ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.4, April 2010, pp.838−848
doi: 10.3724/SP.J.1001.2010.03584

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

攻击图的两种形式化分析*

陈 锋[1,2+], 张 怡[1], 苏金树[1], 韩文报[3]

[1](国防科学技术大学 计算机学院,湖南 长沙 410073)

[2](第二军医大学 网络信息中心,上海 200433)

[3](解放军信息工程大学 信息工程学院,河南 郑州 450002)

Two Formal Analyses of Attack Graphs

CHEN Feng[1,2+], ZHANG Yi[1], SU Jin-Shu[1], HAN Wen-Bao[3]

[1](School of Computer Science, National University of Defense Technology, Changsha 410073, China)

[2](Network Information Center, Second Military Medical University, Shanghai 200433, China)

[3](Institute of Information Engineering, PLA Information Engineering University, Zhengzhou 450002, China)

+ Corresponding author: E-mail: chenfeng@nudt.edu.cn

Chen F, Zhang Y, Su JS, Han WB. Two formal analyses of attack graphs. *Journal of Software*, 2010,21(4): 838−848. http://www.jos.org.cn/1000-9825/3584.htm

Abstract: An attack graph is a model-based vulnerability analysis technology, which can automatically analyze the interrelation among vulnerabilities in the network and the potential threats resulting from the vulnerabilities. Since the state-based attack graphs can not be applied to the real large networks for the combinatorial explosion in the number of attack paths, the study is now shifted to attribute-based. Based on attribute-based attack graphs, this paper discusses the loop attack paths and the optimization security measures. For the former, an iterative algorithm is presented to find all the non-loop attack paths to the key attributes with their depth less than the given number *n*. For the latter, it is proved to be an NP-complete problem, and the greedy algorithm is proposed to solve the problem with polynomial time complexity.

Key words: vulnerability; attack graph; valid attack path; optimization security measures; greedy algorithm

摘 要: 攻击图是一种基于模型的网络脆弱性分析技术,可以自动分析目标网络内脆弱性之间的关系和由此产生的潜在威胁.攻击图主要有状态攻击图和属性攻击图两类.前者由于存在状态爆炸问题不适应于大规模网络,目前主要的研究大多是基于后者.基于属性攻击图研究了含圈攻击路径问题和最优弥补集问题.针对含圈攻击路径问题,定义了反映真实攻击想定的 *n*-有效攻击路径,提出了一种计算关键属性集所有 *n*-有效攻击路径的迭代算法;针对最优弥补集问题,在定义了所有的风险源为属性攻击图的初始属性的基础上,将该问题转化为带权重的集合覆盖问题,从而归结为 NP 完全性问题,提出了可应用于大规模攻击图的具有多项式时间复杂度的近似算法.

关键词: 脆弱性;攻击图;有效攻击路径;最优弥补集;贪婪算法

* Supported by the National Natural Science Foundation of China under Grant No.90604006 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2008AA01A325 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2009CB320503 (国家重点基础研究发展计划(973))

Received 2008-08-03; Revised 2008-11-23; Accepted 2009-01-20

中图法分类号: TP393 文献标识码: A

传统的脆弱性扫描技术是一种基于规则的脆弱性评估方法,它孤立地分析目标网络中存在的脆弱性,不能综合评估这些脆弱性相互作用所产生的潜在威胁.攻击图是一种基于模型的脆弱性评估方法,它从攻击者的角度出发,在综合分析多种网络配置和脆弱性信息的基础上,枚举所有可能的攻击路径,从而帮助防御者直观地理解目标网络内各个脆弱性之间的关系、脆弱性与网络安全配置之间的关系,以及由此产生的潜在威胁.

在研究攻击图自动创建问题的过程中,基于全局状态的建模方法可以产生状态攻击图[1−6].在状态攻击图中,节点表示目标网络和攻击者的状态,有向边表示单一攻击行为引起的状态转换.状态攻击图显式地展示了攻击者从初始状态出发逐步利用目标网络中的脆弱性进行攻击的所有可能的攻击路径,但是由于攻击路径随目标网络的主机规模与脆弱性数目的乘积呈指数增长,它无法应用于大规模网络.为了解决状态攻击图中的组合爆炸问题,Ammann 等人首次将攻击者能力的“单调性”假设引入到分析模型中[9],即攻击者在攻击过程中不断地扩大自己的能力而不会失去已有的能力.该假设提出后,研究者转向采用基于属性的建模方法产生属性攻击图[7−12].在属性攻击图中,节点表示系统条件(属性)和原子攻击,有向边表示节点间的因果关系,属性攻击图能够更紧凑地展示所有的攻击路径.研究者利用该方法实现了多个原型系统,如 TVA[10]和 MulVAL[7,8]等.实践表明,属性攻击图具有良好的可扩展性,可以应用于大规模网络.目前,我们已实现的脆弱性综合分析原型系统 CVAM (comprehensive vulnerability assessment model)也是基于属性攻击图的[13].但是,属性攻击图隐式地展示攻击路径方式给网络安全分析提出了挑战.

文献[7]首先提出了属性攻击图中的含圈攻击路径问题并经研究发现,属性攻击图中含圈攻击路径不能简单地通过删除某些原子攻击来解决,否则会丢失一些重要的无圈攻击路径,但没有提出寻找所有无圈攻击路径的方法.文献[6]基于状态攻击图首次提出和解决了最小优弥补集问题,即以最少的安全弥补措施保障目标网络中关键资源的安全,但状态攻击图难以适应大规模网络.文献[11]基于属性状态攻击图提出和解决了最优弥补集问题,即以最少的成本保障目标网络中关键资源的安全,但是该方法没有解决含圈攻击路径问题.文献[12]提出基于逻辑推理的方法,首先把该问题转化为布尔表达式,然后通过求该表达式的析取范式计算出所有的弥补措施集合,在此基础上求最优弥补集.该方法在最坏情况下具有不可避免的指数时间复杂度,无法应用于网络规模较大的真实目标网络.

