基于日志挖掘的 Web service 安全关联规则挖掘算法及应用*

杭正波，杨鹤标，邹盼盼，陈锦富
（江苏大学 计算机科学与通信工程学院，江苏 镇江 212013）

摘　要： 针对传统的 Web service 安全性测试方法存在的低效性和盲目性，提出了一种基于 Web service 日志挖掘的安全关联规则挖掘算法，并阐述了算法的应用环境。通过该算法挖掘出正常行为的关联规则，采用错误注入的方式对 Web service 注入预先设计的构造算子，并把执行后的日志与关联规则进行比较，进而发现 Web service 存在的安全性问题。实验结果表明，该算法较大地提高了日志挖掘的效率及覆盖率，同时应用该算法能较好地检测出 Web service 的安全性问题，进一步表明提出的算法是可行有效的。

关键词： 日志挖掘；关联规则挖掘；安全性测试；错误注入
中图分类号： TP311；TP391　　**文献标志码：** A　　**文章编号：** 1001-3695(2012)05-1802-04
doi：10.3969/j.issn.1001-3695.2012.05.053

Algorithm and application for mining Web service security association rules based on log mining

HANG Zheng-bo, YANG He-biao, ZOU Pan-pan, CHEN Jin-fu
(*School of Computer Science & Telecommunication Engineering, Jiangsu University, Zhenjiang Jiangsu* 212013, *China*)

Abstract: To solve the inefficiency and blindness of the traditional method of security testing, this paper proposed a new log mining algorithm based on Web service and described the applied environment of the algorithm. The association rules were mined from the normal behavior by this algorithm. It injected the mutation operator into the Web service by fault injection method, compared the execution log with the association rules, and then found out the security problems existed in Web service. The results show that the algorithm can greatly improve the efficiency and coverage rate of log mining and detect the problems of Web service better. The proposed algorithm is feasible and effective.

Key words: log mining; association rules mining; security test; fault injection

0　引言

面向服务的体系结构 SOA(service-oriented architecture)已经成为分布式系统的主要发展趋势。Web service 作为 SOA 体系结构的实现，引入了一种全新的 Web 应用开发、部署和集成的模式[1]，Web service 是一种部署在服务器上的软件构件，其服务接口及绑定形式可以通过 W3C 等国际组织制定的基于 XML(extensible markup language)的标准定义、描述、检索和调用[2]。由于 Web service 通常包含应用系统关键的业务，若其安全性出现问题可能会造成重大损失和严重后果[3]。Web service 的安全问题成为制约其广泛应用的主要障碍[4]。Web service 的安全性需求包括数据的机密性、完整性、身份的可鉴别性和不可抵赖性。Web service 的安全性测试不同于其他软件的测试，主要表现在以下方面：a) Web service 的开发环境和应用环境有很大的不同，在发布之前很难预测实际的运行场景；b) Web service 的应用通常涉及到服务提供者、代理者和使用者三种角色，都需要参与到测试的不同阶段；c) We service 对用户不提供源代码，只能进行黑盒测试；d) 多个 Web service 在运行时可以动态地组成一个新的 Web service，需要进行集成测试[5]。为了保证 Web service 的质量，必须对其进行测试。当前 Web service 安全性测试主要依靠测试人员手工生成大量的测试用例，这种方法费时费力，且带有一定的盲目性和倾向性，主要包括基于扩展的 WSDL(Web service description language)文件来生成测试数据的方法[6]、基于合约变异的测试技术可以有效地针对用户无法获得 Web service 实现细节的缺陷[7]、基于 SOAP(simple object access protocol)洪泛的方式进行组合 Web 服务拒绝服务的攻击，以发现存在的安全问题[8]。

测试数据的有效性将直接影响 Web service 的测试效率和测试成本。于是传统的软件测试方法和技术难以适应 Web service 的测试需求，如何为 Web service 选择有效的安全性测试方法成为亟需解决的问题。针对以上问题，本文借鉴入侵检测中对日志进行分析挖掘的思想[9-11]，提出一种从 Web ser-

收稿日期： 2011-10-30；**修回日期：** 2011-11-30　　**基金项目：** 国家自然科学基金资助项目(61063013)；国家教育部博士点专项基金资助项目(20103227120005)

