基于 Stackelberg 攻防博弈的网络系统安全控制机制优化研究

王 震[1,2], 段晨健[2], 吴 铤[2], 郭云川[1], 王 竹[1,3*], 李凤华[1,3]

[1] 中国科学院信息工程研究所 北京 中国 100093
[2] 杭州电子科技大学网络空间安全学院 杭州 中国 310018
[3] 中国科学院大学网络空间安全学院 北京 中国 100049

摘要 企业级网络中存在的漏洞日益增多, 给公司网络系统安全控制机制的优化选择带来了巨大挑战。本文通过对企业网络中漏洞之间的复杂依赖关系进行建模, 构建了漏洞依赖图, 并在此基础上建立了 Stackelberg 攻防博弈模型。同时考虑到传统求解方法无法求解实际的问题规模, 引入双模块算法。实验结果表明, 本文提出的模型和方法是可行的、高效的。

关键词 漏洞依赖图; Stackelberg 博弈; 安全控制机制; 双模块算法
中图法分类号 TP309.2 DOI 号 10.19363/J.cnki.cn10-1380/tn.2019.01.09

Research on Optimizing Security Control Mechanism of Networked System Based on Stackelberg Defender-Attacker Game

WANG Zhen[1,2], DUAN Chenjian[2], WU Ting[2], GUO Yunchuan[1], WANG Zhu[1,3*], LI Fenghua[1,3]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China
[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract The increasing number of vulnerabilities in enterprise-level networks poses a huge challenge to the optimal selection of corporate network system security control mechanisms. This paper models the complex dependencies between the vulnerabilities in these networks by building a Vulnerability Dependency Graph, and model the Stackelberg game on it. At the same time, considering the traditional solution method cannot solve the actual problem scale, a Double Oracle algorithm is introduced. The results show that the proposed model and method are feasible and efficient

Key words vulnerability dpendency graphs; stackelberg game; security control mechanism; double oracle algorithm

1 引言

随着信息技术的高速发展, 企业级网络面临的安全问题日益严重, 网络安全面临严峻挑战[1-3]。网络中的各种软件和服务普遍存在的漏洞是导致网络安全事件发生的重要原因。大部分公司都会使用 Microsoft, Oracle, Cisco 等公司的软件, 同时他们也有自己的软件产品, 并向用户提供服务, 这些服务或者软件中的漏洞给攻击者提供了很多的攻击机会。美国国家标准与技术研究院(National Institute of Standards and Technology, NIST) 的漏洞数据库(National Vulnerability Database, NVD)[4]记录了超过 50000 个漏洞, 几乎任何一个软件或者服务中都存在漏洞。另外, 随着网络攻击技术的智能化和复杂化, 攻击者会充分利用目标网络中的漏洞然后发起攻击。攻击者通常会首先利用一些网络扫描工具如 Nessus[5]、X-scan[6]等对公司内部软件存在的漏洞进行扫描, 然后选择最优的路径发起攻击。因此, 如何

通讯作者: 王竹, 博士, 副研究员, wangzhu@iie.ac.cn。
本课题得到国家重点研发计划基金资助项目（No. 2016YFB0800700）和国家自然科学基金项目（No. 61872120, No. 61672515）的资助。
收稿日期：2018-09-30; 修改日期: 2018-11-24; 定稿日期: 2018-12-14

选择最优的安全控制机制(如: 修补漏洞或动态地关闭部分服务)来降低企业网络系统遭受攻击的风险是一个值得研究的问题。

网络系统安全控制机制的优化选择问题存在诸多挑战。首先, 一个公司的所有软件和服务中的漏洞存在复杂的相互依赖关系, 比如一个服务器上 FTP 服务的漏洞必须在某个主机上的远程访问漏洞被利用之后才可以被利用, 因此在进行安全控制机制选择时必须要合理地建模这些漏洞之间的复杂依赖关系。其次, 企业通常可以选择修复漏洞或者动态地关闭部分服务来进行安全控制, 但两种机制通常都有资源限制: 企业不可能修补所有漏洞, 所以企业要在考虑漏洞的依赖关系、补丁可用性、修补花费等因素的基础上选择合适的补丁进行修补; 另外企业在选择动态关闭部分服务时, 也要考虑不能影响正常业务的运行, 因此如何在考虑资源受限的情况下进行最优的安全控制机制选择是另一个挑战。最后也是最重要的是, 攻击者通常会在发起攻击前, 通过漏洞扫描工具发现网络中的剩余漏洞及其关联关系, 然后选择最优的漏洞利用路径发起攻击, 因此企业在指定安全控制机制时, 必须要考虑到攻击者的策略性响应行为。

针对以上挑战, 本文首先利用漏洞依赖图(Vulnerability Dependency Graphs, VDG)对漏洞之间的依赖关系进行建模, 然后将企业防护者(防守者)和潜在黑客(攻击者)之间的交互关系建模为 VDG 上的 Stackelberg 攻防博弈。这里考虑到防守者采用动态地关闭部分服务作为安全控制机制, 这样防守者的纯策略为选择若干软件或服务(以及其在 VDG 上对应的漏洞节点)关闭; 而攻击者的纯策略为选择 VDG 上从原发性漏洞到内部任意漏洞节点的一条路径。防守者的优化目标对应为求解该博弈中的最优防守者混合策略。本文将整个博弈过程建模为零和博弈, 但因为双方的策略集会随着 VDG 的网络规模指数集增长, 所以传统求解方法无法求解实际的问题规模, 为此引入了一种双模块算法进行求解, 并对部分算法模块的计算复杂性和算法近似度给出了理论证明。最后, 结合人工构造的网络结构和通用安全漏洞评分系统[7](Common Vulnerability Scoring System, CVSS)的漏洞数据库提供的数据构造了不同规模的算例进行实验, 实验结果证实双模块算法可以求解实际的问题规模, 并且解质量明显优于其他多种启发式解决方案。

2 相关工作

本文所用到的漏洞依赖图属于攻击图的一种, 国内外已经有许多针对攻击图的工作。20 世纪 90 年代, Philips 和 Swiler 等人[8]在进行网络脆弱性分析时首次提出了攻击图的概念。Kotenko 和 Stephashkin 等人[9]根据网络配置、安全规则等信息对攻击者的行为进行建模, 并基于此来构建攻击图。Sekar 等人[10]通过枚举所有攻击者可能利用的漏洞来对攻击图进行建模。Ammann 等人[11]利用模型检测来分析攻击者是否可以从初试状态到达目的状态。国内研究方面, 叶子维等人[12]对攻击图的技术应用做了详细综述, 其中对攻击图的基本构成, 类型、应用现状等方面进行分析, 并给出了未来可以研究的方向。陈峰等人[13]也做了关于攻击图技术综述, 总结了攻击图技术的发展现状, 阐述了它的巨大应用前景, 最后分析了该技术目前所面临的主要挑战。本文所使用的漏洞依赖图是一种特殊的攻击图, 忽略了传统攻击图中的具体状态之间变化的过程, 只保留了其中的漏洞节点, 从漏洞之间关联的角度对网络环境进行分析。

在通过各种方法构建攻击图后, 开始有人用博弈论的方法对攻击图进行研究。Kashyap 等人[14]用博弈论的方法进行分析, 优化表现指数。Zhu 等人[15]使用博弈论来研究干扰攻击。Han 等人[16]用到的是 Stackelberg 博弈。Dewei 和 Poolsappasit 等人[17]通过帕累托分析, 考虑到了这个问题中对漏洞打补丁会带来的成本, 他们找到一种解决方案, 能够对成本和风险进行权衡。Edoardo Serra 等人[18]的工作与本文的网络模型一样, 都是利用漏洞依赖图进行分析, 他们提出了算法为防守者寻找帕累托最优, 最大化了生产力的同时最小化了修补漏洞的成本, 但是遗憾的是这个工作并没有给出纳什均衡的解法。

现有的关于攻击图的工作中并没有考虑到网络规模增大时带来的策略空间爆炸的问题, 而这一问题在其他领域已经有比较成熟的解法, 即双模块(Double Oracle)算法。它经常被用来解决博弈中策略空间很大的问题。比如 Manish Jain 等人[19]研究了在城市中攻击者沿着道路逃跑的问题, 其中防守者需要在路口设置检查点来拦截他们, 由于真实的城市交通网络非常庞大, 因此首次提出了双模块算法来求解均衡。接着, Manish Jain 等人[20]又在自己之前工作的基础上, 对最基础的双模块进行改进, 添加了热启动过程和近似模块, 利用贪心算法寻找有效策略, 大大提高了算法的运行速度。双模块给出了大规模博弈均衡的求解方案, 并得到了广泛应用。Qingyu Guo 等人[21]对双模块算法进行了适当修改, 解决美国墨西哥边境上非法武器和毒品交易等问题。作者王震等人[22]也曾利用双模块算法研究的是安全机构

对恐怖分子网络的监视问题。虽然上述工作的博弈场景各不相同, 但都是从网络结构的角度来对博弈进行建模, 然后再利用双模块算法求解的, 本文所用到的漏洞依赖图, 本质上也是一直图结构, 因此双模块算法同样适用于本文的场景。

本文中在漏洞依赖图的基础上建立了博弈模型, 给出了攻防双方具体策略和收益函数的数学定义, 同时用整数规划建立了纳什均衡的表达。同时考虑到网络规模增大时双方策略空间会指数级增加的问题, 利用双模块框架来求解纳什均衡。

