第 38 卷 第 3 期
Vol.38 No.3

计 算 机 工 程
Computer Engineering

2012 年 2 月
February 2012

• 博士论文 •

文章编号：1000—3428(2012)03—0019—03 文献标识码：A 中图分类号：TP301.6

有环攻击图中的节点风险概率算法

朱随江[1,2]，刘宝旭[1]，刘 宇[1,2]，姜政伟[1,2]

(1. 中国科学院高能物理研究所计算中心，北京 100049；2. 中国科学院研究生院，北京 100049)

摘 要：在攻击图的风险概率计算中，没有针对环路节点的处理。为此，提出一种有环攻击图中的节点风险概率算法。给出带有环路的攻击图示例，介绍有环、无环节点风险概率的计算方法。根据不回溯性假设，确定循环路径，通过移除节点在环路中的出口边及不可达节点打破环路。实验结果表明，该算法能获得较精确的计算结果，且时间复杂度较低。

关键词：攻击图；目标节点；风险概率；循环路径

Nodes Risk Probability Algorithm in Attack Graph with Cycles

ZHU Sui-jiang[1,2], LIU Bao-xu[1], LIU Yu[1,2], JIANG Zheng-wei[1,2]

(1. Computing Center, Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049, China;
2. Graduate University of Chinese Academy of Sciences, Beijing 100049, China)

【Abstract】The calculation of risk probability in the attack graph lacks cycles nodes processing. In order to sovle this problem, this paper proposes a nodes risk probability algorithm in attack graph with cycles. It gives examples of the attack graph with cycles, and introduces the risk probability calculating method with cycles or no cycles. It gives the not retrospective hypothesis to make sure the cycle path, and breaks the cycles through removing the export edge of the nodes in the cycles, and unreachable nodes. Experimental results show that this algorithm can obtain accurate calculation results, and its time complexity is low.

【Key words】attack graph; target node; risk probability; cycling path

DOI: 10.3969/j.issn.1000-3428.2012.03.007

1 概述

传统的漏洞扫描工具，如 Nessus、Retina、ISS 等，只是孤立地给出单个主机内各个漏洞的评述，缺乏对网络内各个漏洞的关联分析，不能识别多阶段、跨主机的网络攻击。攻击者利用网络中多个很轻微的安全漏洞逐步渗透，可能对目标网络造成很严重的损害后果。因此，识别网络内的各种攻击路径生成攻击图成为一个热门的研究领域。国外学者在识别漏洞生成攻击图方面做了大量工作[1-3]，国内学者也有研究，如基于贪心策略生成攻击图[4]，但多数没有实现成熟可大规模应用的系统。世界上的组织如 Bugtraq、NVD、CVE、CNVD、Symantec 等，每天都披露出几十个安全漏洞。由于漏洞补丁的开发需要一定成本和时间，为保证网络服务及软件的可用性，许多漏洞在被发现之后依然存在，因此识别网络资产面临的安全风险，对其被渗透的可能性进行概率评估显得十分重要。基于上述内容，本文提出一种有环攻击图中节点风险概率算法。

2 相关研究

文献[5]指出攻击图中的环路不能单纯通过删除来解决，否则会丢失重要的攻击路径。文献[6]提出消除环路对风险概率计算影响的思想，但没有给出计算各节点风险概率的详细算法。文献[7]通过引入中间节点而等价展开攻击图中的环路，攻击图的等价展开使攻击图变得十分复杂，中间节点的引入导致计算复杂度很高，不适用于大规模网络。

文献[8]提出一种基于攻击图的网络安全概率计算方法，利用到达条件节点的所有路径中，其最大概率来回避各渗透节点相关性，以及通过在攻击图中，环路导致的概率计算复杂性。该方法虽然能简化计算，但忽略其他攻击路径给节点带来的风险，实际上，会错误降低节点发生的概率，节点之间的差错传递，会导致对最终目标节点的风险概率评估出现严重偏差。以文献[8]的攻击图为例，对目标节点的评估结果与准确的评估结果偏差 23%，可信度大大降低。