本文基于属性攻击图研究了含圈攻击路径问题和最优弥补集问题.针对第 1 个问题,本文提出了反映真实攻击想定的 n-有效攻击路径概念,即路径长度不大于常量 n 的无圈攻击路径,并且提出了迭代算法计算破坏关键属性集安全性的所有 n-有效攻击路径.针对第 2 个问题,本文在定义了所有的风险源为属性攻击图的初始属性的基础上,将最优弥补集问题转化为带权重的集合覆盖问题,从而归结为 NP 完全性问题;提出了多项式时间复杂度的近似算法,可以应用于对大规模攻击图的分析.

1 攻击图的定义

属性攻击图隐式地描述了攻击者利用目标网络中各脆弱点进行逐步入侵的所有攻击路径[7−13],它可以形式化描述如下:

定义 1(属性攻击图). 属性攻击图 $AG=(A_0\cup A_d,T,E)$,其中:A_0 表示初始节点集合,对应网络和攻击者的初始属性集合;A_d 表示可达节点集合,对应网络和攻击者在攻击被逐步实施后可达的属性集合;T 表示原子攻击节点集合;E 为有向边集合.AG 满足下列约束:

(1) $E\subset((T\times A_d)\cup((A_0\cup A_d)\times T))$,即攻击图的两个节点之间的关系仅包含 $A_0\to T,A_d\to T,T\to A_d$,其中:$T\to A_d$ 为原子攻击的后果边;$A_0\to T,A_d\to T$ 为原子攻击的前提边;

(2) 对 $\forall \tau\in T$,令 $Pre(\tau)$表示 τ的父节点集合,$Post(\tau)$表示 τ的子节点集合,则父节点之间存在“与”关系,且满足$(\wedge Pre(\tau))\Rightarrow(\wedge Post(\tau))$,表示当原子攻击的所有前提都被满足后,该原子攻击成功,从而使其后果被满足;

(3) 对$\forall a \in A_d$,令 $Parent(a)$是 a 的父节点集合,则父节点之间存在"或"关系,且满足$(\vee Parent(a)) \Rightarrow a$,表示 $Parent(a)$中任何一个原子攻击成功实施都会使属性 a 被满足.

简单起见,本文后续将属性攻击图简称为攻击图.

从定义 1 可以看出,攻击图中包含两类节点:属性节点和原子攻击节点.属性节点代表目标网络和攻击者能力的条件,原子攻击节点代表攻击者利用单个脆弱性进行的一次攻击.图 1 展示了一个目标网络对应的攻击图.该目标网络存在 2 台主机,分别编号为 1 和 2;在目标网络外部有一台恶意主机且与该网络相连,编号为 0.在该攻击图中,文本文字表示目标网络和攻击者的属性,详细描述见表 1,椭圆表示原子攻击,详细描述见表 2.

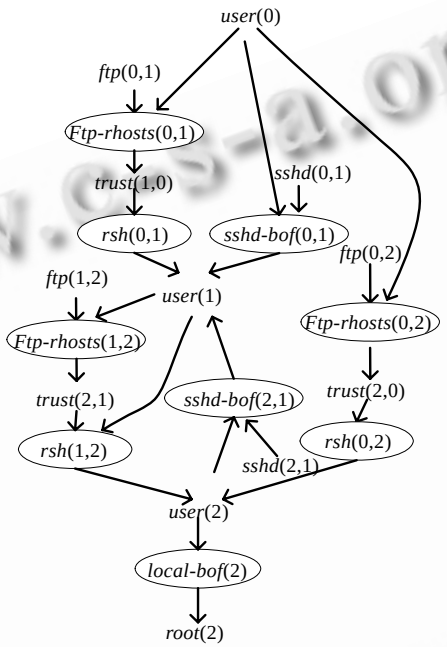

Fig.1 Example of attack graph

图 1 攻击图实例

Table 1 Attributes in the example of attack graph

表 1 攻击图实例中的属性

Attribute	Description
ftp(*a*,*b*)	The service *ftpd* on host *b* is accessible from host *a*.
trust(*a*,*b*)	Host *a* trusts host *b*.
user(*a*)	Attacker has the user privilege on host *a*.
root(*a*)	Attacker has the root privilege on host *a*.

Table 2 Atomic attacks in the example of attack graph

表 2 攻击图实例中的原子攻击

Atomic attack	Description
Ftp_rhosts(*a*,*b*)	Attacker establishes a remote login trust relationship from host *a* to host *b* via the *ftp_rhosts* vulnerability on host *b*.
sshd_bof(*a*,*b*)	Attacker gains the user privilege on host *b* from host *a* using a remote buffer overflow attack on the *sshd* vulnerability of host *b*.
rsh(*a*,*b*)	Using an existing remote login trust relationship between two hosts, the attacker logs in from host *a* to host *b*, getting the user privilege on host *b* without supplying a password.
local_bof(*a*)	Attacker achieves the root privilege on host *a* using a local buffer overflow attack on host *a*

2 基于攻击图的安全分析

2.1 *n*-有效攻击路径

根据容侵的网络安全保障思想,安全管理员可以在攻击图中指定重要的属性集合 $A_c \subseteq A_d$,重点考虑保障这些属性不被攻击者破坏.我们称 A_c 为关键属性集,A_c 是安全的当且仅当$\forall a_c \in A_c$ 不被攻击者破坏.

定义 2(攻击路径). 设 $path=\perp \to \tau_1 \to \tau_2 \to \ldots \to \tau_l$ 是给定攻击图 AG 中的原子攻击序列,其中,符号"⊥"是该序列的起始标识符,$\tau_i \in T(1 \le i \le l)$是原子攻击.若该序列满足下面约束,则称它为 A_c 的一条攻击路径,该攻击路径的长度记为 $L(path)=l$:

(1) 后面任意原子攻击的前提条件都是前面原子攻击的后果或是初始属性,即$\forall a \in Pre(\tau_i)$,

$$a \in \bigcup_{k=1}^{i-1} Post(\tau_k) \cup A_0,$$

其中,$\tau_i \in T, 1 \le i \le l$;

(2) 前面任意原子攻击的后果是后面原子攻击的前提,即$\exists b \in Post(\tau_i), b \in \bigcup_{k=i+1}^{l} Pre(\tau_k)$,其中,$\tau_i \in T, 1 \le i < l$;

(3) 该序列中最后一个原子攻击产生的后果集与关键属性集 A_c 存在非空交集,即 $Post(\tau_l) \cap A_c \ne \varnothing$.

