队列遍历算法的加权策略在蠕虫扩散中的应用

周佳骏[1,3],汪婷婷[2,3],韦 刚[3]
(1.安庆师范学院,安徽 安庆 246003; 2.安阳工学院,河南 安阳 455000;
3.广西师范大学,广西 桂林 541004)

摘 要:针对一般蠕虫不能感知目标环境及其改变,从而不能选择有效的扩散策略问题,采用加权树建模方法,给出一种蠕虫自动扩散模型,描述蠕虫"智能"扩散的本质特征和执行过程。基于队列遍历提出一种加权策略算法。分析和仿真实验结果表明,该模型具有较高的扩散效率,能够灵活描述和实现较为普遍的扩散。

关键词:加权策略;队列遍历;算法;蠕虫;扩散
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2009)05-0166-04

Application for Weighted Strategy of Queue-Traversal Algorithm in Diffusion of Worm

ZHOU Jia-jun[1,3], WANG Ting-ting[2,3], WEI Gang[3]
(1. Anqing Teachers College, Anqing 246003, China;
2. Anyang Institute of Technology, Anyang 455000, China;
3. Guangxi Normal University, Guilin 541004, China)

Abstract: To deal with the serious problem that worms can usually not select effective diffusion strategy as target environment changed. Adopt the method of weighted tree, give an auto diffusion model and describe the essence character and execution process of wisdom diffusion of worm. Algorithm of weighted strategy has been given based on queue-traversal. Analysis and simulation experiment results illustrate that the new diffusion model of worm has better efficiency, and it is able to describe and achieve more prevalent diffusion process flexibly.

Key words: weighted strategy; queue-traversal; algorithm; worm; diffusion

0 引 言

网络环境下,蠕虫扩散活动的发生频率高,潜伏性强,覆盖面广,造成的损失也更加巨大,因此研究蠕虫扩散行为尤为重要。由于相同或相近网段内的主机存在相似性特点(如具有相同的登录口令、存在相同的操作系统或应用软件漏洞等),如果蠕虫在扩散过程中既能具备"学习"和"记忆"功能,又能动态调整扩散策略、适应扩散环境的变化,就能避免无效扩散,提高扩散效率。文中在研究蠕虫扩散过程中的特定事件之间关系的基础上,提出一种蠕虫自动扩散模型,给出基于队列遍历的加权扩散策略算法,并对其进行分析和仿真实验。

收稿日期:2008-08-10
基金项目:国家"973"重点资助项目(2002CB312105)
作者简介:周佳骏(1974-),男,安徽安庆人,讲师,硕士,研究方向为计算机网络与信息安全。

1 加权策略树

蠕虫扩散是由一些基本的离散事件组合而成的,文中在T. Tidwell提出攻击树的概念基础上[1,2],提出扩散策略树的概念来表示扩散过程中离散事件之间的关联;并对这种策略树进行加权处理,从而能够较好地描述、分析蠕虫自动扩散行为。

1.1 基本概念

策略树的根结点表示扩散的最终目标(根事件),子结点表示其中间步骤(中间事件),任一子结点都可能导致父结点的产生。叶结点表示为了完成每个步骤需要实现的方法(基本事件)。在树型结构中,结点之间的关系可能是"与"和"或"关系。策略树由这些"与"和"或"关系组合而成。加入新的"或"关系会产生新的扩散路径,加入新的"与"关系则使已存在的扩散路径延长。

1.2 加权策略树分析蠕虫扩散行为

采用策略树建模方法,描述扩散的企图、特征和步

骤执行过程及其关系。结合文献[3,4],对具体扩散对应的基本功能进行进一步抽象,如图 1 所示。

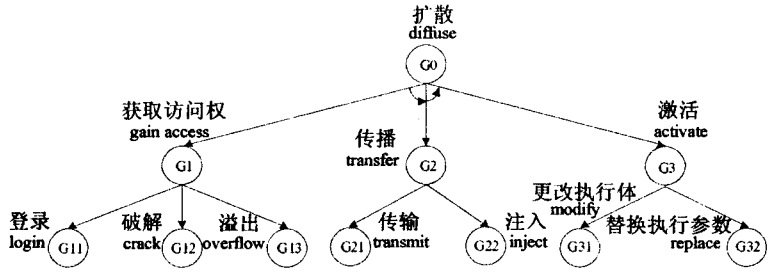

图 1　蠕虫扩散策略树

其中,根结点表示目标,子结点表示完成根结点目标之前要完成的扩散行为或子目标。弧线连接表示子结点之间是 AND 关系,否则为 OR 关系。

在文中,采用对策略树加权方法,在扩散过程中动态调整权值的大小,这样就可以保证扩散过程的每一步都是优化的,从而提高扩散成功的概率。

2　自动扩散模型

基于上述思想,提出一种自动扩散模型,该模型可以模拟"学习"和"记忆"每次扩散的经验结果,动态调整树结点的权值,自动形成优化的扩散策略,实现蠕虫扩散过程的自动化,具有通用性、灵活性和可扩展性。