作者简介： 杭正波(1987-)，男，江苏盐城人，硕士研究生，主要研究方向为数据挖掘、软件测试(hangzbo@163.com)；杨鹤标(1960-)，男(回族)，江苏淮安人，教授，主要研究方向为软件工程、软件系统架构、数据挖掘；邹盼盼(1989-)，男，湖北黄冈人，主要研究方向为软件工程；陈锦富(1978-)，男，江西信丰人，讲师，博士，主要研究方向为软件测试、数据库系统.

vice 执行日志中挖掘出隐藏在其中的安全关联规则的算法 WSLMA(Web service log mining algorithm)。根据 Web service 的 WSDL 文件上描述的调用接口的特性,设计出不同的错误构造算子,并使用错误注入的方式注入到请求者的调用文件中,并把由此产生的日志记录和挖掘出的关联规则进行比较,据此发现 Web service 的安全性问题。

1　Web service 安全性测试方法的框架

Web service 日志挖掘就是从 Web 服务端产生的日志文件中,挖掘出隐含的、有用的、尚未发现的信息,经过分析后得到能直观地被用户看懂的、有价值的信息。日志中包含发生在系统和网络上的不寻常和不期望活动的证据,通过查看 Web service 执行过程中的日志,能够发现其中不安全的漏洞。基于以上描述,本文提出了一种 Web service 日志挖掘的算法,并运用此算法进行 Web service 的安全性测试。该方法的框架如图 1 所示。

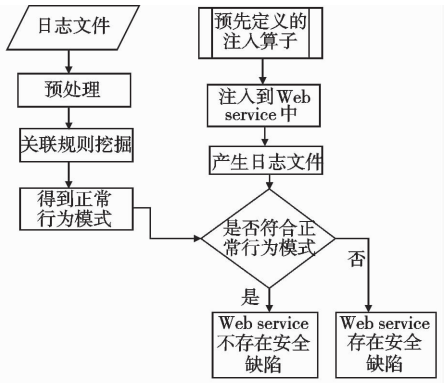

图1　方法框架

图 1 框架中主要分为正常行为关联规则的挖掘、错误构造算子的设计、安全性测试的方法三大模块。对正常行为的日志文件采用 WSLMA 规则提取的算法,找出正常行为的关联规则。对 Web service 执行错误构造算子,把该算子执行后的日志记录和挖掘出的正常行为的关联规则进行比较,如果不符合规则,则 Web service 存在安全性的问题;否则继续执行错误构造算子,直至 Web service 所有的注入点都测试完毕。其中,Web service 日志挖掘算法是整个框架的核心。

2　Web service 日志挖掘算法

2.1　日志文件中字段的选择

Web 服务器日志的原始标准是通用日志格式(common log format,CLF)。该标准包括七个数据元素。而扩展的通用日志格式(extended common log format,ECLF)增加了两个元素,包括以下信息:远程主机域或 IP 地址、请求的日期和时间、方法(如 get、post 等)、用户登录名、HTTP 状态码、传输字节数、引用页的 URL、服务器授权用户名、用户使用的操作系统和浏览器。为了记录更详尽的 Web service 执行过程信息,把 ECLF 进行了扩展,添加了 time-finish、请求的参数 params-in 和返回值 value-return 三个字段,其描述为完成一次服务所需的时间,调用方法输入的参数和方法的返回值。为此,本文截取了对研究有用的几个域,如表 1 所示,以便更加清晰地对日志进行分析。

表 1　日志数据结构表

扩展域	描述	扩展域	描述
remotehost	远程主机域或 IP 地址	status	HTTP 状态码
username	用户登录名	bytes	传输字节数
date	请求的日期和时间	time-finish	完成服务所需的时间,单位 ms
params-in	请求的参数	value-return	服务的返回值

2.2　日志文件的预处理

在获得 Web service 的执行日志后,需要对日志进行数据预处理。尽管使用者每次发起一个请求都会产生一个对应的日志条目,但是 Web service 可能同时为成百上千或者更多的使用者提供服务,因此特定的会话记录并不是连续的。包含一次服务的过程可能分散在日志中,对于这种情况可以采取以下的策略:将 Web service 执行日志中的记录按照 IP 地址和执行时间进行排序,不同的 IP 对应不同的服务请求者。对于同一个 IP 地址,如果一条日志记录与下一条日志记录之间的时间间隔不大于一个事先定义好的阈值(如 10 min),则表示是同一次请求服务。如果日志中有空缺的字段,还需要对日志进行默认值的补全。

