *blend*:

$$(s_6 \wedge s_4 \rightarrow a_6, a_6 \vee a_7 \rightarrow s_9) \Rightarrow (s_6 \wedge s_4 \rightarrow blend, blend \vee s_7 \rightarrow s_9)$$

After this conversion process, each edge in the converted B_NAG represents an attack behavior and has a weight that describes the correlation between the two resource state nodes connected by that edge. It can be observed from the converted B_NAG shown in Fig. 3 that *blend* is a mixed resource state node representing the combination of s_4 and s_6, so there must be directed edges from s_4 and s_6 to *blend*, namely, $P(blend|s_4, s_6) = 1$. The relationships between the nodes do not change upon conversion into a B_NAG, and only the resource state nodes will be included. All attack behaviors are represented by the directed edges of the B_NAG, and the only possible relationships are *and* and *or*.

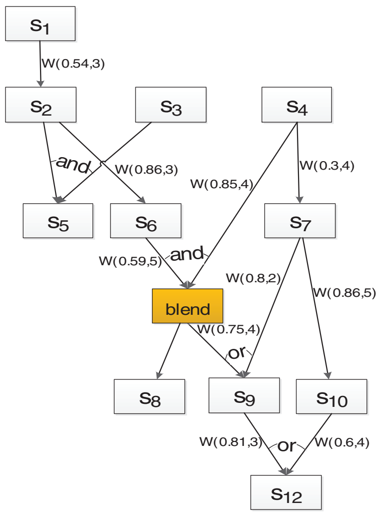


Figure 3: B_NAG

To clarify the process, the conversion algorithm is proposed as follows.

Algorithm 2: Attack graph conversion algorithm, $Alg - AGTrans(Ac\_RAG)$

```
Input: Ac_RAG
Output: B_NAG
1. For each $s_i \in S$ AND $a_i \in A$
2. IF DPre($s_i$) $\neq \varnothing$ AND IF DPre($a_i$) $\neq \varnothing$
3. Else If Num(DPre($a_i$)) = 1 AND Num(DPre(DNex($a_i$))) = 1;
4. $e_{ij} = < s_i, DNex(a_i) >$;// the edge $e_{ij}$ links two nodes
5. $e_{ij} \leftarrow W(i, j)$;
6. Delete($a_i$); //delete the attack node
7. Else If Num(DPre($a_i$))> 1
8. $e_{ij} = < DPre(a_i), DNex(a_i) >$; // there are many incoming edges here
9. $e_{ij} \leftarrow W(i, j)$;
10. Delete($a_i$);
11. The relationship of $\forall e_{ij}$ is AND;
12. Else If Num(DPre($s_i$))> 1
13. $e_{ij} = < DPre(DPre(s_i)), s_i >$;
14. $e_{ij} \leftarrow W(i, j)$;
15. Delete(DPre($s_i$));
16. The relationship of $\forall e_{ij}$ is OR.
17. Else If DPre($a_i$)> 1 AND Num(DPre(Nex($a_i$)))> 1
18. $a_i \in$ Ble; $a_i$= blent; // introduction of a mixed node blend;
19. $e_{ij} = < DPre(blent), blent >$;
20. $e_{ij} \leftarrow W(i, j)$;
21. The relationship of $\forall e_{ij}$ is AND;
22. $e_{ij} = < blent, DNex(blent) >$;
23. $e_{ij} \leftarrow W(i, j)$;
24. The relationship of $\forall e_{ij}$ is OR.
25. End If;
26. Go to For;
27. Return B_NAG;
```

5.2 Calculation of the Probability of Reaching a Node Based on the B_NAG

The direct parent nodes of node S are denoted here by $DPre(S)$, and the attack probability $P_a(S)$ of the target node can be calculated:

$$\begin{aligned} P_a(S) &= P(S|\Pr e(S))P(\Pr e(S)) \\ &= P(S|D\Pr e(S))P(D\Pr e(S)) \end{aligned} \tag{3}$$