3 有环攻击图中节点风险概率

在子网内部的主机之间，由于其相互连接，因此会导致攻击图中出现循环路径，如图 1 所示，c 代表条件节点，e 代表渗透节点，为简便起见这里没有列出节点的具体含义。在计算节点 c_3 遭受攻击的发生概率时，存在 3 条路径，即 c_1,e_1,c_3 、 c_2,e_2,c_4,e_4,c_3 、 $c_1,e_1,c_3,e_3,c_4,e_4,c_3$。因为第 3 条路径是循环路径，所以节点概率会重复不合理的计算。

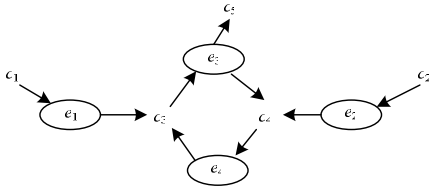

图 1 带有循环路径攻击图示例

基金项目：中国科学院知识创新工程重要方向基金资助项目(YYYJ-1013)；国家科技支撑计划基金资助项目(2009BAH52B06)；中国科学院研发实验服务基地测试基金资助项目(201004)

作者简介：朱随江(1983－)，男，博士研究生，主研方向：网络安全；刘宝旭，研究员、博士；刘 宇，博士研究生；姜政伟，硕士研究生

收稿日期：2011-07-18 **E-mail：**zhusj@ihep.ac.cn

定义 1 设攻击图是一个有向图为 $G(C\cup E,R)$，其中，C 是条件节点集合；E 是渗透节点集合；$R\subseteq(C\times E\cup E\times C)$ 是边集合。G 满足如下约束：

(1)对渗透节点 $e\in E$，记 $pre(e)$ 为 e 的父节点 $c_1,c_2,\cdots,c_m$ 集合，满足 $c_1\wedge c_2\wedge\cdots\wedge c_m\Rightarrow e$，并且记 $post(e)$ 为 e 的子节点 $c_p,c_{p+1},\cdots,c_q$ 集合，满足 $e\Rightarrow c_p\vee c_{p+1}\vee\cdots\vee c_q$。

(2)对条件节点 $c\in C$，记 $pre(e)$ 为 c 的父节点 $e_1,e_2,\cdots,e_n$ 集合，满足 $e_1\vee e_2\vee\cdots\vee e_n\Rightarrow c$。

3.1 无环攻击图中节点风险概率

假设 1 渗透事件独立性表示攻击者执行攻击渗透事件之间的相互独立。

此假设的目的在于，去除渗透之间条件相关性带来的影响，在以后的研究工作中，解决对于绝大部分攻击图中不存在条件相关性的问题。

对每一个条件节点 c，令 $p(c)$ 表示条件节点 c 自身发生的概率；令 $P(c)$ 表示条件节点 c 的累计发生概率，即条件节点 c 可达的可能性。对每一个渗透节点 e，令 $p(e)$ 表示在渗透节点 e 的所有前提条件都满足时，自身发生的概率；假设 $P(e)$ 表示渗透节点 e 的累计发生概率，即渗透节点 e 可达并且执行的可能性。每一个条件节点 c 或为初始条件，或者可以由一个或多个渗透节点满足，故指定其自身发生概率 $p(c)$ 为 1。

定义 2 设给定有向无环攻击图为 $G(C\cup E,R)$，渗透节点累积概率为：

$$P(e)=p(e)\prod_{c_i\in pre(e)}P(c_1)P(c_2)\cdots P(c_m)$$

条件节点累积概率为：

$$P(c)=P(e_1\cup e_2\cup\cdots\cup e_n)$$

其中，$c\in post(e_i),i=1,2,\cdots,n$。

3.2 循环路径

假设 2 不回溯性即攻击者不会从一个节点出发经过若干路径返回出发节点。

攻击者以现有获得条件或权限为基础，通过执行其他攻击而获得更多条件，或更大的权限，而不会通过执行其他攻击重复获得已有条件或权限，即攻击路径具有不回溯性，从而得到这个假设具有合理性。