2.3　日志关联规则的提取算法

对日志中的信息进行安全性的判断需要一个安全性的标准,日志记录了 Web service 执行时的一些属性值,只有这些属性值满足一定的规则,该次服务才是安全的。为了获得安全性的标准,本文借鉴了数据挖掘中的顺序覆盖算法,采用直接从训练数据中提取 IF-THEN 规则的方法。该训练数据就是安全的数据执行 Web service 后产生的日志文件。

规则质量的度量要同时考虑到准确率和覆盖率。假定当前的规则是 R:IF condition THEN class = c。将给定属性测试逻辑合取到 condition 后的新条件是 condition',则 R′:IF condition′ THEN class = c 是一个可能的新规则。如何判断 R′比 R 更好,为此引入了信息增益的度量。在机器学习中,正用于学习规则的类的元组称为正元组,而其余的元组为负元组。设 pos(neg)为 R 覆盖的正(负)元组数。设 pos′(neg′)为 R′覆盖的正(负)元组数,为了度量两个规则的好坏,本文选用比较常用的计算式[12]:

$$\text{Rule\_Gain} = \text{pos}' \times \left(\log_2 \frac{\text{pos}'}{\text{pos}' + \text{neg}'} - \log_2 \frac{\text{pos}}{\text{pos} + \text{neg}}\right)$$

它可以计算出具有高准确率且覆盖许多正元组的规则。规则以从一般到特殊的方式增长,从空规则开始,然后逐渐地向它添加属性测试。添加属性测试作为规则前件当前条件的逻辑合取。例如对 Web service 的日志进行规则的挖掘,可以从 IF THEN service is security 开始,然后考虑每个可以添加到该规则中的可能的属性测试,这些属性从表 1 中查找。如果对于属性是一种值对的形式(key,value),可以考虑诸如 attr = val,attr ≤ val,attr > val 等测试。因为选取了多个日志的字段进行挖掘,每个属性都有可能的值,因此为了找到最优规则集,本文采用一种贪心的深度优先策略,每当面临添加一个新的属性测试到当前规则时,根据日志记录选择最能提高规则质量属性的测试。假定发现 bytes 在 254 ~ 374 Byte 最大限度地覆盖了当前日志中的记录,就将它添加到条件中,当前规则变为 IF bytes > 254 AND bytes < 374 THEN service is security。每添加一个测试

属性到规则时,都需要计算 Rule_Gain 的值,查看结果规则是否更合适,在下一次迭代中,再次考虑可能的属性测试。对 Web service 日志进行 IF-THEN 规则挖掘的算法 WSLMA 用伪代码描述如下:

```
输入:日志文件中的记录集合 C、所有属性与它们可能值的集合 Attrs。
输出:IF-THEN 规则。
rule_set = { }; //规则的初始集为空
while( C! = null) { //日志记录不为空
    if( Attrs! = null) {
    //将每个属性都加入到 conditon 中进行判断
        for( cl in C) { // 对日志的每条记录进行遍历
            dl = addOneAttr( rule);
            //贪心的选择覆盖率最大的属性 dl
            rule′ = rule + dl;
            //加入到旧规则 rule 中形成新规则 rule′
            gain = Rule_Gain( rule′,rule);
            //每加入一个属性都要计算新规则的 gain
            if( gain >0) {
            //属性加入后能覆盖更多的规则
                    rule = rule′;
                    //将更新后的规则 rule′赋给 rule
                    Attrs′ = Attrs-dl;
                    //下次从剩下的属性集中贪心选择
                        }
                    rule_set = rule_set + rule;
                    //把符合条件的 rule 加入到规则集中
                    从日志 C 中删除 rule 覆盖的记录
                        }//end for
                            } //end while
                            //直到找不到合适的规则
```

函数 addOneAttr 是采用贪心算法选择能够最大覆盖日志记录的属性,其描述如下:

```
function addOneAttr( rule) {
    for( dl in Attrs) {
    //对属性集中的每个属性
        for( cl in C) {
        //对日志中的每条记录
            if( cl contains dl) {
            //如果记录 cl 中包含属性 dl
            num[ dl] + +;
            //属性 dl 覆盖日志记录的数目加 1
            } } }
返回 num 数组中覆盖日志记录最多的属性
}
```

算法的时间复杂度分析如下:

m 为属性集中属性的个数;n 为 Web service 中日志记录的条数。

关联规则提取算法的时间复杂度 $\leqslant \max(m \times n^2) \leqslant Cn^2$,故算法的时间复杂度为 $O(n^2)$。

