对应用软件进行安全测试的对手模式及其应用*)

贺 红[1,2] 徐宝文[1] 袁胜忠[3]

(东南大学计算机科学与工程系 南京 210096)[1] (山东大学威海分校信息工程学院 威海 264209)[2]

(山东大学威海分校现代教育技术部 威海 264209)[3]

摘 要 基于网络应用软件对安全性需求的日益加强,提出将安全性测试增加到软件功能性测试之中的观点,并且给出对应用软件进行安全测试的对手模式,以及为快速高效实施对手模式进行软件分解的方法和对手模式的应用范围。基于对手模式给出一种应用软件系统的抗攻击定理,该理论从安全性角度导出了一种实用的估算网络应用软件寿命的方法。

关键词 安全性测试,攻击,对手模式,随机算法,软件寿命

The Adversary Pattern and Attack-proof Theorem in Application Software's Security Testing

HE Hong[1,2] XU Bao-Wen[1] YUAN Sheng-Zhong[3]

(Department of Computer Science and Technology, Southeast University, Nanjing 210096)[1]

(School of Information Engineering, Shandong University at Weihai, Weihai 264209)[2]

(Department of Modern Education Technology, Shandong University at Weihai, Weihai 264209)[3]

Abstract The security of application software is more important in network days. A kind of adversary pattern to find out security bugs in a software is raised which can be used in all testing steps. We discuss the software decomposition ways and application of the adversary pattern, put forward an attack-proof theorem. It shows a method of computing average number of attack times before a software system having concealed bugs becomes invalid. We also give a new way to estimate the software's lifetime under various randomized attacks.

Keywords Security testing, Attack, Adversary pattern, Randomized algorithm, Software's lifetime

1 前言

人们相信在网络级使用防火墙、入侵检测系统等能够解决信息系统的安全问题,这种信任已经催生了一个产业。然而即使某个组织的网络拥有世界上最好的防火墙,只要允许人们穿过防火墙来访问应用软件,并且这个应用软件的代码又有漏洞可以远程利用,那么防火墙就不起任何作用了。密码技术也是一样。根据美国国家研究委员会信息系统可信任委员会发表的"Trust in Cyberspace"报告(National Academy Press,1999),截止到 1998 年,CERT 安全警告提到的问题中有 85%都是无法用密码技术来保护的[3]。计世咨讯 2004 年的一份调查报告显示,中国内地由于系统漏洞问题所造成的经济损失在 2004 年达 400 亿元人民币。2005 年 12 月 13 日,国内领先的反病毒软件厂商江民科技发布中国大陆地区计算机病毒疫情报告显示,2005 年虽然没有特别大规模的病毒爆发,但病毒数量急剧上升,而且病毒变种速度快,传播范围小,更加隐蔽,目的性更强。所以,我们不能认为靠密码技术和防火墙可以解决所有的安全问题。面对层出不穷的恶意攻击和病毒,人们把防火墙越砌越高、入侵检测越做越复杂、病毒库越做越大、误报率也随之增多,维护与管理变得更加复杂和难以实施,信息系统的使用效率大大降低。这种形势使我们认识到:我们走入了安全误区。要保障信息系统的安全,我们应该从盲目的事件应对方式转向对系统主动进行软件和硬件整体安全规划的方式,其中,系统安全性的根本就在于软件的安全性。

在网络时代,软件的安全性比以往任何时候都更加重要。有时可以采取一些措施保护软件薄弱点不受攻击,有时可以设置防火墙来阻止某些类型的数据分组或邮件,或者只允许来自可信任源的连接,有时入侵检测系统可以设置在某个漏洞被人利用时拉响警报,有时托管的安全监视服务能够捕获正在进行中的漏洞挖掘从而实时地阻止入侵。在所有这些情况下,问题都起源于软件。如果软件的安全性不是如此脆弱,我们就用不着花这么多时间、金钱和精力在网络安全上了。支持 Internet 功能的应用软件,包括那些企业内部开发的,都是目前安全风险最大的地方。

软件安全风险的两种最基本类型是代码实现上的错误(implementation error)和架构上的弱点(architectural weakness)。安全测试定义为:验证系统是否具有一定程度的保护机制防止系统或数据受到非法侵入。随着软件内在复杂性的不断提高,借助于成熟的测试经验和强大的自动化测试工具,代码实现上的错误一般能够被发现。但是在架构层自动分析软件安全的技术远远落后于实现工具,所以发现软件架构上的弱点仍然是一项很困难的工作,应该作为软件测试的重要组成部分由专业人员来完成。