定义 3 设给定一个攻击图为 $G(C\cup E,R)$，且 $A(G,e)$ (或 $A(G,c)$)表示删除渗透节点 e (或条件节点 c)的出口边、随后而产生的不可达渗透节点，及条件节点后得到的攻击图。

定理 在不回溯性的假设下，在 $G(C\cup E,R)$ 中，$P(e)$ (或 $P(c)$)等于 $A(G,e)$ (或 $A(G,c)$)中的 $P'(e)$ (或 $P'(c)$)。

假设路径不回溯，计算 $G(C\cup E,R)$ 中的 $P(e)$ (或 $P(c)$)将不考虑从节点 e (或 c)出发，且重新回到节点 e (或 c)路径对节点概率的影响，而删除节点 e (或 c)出口边及随后产生的不可达渗透节点和条件节点后，从根本上解决了从节点 e (或 c)出发返回到节点 e (或 c)的所有路径，即等于 $A(G,e)$ (或 $A(G,c)$)中的 $P'(e)$ (或 $P'(c)$)。

以图 1 为例，基于不回溯性假设，到达节点 c_3 的路径有 2 条，即 c_1,e_1,c_3 和 c_2,e_2,c_4,e_4,c_3。路径 $c_1,e_1,c_3,e_3,c_4,e_4,c_3$ 不符合不回溯性假设，因为攻击者不会在已获取条件 c_3 的情况下再通过路径 c_3,e_3,c_4,e_4,c_3 而重复获得 c_3，所以计算节点 c_3 的累积概率时不考虑此条路径。

3.3 有环攻击图中节点概率算法

在改进图论的宽度优先搜索算法中，只有当进入节点的边数等于节点的入度时，才计算节点累积概率并将其入队。当遇到环路时，按照定理计算环路中的节点累积概率，下面给出详细算法：

输入 一个指定每个节点自身概率的攻击图 G

输出 攻击图 G 中所有节点的累积概率

```
1. For each node k∈C∪E do
2.     Indegree[k]=pre(k)的个数;/*计算每个节点的入度*/
3.     if  k 是初始条件 do
4.         P[k]=p[k]=1,Mark[k]=true;/*初始条件节点累积概率为 1，Mark[k]标记节点 k 的累积概率是否计算完毕，P[k]表示累积概率 P(k)，p[k]表示自身概率 p(k)，条件节点自身概率为 1*/
5.         Insert(Q,k);/*计算完毕的节点入队*/
6.     Else
7.         Mark[k]=false;P[k]=p[k];Count[k]=0; /* Count[k]标记已进入节点 k 的有向边个数*/
8. End // For each node k∈C∪E
9. For each node m=delete(Q) Do //出队
10.    For each node q∈post(m) Do
11.        Count[q]=Count[q]+1;
12.        If q∈E then //q 为渗透节点
13.            P[q]=P[q]×P[m];
14.            If  Count[q]=Indegree[q] then
15.                Mark[q]=true;
16.                Insert(Q,q);/*节点 q 入队*/
17.        If q∈C  then //q 为条件节点
18.            If  Count[q]=Indegree[q] then
19.                利用定义 2 计算 P(q);
20.                Mark[q]=true; Insert(Q,q);
/*对于那些入边数非 0 并且小于其入度的节点即是环路的入口节点*/
21. For each node g∈C∪E do
22.    cycles={}/*初始化攻击图中环路集为空*/
23.    If  0<Count[g]<Indegree[g] then
24.        将从入口节点 g 出发的循环路径 loop_path 加入 cycles; /*含有相同节点的循环路径算作一条环路，每条环路都是从入口节点开始，可能包括多个入口节点*/
25. End //For each node g∈C∪E
26. For each loop_path∈cycles Do
27.    For each 入口节点 g∈loop_path Do
28. Calculate_EntryNode_In_Cycle(g,loop_path);
29.    End //For each 入口节点 g∈loop_path
    /*环路 loop_path 中任意一个入口节点 e 开始，计算环路中的其他非入口节点累积概率*/
30.    Calculate_None_EntryNode_In_Cycle(e, loop_path);
31. End// For each loop_path∈cycles
32. While there exists mark[k]=false Do
33. 利用定义 2 计算 P[k];
```