2.1　结构框架

经过前面的分析,初步形成了面向安全仿真的计算机蠕虫自动扩散模型的设计思路:采用加权策略树建模方法,建立策略树模型,该模型调用扩散仿真器中的接口函数,由扩散仿真器调用 GTNets 底层平台函数实现扩散。该系统的结构框架如图 2 所示。

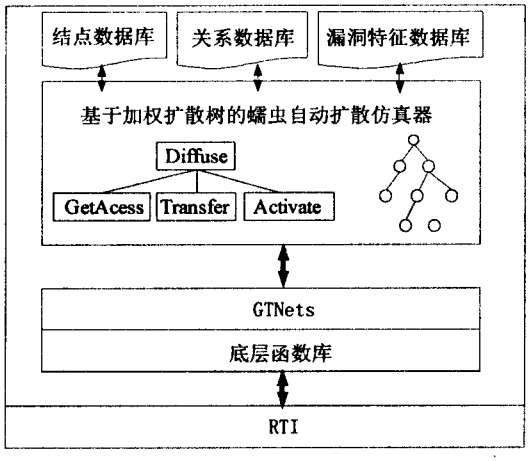

图 2　自动扩散模型结构框架

由此可知,该模型由自动扩散仿真器、各种数据库(如结点描述数据库、结点关系数据库、漏洞特征数据库等)、底层函数库构成。

在功能实现上,采用插件形式(如口令破解插件、溢出获权插件、程序和数据传输插件、注入插件等)完成,具有通用和灵活的特点,易更新。

2.2　结点描述与结点关系数据库

根据实际需求,结点描述数据库由二维表构成,如表 1 所示。其中第一列是结点编号,用于表示结点之间的层次。下标为一位数字(除 G_0)表示根结点以下第一层,以下类推。第二列是描述语言,用来描述结点状态或功能。当有新的结点加入,可以按照其编号下标插入到表中相应位置。

表 1　结点描述表

结点	描述
G_0	根结点,表示扩散最终效果
G_1	获取目标主机的控制权限
G_2	传送蠕虫代码和数据到目标主机
G_3	在目标主机上激活自身副本
G_{11}	利用登录口令获取控制权
G_{12}	破解密码获取控制权
G_{13}	利用缓冲区溢出漏洞获取控制权
G_{21}	通过传输程序(tftp,ftp 等)进行传播
G_{22}	通过注入法进行传播
G_{31}	更改执行体激活副本
G_{32}	替换执行参数激活副本
…	

结点关系数据库也是由二维表构成,如表 2 所示。

表 2　结点关系表

	G_0	G_1	G_2	G_3	G_{11}	G_{12}	G_{13}	…
G_0	0	∩	∩	∩	0	0	0	
G_1	∩	0	∩	∩	0	0	0	
G_2	∩	∩	0	∩	0	0	0	
G_3	∩	∩	∩	0	0	0	0	
G_{11}	0	0	0	0	0	∪	∪	
G_{12}	0	0	0	0	∪	0	∪	
G_{13}	0	0	0	0	∪	∪	0	
…								

其中"∩"表示两个结点之间是"与"关系,"∪"表示两个结点之间是"或"关系。同父亲的结点之间才会具有以上两种关系之一。

2.3　漏洞数据库

漏洞库的来源主要是 CVE 和 Bugtraq 所收集整理的漏洞信息库[5]。为了快速、有效地进行网络漏洞扫描[6~8],从而获取管理员权限,可将漏洞库中的漏洞数据分类,如 FTP 漏洞、CGI 脚本漏洞、后门漏洞、Windows 系统漏洞等类型。每个类型形成一个类函数,其中存放具体的功能,以插件形式被调用。

2.4　工作流程

蠕虫自动扩散模型工作流程的核心步骤如下:

①在结点描述数据库中选择相应结点,构造原始

的完整策略树,并赋予每个结点初始的权值。

②根据规则,查询结点关系数据库提供的结点之间的"与""或"关系,选择权值最大的结点构成一棵新的优化扩散子树(权值大表示由该路径成功实现扩散的概率大)。

③执行该策略子树的各个结点任务,若完成,则该结点的权值加 1,并判断是否为最终结点;否则减 1。

④若不是最终结点,跳转②;若是最终结点,判断扩散任务是否完成,若未完成,寻找下一个可达目标主机,跳转②;若完成,结束程序。

2.5 算法设计

根据前面描述的自动扩散模型,从策略树存储结构的定义、策略树的初始化、扩散策略的选择三个方面对扩散算法进行设计。

2.5.1 存储结构的定义

我们知道,采用 father 链接可以很容易唯一确定一棵树的结构,便于实现树的初始化。但是 father 链接结构存储的树不容易实现遍历[9],所以笔者改进了 father 链接结构,如下所示:

|father|node|child$_1$|…|child$n$ |

指针 father 用于树的初始化,指针 child 用于树的遍历。较好实现了树的构造和遍历的统一。结点类 TreeNode 声明如下:

```
Template <class DT>
Classe TreeNode
{
  friend class Tree;
  private:
  DT data;
  TreeNode <DT> * father, * child;
  // 构造函数
  TreeNode (DT value = 0, TreeNode <DT> * L = NULL,
TreeNode <DT> * R = NULL )
  :node(value),father(L),child(R)
  {}
}
```

2.5.2 策略树的构造以及权值初始化

策略树的构造和权值初始化过程如下:

①在以 G_0 为根的树中,搜索结点描述数据库中结点 G_i,(i = 1,2,…,m; m 是结点数),根据结点存储结构找到其父结点;

②若找到,则将当前指针指向该结点的父结点,赋予该结点初始的权值 w;

③判断数据库中结点是否为空,非空则重复步骤①;否则,初始化树成功。

算法的 ADL 描述:

```
DiffusionTree (t) // 指针 t 指向新建立的树根
L1:[创建一个辅助队列]
  Creat(Q);
L2:[根结点入队]
  Q←t;. .
L3:[利用队列进行层次遍历]
// 当队列 Q 非空,结点出队
  WHILE NOT (IsEmpty(Q)) DO
  (
  p←Q;
  WHILE p< >NULL DO
// 根据 p 的左指针 father 找到其父结点,并且插入到父结点下层
  (
  Insert( p);
// 赋给 p 权值 w
  Weighted(p,w);
// 若 p 有孩子,则 p 所有的孩子入队列
  WHILE child(p)< >NULL DO
  (
    q→firstchild(p);
    Q←q;
    q→q. nextchild;
    )
  )
)
```

2.5.3 扩散策略的选择

同样采用按层次遍历的算法实现策略的选择。利用两个队列 P、Q 作为辅助空间,根结点设为当前结点,其步骤如下。将

① 所有结点根据结点描述表依次压入队列 P,P 的第一个结点设为当前结点。

②若是根结点,从队列 P 中弹出,压入队列 Q。

③根据队列 P 当前结点 father 指针寻找其父结点,再根据其父结点的 child 指针找到其所有兄弟结点,查询结点关系表,若存在与之相 OR 关系并且权值大于它的兄弟结点,当前结点从队列 P 弹出,重复③,直到 P 为空;否则,当前结点从队列 P 弹出,并且压入队列 Q,重复③,直到 P 为空。

④将队列 Q 中的结点,重新构造一棵新树,此时新树是原始树的子树,并且具有完成整个扩散任务的能力,相对其它子树而言,具有权值最大的特征,是一棵优化的加权策略树。

3 验 证

选取某校园两个实验计算机房的网络拓扑结构作为实验拓扑。网络中易感主机 N = 100。在仿真实验

中,硬件采用 Intel X86(Pentium Ⅳ 2.7G,1GB 内存)计算机作为平台。软件采用 Red Hat 9.0 操作系统,GTNets 仿真平台,底层 RTI 实现采用 libSynk,gc++编译环境。实际拓扑结构如图 3 所示。