使用防火墙,在网络层进行渗透测试亦非解决之道。将几百个代表已知漏洞的自动化脚本简单地合起来使用的方法

*)国家自然科学基金资助项目(60373066)。贺 红 博士后,副教授,主要从事算法分析与设计,网络安全测试技术的研究。徐宝文 教授,博导,主要研究方向为软件工程与理论。袁胜忠 软件工程师,主要从事网络软件开发与安全测试工作。

使安全测试人员只能了解已知的、重复出现的缺陷，新的攻击类型，以及攻击新变种不断出现，它们在被利用之前安全测试人员是不知道的。因此，根据查找已知问题来避免明显陷阱的检查清单法已经不够了。（当然，关于一些众所周知的风险比如缓冲区溢出、竞争条件等等的知识仍然能够帮助我们编写更好的代码。）安全测试通常意味着要模仿攻击者通过网络对整个系统进行攻击。这一点从各种层出不穷的渗透测试工具，如 SATAN、SAINT 和 Retina 等就能明显地看出来。

本文第 2 节系统地给出一种能应用于测试的各个阶段的发现软件中安全 bug 的对手模式，描述了为快速高效地实施对手模式进行软件分解的方法，讨论对手模式的应用范围，并且基于对手模式给出企业门户系统的抗攻击定理，它能估算出一个结构松散耦合的存在安全 bug 的企业门户系统平均经过多少次随机性攻击会完全失效。使用对手模式进行安全测试取得的数据不仅可以用来构建威胁模型[1]，而且能用于估算软件寿命的实验与分析。我们从安全性角度导出了一种实用的估算软件寿命的方法。最后是结论。

在后面的论述中，软件寿命是指某个应用软件实际的生存时间，或是从某特定时间算起的能正常工作的存活时间。当一个软件不能正常工作时，我们称之为“发生故障”或“失效”了。

2 对手模式——发现软件安全 bug 的随机模式

2.1 对手模式的设计思想

对手模式（adversary pattern）是来自随机算法的一种发现软件安全 bug 的模式。所谓随机算法，就是在执行过程中要做出随机选择的算法，它有两个优势：简单和快速。我们使用随机算法，目的是借助其优势处理软件安全测试和估算软件寿命这两个含有太多不确定性的复杂问题。不失一般性，我们规定：(1)来自网络攻击者的对应用软件的攻击是随机的，攻击的随机性是指任何两次攻击事件之间是互相独立的。(2) 攻击就是软件安全的“对手（adversary）”。本文只研究常见的攻击类型，即利用对方信息系统自身存在的安全 Bug，通过使用网络命令和专用软件进入对方系统，攻击对方的应用软件以达到破坏对方网络系统的目的。

对手模式可以发现软件安全 bug 的理论基础是：一个有安全缺陷的系统可能“挫败”一部分确定性攻击，但它无法总是“挫败”一个随机选取的攻击[2]。

对手模式的设计思想来源于有经验的安全测试人员的直觉：欲知应用软件的安全性能，让攻击者从网上攻击一下试试便知道。对于一个确定的被测试系统，当对它进行绝对行为的安全性分析无意义时，可以寻找一个对手来攻击它，以使系统的安全性能表现最坏。事实上，应用软件在发布之后，可能会暴露在成千上万人同时的攻击面前。试图通过安全性设计保护系统抵抗所有类型的攻击是不切实际的[5]。但是如果安全测试者独立于开发小组，在正常的程序测试渠道之外，以攻击者的身份，对应用软件发起专注而且密集的随机攻击，就会比入侵者更早、更高效、更准确地发现应用软件中的安全 bug。

对手模式的测试成本较高，测试重点为应用软件中特别容易遭受网络攻击的部分，也可以为检验软件对某些攻击类型的防御能力设计专门的测试。

测试模式确定为对手模式后，过程主要包括：目标信息获取、攻击构造、强度控制、测试实施、测试反馈等几个模块。为保证对手模式正常实施，我们使用分布式多线程框架分别负责完成不同模块的任务。各模块说明如下：