3 模型

在本节中介绍了使用到的漏洞依赖图并在该网络的基础上建立了 Stackelberg 博弈, 对攻击者和防守者的策略、收益进行了详细定义。

3.1 漏洞依赖图

本文中利用攻击图来进行网络建模。攻击图展示了攻击者可能发动的攻击顺序和攻击效果, 由顶点和边两部分构成。理论上攻击图可以构建完整的网络安全模型, 反映网络中各个节点的脆弱性并刻画出攻击者攻陷重要节点的所有途径。本文所用到的漏洞依赖图是一种特殊的攻击图。VDG 主要用来描述漏洞之间的依赖关系。比如, 可以假设这样一个攻击场景: 在一个局域网环境中, 一台服务器运行 FTP 服务, 其中存在漏洞 CVE-2004-0148(remote-to-root), 还有一台主机中存在漏洞 CVE-2004-0836 (remote-to-user), 攻击者可以直接利用主机中的漏洞, 通过访问服务器的 FTP 服务漏洞获得服务器的用户权限, 此时对攻击者来说, CVE-2004-0148 依赖于 CVE-2004-0836, 也就是说只有在 CVE-2004-0836 被利用之后, CVE-2004-0148 才可以被利用[22]。

在漏洞依赖图 $G(V,E)$中, 点集 V 表示漏洞集合, 每个点代表一个漏洞, 边集 E 表示漏洞之间的关系。显然, VDG 是有向图, 用 $d_{in}(v)$和 $d_{out}(v)$来表示点 v 的入度和出度。相应地, 用 $pre(v)$和 $post(v)$表示点 v 的前驱邻居和后继邻居。

首先可以做一个基本假设, 一个漏洞只能属于一个软件或服务, 而一个软件或服务中存在至少一个漏洞。用*Serven*表示所有的服务, 其中的每个服务 $s \in Serven$ 在一个主机上运行。在这个基础上, 定义一个漏洞与服务之间的二元关系 $R=\{<v,s> | v \in V \wedge s \in Server\}$。用$S(v)$表示漏洞属于的软件, 用 *Vuln*$(v)$表示该服务中包含的所有漏洞。对于每个漏洞, 需要估计其对企业网络系统的重要程度, 也可以理解为该漏洞被攻击者利用后所带来的影响。这里使用NIST的CVSS标准来对该值进行估计。CVSS是测量漏洞严重性的标准, 可以用来比较漏洞的严重程度然后决定修复它们的优先级。CVSS的评价得分主要基于以下几个指标: *attackVector*、*attackComplexity*、*confidentiality-Impact*、*integrityImpact*、*availabilityImpact*、*impactScore*。本文选择其中的*impactScore*来确定每个漏洞的影响, 用*impact*(v)表示。在图1中给出了VDG的例子。

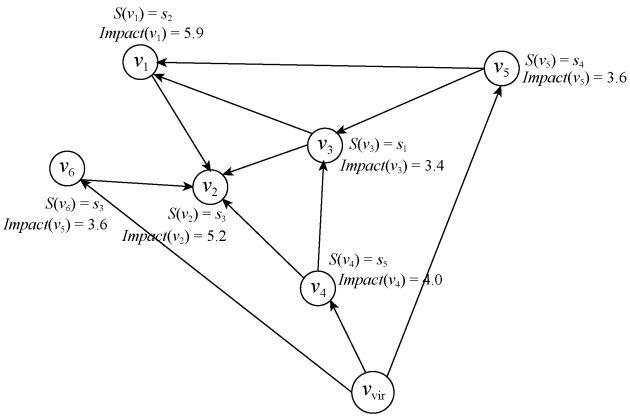

图 1 VDG 示例

Figure 1 An example of VDG

在图1的例子中, $V=\{v_1,\cdots,v_6\}$, $Server=\{s_1,\cdots,s_5\}$, $R=\{<v_1,s_2>,<v_2,s_3>,<v_3,s_1>,<v_4,s_5>,<v_5,s_3>\}$, 每个漏洞的*impact*分别为5.9、5.2、3.4、4.0、3.6、3.6。

3.2 Stackelberg 博弈

本节中将这个攻击场景建模为 Stacklberg 博弈[18], 然后定义了攻击者和防守者的策略。在 Stacklberg 博弈中, 其中有一方作为领导者, 先选择自己的策略, 而另一方作为跟随者也知道领导者所选择的策略, 并且会对该策略做出自己的最佳响应。

在给出双方策略的定义之前, 可以进行一个合理的假设: 攻击者可以使用一些漏洞分析工具来检测一个公司所使用的软件中漏洞之间的关系, 然后推测出网络的拓扑结构, 因此, 攻击者了解 VDG 的所有信息。在这个博弈模型中, 防守者会首先根据自己的策略把一些服务关闭, 因此防守者是领导者, 攻击者是跟随者。攻击者可以随时探测网络的拓扑结构来推断防守者关闭了哪些服务, 以便做出针对防守者策略的最佳响应。

3.2.1 策略

防守者策略: 公司的安全人员作为防守者需要选择一些服务, 然后将它们关闭, 这样就可以保证其中的漏洞不会被攻击者所利用。防守者的一个纯

策略用 *ds* 来表示, *ds* 是一些服务的集合, 但是防守者不可能关闭大量的服务, 这会对公司的正常业务造成影响, 所以假设防守者有一个安防预算, 用 *B* 来表示, 即保证 $|ds| \leqslant B$。在图 1 给出的示例中, 当防守者的纯策略为 $ds=\{s_1,s_2\}$ 时, 漏洞 v_1、v_3 就可以得到保护。在给定防守者策略 *ds* 后, 用 *unprotect*(*ds*)来表示策略 *ds* 不能保护到的漏洞, 用 *protect*(*ds*)表示可以保护到的漏洞。防守者的纯策略空间用 *DS* 表示, 混合策略 $\mathbf{x}=<x_{ds}>$ 是 *DS* 上的概率分布, 即防守者以 x_{ds} 的概率使用 *ds*。

攻击者策略: 在这个攻击场景中, 可以认为攻击者的目的是利用目标漏洞窃取数据或者非法登录等。但是攻击者在利用漏洞时必须满足漏洞之间的依赖关系。在 VDG 中, 入度为 0 的点 $d_{in}(v)=0$ 不需要依赖于其他任何漏洞, 可以直接被攻击者利用。用 *start*(*G*)来表示 VDG 中所有入度为 0 的点。如果一个漏洞 *v* 的任意一个前驱漏洞 $u \in pre(v)$ 被利用, 那么 *v* 就可以被利用。因此, 攻击者从那些入度为 0 的点开始会形成一条攻击路径。任何一个入度为 0 的点都可能成为攻击者的攻击起点, 当网络规模很大时, 数量众多的起点会使计算变得非常复杂, 所以为了简化问题, 加快算法的计算速度, 在 VDG 中添加一个虚拟节点 v_{vir} 作为攻击者统一的起点, 该节点与网络中所有的起点 *start*(*G*)相连, 而且该节点不能被防守者保护到, 如图 1 中的 v_{vir} 所示。用 *as* 表示一个攻击者的一个纯策略, 其中 $as \subset V, as \bigcap start(G) \neq \varnothing$。与防守者的表示相同, *AS* 表示攻击者的纯策略空间, 混合策略 $\mathbf{y}=<y_{as}>$ 是 *AS* 上的概率分布。

3.2.2 收益

假设该博弈模型是零和博弈。对于防守者来说, 关闭服务势必会带来一定的影响, 比如降低了生产力, 而且还有一些操作成本产生。但是如果攻击成功, 造成的损失会非常巨大, 而防御成本与此相比显得微不足道, 所以本文不对防守者的成本进行定量分析。同样地, 对于攻击者来说, 虽然发起攻击也需要一定的成本, 但是如果攻击成功, 比如成功窃取数据, 然后将这些非法得到的数据在黑市出售, 他们的收益也是相关客观的, 所以本文中也没有定量分析攻击者的成本。

在给定攻击策略 *as* 和防守策略 *ds* 之后, 如果 $protect(ds) \bigcap as = \varnothing$, 表明攻击者利用的那些漏洞没有被防守者保护到, 此时攻击者和防守者的收益分别为 *P*(*as*)和–*P*(*as*)。如果 $protect(ds) \bigcap as \neq \varnothing$, 说明防守者通过保护漏洞切断了攻击路径, 此时攻击失败, 但是由于双方都付出了一定的成本, 所以收益均为 0。收益函数 *P*(*as*)的定义如下:

$$P(as)=\sum_{v \in as} impact(tas) \times length(as) \tag{1}$$

当攻击路径越长时, 攻击者得到的收益显然是越大的, 因此在公式(1)中添加了 *length*(*as*)这一项, 表示所有起点 *start*(*G*)中到该策略目标点的最短路径长度的最小值, *tas* 表示该策略的目标节点。

给定防守者混合策略 **x** 和攻击者策略 *as*, 此时攻击者的期望收益为:

$$U_a(\mathbf{x},as)=P(as)\sum_{ds \in DS}(1-z_{x,as})x_{ds} \tag{2}$$

其中 $z_{x,as}$ 是标志变量, 如果攻击成功, *z*=0, 否则, *z*=1。

给定攻击者的混合策略 **y** 和防守者的策略 *ds*, 攻击者的期望收益为:

$$U_a(ds,\mathbf{y})=\sum_{ds \in DS}(1-z_{y,ds})y_{as}P(as) \tag{3}$$

当双方都是混合策略时, 攻击者的期望收益为:

$$U_a(\mathbf{x},\mathbf{y})=\sum_{ds \in DS}x_{as}U_a(ds,\mathbf{x})=\sum_{ds \in AS}y_{as}U_a(\mathbf{y},as) \tag{4}$$