2.4 Web service 安全性测试的方法

定义 1 错误构造算子 Γ_i 是指针对错误注入点生成测试用例的规则,对任意的错误注入点 α_i 都可以选用适当的构造算子 Γ_i,生成测试用例集。

设计错误构造算子的关键是根据错误注入点的特性来构造最有效的测试用例,使其能够尽可能地引发 Web service 的安全漏洞。Web service 描述语言(WSDL)是 Web service 技术重要组成部分,它描述了分布在网络环境中服务操作的抽象定义接口和服务的具体实现端口。通过对 WSDL 特性的分析和对各种 Web service 安全漏洞的分析,以及对攻击者利用漏洞入侵系统的各种手段的研究,可以归纳出对 Web service 有效的错误构造算子。错误构造算子如表 2 所示。

表 2 错误构造算子

算子	生成规则	例子描述
Γ_{seq}	打乱参数传递顺序	对于规定的参数顺序(p_1,p_2,p_3,…,p_n),执行(p_1,p_3,p_2,…,p_n)等。
Γ_{sql}	在数据中加入 sql 语句	“name’ OR ‘a = a”); DELETE FROM Users;等
Γ_{com}	在数据中加入系统命令	“;rm -rf/”“ls user. txt”等
Γ_{lon}	超长字符串	构造超长字符串“AAA…(256、512、1 024 个等)”
Γ_{net}	修改网络属性	对 IP 地址、域名等的格式和内容进行修改
Γ_{rep}	暴力猜测	对跳转的 URL、用户名猜测等
Γ_{xpa}	在数据中加入 Xpath 查询	‘OR 1 = 1 OR ‘ ’ = ’
Γ_{bou}	参数边界值变异	min、max、max + 1、min – 1 等
Γ_{null}	将节点 n 的值设置为 null	传递 null 值

运用表 2 中的错误构造算子,可以针对 WSDL 的各种不同的注入点来构造错误注入测试用例集。通常对一个错误注入点要根据其特性综合运用不同的错误构造算子来生成测试用例集,以保障测试的充分性。在运行完测试用例集后,查看 Web service 的日志文件,把日志文件中的记录信息和挖掘出来的 IF-THEN 规则集进行比较,如果该日志记录符合规则集中的规则,则说明不存在安全问题,反之则存在安全性问题。测试算法的步骤如下:

输入:构造出的测试用例,Web service 程序。

输出:安全性缺陷个数 n。

a)初始设置 n 为 0。

b)查找 Web service 中的所有注入点,计算总数为 m。

c)处理任一注入点,查找表 2,判断该注入点的具体类型。

d)根据错误注入点的具体类型构造测试用例。

e)注入测试用例。

f)查看日志中记录的信息,与 IF-THEN 规则集进行比较,判断是否存在安全问题,若存在安全问题,设置 $n = n + 1$。

g)判断该注入点是否还可以构造其他测试用例,如果可以,则重复步骤 c) ~ g),否则进行下个注入点的处理。

h)输出安全性缺陷的个数 n,评价 Web service 的安全性。

3 实验分析

实验以一个模拟比较两家航空公司票价差异的 Web service 作为测试对象。该 Web service 包括模拟查询航空公司 A

的票价(searchPriceA)、航空公司 B 的票价(searchPriceB)和比较两家航空公司票价(comparePrices)的三种方法,其状态转换如图 2 所示。使用代码可用跨平台的 Java 语言编写 Web service,然后将服务部署到 Tomcat 服务器上,这样就能够利用浏览器访问服务器端所提供的 Web service。使用自动测试工具进行业务流程完备测试,并把每次服务的执行情况记录在日志文件中,产生了 3 638 K 的日志记录,对日志记录进行预处理后剩下 3 194 K 的日志记录,将预处理后的日志记录作为实验数据集。实验进行的环境为 Intel® Pentium® Dual E2180@ 2.00 GHz CPU/ 1.00 GB/ Windows XP SP3。将本文实验与文献[11]所提出的 PERIO 算法进行测试比较。

3.1　算法效率分析

为了测试算法的效率,本文进行了算法执行时间的比较实验,比较的方法是对不同数目的日志记录分别运行 WSLMA 和 PERIO 算法,统计出相应的执行时间,结果如图 3 所示。另外还进行了算法准确性的实验比较,比较的方法是从日志记录中随机抽取部分记录作为训练集,然后采用两种算法进行处理,并根据算法结果和手工统计结果计算相应的准确度,结果如图 4 所示。