The state transition index $P_m(\cos t_i)$ can be denoted as the probability of the conversion from S_{i-1} to S_i. Because of the correlation between one resource and its parent nodes, the weights W must be considered when the state transition indexes of the parent nodes are calculated. If a sufficiently high cost is paid, the attack will be guaranteed to be accomplished; namely, if $\cos t \to \infty$, then $P_m(\cos t) = 1$. If no cost is afforded, any target can't be attacked successfully; that is, when $\cos t = 0$, $P_m(\cos t) = 0$. If an attack on a node fails, the state of this node remains unchanged. For the state transition index $P_m(\cos t_i)$, its value follows a certain distribution. Thus, $P_m(\cos t_i)$ is calculated as follows:

$$P_m(\cos t_i) = P(\cos t_i < \mathrm{Cos}\ t) = 1 - e^{-depcoef \times \cos t_i} \tag{4}$$

Here, $\cos t$ refers to the cost required to perform an attack, that is, the knowledge, experience, and resources needed to complete the attack. Cos t means the average cost required to complete the final attacks; it's a default value and relies on the resources, knowledge, attack tools and time. *depcoef* is the correlation degree:

$$depcoef = \frac{1}{Aga\_Dif}\ (0 < depcoef < 1) \tag{5}$$

Accordingly, the state transition index $P_m(\cos t_i)$ is calculated as

$$P_m(\cos t_i) = 1 - e^{-\frac{\cos t_i}{Aga\_Dif}} \tag{6}$$

It can be concluded from Eq. (6) that resource state nodes in the B_NAG interact with each other, so the probability of reaching a given node cannot be analyzed only by traditional inference in the vulnerability analysis; instead, these state transitions must also be considered deeply. To solute this problem, the index of state transition is used to consider the probability of node state transitions when assessing vulnerabilities of the network. P_{end} denotes the probability of reaching a target node:

$$\begin{aligned} P_{end}(s_i) &= P_m(\cos t_i) \times P_a(s_i) \\ &= (1 - e^{-\frac{\cos t_i}{Aga\_Dif}}) \times\ P(s_i|D\Pr e(s_i))P(D\Pr e(s_i)) \end{aligned} \tag{7}$$

Here, $P_m(\cos t_i)$ is the state transition index of the target node s_i; $\cos t$ and *depcoef* are the attack cost and the correlation degree between node s_i and its parent nodes, respectively; and $P_a(s_i)$ denotes the Bayesian probability of s_i is attacked. Eq. (7) gives the probability of reaching a single target node; the probability of reaching a whole path will be obtained by iterating Eq. (7) accordingly. The related iterative algorithm is provided below.

In Algorithm 3, all nodes are traversed firstly. The parent nodes $Pre(s_i)$ are pushed onto their respective stack q in accordance with the number of direct parent nodes $DPre(s_i)$ of s_i, and it must make sure that the start node in the path of $DPre(s_i)$ is finally pushed onto q. Then, the nodes are each removed in proper order based on the "last in first out" principle, and their reachability probability can be calculated to finally determine the whole path's probability being reached.

Algorithm 3: Iterative algorithm for calculating the probability of reaching a whole path in a graph, $IterAlg - ReaPro(B\_NAG, W)$

```
Input: converted $B\_NAG$ and weights $W = (depcoef, \cos t)$ between nodes
Output: final probability of an attack reaching the whole path $P_{end}(s_i)$
1. For each $s_i \in S$
2. Number of Count = $DPre(s_i)$;
3. InitStack(&q);
4. for each $DPre(s_i) \in S$ And $DPre(s_i) \neq \varnothing$
5. root = $DPre(s_i)$; // number of direct parent nodes of the target node
6. Push Stack(root) → q;
7. End for
8. End for
9. For $q \neq \varnothing$
10. $s_i$ = PopStack(root);
11. $Pa = Pa(s_i)$;
12. $P_m = P_m(cost_{dpre(si)})$;
13. $P_{end} = P_a \times P_m$;
14. End for;
15. return $P_{end}$;
```