3.2.3 求解均衡

在 Stackelberg 博弈中, 因为是零和博弈, 所以 Stackelberg 均衡和纳什均衡是等效的。在这个模型中, 防守者的目的是最大化自己的最小收益, 攻击者则是最小化自己的最大收益。可以用线性规划来找到博弈中的最优策略, 确定双方的最优策略:

$$\max U \tag{5}$$

$$\text{s.t.} \quad U \leqslant Ud(\mathbf{x},as) \quad as \in AS \tag{6}$$

$$\sum_{ds \in DS} x_{ds}=1 \tag{7}$$

$$x_{ds} \geqslant 0 \tag{8}$$

当双方的策略空间都很小时, 可以通过求解(5)~(8)中的规划方程得到最优解, 但是随着网络规模的不断扩大, 防守者的策略空间会随着预算值 *B* 指数级增加, 攻击者的策略空间也会同样随着|*V*|增加。此时用数学规划很难在短时间内找到最优解, 所以需要设计新的算法求解双方的最优策略。

4 算法

为了求解上节中提出的问题, 本文引入了双模块算法框架, 该框架在文献[19]中首次提出。双模块算法框架可以在博弈的策略空间很大时有效地减少计算最优解所用的时间, 接下来首先给出算法的整体框架[10]。

4.1 算法总览

在图 2 中给出双模块算法的整体流程图。首先在给定初始策略后, 利用公式(5)~(8)求解小规模博弈的均衡, 然后分别调用防守者和攻击者模块求解双方的策略(调用顺序不影响计算结果), 如果双方都没有找到好的策略, 那么算法终止, 最终结果收敛到了全局最优解。

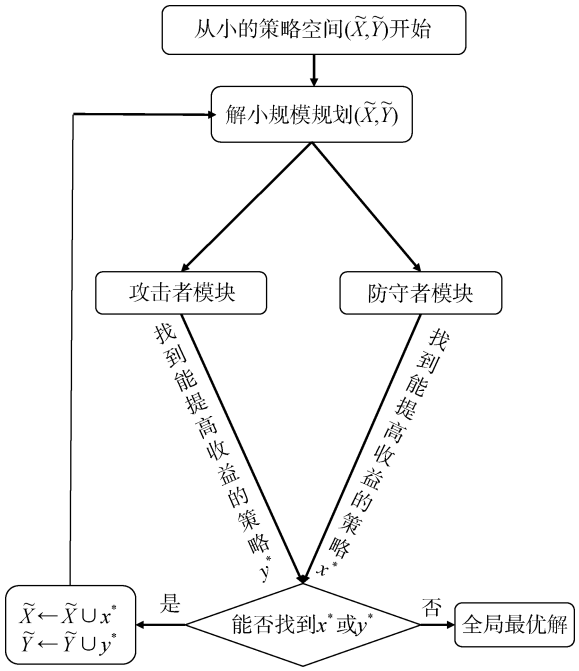

图 2 双模块算法流程图

Figure 2 Double Oracle flowchart

下面用伪代码的形式对算法进行详细描述, 如算法 1 所示。

算法 1: 双模块算法总框架

输入: VDG、防守者预算 B

输出: 双方的混合策略$(\mathbf{x},\mathbf{y})$

1 初始化 $\tilde{X}$ 、 $\tilde{Y}$

2 REPEAT

3 $(\mathbf{x},\mathbf{y})\leftarrow \text{CoreLP}(\tilde{X},\tilde{Y})$

4 $x^*\leftarrow$防守者近似模块$(\mathbf{y})$

5 IF $x^*=\varnothing$ THEN

6 $x^*\leftarrow$防守者最优模块$(\mathbf{y})$

7 $\tilde{X}\leftarrow\tilde{X}\cup x^*$

8 $y^*\leftarrow$攻击者近似模块$(\mathbf{x})$

9 IF $y^*=\varnothing$ THEN

10 $y^*\leftarrow$攻击者近似模块$(\mathbf{x})$

11 $\tilde{Y}\leftarrow\tilde{Y}\cup y^*$

12 UNTIL and

13 返回 $(\mathbf{x},\mathbf{y})$

在第 1 行的初始化阶段中, 假设攻击者每个初始策略包括虚拟节点 v_{vir}, 每个 $start(G)$以及它的第一个出度邻居。防守者则是随机选择 B 个服务作为初始策略。在第 3 行中使用 CoreLP 即公式(5)~(8)来求解 Stackelberg 均衡和双方收益。在第 4~8 行, 防守者近似模块使用一个启发式算法尝试寻找一个收益更高的策略, 如果不能找到这样一个策略, 那么使用防守者最优模块中的混合整数线性规划(Mixed Integer Linear Programming, MILP)来求解防守者的最优策略。第 9~13 行对于攻击者来说同样的过程。当攻击者和防守者都不能找到更好的策略时, 算法结束。

4.2 防守者模块

防守者模块包括两部分: 最优模块(Defender Best Oracle)和近似模块(Defender Better Oracle)。最优模块负责求解防守者对攻击者混合策略的最佳响应, 即最优策略。近似模块负责寻找能提高防守者收益的策略。

4.2.1 防守者最优模块

给定攻击者的混合策略后, 防守者需要找到一个纯策略作为自己的最佳响应。当防守者的策略 ds 满足 $U_a(ds,\mathbf{y})>U_d(\mathbf{x},\mathbf{y})$, 那么该策略就是有效的, 就把它进入防守者的策略空间 $\tilde{X}$ 中。可以使用下面的 MILP 来找到最大化防守者收益的纯策略:

$$\max -\sum_{as\in Y}(1-z_{as})y_{as}P(as) \tag{9}$$

$$\text{s.t.}\quad z_{as}\leqslant\sum_{v\in V}A_vS_v\forall as\in Y \tag{10}$$

$$\sum_{p\in Server}S_p\leqslant B \tag{11}$$

$$S_v=1\,M[p][v]=1\forall v\quad\forall S_p=1 \tag{12}$$

$$z_{as}\in[0,1] \tag{13}$$

$$S_p=\{0,1\} \tag{14}$$

$$A_v\ \ S_v=\{0,1\} \tag{15}$$

其中 S_p 和 S_v 是防守者关于服务和顶点的选择变量。A_v 是攻击者关于节点的选择变量。如果防守者的策略 ds 切断了攻击路径 as, 即 $ds\cap as\neq\varnothing$, 那么 z_{as}=1, 否则为 0。公式(12)中的 $M[Server][|V|]$ 是服务和顶点的关联矩阵, 当防守者选择任意一个服务 p 时, 只要有与节点 v 相关联的漏洞在服务 p 中, 那么 $M[p][v]=1$。

但是防守者最优模块是 NP 难问题[22]。为了减少

运算时间, 需要用近似模块来找到一个能提高防守者收益的纯策略即可, 而不必每个迭代步中都找出最优解。只有当近似模块无法找到这样的策略时, 再调用最优模块。

定理 1. 防守者最优模块是 NP 难问题

证明. 通过问题规约到同样是 NP 难的集合覆盖问题上(Set Cover Problem)可以实现对该问题的证明。集合覆盖问题是给定一些元素集合 U, 还有一些集合 $S=\{S_1,S_2,\cdots,S_i\}$, 集合覆盖问题的目标是想找到 k 个子集, 使它们能够包含 U 中所有的元素。

首先构造一个 VDG。对于 U 中的每个元素 $u\in U$ 和每个非单元素集合 $S_i\in S$, 向网络中添加一个节点 v。然后为每个点添加一个标签 $L(v)$用来表示与该点相关的元素或集合。需要注意的是, 一个软件中可能存在多个漏洞, 因此有时需要为一个元素添加多个点。接下来, 添加 VDG 中的边。如果一个点的标签是$\{u\}$, 另一个是 $u\in S$, 那么这两点之间添加一条边。最后构造攻击者的混合策略。为每个元素 $u\in U$ 构造一个攻击者的纯策略 $as\in\bigcup_{u\in S_i},_{S_i\in S}L^{-1}(S_i)$, 其中 $L^{-1}(S_i)$ 表示标签为 S_i 的点。

在图 3 中给出了一个例子, 其中 $U=\{1,2,3\}$, $S=\{\{1\},\{2\},\{3\},\{1,2\},\{2,3\}\}$。

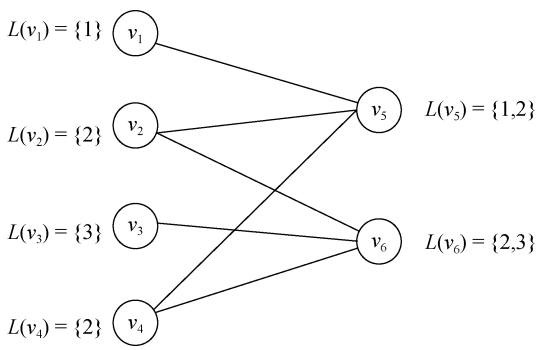

图 3 示例网络

Figure 3 Sample network

攻击者的纯策略有 3 个, 与元素$\{1\}$相关的 $as_1=\{v_1,v_5\}$, 与$\{2\}$相关的 $as_1=\{v_2,v_4,v_5,v_6\}$, 与$\{3\}$相关的 $as_3=\{v_3,v_6\}$。只需证明当且仅当防守者通过保护 VDG 中的 k 个服务切断攻击路径时, U 被 k 个子集所覆盖。

首先, 假设可以通过保护 VDG 中的 k 个服务切断攻击路径, 用 P 来表示与这 k 个服务相关的点。那么只需要证明集合 U 可以被 $L(v)$覆盖即可。

充分性证明: 如果 U 不能被 $L(v),\forall v\in P$ 覆盖, 那么一定会存在一个元素 $\hat{u}\notin L,\forall v\in P$。但是根据攻击者策略的定义, 攻击者的策略包括了所有的标签, 因此 $\hat{u}$ 不在任意一个攻击者策略中, 这与假设矛盾。