(1) 目标信息获取模块负责自动收集测试对象的相关参数。

(2) 攻击构造模块根据对手模式思想和目标信息获取模块得到的参数构造相应的攻击。

(3) 强度控制模块可以调整测试强度，如测试用例的攻击范围、发送速度等。

(4) 测试实施模块按照规定的测试强度，将攻击构造模块生成的攻击发送给待测试软件。该处的关键是保证强度控制模块规定的攻击强度。

(5) 测试反馈模块用于收集测试结果。

对手模式不仅适用于集成测试或系统测试阶段，也同样适用于单元测试阶段。它不仅包含了检查清单法所具有的功能，而且可以按照系统需求分析的安全设计要求，有针对性地对应用软件的某些部分做特别的强化测试。另外，由于解决同一个问题存在不同的算法，不同的算法对应的对手一般是不同的。可以通过算法面临对手攻击时的表现选择算法而使系统安全性能达到最佳。

由于不可能发现软件系统所有的 Bug，所以安全测试需要某种停止标志。当系统的每一个部分都在可以接受的风险阈值内时，就能够停止风险处理了——当然，什么是可以接受的要取决于具体的业务性质。

2.2 为实施对手模式进行的应用软件分解

对手模式的主要目标，就是快速找出软件哪些区域最可能存在漏洞。为了合理分配安全测试资源，让对手模式得到高效率实施，我们将应用软件分解成多个特性，由不同测试小组分别对这些特性进行安全测试，对这些特性进行安全评分，最终提交包含所有特性测试结论的测试报告。文[4]说明如何找出各种可能发现安全 bug 的特性。

应用软件的分解就是指将应用软件的特性分解为多个可以使用对手模式的测试区域。具体分解方法可能会变化，理论上，有以下两个分解特性可遵循：

(1)区域大小的特性——一个人或者一个小组能够在预计的相对较短的时间内研究其功能并进行测试吗？

(2)这个特性是否为其中包含的大多数功能形成了自然的分区，它与应用软件其它部分之间是否有接口？

例如，一个可以播放来自网络和存储在本地或者远程计算机上文件的流媒体播放器，可以简单分解为：①从本地文件系统读入文件；②通过网络与流媒体通信；③图形用户界面；④存储收藏用户偏好数据。

大型网络应用软件可以用三种方式分解[7]：

(1)按步骤分解。某些应用软件由一系列处理构成，这些处理分布到网络上不同计算机上完成，就像工厂的流水线一样，可以按不同步骤分别分配测试资源。

(2)按功能分解。某些系统能够分解成相互基本独立的不同功能单元，可以分别测试各个功能单元。

(3)按数据分解。有许多应用的数据量极大，需要把不同数据块分配到不同的计算机上处理。可以按数据块之间耦合度大小来划分成不同测试单元。

应用软件可能有成百上千的特性，在特性之间分配测试资源的标准很多，各有利弊。根据输入的数量，或者代码行数，或者使用此特性的用户比例划分都是合理的分解标准，分解目标都是增加用户遇到 bug 的可能性。而且，并不是所有

的大规模应用软件都能轻松地被分解。安全性测试的资源是有限的。能使有限的资源发挥最大作用的分解方法就是最好的特性分解法。因为对手要在真正的攻击者之前发现安全bug,必须进行短暂、密集的测试。所以,应该把更多测试资源分配到更可能包含缺陷的构件上。

Gary McGraw & John Viega 提出了软件安全的十大原则[3],确认并强调了安全的软件系统设计和构造中最重要的目标。当然,这些原则并没有涵盖未来潜在的所有可能的新缺陷,因为没有什么原则能一劳永逸地解决软件安全问题;相反,他们提供了很好的"90、10"策略——通过遵循 10 条简单的原则,并且作巧妙的权衡,可以避免 90%的潜在问题。

2.3 对手模式的应用

2.3.1 *应用范围* 作为安全测试模式,对手模式的成本是比较高的。但是在纷繁复杂的网络环境下,它像反证法一样,能够支持开发者对应用软件的安全保证。所以,本节通过一个实例来讨论对手模式的应用范围。

实例:宝洁公司门户系统的安全测试。宝洁公司在全球 100 多个国家拥有 300 多种品牌的商品和超过 50 亿的消费者。它建立统一的企业门户系统的目的是,让企业外部用户可以和企业内部的用户一样访问企业的技术和信息资源。在网络环境下,越来越多的公司开始认识到公司门户系统是非常重要的基础结构应用系统,它可以通过整合多个应用程序,建立一个能依据应用程序的要求进行调节的有弹性的框架而形成,这也意味着对企业的基于网络的应用程序,旧版本的数据库驱动的系统以及其它以文档为中心的组件的广泛整合。整合常犯的错误之一是认为整合应用程序的范围应该足够大以适合所有的情况。而宝洁公司客观地认为技术仅仅是一个帮助它们实现理想方案的工具,所以,他们在开发和部署其集中式门户系统的过程中根据用户的喜好程度决定是否将某个应用程序整合在内,并不断添加硬件服务器以处理独立运行的门户系统单元。大多数门户系统部署是使用单一服务器模式,通过服务器对门户组件进行分布式处理,可以在任何需要的时候对系统的基础结构进行升级。