上述算法的每一步都有详细注释，给出如下简要说明：算法 1 行~8 行遍历攻击图中的所有节点，记录每个节点的入度，将初始条件节点插入队列 Q。算法 9 行~20 行处理无环路节点风险概率；当队列 Q 为空时，如果存在入边数大于 0 并且小于入度的节点时，攻击图存在环路，且这些节点全部是循环路径入口节点。算法 21 行~25 行计算所有循环路径；算法 28 行计算每条循环路径的入口节点累积风险概率；算法 30 行计算每条循环路径的非入口节点累积风险概率。如果入口节点是条件节点，只需要考察其后继渗透节点在环路中的

子节点(条件节点)是否有其他路径可达，如图 1 中的 c_3，算法只需要考察其后继渗透节点 e_3 在环路中的子节点 c_4 是否有其他路径可达。如果存在其他可达路径，临时移除边(e_3, e_4)打破环路，计算入口节点累积概率。如果入口节点存在非环路中的出口边，累积概率计算完毕时将入口节点插入队列。如果入口节点是渗透节点，算法不但需要考察其在环路中的后继条件节点是否有其他路径可达，还需要考察其在环路中的父节点(条件节点)是否有非环路入口边。对于入口节点和环路中非入口节点的处理，参见函数 *Calculate_EntryNode_In_Cycle* 和函数 *Calculate_None_EntryNode_ In_Cycle*。

Calculate_EntryNode_In_Cycle 函数的算法如下：

```
1.If j∈C then{//入口节点 j 是条件节点
2.    记 t 为 post(j)中在循环路径 loop_path 中的节点;/*t 为渗透节点*/
3.    记 s 为 post(t)中在循环路径 loop_path 中的节点;/*s 为条件节点*/
4.    If 0<Count[s]<Indegree[s] then{
/*如果 s 为入口节点，删除 t 到 s 的边后，节点 s 仍然为其他到达入口节点 j 的路径中的一个节点;*/
5.        忽略 t 到 s 的边，按照定义 2 计算 P[j];Mark[j]=true;
6.        If 节点 j 有不在环路中的出口边 then
7.            Insert(Q,j);/*节点 j 与环路外的节点有联系，将其入队*/
8.    }Else{/*如果 s 不是入口节点，删除节点 j 的出口边后无其他路径可达节点 j*/
9.        按照定义 2 计算 P[j];Mark[j]=true;
10.       If 节点 j 有不在环路中的出口边 then
11.           Insert(Q,j);
12.   }//End If 0<Count[s]<Indegree[s]
13.}//End If j∈C 入口节点 j 是条件节点
14.If j∈Ethen{//入口节点 j 是渗透节点
15.   记 n 为 post(j)中在环路中的条件节点;
16.   If 0<Count[n]<Indegree[n] then{
/*删除 j 的出口边后如果 post(j)中在环路中的条件节点 n 是入口节点，那么有路径可达渗透节点 j;*/
17.       忽略 j 到 n 的边，按照定义 2 计算 P[j];
      Mark[j]=true;
18.       If 节点 j 有不在环路中的出口边 then
19.           Insert(Q,j);
20.   }Else{/*如果后继不是入口节点，检查 pre(j)中在环路中的条件节点*/
21.           记 d 为 pre(j)中在环路中的节点;
22.           If 0<Count[d]<Indegree[d] then{/*前驱 d 为入口节点*/
23.               忽略环路中到节点 d 的边，按照定义 2 计算 P[j];Mark[j]=true;
24.               If 节点 j 有不在环路中的出口边
25.                   Insert(Q,j);
26.           }Else{
27.               P[j]=0;Mark[j]=true;
28.               If 节点 j 有不在环路中的出口边
29.                   Insert(Q,j);
30.           } //End If 0<Count[d]<Indegree[d]
31.   }//End If 0<Count[n]<Indegree[n]
32.}//End If j∈E 入口节点 j 是渗透节点
```