必要性证明: 如果假设 U 可以被 k 个集合覆盖。那么可以推断出如果标签为 $u\in L(V)$ 的顶点被覆盖, 那么与 u 相关的攻击者策略也一定会被切断。当 U 被 k 个集合覆盖时, 意味着对于每个 $u\in U$, 至少是 k 个子集中某个的成员, 因此所有攻击者的策略都会被防守者选择保护的 k 个服务切断, 证毕。

既然防守者的最优模块是 NP 难的, 所以需要利用近似模块中的算法来尝试寻找一个能提高防守者收益的纯策略。当近似模块无法找到这样的策略时, 再调用最优模块。

4.2.2 防守者近似模块

在本节中提出一个贪心算法来计算可以提高防守者收益的纯策略[18], 即 $U_d(ds,\mathbf{y})>U_d(\mathbf{x},\mathbf{y})$。首先需要证明收益函数是次模函数, 这样用贪心算法就可以得到一个较好的解。

定理 2. 贪心算法可以保证防守者最优模块有 $1-\frac{1}{e}$ 的精确度。

证明. 首先需要定义标准的收益函数 F:

$$U_d(ds,\mathbf{y})-U_d(\varnothing,\mathbf{y})=\sum_{as\in y|as\bigcup ds\neq\varnothing}y_{as}P(as) \tag{16}$$

接下来, 需要证明 $F(ds)$是次模的。设 V_1 和 V_2 是两个根据防守者选择服务确定的两个集合, $V_1\subseteq V_2$。$\overline{Y_1}$ 和 $\overline{Y_2}$ 是两个不能被 V_1 和 V_2 切断的攻击策略, 即 $\overline{Y_1}\in y\,|\,\overline{Y_1}\bigcap V_1=\varnothing$, $\overline{Y_2}\in y\,|\,\overline{Y_2}\bigcap V_2=\varnothing$。由于 V_1 能切断的攻击者策略一定会被 V_2 切断, 因此可以得出 $\overline{Y_1}\subseteq\overline{Y_2}$。只要函数 $F(ds)$是次模的, 那么一定会满足下面这个式子:

$$F(V_1\bigcup v)-F(V_1)\geqslant F(V_2\bigcup v)-F(V_2) \tag{17}$$

然后可以得到:

$$\begin{aligned}F(V_1\bigcup v)-F(V_1)=&\sum_{as\in y|as\bigcap(V_1\bigcup v)\neq\varnothing}y_{as}P(as)\\&-\sum_{as\in y|as\bigcap V_1\neq\varnothing}y_{as}P(as)\\&-\sum_{as\in\overline{Y_1}|as\bigcap v\neq\varnothing}y_{as}P(as)\end{aligned} \tag{18}$$

对于(17)式的右边, 同样可以得到:

$$F(V_2 \cup v) - F(V_2) = \sum_{as \in \overline{Y_2} | as \cap v \neq \varnothing} y_{as} P(as) - \sum_{as \in y | as \cap V_1 \neq \varnothing} y_{as} P(as) - \sum_{as \in \overline{Y_1} | as \cap v \neq \varnothing} y_{as} P(as) \tag{19}$$

由于 $\overline{Y_1} \in \overline{Y_2}$, $y_{as} P(as) > 0$, (17)式得证。

由于 $F(ds)$是次模函数, 所以算法 2 能够保证 $1 - \frac{1}{e}$的精确度。

下面用伪代码详细描述防守者近似模块的贪心算法, 如算法 2 所示。

算法 2: 防守者近似模块

输入: 攻击者混合空间 **y**
输出: 防守者纯策略 *ds*

```
1   初始化 ds←∅
2   初始化 Y ← y
3   WHILE |ds < B and Ȳ ≠ ∅ DO
4       w_i = 0 ∀s_i ∈ Server
5       FOR v_j ∈ Ȳ DO
6           FOR v_j ∈ ȳ_j DO
7               w_s(i) ← w_s(i) + y_i * T(ȳ_i)
8       s* ← argmax w_s(i)
9       ds ← ds ∪ v(s*)
10      FOR ȳ_i ∈ Ȳ DO
11          IF v(s*) ∈ ȳ_i THEN
12              Ȳ ← Ȳ − v(s*)
13  WHILE |ds| < B DO
14  ds ← ds ∪ v(s) choose s ∈ Server randomly
15  IF U_d(ds, y) > U_d(x, y) THEN
16      返回 ds
17  ELSE
18      返回 空
```

该贪心算法会在每个迭代步内尝试找出一个有效的防守者策略。输入 **y** 是攻击者的策略空间, 但是只关注其中概率不为 0 的策略(支撑集)。第 2 行中的 $\overline{Y}$ 表示不会被防守者切断的攻击者策略。第 3 行的防守者策略 *ds* 在开始时为空, 每次需要向其中添加与所选择的服务相关的点。w_i 是每个点上的权值, 防守者根据 w_i 来选择要关闭的服务, 其表示该点被当前输入的攻击者混合策略 **y** 通过的概率总和, 在第 5~7 行进行更新。第 7 行的 $T(y)$表示攻击者策略的收益。$S(v)$和 $V(s)$表示顶点和服务之间的对应关系。在第 8 行, 该算法总是会选择权值最大的服务, 然后在第 9 行将网络中与其相关的点加入防守者策略 *ds* 中, 并把该点从攻击者策略中移除(10~12 行)。当没有服务满足条件时, 那么随机选择一些服务, 然后生成一个策略(13~14 行)。最后, 需判断要找到的策略是否有效, 如果有效, 则返回, 否则返回空。

4.3 攻击者模块

攻击者模块与防守者模块类似, 同样由两部分组成: 最优模块(Attacker Best Oracle)和近似模块(Attacker Better Oracle)。最优模块负责求解攻击者对防守者混合策略的最佳响应, 即最优策略。近似模块负责寻找能提高攻击者收益的策略。

4.3.1 攻击者最优模块

同样地, 利用下面的 MILP 来计算攻击者的最佳策略, 以此作为对防守者混合策略的最佳响应。

$$\max impact(t) * d(t) * \sum_{dsEx} x_{ds}(1 - Z_{ds}) \tag{20}$$

$$\text{s.t.} \sum_{u \in post(v)} y(v,u) = \sum_{u \in pre(v)} y(u,v) \forall v \neq s,t \tag{21}$$

$$\sum_{u \in post(s)} y(s,u) = 1 \tag{22}$$

$$\sum_{u \in pre(t)} y(u,t) = 1 \tag{23}$$

$$Z_{ds} \geqslant X_{ds,v} * S_v \ \forall ds \forall v \tag{24}$$

$$\sum_{u \in post(v)} y(v,u) + \sum_{u \in pre(v)} y(u,v) - 1 \leqslant Sv \leqslant \sum_{u \in post(v)} y(u,v) \tag{25}$$

$$Z_{ds} = [0,1] \tag{26}$$

$$S_v = \{0,1\} \tag{27}$$

$$X_{ds,v} = \{0,1\} \tag{28}$$

$$y(v,u) \in \{0,1\} \tag{29}$$

其中, $y(v,u)$和 S_v 分别是关于边和点的选择变量。S_v 在(25)式中通过 $y(v,u)$来约束。如果防守者策略 *ds* 保护到了节点 *v*, 那么 $X_{ds,v}$ 为 1, 否则为 0。该规划的输入是攻击路径的起点 *s* 和终点 *t*。理论上需要遍历所有路径的起点和终点, 但是由于虚拟节点的存在, 只需要考虑虚拟节点和所有终点即可, 这样可以减少很多计算量。但即使这样, 攻击者最优模块仍然是一个 NP 难问题, 在网络规模比较大时不能很快得到结果, 因此需要攻击者近似模块来计算攻击者策略。

下面对攻击者的最优模块的计算复杂度进行证明[19]。

定理 3. 攻击者最优模块是 NP 难问题

证明. 证明的思路是将这个问题规约到 3-SAT 问题上来。3-SAT 是给定一个有穷布尔变量集合, $x=\{x_1,x_2,\cdots,x_n\},|x|=n$, 每个变量都能取值 0 或 1, 一组子句, $C=\{C_1,C_2,\cdots,C_k\},|C|=k$, 一个公式, $E=C_1\wedge C_2\cdots\wedge C_k$, 每个 C_i 是由三个变量组成的析取范式, 即 $x_1\vee x_2\vee x_3$。3-SAT 问题在于是否存在一个成真赋值, 使公式 E 为真。

下面给出一个例子:

$$E=(x_1\vee -x_2\vee -x_3)\wedge(x_1\vee x_2\vee x_4) \tag{30}$$

其中公式 E 包含了 n=4 个变量和 k=2 个子句。

接下来通过以上信息构造图。这个图中包括 $n+k+1$ 个节点 $v_0,v_1,\cdots,v_{n+k}$, 其中路径的起点 $s=v_0$, 终点 $t=v_{n+k}$。每条边连接其中的一对节点, 即每条边对应 3-SAT 表达中的一个文字, 即取值为 x_i 或 $-x_i$。因为每个子句由三个变量构成, 所有对于 $i=1,\cdots,k$ 的点 v_{i-1} 和 v_i 来说, 它们之间有三条边相连, 这三条边与第 i 个子句相对应。另一方面, 对于 $j=1,\cdots,n$ 的点 v_{k+j-1} 和 v_{k+j} 来说, 它们之间有两条边, 分别与文字 x_j 和 $-x_j$ 对应。下面给出一个与公式(30)对应的一个例子, 如图 4 所示。

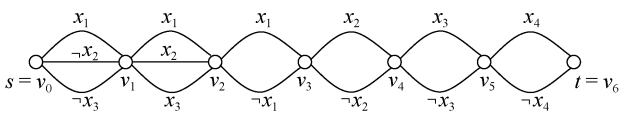

图 4 与公式(30)相关的示例网络

Figure 4 Sample network about formula (30)