在图 1 的攻击图实例中,令 $A_c=\{user(1)\}$,原子攻击序列 $path_1=\perp \to Ftp_rhost(0,1) \to rsh(0,1)$和 $path_2=\perp \to sshd_bof(0,1)$是 A_c 的两条攻击路径.$path_3=\perp \to Ftp_rhost(0,1) \to rsh(0,1) \to Ftp_rhost(1,2) \to rsh(1,2) \to sshd_bof(2,1)$也是 A_c 的一条攻击路径,它是一条含圈的攻击路径.但在实际的攻击过程中,该攻击路径不会发生,因为攻击者的攻击过程具有单调性,即不会再去获取已经具有的能力.因此在图 1 的例子中,攻击者获得主机 1 的 user 权限后,不会再通过获取主机 2 的 user 权限重复地去获取主机 1 的 user 权限.但在分析攻击图时,不能删除 $sshd\_bof(2,1)$,否则就会删除 A_c 的合理攻击路径 $path_4=\perp \to Ftp\_rhost(0,2) \to rsh(0,2) \to sshd\_bof(2,1)$.攻击图中含圈路径的存在,大大增加了攻击图分析的复杂性.

定义 3(有效攻击路径). 设 $path=\perp \to \tau_1 \to \tau_2 \to \ldots \to \tau_l$ 是给定攻击图 AG 中的一条攻击路径,若该攻击路径不含圈,即$\forall \tau_i, \tau_j \in T(1 \le i < j \le l), Pre(\tau_i) \cap (\bigcup_{j=i+1}^{l} Post(\tau_j)) = \varnothing$,则称该攻击路径是有效攻击路径.

有效攻击路径中,任意原子攻击的前提不是它后面原子攻击的后果,反映了攻击者真实可能的攻击路线.要准确识别攻击者的意图,必须识别出所有的有效攻击路径.

研究中发现,在一般的攻击图中有大量的有效攻击路径包含了完全相同的原子攻击,虽然这些攻击的顺序不同,但反映了相同的原子攻击之间的依赖关系.为了简化和方便理解,我们定义这类有效攻击路径是等价的.例如在图 2 的攻击图中,原子攻击 τ_3 依赖于 τ_1,而 τ_2 与 τ_3 和 τ_1 都不存在依赖关系,故$\perp \to \tau_1 \to \tau_2 \to \tau_3 \to \tau_4$,$\perp \to \tau_1 \to \tau_3 \to \tau_2 \to \tau_4$和$\perp \to \tau_2 \to \tau_1 \to \tau_3 \to \tau_4$等都是等价的有效攻击路径.不失一般性,在下面的研究中,对于这一类有效攻击路径只选取其中的一条作为代表.

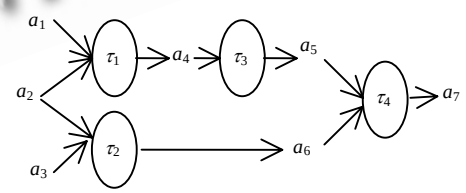

Fig.2 Example of equivalent attack paths

图 2 等价攻击路径的例子

在进一步的研究中发现,利用攻击图生成工具为真实的较大规模目标网络生成的攻击图中,可能存在超长的有效攻击路径(长度超过 20);但是经过统计大量真实攻击事件发现,真实攻击者实施的有效攻击路径的长度绝大部分在 3 以内,且没有发现长度超过 10 以上的.因此,分析不超过指定长度 n 的有效攻击路径,对识别和分析攻击者真实、可能的攻击路线更具实际意义.

定义 4(*n*-有效攻击路径). 设 $path=\bot\rightarrow\tau_1\rightarrow\tau_2\rightarrow\ldots\rightarrow\tau_l$ 是给定攻击图 AG 中的一条有效攻击路径,$n\in N$ 是给定的常数,若攻击路径的长度 $l\le n$,则称该有效攻击路径为 n-有效攻击路径.

对于攻击图 AG,若 $a\in Post(\tau_i)$,我们称 $path^{(i)}(a)=\bot\rightarrow\tau_1\rightarrow\tau_2\rightarrow\ldots\rightarrow\tau_l$ 是可达 a 的第 i 条攻击路径;令 $PATHS(a)=\{path^{(i)}(a)\}$ 是所有可达 a 的攻击路径集合.若 $\tau=\tau_m$,称 $path^{(j)}(\tau)=\bot\rightarrow\tau_1\rightarrow\tau_2\rightarrow\ldots\rightarrow\tau_m$ 是可达 τ 的第 j 条攻击路径.令 $PATHS(\tau)=\{path^{(j)}(\tau)\}$ 是所有可达 τ 的攻击路径集合.

定义 5(函数⊕). 对于任意两条攻击路径 $path_1$ 和 $path_2$,定义函数 $path_1\oplus path_2$:$path_2$ 中的每个原子攻击 τ_i,若对 $path_1$ 中的任意原子攻击 τ_j,满足 $\tau_i\neq\tau_j$,则把 τ_i 增加到 $path_1$ 的尾部,最后返回 $path_1$ 作为该函数的结果.