这类分布规模庞大、管理内容众多、内部结构松散耦合的应用软件系统,可以伴随系统部署的过程使用对手模式进行灵活的安全测试。

2.3.2 *门户系统的抗攻击定理* 考虑一个有 n 个独立运行的门户系统单元的软件系统,它受到随机发生的网络攻击。一次攻击对系统安全造成的攻击程度用一个数值表示,记做攻击值,由概率的意义,设攻击值的定义区间是(0,1)。假定每次攻击的大小服从分布 G,与其它的攻击值相互独立。若发生一次值为 x 的攻击,则攻击到达时,正在运行的各单元相互独立地以概率 x 在瞬时间失效。我们研究直到软件系统的全部单元都失效所必需的攻击次数 N 的分布。

定理 1 在对手模式下,要使软件系统的全部单元都失效所必需的攻击次数 N 的均值为 $E[N]=\sum_{i=1}^{n}\binom{n}{i}\frac{(-1)^{i+1}}{1-p_i}$,其中,$n$ 是软件系统的单元个数,p_j 是任意一次攻击后,指定的 j 个单元全部完好的概率。

证明:为计算 $P\{N>k\}$,以 $E_i(i=1,2,\cdots,n)$ 记部件 i 在前 k 次攻击后仍幸存完好这一事件,则

$$P\{N>k\}=P(\bigcup_{i=1}^{n}E_i)=\sum_{i=1}^{n}P(E_i)-\sum_{i<l}P(E_iE_l)+\cdots+(-1)^{n+1}P(E_1E_2\cdots E_n)$$

为了计算此概率,以 p_j 记任意一次攻击后指定的 j 个单元全部完好的概率,以攻击值为条件得

$$p_j=\int P\{\text{指定的 } j \text{ 个幸存完好}\mid\text{攻击值是 } x\}\,dG(x)=\int(1-x)^j\,dG(x)$$

因为

$$P(E_i)=p_1^k$$

$$P(E_iE_l)=p_2^k,\cdots,P(E_1E_2\cdots E_n)=p_n^k$$

所以

$$P\{N>k\}=np_1^k-\binom{n}{2}p_2^k+\binom{n}{3}p_3^k-\cdots+(-1)^{n+1}p_n^k$$

由此可以计算 N 的均值如下:

$$E[N]=\sum_{k=0}^{\infty}P\{N>k\}=\sum_{k=0}^{\infty}\sum_{i=1}^{n}\binom{n}{i}(-1)^{i+1}p_i^k=\sum_{i=1}^{n}\binom{n}{i}(-1)^{i+1}\sum_{k=0}^{\infty}p_i^k=\sum_{i=1}^{n}\binom{n}{i}\frac{(-1)^{i+1}}{1-p_i}$$

证明中使用了恒等式 $E[N]=\sum_{k=0}^{\infty}P\{N>k\}$,它对一切非负整数随机变量 N 成立[6]。

2.3.3 *用对手模式测得的数据估算软件寿命的一个实验* 从定理的结论可以看出,要求出让软件系统的全部构件都失效所必需的攻击次数 N 的均值,就需要统计计算 p_j。一般地,p_j 可以通过在一段连续时间内对网络状态和软件工作状态进行监测得到比较准确的经验数据。但是我们关心的不是软件系统的全部构件都失效所必需的攻击次数,而是软件在一般的网络攻击下能生存的大致时间或寿命。

为了计算软件寿命,我们可以使用两种方法:一种是根据经验数据应用某个指数分布模型来精确计算,在涉及寿命问题时都广泛提倡用威布尔分布[7],它是一种"无记忆性"的分布,对许多类型的寿命数据都能给出很好的描述。另一种是使用拟合生存分布的图方法进行估算。图方法既简单又有效,可用于代替数字分析,数据图还可以同时服务于若干目的,而某些目的是不能用数字方法解决的。