然后, 确定共有 $2n$ 个防守者纯策略, 每个被使用的概率为 $1/2n$, 每个防守者纯策略与一个文法相对应, 然后根据这些文法对应的边来确定这条边是否被防守者切断。在图 4 给出的例子中, 共有 4 个变量, 8 个文法, 因此防守者共有 8 个纯策略, 并且每个被使用的概率都是 1/8。当防守者纯策略与文法 x_1 相关时, 一共有 3 条边可以被切断, 分别为 v_0 与 v_1 之间最上面的边, v_1 与 v_2 之间最上面的边, 和 v_2 与 v_3 之间最上面的边。当防守者纯策略与文法 $-x_4$ 相关时, 那么只有 1 条边可以被切断, 即 v_5 与 v_6 之间最下面的边。

接下来需要证明当且仅当防守者以至多 1/2 的概率切断从 s 到 t 的一条路径时, 就会存在一个关于变量的成真赋值使 3-SAT 问题中的公式为真。

充分性证明: 假设一条从 s 到 t 的路径 p 以至多 1/2 的概率被切断。可以注意到对于所有 $i=1,\cdots,n$, 任何一条从 s 到 t 的路径都可以被与 $\{x_i,-x_i\}$ 相关的至少一个策略切断。对于一些变量, 如果这条路径会同时通过标签为 x_i 和 $-x_i$ 的边, 那么这条路径被切断的概率就至少为 $\frac{n+1}{2n}>\frac{1}{2}$, 这与假设矛盾, 对于路径 p 来说是不可能发生的。因此, 需要为与路径 p 上的边相关的文字分配真值为 1, 不在路径上的则为 0。这个过程就等价于构造了一个 3-SAT 问题的解。因为每个子句都包含了与路径中的边相关的文字, 而它们被分配的真值为 1。

必要性证明: 假设已经存在一个使 3-SAT 问题中公式成真的赋值。考虑一条从 s 到 t 的与赋值为 1 的子句相关的边构成的路径 p。由于成真赋值满足每个子句, 因此这样的路径一定存在。进一步说, 这条路径只能被与真值为 1 的文字相关的防守者策略切断, 而这些文法恰好有 n 个。因此, 这条路径被切断的概率为 $\frac{n}{2n}=1/2$, 证毕。

既然攻击者的最优模块是 NP 难的, 所以与防守者类似, 需要利用近似模块中的算法来尝试寻找一个能提高攻击者收益的纯策略。当近似模块无法找到这样的策略时, 再调用最优模块。

4.3.2 攻击者近似模块

在本节中提出了一个攻击者的启发式算法[18]。伪代码如算法 3 所示。

算法 3: 攻击者近似模块

输入: 起点 s、终点 t、防守者混合策略 $\mathbf{x}$

输出: 攻击者纯策略 as

```
1   初始化 weight[v] ← ∞ v ∈ V
2   weight[s] ← 0
3   X̄5 ← x
4   add s to Queue Q
5   WHILE Q is not empty DO
6       u ← arg min_{v∈Q} weight[v]
7       Remove u from Q
8     FOR i ∈ out(u)DO
9         g ← Σ_{dse x̄|vi∈ds} x_i
10  IF weight[u] + g ⩽ weight[i]THEN
11          weight[u] + g ← weight[i]
12          pre[i] ← u
```

```
13            X_i ← X_u − {ds | i ∈ ds}
14            add i to Q
15  FOR vi ∉ start(G)DO
16      payoff[v] ← (1 − weight[t]) * impact(t)
17    t* ← arg max payoff[t]
18    as ← prev[t*]
19  IF Ua(x, as) > Ua(x, y)THEN
20      返回 as
21    ELSE
22      返回 NULL
```

算法 3 主要是以 Dijkstra 算法为基础。算法的输入是路径的起点 s 和终点 t, 在本论文中, 起点为虚拟节点。算法 3 将传统 Dijkstra 算法中根据边的权值修改为根据 $weight[v]$ 进行计算, 其表示防守者能够切断攻击者从 s 到 v 路径的概率。与此相对应, 用 $\overline{X_v}$ 来表示攻击者从 s 到 v 路径中不会被防守者切断的策略。攻击者移动过程中, 需要对每个点的后继节点进行计算, 如果到某个后继节点被切断的概率更低, 那么就对该点的权值进行更新, 并对后者的 $\overline{X_v}$ 进行更新(10~14 行)。同时在计算过程中, 用 $pre[i]$记录节点的前驱信息。在对整个网络的节点计算之后, 在第 15~16 行计算了每个目标节点的收益, 选择出使攻击者收益最大的目标节点, 那么攻击者的策略是从虚拟节点到该目标节点的一条路径。在 18 行通过 $pre[i]$ 中保存的前驱节点信息倒退出整条路径。最后需要判断该策略是否有效, 如果有效, 则返回, 否则返回空。

5 实验结果

在本节中将给出实验结果。因为很难搭建大规模的真实网络环境, 获取真实漏洞依赖图的数据比较困难, 所以这里使用无标度(Scale-Free)网络来模拟 VDG。这里的无标度网络满足 Barabási-Albert 模型[24]。无标度网络中的度分布满足幂律分布, 其中包含大量度小的节点和小部分度大的节点, 构造过程如下:

(1) 增长阶段。从包含 n_0 个节点的初始网络 G_0 开始, 每次都添加若干节点, 每个新节点与 $m(m \leqslant n_0)$ 个已经存在的节点相连。

(2) 优先连接。当新节点要与已有节点 n_i 进行连接时, 连接的概率取决于已有节点的度:

$$P_i = \frac{d(n_i)}{\sum_{j-1}^{n} d(n_j)} \tag{31}$$

可以通过修改参数来生成不同规模的网络, 以便评价所提出的算法在平均度不同的无标度网络中的表现。如 BA(2,2)表示初始节点有 2 个, 每个新节点与 2 个已有节点相连的无标度网络。但是由于传统的无标度网络是无向图, 因此需要对构造过程做一些改动再生成 VDG。具体过程如下:

(1) 构造 Scale-Free 网络。这里需要注意的是当新节点 v_{new} 添加到网络中, 与已经存在的点 v_{exist} 建立连接时, $v_{new} \to v_{exist}$ 之间的边是有向的。

(2) 预处理漏洞数据。本文使用的是现有的漏洞数据库中(Common Vulnerabilities and Exposures, CVE)发布的漏洞信息。具体来说使用的是来源于文献[25]中的 JSON Feed CVE-2018 数据。从数据集中提取出每个漏洞的 CVE-ID, *product_name*, *impactScore*。这里的 *impactScore* 即 3.1 节中提到的每个漏洞的 *impact* 值, 取值为[0,10]。CVSS 有两个版本的标准: baseMetricV3 和 baseMetricV2, 这里优先选择使用基于 baseMetricV3 的 *impact-Score*。如果 V3 的数据缺失, 用 base-MetricV2 来进行替代。此外, 一些漏洞并不是存在于服务中, 比如 CVE-2018-0791 是在微软 office 软件中的一个漏洞, 但是要求公司停止使用 office 是很不现实的, 因此在实验中从数据集中移除了类似的漏洞, 只考虑服务中的漏洞。

(3) 随机关联。这里将漏洞与网络中的顶点进行一对一随机关联, 然后攻击者和防守者根据各自的策略选择网络中的点。

所有实验在主频为 3.60GHz 的双核 CPU 和 8.00GB 内存的 PC 机上进行。所有的线性规划都使用规划求解软件 CPLEX(版本为 12.6)进行求解。由于模型中多处涉及到随机性, 对于每个结果, 都进行 50 次实验, 取平均值做为最终的结果。

首先, 运用双模块算法求解图 1 中给出的示例, 展示算法的执行结果。其中用到的漏洞信息如表 1 所示。

在表 2 中, 给出了每个时间步内双方策略所包括的节点(忽略虚拟节点)和攻击者的收益。时间步为 1 时是初始化策略的过程。接下来每个时间步分布调用防守者模块和攻击者模块计算双方的策略, 最后计算当前策略下双方的纳什均衡

接下来给出一般网络的实验结果。作为对比, 本文称直接利用公式(5)~(8)中数学规划求解全局最优解的过程为 FullLP。这个过程中枚举攻击者和防守者所有策略的时间也被计算在内。图 5、6、7 分别是两种方法在不同资源数下的运行时间对比图。

表 1　节点-漏洞信息表

Table 1　The information of vertices and vulnerabilities

节点	相关服务	相关漏洞	impact
v_1	s_2	CVE-2018-0088	5.9
v_2	s_3	CVE-2018-0092	5.2
v_3	s_1	CVE-2018-0087	3.4
v_4	s_5	CVE-2018-0086	4.0
v_5	s_4	CVE-2018-0089	3.6
v_6	s_3	CVE-2018-0090	3.6

每个时间步的求解结果如表 2 所示。

表 2　双模块算法求解过程

Table 2　The solution process of Double Oracle

时间步	攻击者策略(概率)	防守者策略(概率)	攻击者收益
1	v_1,v_4	v_2,v_4 v_1,v_5 v_1,v_6	
均衡	v_1,v_4 (1)	v_2,v_4 (1)	0
2	v_1,v_4	v_2,v_4 v_1,v_5 v_1,v_6 v_2,v_3,v_5 v_3,v_5	
均衡	v_1,v_4 (1)	v_2,v_3,v_5 (1)	5.2
3	v_1,v_4 v_3,v_4	v_2,v_4 v_1,v_5 v_1,v_6 v_2,v_3,v_5 v_3,v_5 v_2,v_6	
均衡	v_1,v_4(0.12) v_3,v_4(0.88)	v_2,v_6 (1)	5.2
4	v_1,v_4 v_3,v_4 v_1,v_3,v_6	v_2,v_4 v_1,v_5 v_1,v_6 v_2,v_3,v_5 v_3,v_5 v_2,v_6	
均衡	v_1,v_4 (0.53) v_3,v_4 (0.47)	v_1,v_5(0.47) v_2,v_3,v_5(0.53)	2.76396
5	v_1,v_4 v_3,v_4 v_1,v_3,v_6 v_3,v_5	v_2,v_4 v_1,v_5 v_1,v_6 v_2,v_3,v_5 v_3,v_5 v_2,v_6	