定义 6(函数⊗). 对于属性节点 $a,b\in A_0\cup A_d$,定义函数

$$PATHS(a)\otimes PATHS(b)=\{path^{(i)}(a)\oplus path^{(j)}(b)|path^{(i)}(a)\in PATHS(a),path^{(j)}(b)\in PATHS(b)\}.$$

为了计算可达指定节点 v 的所有 n-有效攻击路径,我们提出了迭代算法 $obtain\_path(v,n)$,算法细节如图 3 所示.该算法的主要思想是,从指定的节点 v 出发,采取前向搜索的方式和深度优先的搜索策略迭代计算 v 的所有可达其父节点的(n–1)-有效攻击路径(若 v 为属性节点,则计算 v 的所有可达其父节点的 n-有效攻击路径),然后,在此基础上计算可达 v 的 n-有效攻击路径.在计算过程中,为了保证产生的攻击路径不含圈,引入了节点集合 $trace$ 来存放已搜索的属性节点的轨迹.从 v 节点出发,在深度搜索迭代之前,如果该节点是属性节点,则加入到该轨迹中,迭代结束后,将该节点从轨迹中擦除.迭代前若该节点已在轨迹中,则说明继续该次迭代会产生含圈的攻击路径,故终止该次迭代.

```
Input: The given node v in the attack graph AG;
Output: the set of n-valid attack paths reachable to v.
Procedure obtain_path(v,n)
(1)   If v∈A0 Then
(2)       Return PATHS(v)={⊥};
(3)   If v∈Ad Then
(4)       If n==0 Then
(5)           Return PATHS(v)=∅;
(6)       If v∈trace Then
(7)           Return PATHS(v)=∅;
(8)       trace=trace∪{v};
          //{τ1,…,τm} is the set of v's parent nodes
(9)       For each τi Do
(10)          PATHS(τi)=obtain_path(τi,n);
(11)      PATHS(v)=(∪PATHS(τi));
(12)      trace=trace\{v};
(13)  If v∈T Then
          //{a1,…,an} is the set of v's parent nodes
(14)      For each ai Do
(15)          PATHS(ai)=obtain_path(ai,n–1);
(16)      PATHS(v)={path→v|path∈PATHS(a1)⊗…⊗PATHS(an),L(path→v)≤n};
(17)  Return PATHS(v);
```

Fig.3 Detailed description for the algorithm $obtain\_path(v,n)$

图 3 $obtain_path(v,n)$算法的详细描述

定理 1. 算法 $obtain_path(v,n)$产生可达 v 的所有 n-有效攻击路径.

证明:若 v 为原子攻击 τ,假设 $a_1,\ldots,a_n$ 是它的所有父节点,即 $a_i\in Pre(\tau)$,其中 $0\le i\le n$,则根据定义 1 中的约束 2 可知,可达 τ 的 n-有效攻击路径是由它与可达其父节点的所有(n–1)-有效攻击路径组合构成,即 $PATHS(\tau)=\{path\rightarrow\tau|path\in PATHS(a_1)\otimes\ldots\otimes PATHS(a_n)$,且 $L(path\rightarrow\tau)\le n\}$.若 v 为属性节点 a,假设 $\tau_1,\ldots,\tau_m$ 是它的所有父节点,即 $a\in(\cap Post(\tau_i))$,其中 $0\le i\le m$,则 $trace$ 存放了 a 的所有后继原子攻击的后果.根据定义 3,若 $a\in trace$,则说明把 τ_i 的有效攻击路径加入到该路径中会产生圈,故终止该迭代;否则,根据定义 1 中的约束 3 可知,可达 τ_i 的 n-有效攻击路径都是可达 a 的 n-有效攻击路径,即 $PATHS(a)=(\cup PATHS(\tau_i))$. □

下面说明如何利用算法迭代算法 $obtain\_path$ 计算关键属性集 A_c 的所有 n-有效攻击路径.在攻击图 $AG=(A_0\cup A_d,T,E)$中,引入伪原子攻击 δ_i 和伪可达属性节点 ε.令 $A_d^0=A_d\cup\{\varepsilon\}$,$T^0=T\cup\{\delta_i\}$,$E^0=\{\delta_i\rightarrow\varepsilon,a_i\rightarrow\delta_i|a_i\in A_c\}$,其中

$1\le i\le|A_c|$,则新的攻击图 $AG^0=\{A_0\cup A_d^0,T^0,E^0\}$.显然,攻击图 AG^0 中每条可达 ε 的 $(n+1)$-有效攻击路径删除 δ_i 后,就是攻击图 AG 中关键属性集 A_c 的 n-有效攻击路径.因此,通过在攻击图 AG^0 中应用 $obtain\_path(\varepsilon,n+1)$ 可以获得 A_c 的所有 n-有效攻击路径.当取 n 为足够大时,该算法可以计算出 A_c 的所有有效攻击路径.

2.2 最优弥补集

为了保障关键属性集 A_c 的安全性,管理员可以采取各种安全弥补措施通过消除某些初始属性节点来阻断 n-有效攻击路径,如改变网络连接、为某些脆弱性打补丁等.每种安全弥补措施都需要付出一定成本代价.例如,给 Web 服务器打补丁时可能需要中断服务,而购买或制作补丁本身也需要经济成本.因此,我们考虑最优弥补集问题,即如何用最小的代价来保障关键属性集 A_c 的安全性.

定义 7(风险源). 设 $path=\perp\to\tau_1\to\tau_2\to\ldots\to\tau_l$ 是攻击图的一条 n-有效攻击路径$(l\le n)$,称该路径上出现的所有初始属性集合为 $path$ 的风险源,即 $A^r=(\cup Pre(\tau_i))\cap A_0$,其中 $1\le i\le l$.

定义 8(弥补集). 设 $A_1^r,A_2^r,\ldots,A_l^r$ 是关键属性集 A_c 的所有风险源,称 A^m 是弥补集当且仅当 $A^m\cap A_i^r\neq\varnothing$ $(0\le i\le l)$.

显然,任意攻击图的 A_0 都是弥补集,消除该弥补集中的所有初始属性节点将消除所有的风险源,即阻断所有的 n-有效攻击路径.

定义 9(最优弥补集). 成本函数 $Cost:A_0\to^+\Re$ 标识了管理员采取安全弥补措施移除每个初始属性花费的成本,对任意弥补集合 A^m,令 $cost=\sum Cost(a_i)$ 是该弥补集的总成本,其中 $a_i\in A^m$.我们称弥补集 A^m 是最优的当且仅当在所有的弥补集中它的总成本最小.

定理 2. 最优弥补集 A^m 问题是 NP 完全性问题.