我们实验用的是作概率图估计软件寿命的方法。概率图要求完全的样本,我们使用的样本是用对手模式进行安全测试获得的数据。

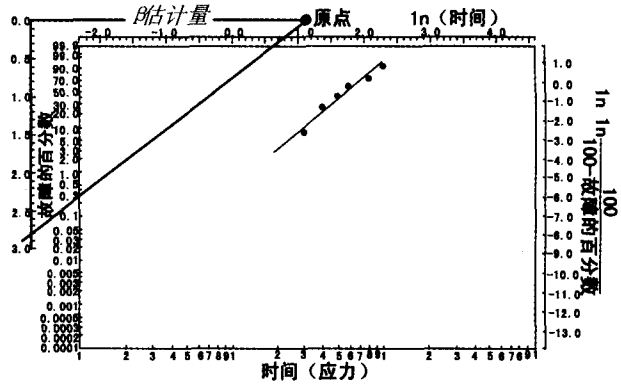

图 1 6 次对 P 进行安全测试所得的 P 寿命的威布尔概率图

对某个网络应用软件 P,先后 6 次使用对手模式进行安全测试,从某一时刻(比如 P 开始试运行)算起,每一次 P 被对手随机攻击失效后立刻记录 P 的存活时间,并且形成详细安全测试报告提交给系统测试组,由设计人员对发现的缺陷做相应改进。重复上述步骤。6 次实验测得 P 按周计的寿命:3,4,5,6,8,10。在威布尔概率纸上,做出点$(t(i),100(i-0.5)/n)$,$i=1,\cdots,6$。如图 1 所示,然后在威布尔概率图纸上

用眼睛拟合出一条直线。63.2%分位点是近似等于6.5个周，它是1/λ的估值，于是λ的估值是0.154。从顶部的原点出发画一条平行于拟合直线的直线，该直线与在左侧的辅助刻度直线的交点给出γ在图上的估值，它近似等于2.75。有了这些估值，P的寿命均值能用威布尔分布的均值公式估算出来：

$$\mu=\frac{\Gamma(1+1/\gamma)}{\lambda}$$

即$\mu=\Gamma(1.364)/0.154=0.8897/0.154=5.78$周。这里$\Gamma(\gamma)$是著名的Γ函数，定义为

$$\Gamma(\gamma)=\int_0^\infty x^{\gamma-1}e^{-x}\mathrm{d}x$$

注意，$\Gamma(x)$的值能在Abramowitz和Stegun(1964)的文章里找到。

最后，将实验环境下估算的P的寿命$\mu=5.78$周，用下式进行换算求出P的估计寿命：

$$L_P=\mu\times\frac{f_{adversary}}{f_{general}}$$

其中，$f_{adversary}$表示使用对手模式进行安全测试时对P发动攻击的频率；$f_{general}$表示一般网络状况下P遭受攻击的频率。

本次实验中，P每周遭受的测试小组发起的攻击频率为120次/周，根据网络安全状况调查报告的统计数据估计一般网络状况下P遭受攻击的频率约为5天/周×1次/天×(P受到攻击的可能性)50%=2.5次/周，可以计算出P的估计寿命年数为：

$$L_P=\mu\times\frac{f_{adversary}}{f_{general}}=5.78\times\frac{120}{2.5}=277.44(\text{周})\approx5.34(\text{年})$$

当然，在估算过程中我们用了一些网络统计结果和经验数据，上述实验用的对手是250个已知的攻击类型，对于未知的新的攻击类型无法加以测试，这些因素都可能导致最后估算出来的软件寿命数据不够精确。但是，由于应用软件的寿命本身就不是一个精确值，它不仅受软件本身抗攻击性能的影响，而且主要受到组织业务变动或机构调整的影响。上述估算结果与我们的经验数据拟合较好，可以作为软件购买组织进行该软件的投入—产出成本分析或者软件生产组织估算软件维护成本的一种新简易方法。

结论 在上述数据分析中，模型所发挥的作用应该随着讨论的问题以及针对问题所采用的方法的不同而不同，模型和有关统计方法的复杂性将随着应用的不同而有很大不同，一个应用软件很可能要受若干"失效"原因中任一个的影响，等等。所以，在涉及软件寿命的问题中，数据的产生过程经常是难于弄清楚的，这也是使问题复杂化的一个因素。也正是由于问题的复杂性，我们才不再划分各种网络攻击对应用软件寿命的不同影响，抽象到更高的层次，把所有攻击及其形形色色的新变种统统放到一个攻击环境里来进行研究。