续表

时间步	攻击者策略(概率)	防守者策略(概率)	攻击者收益
均衡	v_1,v_4 (0.53) v_1,v_3,v_6(0.47)	v_1,v_5(0.47) v_2,v_6 (0.53)	2.76396
6	v_1,v_4 v_3,v_4 v_1,v_3,v_6 v_3,v_5 v_1,v_2,v_6	v_2,v_4 v_1,v_5 v_1,v_6 v_2,v_3,v_5 v_3,v_5 v_2,v_6	
均衡	v_1,v_4 (0.04) v_1,v_3,v_6 (0.27) v_3,v_5 (0.27) v_1,v_2,v_6 (0.43)	v_1,v_5 (0.27) v_1,v_6 (0.27) v_3,v_5 (0.46)	1.57953

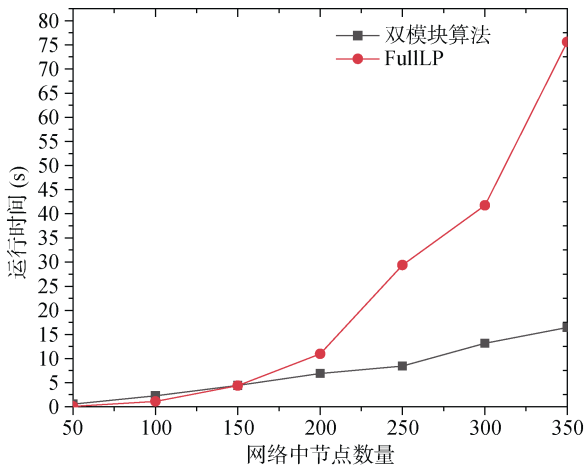

图 5　*B*=2 时双模块与 FullLP 对比图

Figure 5　Comparison of Double Oracle and FullLP when *B*=2

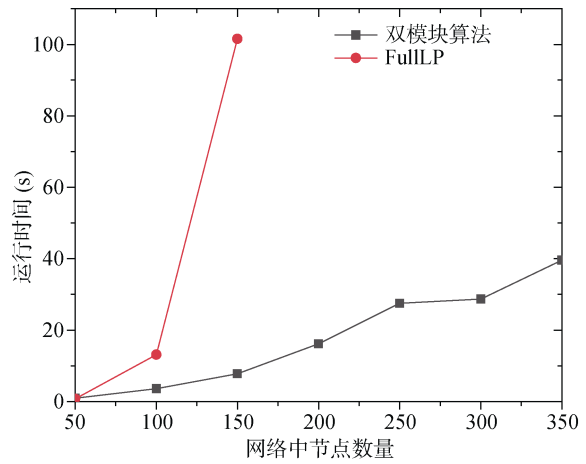

图 6　*B*=3 时双模块与 FullLP 对比图

Figure 6　Comparison of Double Oracle and FullLP when *B*=3

很明显可以看出, 在网络规模和预算比较小时(点的数量小于 100, B=2), FullLP 比双模块算法要快, 但是双模块算法在大规模网络时明显有更好的表现。在图 6 和图 7 中, 没有给出完整的 FullLP 求解的结果, 是因为当网络规模和预算值比较大时, 很难再短时间内得出结果。

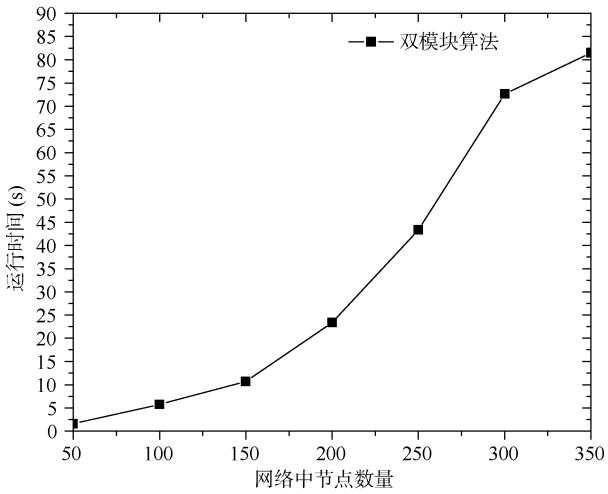

图 7 B=4 时双模块与 FullLP 对比图

Figure 7 Comparison of Double Oracle and FullLP when B=4

为了说明近似模块的必要性, 将双模块算法有无近似模块时的情况做了对比, 结果分别如图 8~图 10 所示。

可以很明显的看出没有用到近似模块时算法明显要慢, 而且随着网络规模的增大, 这种差距也逐渐变大。在预算值为 4 时, 由于可选择的防守者的策略数量非常庞大, 因此出现了有近似模块和比无近似模块慢的情况。为了具体说明近似模块的必要性, 在无标度网络为 BA(2,2), 节点数量为 200, 预算值 B=3 时, 将双模块算法在都有攻击者近似模块和防守者近似模块、只有攻击者近似模块、只有防守者近似模块以及两者都没有的情况下进行运算时间的对比, 结果如图 11 所示。

在上面的基础上, 紧接着分别比较了上面四种情况下近似模块中算法的调用比例。显然, 如果没有近似模块的情况下, 那么调用比例为 0, 其他三种情况如图 12 所示。

可以看出, 近似模块的调用比例并不高, 双方近似模块的比例分别为 0.19 和 0.28, 也就是说, 近似模块在很多时候并不能找到一个好的策略, 需要用最优模块计算最优策略。