In the example shown in Fig. 3, if s_{12} is attacked, it is obvious that the attack target can be accomplished by the below three paths:

$$Path_1 = < s_1 \rightarrow s_2 \rightarrow s_6 \rightarrow \text{blend}(s_6 \wedge s_4) \rightarrow s_9 \rightarrow s_{12} >$$

$$Path_2 = < s_4 \rightarrow s_7 \rightarrow s_9 \rightarrow s_{12} >$$

$$Path_3 = < s_4 \rightarrow s_7 \rightarrow s_{10} \rightarrow s_{12} >$$

If the weights W, the values of *depcoef* and the attack costs are shown in Fig. 3, then the prior probabilities of s_4 and s_1 are 0.3 and 0.2, respectively. For instance, the related steps for $Path_1$ are described below:

$$P(s_2) = P_m(\cos t_1) \times P_a(s_2) = P_m(\cos t_1) \times P(\Gamma(s_2) = 1|\Gamma(s_1) = 1) \times P(s_1) = 0.0866$$

$$P(s_6) = P_m(\cos t_2) \times P_a(s_6) = P_m(\cos t_2) \times P(\Gamma(s_6) = 1|\Gamma(s_2) = 1) \times P(s_2) = 0.0684$$

$$P(blend) = P_m(\cos t_6) \times P(\Gamma(blend) = 1|\Gamma(s_6) = 1, \Gamma(s_4) = 1) \times P(s_6) \times P(s_4) = 0.0382$$

In a similar way, the following is obtained from Eq. (7): the reachability probabilities of s_9 and s_{12} by following $Path_1$ are $P(s_9) = 0.0272$ and $P_{end}(s_{12}) = 0.0201$, respectively; the reachability probability of s_{12} by following $Path_2$ is $P_{end}(s_{12}) = 0.0648$; and the reachability probability of s_{12} by following $Path_3$ is $P_{end}(s_{12}) = 0.0573$. If the administrator knows that a_1 has been targeted, namely, $P(s_1) = 1$, then the reachability probability of s_{12} by following $Path_1$ can be recalculated as $P_{end}(s_{12}) = 0.0631$. This indicates that when the resource state condition corresponding to s_1 is met, s_{12} is more possible to be attacked, which is the same as expected.

5.3 Posterior Probability Calculation Based on the B_NAG

In a B_NAG, it is not possible to monitor changes in the network security conditions in real time when the probabilities of resource state nodes attacked by attackers are calculated. Based on the detected the precondition and the available information of security incidents, the posterior probabilities should be calculated, and these related node probabilities can then be updated to achieve real-time monitoring. The equation for calculating a posterior probability is as follows:

$$P_o(S_i|O) = \frac{P(O|S_i) \times P(S_i)}{P(O)} \tag{8}$$

Suppose that O_1 in Fig. 3 can be detected, and the probability of s_{12} is 1. Then, the posterior probability of s_9 is calculated as follows by Eq. (8).

$$\begin{aligned}
P_o(s_9|s_{12}) &= P(s_{12}|s_9) \times P(s_9)/P(s_{12}) \\
&= (P(s_{12}, s_{10}|s_9) + P(s_{12}, {}^\wedge s_{10}|s_9)) \times P(s_9)/P(s_{12}) \\
&= (P(s_{12}|s_{10}, s_9) \times P(s_{10}|s_9) + P(s_{12}|{}^\wedge s_{10}, s_9) \times P({}^\wedge s_{10}|s_9))/P(s_{12}) \\
&= (P(s_{12}|s_{10}, s_9) \times P(s_{10}) + P(s_{12}|{}^\wedge s_{10}, s_9) \times P({}^\wedge s_{10}))/P(s_{12}) \\
&= 0.082
\end{aligned}$$

In this case, the probability of reaching node s_9 changes from 0.0272 to 0.082. When certain attacks occur, the corresponding posterior probabilities in the B_NAG can effectively discover the potential risk. The real-time calculation of the risk values of nodes in the B_NAG is of great significance for the assessment of vulnerabilities.

6 Experimental Analysis

6.1 Experimental Network Environment

To verify that the given method is feasible and effective, the experimental environment shown in Fig. 4 was created. The experimental network includes five hosts: the attacking machine, a web server, a file server, an e-mail server, and a database server. For ease of description, these hosts are represented by the letters A, W, F, E and D, respectively. W opens the telnet service, F opens the File Transfer Protocol (FTP) service, E opens the FTP and Hypertext Transfer Protocol (HTTP) services, and D opens the Oracle service. The final aim of attacker A is to obtain root permissions for host D, but the firewall allows the foreign host A access to only the telnet service of host D and denies other external access. Similarly, host E is allowed access to only the Oracle service of host D, while the other three hosts can openly gain access to each other's services. Host W can directly access host E; when it obtains access to the two services provided by host E, it can, in turn, gain direct access to the Oracle service of host D.