证明:数学描述最优弥补集 A^m 问题如下:设$|A_0|=k$,令 $A_1^r,A_2^r,\ldots,A_l^r$ 是关键属性集 A_c 的全部风险源.设变量 $x_j\in\{0,1\}$,若 $a_j\in A^m$,则 $x_j=1$;否则,$x_j=0$.为了使 A^m 是最优弥补集,最终的总成本需要最小化,即

$$cost^*=\mathrm{Min}\left(\sum_{j=1}^{k}x_j cost(a_j)\right) \tag{1}$$

并且满足下面的约束:

(a) $\sum_{a_j\in A_i^r}x_j\ge 1$,其中,$0\le i\le l$;

(b) $x_j\in\{0,1\}$.

约束(a)说明,每个风险源与弥补集有交集.该问题的数学描述与带权重的集合覆盖问题(set cover problem-weighted version)的数学描述相同,而带权重的集合覆盖问题是 NP 完全性完全问题,故最优弥补集 A^m 问题也是 NP 完全性完全问题. □

由于最优弥补集问题是 NP 完全性问题,精确求解算法的时间复杂度为指数级.而真实目标网络的攻击图一般都很庞大,因此需要有多项式时间复杂度的近似算法.本文设计了贪婪算法 weighted-Greedy 来计算近似最优弥补集 A^m.

令 A^R 是风险源集合,$A(a_j,A^R)$是 A^R 中包含 a_j 的风险源集合;函数 $w(a_j,A^R)=|A(a_j,A^R)|$表示 a_j 在风险源集合 A^R 中出现的次数;函数 $\gamma(a_j,A^R)=Cost(a_j)/w(a_j,A^R)$表示 $A(a_j,A^R)$中每个风险源的均摊成本.图 4 给出了算法 weighted-Greedy 的详细描述.

由于贪婪算法计算出的结果是近似最优弥补集,下面讨论近似最优弥补集所需成本与最优弥补集所需成本的性能之比.

Input: All the risk sources{ A_i^r |(0≤i≤l)}, cost function $Cost(a_j)(0\le j\le k)$;
Output: Approximative optimization security measures A^m.

Procedure weighted-Greedy
(1) $A^m=\varnothing$;
(2) $A^R=\{A_i^r \mid 0\le i\le l\}$;
(3) **While** ($A^R\neq\varnothing$)
(4) Find a_j in A_0-A^m, which lets the value of $\gamma(a_j,A^R)(0\le j\le k)$ be minimal;
(5) $A^m=A^m\cup\{a_j\}$;
(6) $A^R=A^R-A(a_j,A^R)$;

Fig.4 Detailed description for the algorithm weighted-Greedy

图 4 Weighted-Greedy 算法的详细描述

设 $cost_g$ 为由贪婪算法计算出的近似弥补集对应的总成本,$t=|A^m|$为最优安全弥补措施的个数;记 $y_i(0\le i\le l)$为风险源 A_i^r 的贪婪成本,它为算法的第(4)行计算所得的最小均摊成本,即 $y_i=\mathrm{Min}(\gamma(a_j,A^R))$, $A_i^r\in A^R$.由此可知,下面的等式成立:

$$\sum_{i=1}^{l} y_i=\sum_{j=1}^{t} cost(a_j)=cost_g \tag{2}$$

令 $A_*^R=\{A_i^r \mid 0\le i\le l\}$,记 $A_*(a_j)=A(a_j,A_*^R)$ 是算法初始化时弥补措施 a_j 所能消除的所有风险源,函数 $w_*(a_j)=|A(a_j,A_*^R)|$表示 a_j 所能消除的风险源最大个数.

引理 1. a_j 所能消除的所有风险源的贪婪成本之和不大于 $Cost(a_j)$的 $H(d)$倍,即

$$Cost(a_j)\cdot H(d)\ge\sum_{A_i^r\in A_*(a_j)} y_i,$$

其中,$d=w_*(a_j)$,$H(d)=(1+1/2+\ldots+1/d)$.

证明:令 $A_*^{(0)}(a_j)=A_*(a_j)$ 且 $d^{(0)}=d$,a_{r-1} 是贪婪算法第 $r-1$ 次迭代后选择的弥补措施.在第 r 次迭代开始时,$A_*^{(r)}(a_j)=A_*^{(r-1)}(a_j)-A_*(a_{r-1},A^R)$,此时$|A_*^{(r)}(a_j)|=d^{(r)}$.令 $y^{(r)}=Cost(a_r)/|A^{(r)}(a_r)|$是此次迭代时产生的贪婪成本,若 $Cost(a_j)/d^{(r)}>y^{(r)}$且$|A_*^{(r)}(a_j)\cap A_*^{(r)}(a_r,A^R)|=l^{(r)}\neq 0$,则 $y^{(r)}$是交集中各个风险源的贪婪成本,且满足 $l^{(r)}\times y^{(r)}\le l^{(r)}\times\frac{Cost(a_j)}{d^{(r)}}\le\frac{Cost(a_j)}{d^{(r)}}+\ldots+\frac{Cost(a_j)}{d^{(r)}-l^{(r)}+1}$;若 $Cost(a_j)/d^{(r)}=y^{(r)}$, 则 $l^{(r)}\times y^{(r)}=Cost(a_j)$, 且 $A_*^{(r+1)}(a_j)=\varnothing$.由于 $d^{(r)}=d^{(r-1)}-l^{(r-1)}$,故 $\sum_{A_i^r\in A_*(a_j)} y_i\le(1+1/2+\ldots+1/d)Cost(a_j)=Cost(a_j)\cdot H(d)$. □

定理 3. 算法 weighted-Greedy 的理论最差性能比为 $H(g)$,即 $H(g)\ge cost_g/cost^*$,其中,$g=\mathrm{Max}_{j=1}^{k}(w_*(a_j))$, $H(g)=1+1/2+\ldots+1/g$.