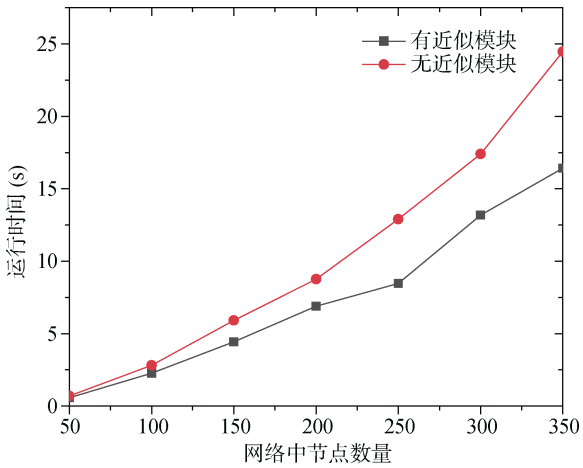

图 8 B=2 时有无近似模块对比图

Figure 8 Comparison of Double Oracle with or without Better Oracle when B=2

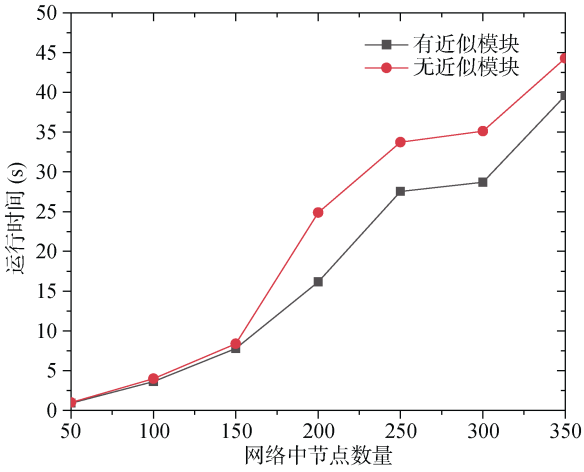

图 9 B=3 时有无近似模块对比图

Figure 9 Comparison of Double Oracle with or without Better Oracle when B=3

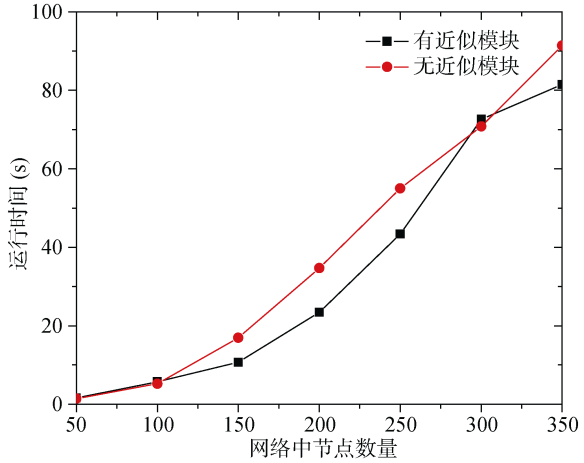

图 10 B=4 时有无近似模块对比图

Figure 10 Comparison of Double Oracle with or without Better Oracle when B=4

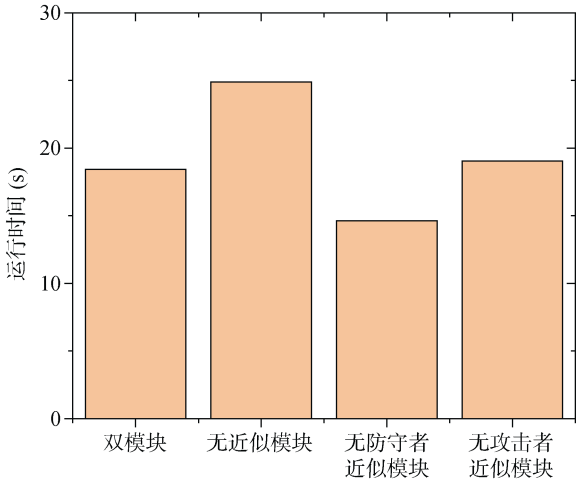

图 11 不同组合下双模块运算时间图

Figure 11 The runtime of Double Oracle with different combinations

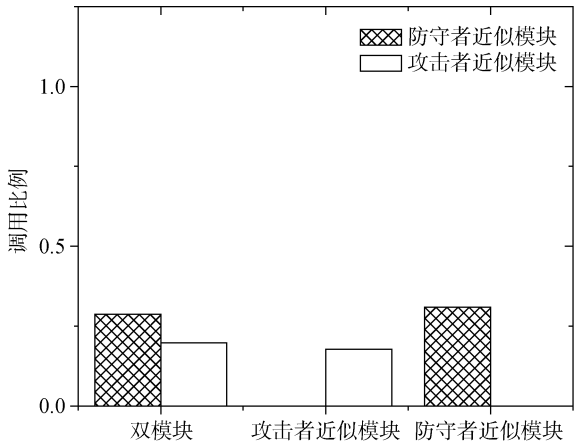

图 12 各组合下近似模块调用比例图

Figure 12 The called rate of Better Oracle with different combinations

此外, 本文还将双模块算法在平均度不同的无标度网络下进行测试, 结果分别如图 13~图 15 所示。

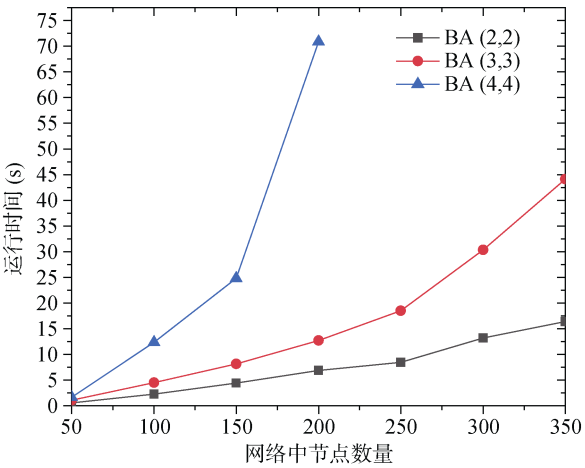

图 13 *B*=2

Figure 13 The runtime of Double Oracle in different average degree when *B*=2

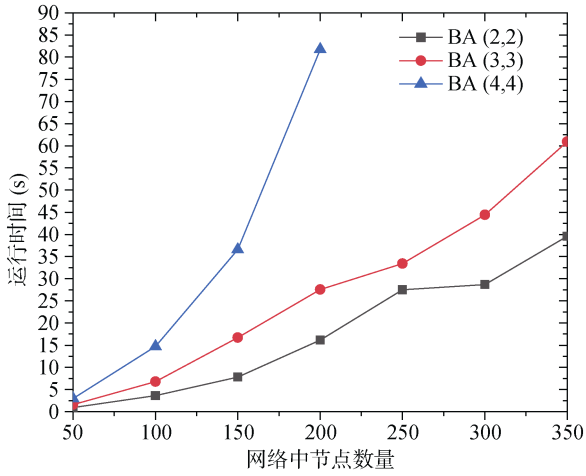

图 14 *B*=3

Figure 14 The runtime of Double Oracle in different average degree when *B*=3

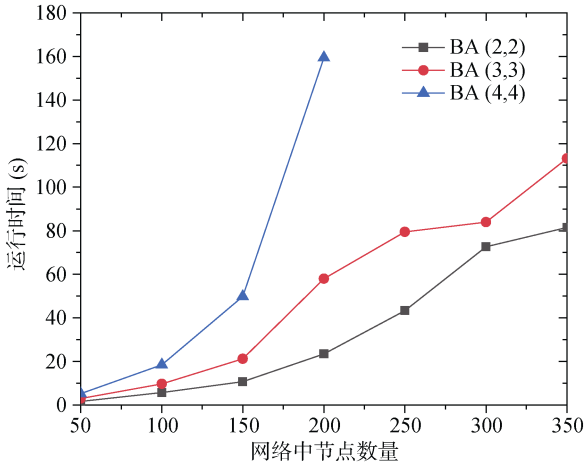

图 15 *B*=4

Figure 15 The runtime of Double Oracle in different average degree when *B*=4

从结果可以看到, BA(3,3)比 BA(2,2)网络要密集很多, 所以运行时间要增加不少, 但是在 BA(4,4)时, 网络变得更加稠密, 此时攻击者可以选择的攻击路径急剧增加, 策略数量都非常庞大, 所以即使是双模块算法, 运算时间有了明显的增多。加上实验中有随机性, 某些情况下稠密的网络计算时间会非常长, 因此在图 14 和 15 中并没有给出计算结果。

接着固定预算值 *B*=3, 在不同网络规模下对上文提到的四种不同模块组合情况的攻击者收益做了分析, 由于是零和博弈, 所以防守者的收益为攻击者收益的相反数。结果如图 16 所示。

通过对比发现, 每种情况下攻击者的收益随网络规模的变化趋势相同, 彼此相差并不大。

为了说明防守者的安防预算对攻击者收益的影响, 本文在网络规模为 150 个点, BA(2,2)的情况下, 比较了不同预算下攻击者的收益变化, 如图 17 所示。

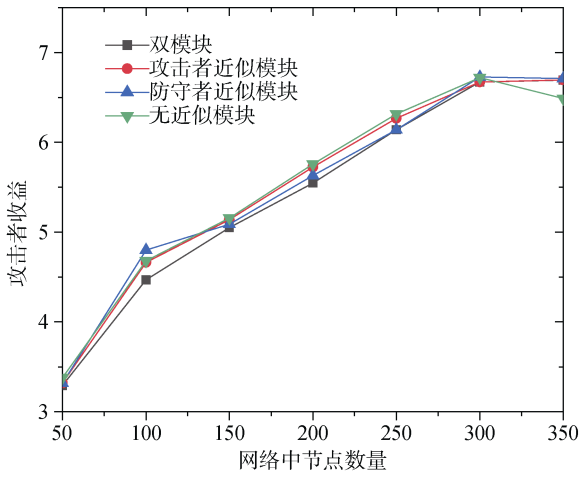

图 16 各个组合下攻击者收益对比图

Figure 16 Comparison of the utility of attacker with different combinations

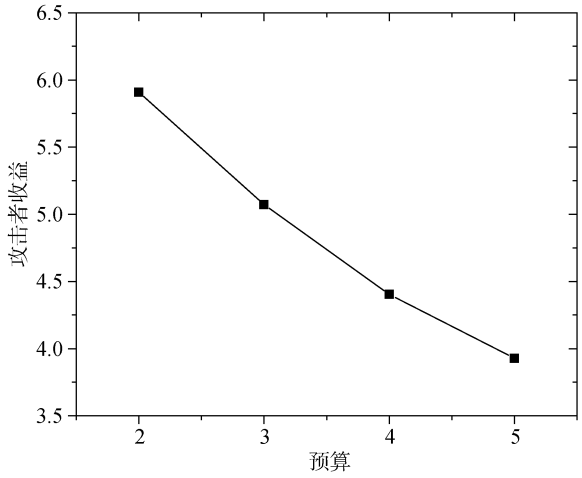

图 17 不同预算下攻击者收益对比图

Figure 17 Comparison of the utility of attacker with different budget

很明显，提高预算可以线性地降低攻击者收益，但是一方面，提高预算意味着需要更多的运算时间，另一方面，对于防守者来说，不能无限制的提高预算，所以需要在成本和收益之间做一个比较的权衡。

为了测试提出双模块算法的解质量，本文设计了两组启发式的对比方案。一是随机选择双方策略。二是防守者根据与服务相关联的节点度的大小选择要保护的服务，然后攻击者根据防守者的纯策略，做出自己的最佳响应。分别将这两种情况下的攻击者收益和双模块算法得到的收益进行对比，结果分别如图 18~图 20 所示。

可以看出，当防守者的策略为随机选择和度优先时，攻击者的收益是比较高的，而最终的优化目标是使攻击者收益尽可能的小，而双模块算法计算的是博弈的最优解，可以得到比其他两种启发式策略更少的攻击者收益，因此双模块算法能够保证很高的解质量。