另外，大多数安全bug的隐蔽本质使得应用软件需要特殊的、专门的安全测试。目前还没有算法(即使是探索性的)能够用来描述已知的各种攻击。新的攻击类型或变种不断被发现，人为地预测这些攻击的工作原理简直是不可能的。我们下一步的研究方向是应用机器学习的相关算法，研究从已知攻击特征预测出新攻击变种的方法，以提高软件对防范攻击的自学习能力和免疫能力。全方位的安全防护措施结合软件对防范攻击的自学习能力可能成为我们增强软件安全性的真正出路。

参考文献

1 Howard M, LeBlanc D. Writing Secure Code. Microsoft Press, 2002
2 Motwani R, Raghavan P. Randomized Algorithms. Cambridge University Press, 1995
3 McGraw G, Viega J. 软件安全十大原则. http://www.nap.edu/catalog/6161.html
4 Thompson H H, Chase S G. 红队程序安全测试. 软件研发, 2004, 2(7)
5 Whittaker J, Thompson H H. How to Break Software Security: Effective Techniques for Security Testing. Addison Wesly, 2003
6 Ross S M. Stochastic Processes. John Wiley, 1983
7 Foster I, Kesselman C. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1998, Chapter 3
8 Lawless J F. Statistical models and methods for lifetime data. John Wiley, 1983

(上接第183页)

+ |U2′|))，其中"≪"表示远远小于。

4.3.2 时间复杂度

对不一致决策表，若不一致对象较多，则算法1和算法2的时间复杂度低于经典的基于差别函数求解属性约简算法，算法的性能有所改进。

可见，算法2可有效地降低存储空间代价，一定程度提高求解属性约简的效率。当然，在实际应用中，用户可根据问题的规模，选择相应的属性约简算法。

结束语 本文提出改进的差别矩阵及其属性约简求解方法，统一考虑决策表一致和不致情况两种情况下的属性约简问题，克服经典的差别矩阵引起的求解属性约简算法不足。同时，提出一种新的差别矩阵浓缩策略，以此提高属性约简的效率，为进行大数据集属性约简提供一条有效的途径。

参考文献

1 Pawlak Z. Rough sets [J]. International Journal of Information and Computer Science, 1982, 11(5): 341～356
2 Pawlak Z. Rough set approach to multi-attribute decision analysis [J]. European Journal of Operational Research, 1994, 72: 443～459
3 刘清. Rough集及Rough推理[M]. 北京：科学出版社，2001
4 杨明，孙志挥. 改进的差别矩阵及其求核方法[J]. 复旦大学学报，2004，43(5)：865～868
5 杨明. 一种基于改进差别矩阵的核增量式更新算法[J]. 计算机学报，2006，29(3)：407～413
6 杨明，杨萍. 一种基于垂直分布的多决策表全局属性核求解算法[J]. 控制与决策，2006(录用待发)
7 Hu X H, Cercone N. Learning in relational databases: A rough set approach [J]. Computational Intelligence: An International Journal, 1995, 11(2): 323～338
8 Jelonek J, Krawiec K, Slowinski R. Rough set reduction of attributes and their domains for neural networks [J]. Computational Intelligence, 1995, 11(2): 339～347
9 叶东毅. Jelonek属性约简算法的一个改进[J]. 电子学报，2000，28(12)：81～82
10 Wang Jue, Wang Ju. Reduction algorithm based on discernibility matrix the ordered attributes method [J]. Journal of Computer Science and Technology, 2001, 16(6): 489～504
11 常犁云，王国胤，吴渝. 一种基于Rough Set理论的属性约简及规则提取方法[J]. 软件学报，1999，10(11)：1206～1211
12 Jensen R, Shen Qiang. Semantics-Preserving Dimensionality Reduction: Rough and Fuzzy-Rough-Based Approaches. IEEE Transactions on Knowledge and Engineering, 2004, 16(12): 1457～1471
13 刘少辉，盛秋戬，吴斌，等. Rough集高效算法的研究. 计算机学报，2003，26(5)：524～529
14 Guan J W, Bell D A. Rough computational methods for information systems [J]. Artificial Intelligences, 1998, 105 (1-2): 77～103
15 苗夺谦，胡桂荣. 知识约简的一种启发式算法. 计算机研究与发展，1998，36(6)：681～684
16 Woblewski J. Finding minimal reducts using genetic algorithm: [ICS Research Report]. 16/95. 1995. Warsaw University of Technology