Calculate_None_EntryNode_ In_Cycle 函数的算法如下：

```
1.记入口节点 e 在环路中的后继节点为 s;
2. If Mark[s]=false then{//s 为非入口节点
3.    If Indegree[s]=1 then{//s 的入度为 1
4.        P[s]=P[pre(s)]×p[s];Mark[s]=true;
5.        If 节点 s 有不在环路 loop_path 中的出口边 then
6.            Insert(Q,s);
7.    }Else{
8. Calculate_EntryNode_In_Cycle(s,loop_path);}
9. }//End If Mark[s]=false then//s 为非入口节点
```

10.令 s 为 post(s)中在循环路径中的节点，继续 2 行~10 行直到环路中所有节点均计算完毕。

3.4　算法的时间复杂度

设攻击图中初始条件节点数为 n_0；入口节点数为 n_1；最大循环路径长度为 L；各循环路径中的总节点数为 n_2；攻击图中总的节点数为 n；节点的最大子节点数为 N。概率算法初始化时间复杂度为 $O(n)$；概率算法 9 行~20 行的时间复杂度为 $O((n-n_0-n_2)N)$；概率算法 21 行~25 行的时间复杂度为 $O(n)$；环路中的入口节点时间复杂度为 $O(n_1L)$；环路中其他节点近似复杂度为 $O((n_2-n_1)L)$；概率算法 26 行~31 行的时间复杂度为 $O(n_2L)$；概率算法 32 行~33 行的时间复杂度为 $O(n)$。概率算法总的时间复杂度表示为：$O((N+3)n+n_2L-(n_0+n_2)N)$，随节点 n 呈线性增长。

文献[8]算法在牺牲精确度的条件下，获得时间复杂度为 $O((n-n_1+e)+n_1\times M^{2L+1})$，其中，$e$ 为有向边的数目；L 为攻击图中最大路径深度；M 为每个节点的最大父节点数目。与文献[8]的算法相比，本文算法并没有显著增加计算复杂度，而获得更高的计算精度，提供更为精确的节点风险概率。与文献[7]算法相比，本文算法具有更低的时间复杂度，且算法的线性时间复杂度适用于大规模网络。

4　实验结果与分析

为测试算法的运行效率，构建 2 个子网，采用乔治梅森大学信息系统安全中心开发的工具 TVA 生成攻击图，参考国际通用漏洞评分系统(CVSS)中的基本分，分配渗透节点自身概率。构建网络详细信息及实验结果参见表 1 所示，其中，*AA* 表示单个主机漏洞数；*BB* 表示攻击图中节点总数；*CC* 表示与文献[8]算法相比节点精度提高；*DD* 表示最大循环路径长度；*EE* 表示文献[8]算法；*FF* 表示文献[7]算法；*GG* 表示本文算法。

表 1　不同算法测试对比

子网	主机数	*AA*	*BB*	*DD*	运行时间/s			*CC*/(%)
					FF	*EE*	*GG*	
A	10	5	1 112	8	18	6	6	17
B	30	10	13 248	22	632	23	24	25

文献[8]算法采用到达节点的最大概率作为节点风险概率虽能简化计算，但会忽略其他可达路径带来的风险，节点风险的差错传递，会随着攻击路径及攻击图中的节点数增多而放大。本文算法在基本不增加时间复杂度的条件下，能获得更为精确的计算结果。文献[7]算法与本文算法有着相同的节点精度，但其对环路节点的处理过于复杂，不适用于大规模网络。

5　结束语

本文提出一种有环攻击图中节点风险概率算法。实验结果表明，该算法能避免节点风险概率重复计算。同已有算法相比，其计算的节点概率精度更高，具有线性时间复杂度，适用于大规模网络。今后将开发有效的攻击图生成工具，进一步研究攻击图中渗透节点的相关性问题。　(下转第 30 页)