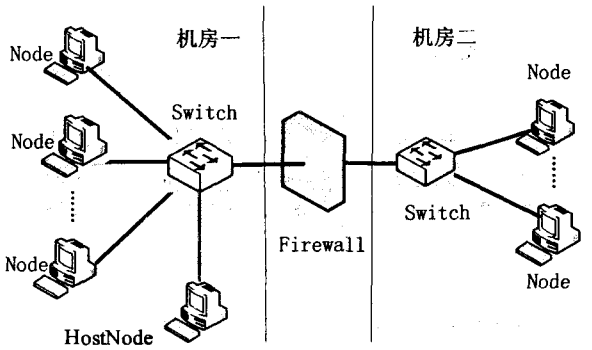

图 3　蠕虫自动扩散的实验拓扑

拓扑中具有 HostNode(扩散发起者)结点,Firewall 结点和 Switch 结点。

实验假设如下:①未进行扩散时,实验中的网络物理链路和通信链路正常工作;②每次扩散成功的状态都能被捕获到。

实验时仿真网络参数配置如下:仿真执行时间为 10s;基本带宽 100Mb。

文中研究中针对采用加权算法和没有采用加权算法的两种策略各进行了 20 次扩散仿真实验,实验结果平均值如图 4 所示。

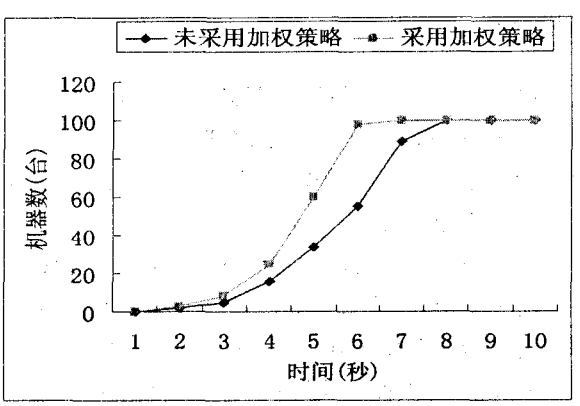

图 4　两种蠕虫扩散算法的效率对比

实验结果表明,两组数据之间存在着较大差异。①没有采用加权算法的蠕虫在第 5s 仅感染了 37 台主机;而此时采用加权算法的蠕虫已经感染了 61 台主机,5s 内扩散的速度效率是前者的 1.65 倍。②没有采用加权算法的蠕虫在第 8s 感染所有主机;而采用加权算法后在第 6s 就完成所有任务,完全扩散的时间效率比前者提高 25%。

由此可见,采用加权策略算法的蠕虫,其扩散速度、扩散时间都得到较大改善。

4　结束语

通过引入结点之间的顺序关系以及权值的概念对扩散树进行改进,较好解决了蠕虫扩散策略和环境之间如何相适应的问题。在此基础之上,设计一种基于加权策略树的自动扩散模型,并给出相应的队列遍历算法来实现。

仿真结果说明该模型既能使蠕虫在相同或相近网段中快速扩散,又能使蠕虫有较好的环境适应能力,能根据环境的变化动态调整扩散策略。

参考文献:

[1] Tidwell T, Larson R, Fitch K, et al. Modeling Internet Attacks[C]//Proceedings of the 2001 IEEE Workshop on Information Assurance and Security. Oakland, CA:[s.n.],2001.

[2] 周　伟,王丽娜,张焕国.一种基于攻击树的网络攻击系统[J].计算机工程与应用,2006(24):38-40.

[3] Kienzle D M, Elder M C. Recent Worms: A Survey and Trends[C]//Proc. of the ACM CCS Workshop on Rapid Malcode. Washington D.C.:[s.n.], 2003.

[4] Gebhart G. Worm Propagation and Countermeasures[R].[s.l.]:SANS Institute, 2004.

[5] Bishop M, Bailey A. Critical Analysis of Vulnerability Taxonomies[D].Davis:Department of Computer Science, University of California, 1996.

[6] 黄家林,姚景周,周　婷.网络扫描原理的研究[J].计算机技术与发展,2007,17(6):147-150.

[7] 丁常福,方　敏,徐　亮.端口扫描技术及防御分析[J].微机发展(现更名:计算机技术与发展),2003,13(6):7-12.

[8] 陈　峰,罗养霞,陈晓江.网络攻击技术研究进展[J].西北大学学报:自然科学版, 2007,37(2):208-211.

[9] Sedgewick R, Flajolet.算法分析导论[M].冯舜玺,等译.北京:机械工业出版社,2006.

(上接第 165 页)

[J].Communications of the ACM, 1997,40(10):88-96.

[6] Foster I,Kesselman C,Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations[J]. International Journal of Supercomputer Applications, 2001,15(3):1-10.

[7] 王煦法,张显俊,曹先彬,等.一种基于免疫原理的遗传算法[J].小型微型计算机系统, 1999,20(2):117-120.

[8] Forrest S, Perelson A S, Allen L, et al. Self-Noneself Discrimination in a Computer[J]. Proceedings of IEEE Symposium on Research in Security and Privacy. Oakland:[s.n.], 1994.