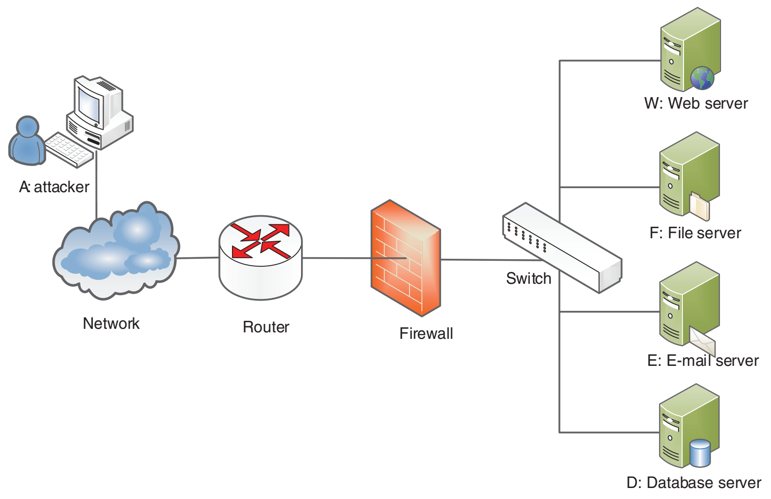


Figure 4: Topological graph of the experimental network

Information about the internal host is shown in Tab. 3.

Table 3: Information about the internal host

Host Name	Service(s)	Vulnerability ID(s)
W	{Telnet}	{12815}
F	{FTP}	{9904,13454}
E	{FTP,HTTP}	{7974,8952}
D	{Oracle}	{14312}

6.2 Experimental Results and Analysis

After loops have been removed as previously described during the generation of the RAG in accordance with the attack graph model and the topological graph of the experimental network, the corresponding descriptions of the attack behavior nodes are as shown in Tab. 4. These attacks are related to the services provided by the hosts and their vulnerabilities.

Table 4: Attack behavior information of the experimental attack graph

vid	Src_id	Att_code	Dst_id	Res
12815	A	tel-rsh(A,W)	W	Trust(W,A)
9904	W	ftp-rhost$_1$(W,F)	F	Trust$_1$(F,W)
13454	W	ftp-rhost$_2$(W,F)	F	Trust$_2$(F,W)
7974	F,W	ftp-rhost(F,E), ftp-rhost(W,E)	E	Trust$_1$(E,F), Trust$_1$(E,W)
8952	F,W	http-rsh(F,E), http-rsh(W,E)	E	Trust$_2$(E,F), Trust$_2$(E,W)
14312	F	oracle(E,D), oracle(W,D)	D	Trust(D,E), Trust(D,W)

After the application of the conversion algorithm based on the topological graph of the experimental network to replace the attack behavior nodes mentioned in Tab. 4 with corresponding edges, the converted B_NAG is as shown in Fig. 5.

As shown in Fig. 5, each host node must win the trust of another host through a service provided by that other host, corresponding to a parallel "and" structure in the graph. When one host opens two services, the trust of that host can be obtained by gaining access to either one of its services, so the relationship between the possible attacks against that host is "or". A node with a mixed relationship can directly access the service provided by another host by crossing over the host it is attacking once it gains access to both services of the target host. A blend node is introduced to address the corresponding mixed relationship in this graph.

There are 5 paths in Fig. 5 through which the target host D can be reached. The attack path information and the probabilities of reaching each whole path are shown in Tab. 5. P_1 denotes the probability of reaching the whole path as calculated by considering the state transition index as proposed in this paper, while P_2 is the probability of reaching the whole path calculated without considering the state transition index.

Based on Tab. 5, the probability of reaching each host node is plotted in Fig. 6.