证明:由引理 1 和公式(1)的约束(a)以及公式(2)可知,

$$\sum Cost(a_j)\cdot H(g)x_j=\sum Cost(a_j)\cdot H(w_*(a_j))x_j=\sum(\sum_{A_i^r\in A_*(a_j)} y_i)\cdot x_j=\sum(\sum_{a_j\in A_i^r} x_j)\cdot y_i\ge\sum y_i=cost_g.$$

因为 $cost^*=\mathrm{Min}(\sum Cost(a_j)x_j)$,故 $H(g)\ge cost_g/cost^*$. □

定理 3 表明,由 weighted-Greedy 计算所得的近似最优弥补集的总成本不超过实际最优弥补集总成本的 $H(g)$倍,其中,g 为单个弥补措施所能消除的风险源最大个数.当 $g=1$ 时,可知 $H(g)=1$,即 $A_i^r\cap A_j^r=\varnothing$ $(1\le i<j\le l)$,此时,贪婪算法所得结果为最优弥补集.

2.3 算法时间复杂度分析

定理 4. 给定攻击图 $AG=(A_0\cup A_d,T,E)$以及常数 $n\in N$,假设每个节点 $u\in A_d\cup T$ 最多具有 M 个父节点,即 $M=\mathrm{Max}_{u\in A_d\cup T}(|Parent(u)|)$,则计算所有可达 $v\in A_d$ 的 n-有效攻击路径算法 $obtain_path(v,n)$的时间复杂度为 $O(M^{2n+1})$.

证明:对于算法 $obtain\_path(v,n)$,若 v 为属性节点 a,假设$\tau_1,\ldots,\tau_m$ 是它的所有父节点,即 $a\in(\cap Post(\tau_i))$,其中

$0\le i\le m\le M$,最多需要调用 M 次 *obtain_path*(τ_i,n)计算所有可达 τ_i 的 n-有效攻击路径.若 v 为原子攻击 τ,假设 $a_1,\ldots,a_n$ 是它的所有父节点,即 $a_i\in Pre(\tau)$,其中 $0\le i\le n\le M$,则最多需要调用 M 次 *obtain_path*(a_i,n−1)计算所有可达 a_i 的 (n−1)-有效攻击路径.由于 $n\ge 0$ 且 $v\in A_d$,故上述迭代次数最多为 $2n$+1,因此,该算法的时间复杂度为 $O(M^{2n+1})$. □

由于 n 是指定的常数,故该算法具有多项式时间复杂度.下面证明近似最优弥补集算法 weighted-Greedy 的时间复杂度.

定理 5. 给定攻击图 $AG=(A_0\cup A_d,T,E)$,假设风险源的个数为 l,则计算最优弥补集算法 weighted-Greedy 的时间复杂度为 $O(|A_0|\cdot l)$.

证明:算法 weighted-Greedy 在 $A_0\backslash A^m$ 中寻找 a_j 满足 $\gamma(a_j,A^R)$最小(第 4 行)需要花费 $O(|A_0|)$时间.算法输入共有 l 个风险源,每次迭代操作(第 3 行~第 6 行)至少消除一个风险源,故其时间复杂度为 $O(|A_0|\cdot l)$. □

2.4 实 例

我们将算法应用于图 1 的攻击图实例来验证算法的正确性.令关键属性集 A_c={*user*(1),*user*(2)},n=4.运行算法 *obtain_path* 来计算所有可达 A_c 的 4-有效攻击路径,并根据定义 6 得到各个 4-有效路径对应的风险源,结果见表 3.假设缺少脆弱性的补丁包,管理员只能通过设置防火墙安全策略阻止为编号为 0,1,2 的主机提供相关服务的方式来保证关键属性集 A_c 的安全性.令 a_1=*user*(0),a_2=*ftp*(0,1),a_3=*ftp*(0,2),a_4=*ftp*(1,2),a_5=*sshd*(0,1),a_6=*sshd*(2,1).设采取这些安全弥补措施移除各个初始属性 A_0={a_1,a_2,a_3,a_4,a_5,a_6}对应所需成本为{∞,5,2,8,11,7}.其中,由于管理员无法阻止攻击者具有目标网络外部恶意主机 0 的 user 访问权限,即 a_1=*user*(0),故消除该属性所花费成本为无穷大.表 4 给出了运行算法 weighted-Greedy 计算最优弥补集的过程,可以得到最近最优弥补集为{a_2,a_3,a_5},它的总成本 $cost_g$=18,而通过文献[12]中的算法得到实际最优弥补集总成本 $cost^*$=18.由于采取消除 a_1 的弥补措施能够消除全部风险源,故 g=6,$H(g)$=1+…+1/6.经计算验证它们满足定理 3,即 $H(g)\ge cost_g/cost^*$.

Table 3 Risk sources corresponding to 4-valid attack paths

表 3 4-有效路径及其对应的风险源

No.	4-Valid attack path	Risk source
1	⊥→*Ftp_rhost*(0,1)→*rsh*(0,1)	*user*(0), *ftp*(0,1)
2	⊥→*sshd_bof*(0,1)	*user*(0), *sshd*(0,1)
3	⊥→*Ftp_rhost*(0,2)→*rsh*(0,2)→*sshd_bof*(2,1)	*user*(0), *ftp*(0,2), *sshd*(2,1)
4	⊥→*Ftp_rhost*(0,2)→*rsh*(0,2)	*user*(0), *ftp*(0,2)
5	⊥→*Ftp_rhost*(0,1)→*rsh*(0,1)→*Ftp_rhost*(1,2)→*rsh*(1,2)	*user*(0), *ftp*(0,1), *ftp*(0,2)
6	⊥→*sshd_bof*(0,1)→*Ftp_rhost*(1,2)→*rsh*(1,2)	*user*(0), *sshd*(0,1), *ftp*(1,2)

Table 4 Procedure of the algorithm weighted-Greedy computing optimization security measures

表 4 Weighted-Greedy 计算最优弥补集的过程

The *i*th iteration	Set of risk resource A^R	Minimum cost shared equally Min($\gamma(a_j,A^R)$)	Choice a_j	Optimization security measures A^m
1	{a_1,a_2}, {a_1,a_5}, {a_1,a_3,a_6}, {a_1,a_3}, {a_1,a_2,a_3}, {a_1,a_5,a_4}	2/3	a_3	a_3
2	{a_1,a_2}, {a_1,a_5}, {a_1,a_5,a_4}	5	a_2	a_3, a_2
3	{a_1,a_5}, {a_1,a_5,a_4}	11/2	a_5	a_3, a_2, a_5

3 实验与分析

我们通过两组模拟实验来分析本文所提算法的性能并验证其正确性.模拟实验所在的主机环境如下:Intel Core Duo T7500 2.2GHz CPU,2GBRAM,Windows XP 操作系统.