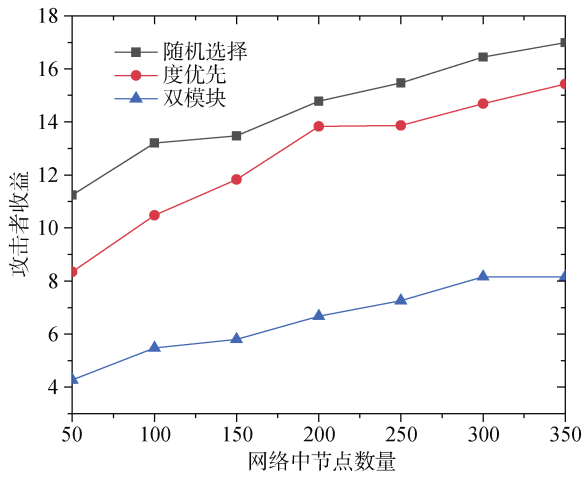

图 18 *B*=2

Figure 18 Comparison of solution quality when *B*=2

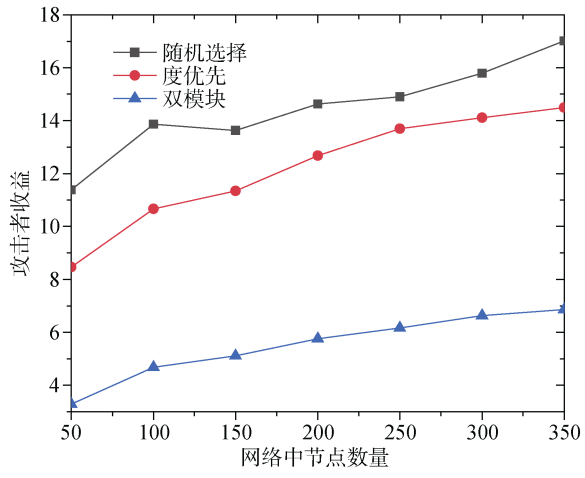

图 19 *B*=3

Figure 19 Comparison of solution quality when *B*=3

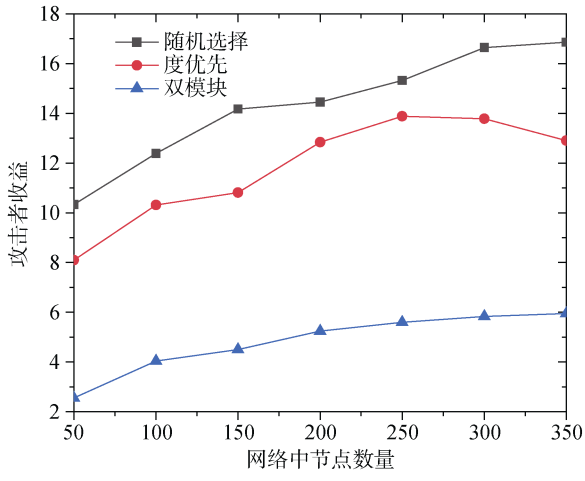

图 20 *B*=4

Figure 20 Comparison of solution quality when *B*=4

6 结论

本文对传统的攻击图进行了适当变形, 考虑在实际攻击场景中漏洞之间的利用关系, 构建了漏洞依赖图。并以CVSS漏洞脆弱性评分的开放性体系框架为数据来源, 建立了攻防双方的Stackelberg博弈模型, 对双方的策略和收益进行了研究, 并用数学规划给出了纳什均衡的求解方法。同时考虑到网络规模变大时双方策略空间指数级增加的特点, 引入了解决这类问题的典型算法: 双模块算法。当网络规模很大时, 可以在短时间内计算出防守者和攻击者的最优策略。

本文所使用的实验数据为模拟的无标度网络, 但是实际的攻击图并不一定满足无标度网络的特性, 因此需要搭建不同规模的网络模拟攻防实验环境, 验证双模块算法在真实场景下的合理性和有效性。本文中存在一些非常理性的假设, 如博弈过程中假设攻击者和防守者都是完全理性的, 网络的信息对攻防双方都是一致的, 但这在真实场景下几乎不太可能发生, 因此未来需要考虑到这些因素对双方策略和收益的影响, 利用人类行为建模等手段, 综合各方面因素进行分析。此外, 需要根据博弈均衡的求解结果, 制定切实可行的防御方案, 最大程度地降低攻击所带来的损失。所以真实的实验环境、非理性的博弈模型是未来工作研究的重点。

参考文献

[1] National Internet Emergency Center, "Report on China Internet network security in 2016," Beijing: Posts and Telecommunications Press, pp. 15-89, 2017.
(国家计算机网络应急技术处理协调中心, "2016 年中国互联网网络安全报告", *北京: 人民邮电出版社*, 2017: 15-89.)

[2] X. Liang, and Y. Xiao, "Game Theory for Network Security," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1, pp. 472-486, 2013.

[3] V. Viduto, W. Huang, and C. Maple, "Toward optimal multi-objective models of network security: Survey," *IEEE International Conference on Automation and Computing (ICAC'11)*, pp. 6-11, 2011.

[4] P. Mell, K. Scarfone, and S. Romanosky, "Common vulnerability scoring system", *IEEE Security and Privacy*, vol. 4, no. 6, pp. 85-89, 2006.

[5] J. Beale, H. Meer, and R. Temmingh, "Nessus Network Auditing," *Rockland: Syngress Publisher*, 1998.

[6] "X-scan", https://x-scan.apponic.com/, 2018.

[7] P. Mell, T. Bergeron, and D. Henning, "Creating a patch and vulnerability management program," *NIST Special Publication*, vol. 800, pp. 40, 2005.

[8] C. Philips, and L.P. Swiler, "A graph-based system for network-vulnerability analysis," *ACM The 1998 Workshop on New Security Paradigms*, pp. 71-79, 1998.

[9] I. Kotenko, and M. Stepashkin, "Attack graph based evaluation of network security," *IFIP International Conference on Communications and Multimedia Security (CMS'06)*, pp. 216-227, 2006.

[10] C.R. Ramakrishnan, and R. Sekar, "Model-based analysis of configuration vulnerabilities 1," *Journal of Computer Security*, vol. 10, no. 1-2, pp. 189-209, 2002.

[11] R.W. Ritchey, and P. Ammann, "Using model checking to analyze network vulnerabilities", *IEEE Security and Privacy*, pp. 156-165, 2000.

[12] Ye Z. W., Y.B. Guo, and C.D. Wang, "Survey on application of attack graph technology," *Journal on Communications*, vol. 38. no. 11, pp. 121-132, 2017.
叶子维, 郭渊博, 王宸东, "攻击图技术应用研究综述", *通信学报*, 2017, 38(11): 121-132.

[13] F. Chen, H.D. Mao, and W.M Zhang, "Survey of Attack Graph Technique," *Computer Science*, vol. 38, no. 11, pp. 12-18, 2011.
陈锋, 毛捍东, 张维明, "攻击图技术研究进展", *计算机科学*, 2011, 38(11): 12-18.

[14] A. Kashyap, T. Basar, and R. Srikant, "Correlated jamming on MIMO Gaussian fading channels," *IEEE Transactions on Information Theory*, vol. 50, no. 9, pp. 2119-2123, 2004.

[15] Q. Zhu, H. Li, and Z. Han, "A stochastic game model for jamming in multi-channel cognitive radio systems," *IEEE International Conference on Communications (ICC'10)*, pp. 1-6, 2010.

[16] Z. Han, N. Marina, and M. Debbah, "Physical layer security game: How to date a girl with her boyfriend on the same table," *International Conference on Game Theory for Networks (GameNets'09)*, pp. 287-294, 2009.

[17] R. Dewri, I. Ray, and N. Poolsappasit, "Optimal security hardening on attack tree models of networks: a cost-benefit analysis," *International Journal of Information Security*, vol. 11, no. 3, pp. 167-188, 2012.

[18] E. Serra, S. Jajodia, and A. Pugliese, "Pareto-Optimal Adversarial Defense of Enterprise Systems," *ACM Transactions on Information and System Security*, vol. 17, no. 3, pp. 1-39, 2015.

[19] M. Jain, D. Korzhyk, and O. Vaněk, "A double oracle algorithm for zero-sum security games on graphs," *International Conference on Autonomous Agents and Multiagent Systems. International Foundation for Autonomous Agents and Multiagent Systems (AAMAS'11)*, pp. 327-334, 2011.

[20] M. Jain, V. Conitzer, and M. Tambe, "Security scheduling for real-world networks," *International conference on Autonomous agents and multi-agent systems. International Foundation for Autonomous Agents and Multiagent Systems (AAMAS'13)*, pp. 215-222, 2013.

[21] Q. Guo, B. An, and Y. Zick, "Optimal interdiction of illegal network flow," *International Joint Conference on Artificial Intelligence (AAAI'16)*, pp. 2507-2513, 2016.

[22] Z. Wang, Y. Yin, B. An, "Computing Optimal Monitoring Strategy for Detecting Terrorist Plots," *International Joint Conference on Artificial Intelligence (AAAI'16)*, vol. 16, pp. 637-643, 2016.

[23] J.C. Ma, and J.Y. Sun, "A vulnerability assessing method based on vulnerability dependence graph," *Journal of Dalian Maritime University*, vol. 36, no. 4, pp. 92-95, 2010.
马俊春, 孙继银, 王勇军, "一种基于脆弱点依赖图的脆弱性评估方法", *大连海事大学学报*, 2010, 36(4): 92-95.

[24] A.L. Barabási, and R. Albert, "Emergence of scaling in random net-

works," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[25] "NIST, Nvd data feeds", https://nvd.nist.gov/, 2018.

王震 于 2016 年在大连理工大学软件工程专业获得博士学位。现为中国科学院信息工程研究所博士后, 杭州电子科技大学网络空间安全学院副研究员。研究领域为网络空间安全、博弈论。研究兴趣包括: 网络安全度量、攻防博弈、博弈优化。Email: wangzhen@hdu.edu.cn

吴铤 于 2002 年获山东大学理学博士学位。2002 年进入杭州电子科技大学计算机学院任教, 2016 年任杭州电子科技大学网络空间安全学院执行院长。主要从事信息安全与保密研究。Email: wuting@hdu.edu.cn

王竹 于 2009 年在中国科学院获得信息安全专业博士学位。现任中国科学院信息工程研究所高级工程师。研究领域为信息安全。研究兴趣包括: 安全协议、人工智能。Email: wangzhu@iie.ac.cn

段晨健 于 2013 年在山东农业大学计算机专业获得学士学位。现在杭州电子科技大学信息安全专业攻读硕士研究生。研究领域为网络安全度量。研究兴趣包括: 网络博弈、网络安全。Email: duanchenjian1018@gmail.com

郭云川 于 2011 年在中国科学院研究生院获得信息安全专业博士学位。现任中国科学院信息工程研究所副研究员。研究领域为访问控制。研究兴趣包括: 物联网安全、形式化方法。Email: guoyunchuan@iie.ac.cn

李凤华 于 2009 年在西安电子科技大学计算机系统结构专业获得博士学位。现任中国科学院信息工程研究所研究员、博士生导师。研究领域为网络与系统安全、信息保护、隐私计算。Email: lifenghua@iie.ac.cn