As shown in Fig. 7, the hosts attacked on Path1 and Path2 (and on Path3 and Path4) are the same; the only difference lies in the service of host E that is accessed. Path1 accesses the FTP service of host E, while Path2 accesses the HTTP service of host E. The final probabilities of reaching the whole path for Path1 and

Path2 are 0.03247 and 0.05784, respectively, as calculated using the proposed algorithm based on the state transition index. Obvious differences can be seen between the two paths in terms of the probability of the attack successfully proceeding from host F to host E, as shown in Fig. 7a. By contrast, when the state transition index is not considered, the final probabilities for Path1 and Path2 are 0.4768 and 0.4789, respectively, and there is no meaningful difference in the probability of proceeding from host F to host E, as shown in Fig. 7b. With the proposed algorithm, although the reference value of the probability for each node decreases, the differences in probability associated with attacking different nodes are fully apparent. Therefore, this approach is effective in enabling network security administrators to perform useful analyses.

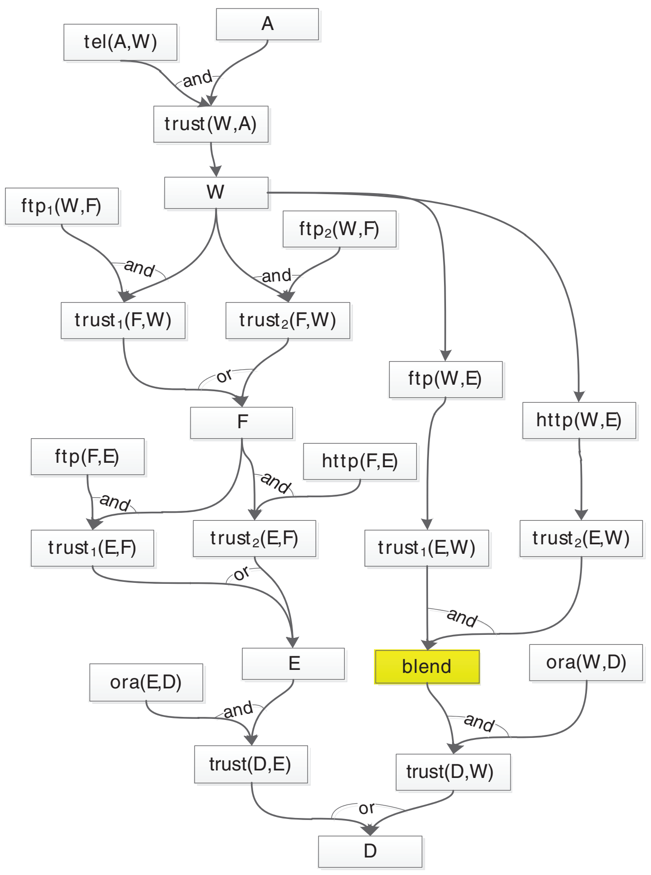
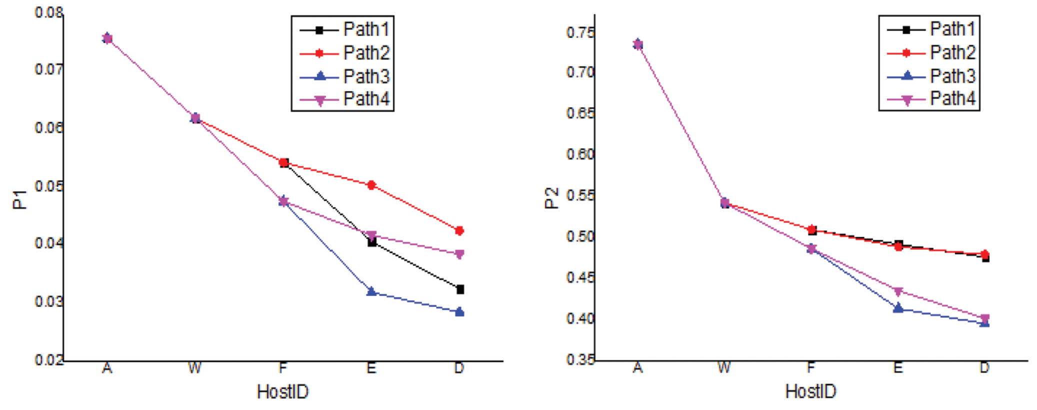