3.1 应用于小规模攻击图

在实验 1 中,分别将本文中算法与文献[12]中算法应用于小规模攻击图,分析它们在计算最优弥补集时 CPU 运行时间的变化趋势,以及本文算法所得结果的实际性能之比.实验模拟产生了 5 个具有含圈攻击路径的小规模攻击图,分别编号为 *A*,*B*,*C*,*D*,*E*;它们的复杂度逐渐递增,具体统计特征见表 5.其中:列$|A_0|$表示它的初始属性节点数;$|A_d|$表示它的可达属性节点数;$|T|$表示它的原子攻击节点数;$|E|$表示它的边数,且每个节点最多具有 4 个父

节点.假设各攻击图的关键属性集 A_c 是它们叶节点属性构成的集合,成本函数 $Cost(a_i)=i,i\in N,a_i\in A_0$.*ValidPath* 表示各攻击图的有效攻击路径条数,并且这些有效攻击路径的长度都低于 10.

Table 5 Statistical characteristics of attack graphs in Experiment 1

表 5 实验 1 中攻击图的统计特征

AG	$\|A_0\|$	$\|A_d\|$	$\|T\|$	$\|E\|$	*ValidPath*
A	18	24	28	59	14
B	21	50	67	159	108
C	24	54	68	167	159
D	51	66	86	221	695
E	76	85	97	254	786

图 5 给出了算法的性能实验结果,其中,L_1 是 *obtain_path* 算法的 CPU 运行时间曲线,它表明,该算法产生所有 10-有效攻击路径的 CPU 运行时间随着攻击图的复杂度增大而以多项式方式逐渐增加.L_2 是 weighted-Greedy 算法性能曲线,它表明,该算法产生近似最优弥补集的 CPU 运行时间随着攻击图的复杂度增大而以多项式方式逐渐增加.L_3 是采用文献[12]中算法产生精确最优弥补集的 CPU 运行时间曲线,它表明,其 CPU 运行时间随着攻击图的复杂度增大而以指数方式增加.令 $h=cost_g/cost^*$ 表示贪婪算法的实际性能之比,图 6 显示了贪婪算法的实际性能比和它的理论最差性能比,并验证了定理 3,即该算法的实际性能比低于它的理论最差性能比.

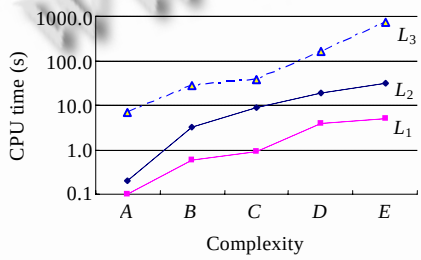

Fig.5 Performance trendlines of each algorithm

图 5 算法性能趋势线

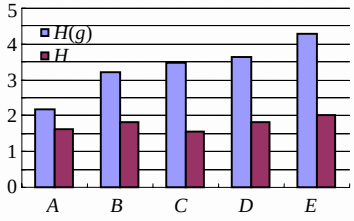

Fig.6 Theoretical and practical performance ratio of the algorithm weighted-Greedy

图 6 算法 weighted-Greedy 的实际性能比及其理论最差性能比

该实验结果表明,文献[12]的算法虽然能够产生精确的最优弥补集,但是它具有指数时间复杂度,故存在可扩展性的局限性,不能应用于大规模的攻击图.本文算法能够以低于其理论最差性能比快速计算出近似最优弥补集.

3.2 应用于大规模攻击图

在实验 2 中,我们将本文算法应用于大规模攻击图来分析它的可扩展性.实验模拟产生了 5 个具有含圈攻击路径的大规模攻击图,分别编号为 *F*,*G*,*H*,*I*,*J*;攻击图中每个节点最多具有 4 个父节点,有效攻击路径的长度都低于 30.它们的复杂度逐渐递增,具体统计特征见表 6.假设各攻击图的关键属性集 A_c 是由其叶节点属性构成的集合,成本函数 $Cost(a_i)=i,i\in N,a_i\in A_0$.运行文献[12]中算法为这些攻击图计算精确最优弥补集,在 1 800s 内都没有产生结果.

Table 6 Statistical characteristics of attack graphs in Experiment 2

表 6 实验 2 中攻击图的统计特征

AG	$\|A_0\|$	$\|A_d\|$	$\|T\|$	$\|E\|$	*ValidPath*
F	201	243	212	776	2 023
G	272	289	245	981	2 876
H	321	387	418	1 167	3 281
I	452	496	486	2 021	4 095
J	676	785	597	2 854	5 986

令 $P(x)$是 *obtain_path* 算法为攻击图 x 产生可达 A_c 的所有 n-有效攻击路径的 CPU 运行时间函数,如图 7 所示.它表明,对于同一个攻击图,所计算的有效攻击路径长度的上界值 n 越大,所需 CPU 运行时间将以指数方式增长.令 $G(x)$是 weighted-Greedy 算法根据可达 A_c 的所有 n-有效攻击路径对应的风险源产生近似最优弥补集所需的 CPU 运行时间函数,如图 8 所示.它表明,对于同一个攻击图,计算近似最优弥补集所需的 CPU 运行时间依赖于所计算的有效攻击路径长度的上界值 n.其原因在于,随着 n 的增大,攻击图的 n-有效攻击路径数目增多,这意味着风险源数目 l 增加.而根据定理 5,weighted-Greedy 算法时间复杂度随 l 线性增加.