Figure 5: Example of a Bayesian network attack graph

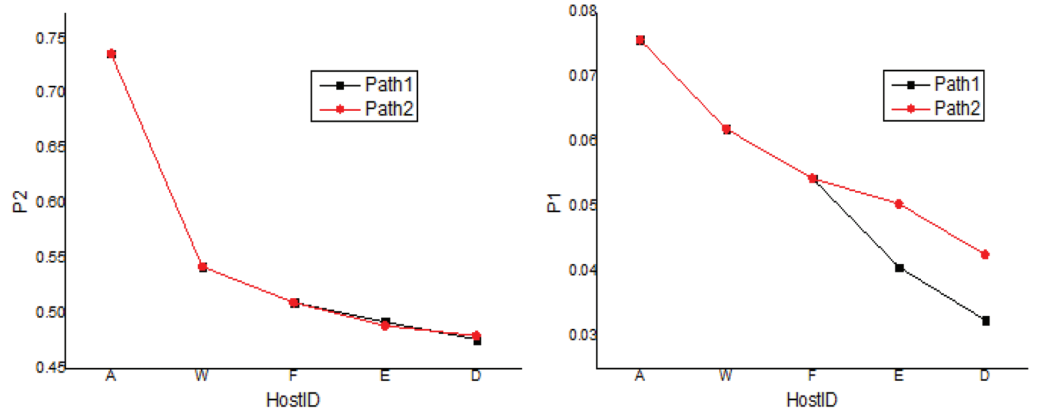
Table 5: Attack path information for the example graph

Path	Src_id	Dst_id And Service	Tar_id	Path_Len	P_1	P_2
$Path_1$	A	(W:12815),(F:9940),(E:7974),(D:14312)	D	9	0.03247	0.4768
$Path_2$	A	(W:12815),(F:9940),(E:8952),(D:14312)	D	9	0.04252	0.4789
$Path_3$	A	(W:12815),(F:13454),(E:7974),(D:14312)	D	9	0.02846	0.3954
$Path_4$	A	(W:12815),(F:13454),(E:8952),(D:14312)	D	9	0.03845	0.4018
$Path_5$	A	(W:12815),(E:7974,8952),(D:14312)	D	8	0.04836	0.3912



(a) (b)

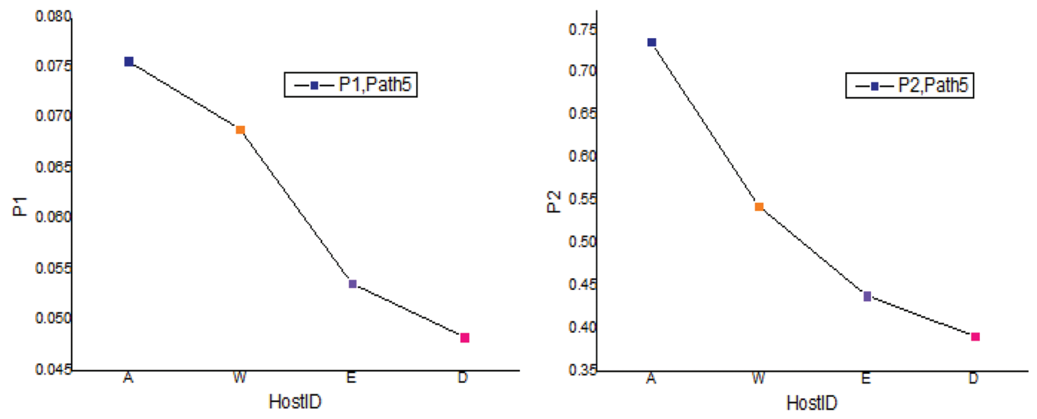
Figure 6: Path probabilities under P_1 and P_2 (a) P_1 (b) P_2



(a) (b)

Figure 7: Probabilities of $Path_{1,2}$ under P_1 and P_2 (a) P_1 (b) P_2

As shown in Fig. 8, the traditional computational method for Path5, which includes a mixed relationship, is to calculate all "and" nodes and "or" nodes individually. This not only requires a large number of calculations but also ignores the correlations between nodes.



(a) (b)

Figure 8: Probability of $Path_5$ under P_1 and P_2 (a) P_1 (b) P_2

The mixed node approach introduced herein provides better calculation results than the traditional method, and it does so with fewer calculations. For the mixed relationship identified when host W attempts to gain access to host E, the probability calculated by considering the state transition index effectively reflects the degree of hazard of the associated vulnerability, making this type of vulnerability more likely to be noticed by the network security administrator.

7 Conclusion

Improving the accuracy of network vulnerability assessments is an important topic in the field of network security. This paper presents a B_NAG model and an associated vulnerability algorithm as well as the algorithm E-Loop to eliminate loops in an attack graph. To effectively capture mixed relationships between nodes during the process of converting a RAG into a B_NAG, the Alg-AGTrans algorithm is also proposed. In addition, the indexes of node attack complexity and node state transition are introduced into the calculation of the probability of reaching each node, and the posterior probabilities are also calculated on this basis. The results of an experimental evaluation show that the model proposed herein can provide an accurate and effective assessment of network vulnerability. However, the proposed algorithm also has some shortcomings that should be addressed. For example, the effects of some factors, such as risk costs, are not considered when calculating the probability of reaching a node.

Funding Statement: This work was partially supported by the National Natural Science Foundation of China (61300216, Wang, H, www.nsfc.gov.cn).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] V. Varadharajan, K. Karmakar, U. Tupakula and M. Hitchens, “A policy-based security architecture for software-defined networks,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 897–912, 2019.

[2] J. Ai, H. Chen, Z. Guo, G. Cheng and T. Baker, “Mitigating malicious packets attack via vulnerability-aware heterogeneous network devices assignment,” *Future Generation Computer Systems-the International Journal of Escience*, vol. 111, no. 2, pp. 841–852, 2020.

[3] A. J. Gallo, M. S. Turan, F. Boem, T. Parisini and G. Ferrari-Trecate, “A distributed cyber-attack detection scheme with application to DC microgrids,” *IEEE Transactions on Automatic Control*, vol. 65, no. 9, pp. 3800–3815, 2020.

[4] Netinfo Security “China Internet Station Development Status and Security Report (2018),” China, 2018. [Online]. Available: https://www.isc.org.cn/editor/attached/file/20180711/20180711201225_67539.pdf.

[5] L. Muñoz-González, D. Sgandurra, A. Paudice and E. C. Lupu, “Efficient attack graph analysis through approximate inference,” *ACM Transactions on Privacy and Security (TOPS)*, vol. 20, no. 3, pp. 1–30, 2017.

[6] L. Munoz-Gonzalez, D. Sgandurra, M. Barrere and E. C. Lupu, “Exact inference techniques for the analysis of bayesian attack graphs,” *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 2, pp. 231–244, 2019.

[7] H. Wang, Z. Chen, J. Zhao, X. Di and D. Liu, “A vulnerability assessment method in industrial internet of things based on attack graph and maximum flow,” *IEEE Access*, vol. 6, pp. 8599–8609, 2018.

[8] X. Sun, J. Dai, P. Liu, A. Singhal and J. Yen, “Using Bayesian networks for probabilistic identification of zero-day attack paths,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2506–2521, 2018.

[9] A. Shameli-Sendi, H. Louafi, W. He and M. Cheriet, “Dynamic optimal countermeasure selection for intrusion response system,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 755–770, 2018.

[10] G. Hu and P. Qiao, “Cloud belief rule base model for network security situation prediction,” *IEEE Communications Letters*, vol. 20, no. 5, pp. 914–917, 2016.

[11] U. Garg, G. Sikka and L. K. Awasthi, “Empirical analysis of attack graphs for mitigating critical paths and vulnerabilities,” *Computers & Security*, vol. 77, no. 4, pp. 349–359, 2018.

[12] B. Li, G. J. Sutton, B. Hu, R. P. Liu and S. Chen, “Modeling and QoS analysis of the IEEE 802.11p broadcast scheme in vehicular *ad hoc* networks,” *Journal of Communications and Networks*, vol. 19, no. 2, pp. 169–179, 2017.

[13] Q. Zhang, C. Zhou, N. Xiong, Y. Qin, X. Li *et al.,* “Multimodel-based incident prediction and risk assessment in dynamic cybersecurity protection for industrial control systems,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 10, pp. 1429–1444, 2016.

[14] S. Wu, Y. Zhang and W. Cao, “Network security assessment using a semantic reasoning and graph based approach,” *Computers & Electrical Engineering*, vol. 64, no. 4, pp. 96–109, 2017.