该实验结果表明,本文算法 *obtain_path* 和 weighted-Greedy 可以为大规模攻击图快速计算近似最优弥补集,但算法总 CPU 运行时间与有效攻击路径的上界值 n 相关,且随 n 以指数方式增长.但真实的攻击者实施的有效攻击路径的长度绝大部分在 3 以内,且一般不会实施长度为 10 以上的有效攻击路径.因此,利用本文所提算法可以快速地计算 10-有效攻击路径来分析攻击者实际可选的有效攻击路径,并在此基础上计算近似最优弥补集,从而有效保证关键属性集 A_c 在真实环境下的安全性.

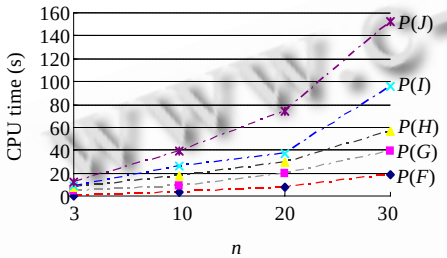

Fig.7 Performance trendlines of the algorithm *obtain_path*(v,n) for the variable n

图 7 *obtain_path*(v,n)算法关于 n 的性能趋势线

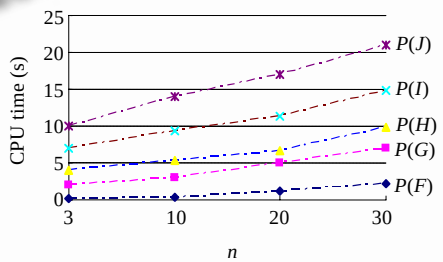

Fig.8 Performance trendlines of the algorithm weighted-Greedy for the variable n

图 8 Weighted-Greedy 算法关于 n 的性能趋势图

4 结 论

本文针对属性攻击图中含圈攻击路径问题和最优弥补集问题进行了研究.针对第 1 个问题,本文提出了迭代算法计算关键安全属性集的所有 n-有效攻击路径.针对第 2 个问题,本文形式化描述了最优弥补集问题,证明了该问题是 NP 完全性问题.为了在大规模攻击图中分析该问题,提出了具有多项式时间复杂度的贪婪算法,并且给出了算法的最差性能比.该贪婪算法是精确计算方法[12]的补充,对于小规模简单攻击图,可采用精确算法,而对于大规模的复杂攻击图,可采用本文的贪婪算法,并且根据定理 3 可确定结果的性能比上界.实验结果表明,迭代算法和贪婪算法可以应用于大规模攻击图,并验证了贪婪算法的实际性能比和它的理论上界.由于属性攻击图隐式地展示攻击路径,不便于直观理解,在后续研究工作中,我们将对属性攻击图的化简和可视化管理进行深入研究.

References:

[1] Swiler LP, Phillips C, Gaylor T. A graph-based network-vulnerability analysis system. Technical Report, SANDIA Report No.SAND 97-3010/1, 1998.

[2] Swiler LP, Phillips C, Ellis D, Chakerian S. Computer-Attack graph generation tool. In: Proc. of the 2nd DARPA Information Survivability Conf. & Exposition. Los Alamitos: IEEE Computer Society Press, 2001. 307−321.

[3] Lippmann RP, Ingols KW. An annotated review of past papers on attack graphs. Technical Report, ESC-TR-2005-054, MIT Lincoln Laboratory, 2005.

[4] Ritchey R, Ammann P. Using model checking to analyze network vulnerabilities. In: Proc. of the 2000 IEEE Symp. on Security and Privacy. Oakland: IEEE Computer Society Press, 2000. 156−165.

[5] Sheyner O, Jha S, Wing JM, Lippmann RP, Haines J. Automated generation and analysis of attack graphs. In: Hinton H, Blakley B, Abadi M, Bellovin S, eds. Proc. of the IEEE Symp. on Security and Privacy. Oakland: IEEE Computer Society Press, 2002. 273−284.

[6] Jha S, Sheyner O, Wing J. Two formal analyses of attack graphs. In: Proc. of the 15th IEEE Computer Security Foundations Workshop. Cape Breton: IEEE Computer Society, 2002. 49−63.

[7] Ou XM, Boyer WF, McQueen MA. A scalable approach to attack graph generation. In: Proc. of the 13th ACM Conf. on Computer and Communications Security. Alexandria: ACM Press, 2006. 336−345.

[8] Ou XM. A logic-programming approach to network security analysis [Ph.D. Thesis]. Princeton: Princeton University, 2005.

[9] Ammann P, Wijesekera D, Kaushik S. Scalable, graph-based network vulnerability analysis. In: Proc. of the 9th ACM Conf. on Computer and Communications Security. New York: ACM Press, 2002. 217−224.

[10] Jajodia S, Noel S, O'Beny B. Topological analysis of network attack vulnerability. In: Kumar V, Srivastava J, Lazarevic A, eds. Proc. of the Managing Cyber Threats: Issues, Approaches and Challenges. Netherlands: Kluwer Academic Publisher, 2003.

[11] Noel S, Jajodia S, O'Berry B, Jacobs M. Efficient minimum-cost network hardening via exploit dependency graphs. In: Proc. of the 19th Annual Computer Security Applications Conf. Las Vegas: IEEE Computer Society Press, 2003. 86−95.

[12] Wang L, Noel S, Jajodia S. Minimum-Cost network hardening using attack graphs. Computer Communications, 2006,29(18): 3812−3824. [doi: 10.1016/j.comcom.2006.06.018]

[13] Chen F, Su JS, Han WB. AI planning-based approach of attack graph generation. Journal of PLA University of Science and Technology (Natural Science Edition), 2008,9(5):460−465 (in Chinese with English abstract).

附中文参考文献:

[13] 陈锋,苏金树,韩文报.一种基于智能规划的攻击图快速构建方法研究.解放军理工大学学报(自然科学版),2008,9(5):460−465.

陈锋(1979－),男,江苏丹阳人,博士生,主要研究领域为信息安全,网络安全,密码学.

张怡(1973－),女,博士,副研究员,主要研究领域为信息安全,网络安全.

苏金树(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为计算机网络,网络安全.

韩文报(1963－),男,博士,教授,博士生导师,主要研究领域为信息安全,密码学.