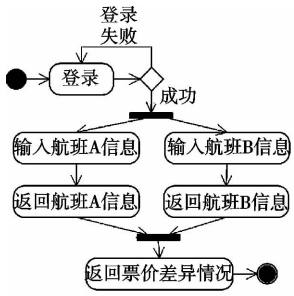

图2　Web service方法转换状态

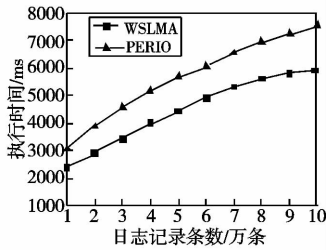

图3　算法执行时间比较

图 3 是算法执行时间的测试结果。从图中可以看出,WSLMA 算法的执行时间要低于 PERIO 的执行时间,同时随着数据量的增多,本文算法 WSLMA 执行时间曲线的上升趋势比较缓慢,说明 WSLMA 算法具有较好的扩展性。图 4 说明 WSLMA 的准确性要优于 PERIO 算法,并且随着日志记录的增多效果越明显。

3.2　算法覆盖率分析

图 5 是针对算法对日志覆盖率的测试结果。从图中可以看出,WSLMA 算法对日志的覆盖率高于 PERIO 算法,同时随着数据量的增多,本文算法 WSLMA 日志覆盖率曲线的下降趋势比较缓慢,说明 WSLMA 具有较高的覆盖率。

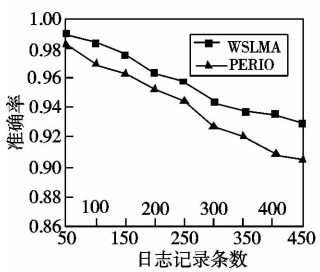

图4　算法准确性比较

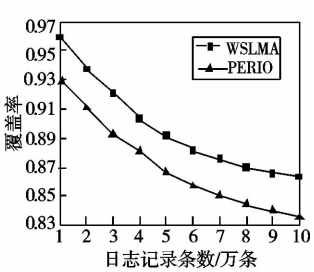

图5　日志覆盖率比较

3.3　对 Web service 安全性测试的效果

为了检测算法对 Web service 安全性测试的效果,对实验中 Web service 的三个操作分别用预先设计的测试用例进行注入测试,并把执行后的日志记录和利用 WSLMA 算法挖掘出的关联规则进行比较。对实验中的 Web service 进行安全性测试的结果如表 3 所示。

表3　发现 Web service 安全性问题的情况

方法名	测试用例数为 10、20、30 下的缺陷数		
	10	20	30
searchPriceA	1	2	4
searchPriceB	1	1	3
comparePrices	2	4	5

从表 3 可以看出,采用 WSLMA 算法进行 Web service 安全性测试的方法能较好地检测出 Web service 的安全性问题。测试用例的个数越多,对 Web service 的安全性测试就越准确。

4　结束语

本文提出的 Web service 日志挖掘算法能较好地挖掘出隐藏在其中的关联规则,并使 Web service 安全性测试过程程序化、系统化、独立化,减少了安全性测试过程对测试人员个人能力的依赖,降低了测试结果的不确定性。实验结果表明,与其他算法相比,其挖掘效率和覆盖率均有较大提高。但是本文算法适用于标准日志格式的数据集,对非标准日志格式数据集的安全性分析有待于进一步的探讨和研究。

参考文献:

[1] 岳昆,王晓玲,周傲英. Web 服务核心支撑技术:研究综述[J]. 软件学报,2004,15(3):428-442.

[2] McLLRAITH S A,SON T C,ZENG Hong-lei. Semantic Web services [J]. Journal of IEEE Intelligent Systems,2001,15(6):46-53.

[3] HANNA S,MUNRO M. Fault-based Web services testing[C]//Proc of the 5th International Conference on Information Technology. New York: IEEE Computer Society, 2008: 471-476.

[4] 李盛钢,丁晓明. 一种基于扩展 WSDL 的测试数据自动生成方法[J]. 西南师范大学学报:自然科学版,2011,36(1):188-192.

[5] 冯细光,刘建勋. Web 服务测试技术综述[J]. 微计算机应用,2010,31(1):21-26.

[6] 姜瑛,辛国茂,单锦辉,等. 一种 Web 服务的测试数据自动生成方法[J]. 计算机学报,2005,28(4):568-577.

[7] 陈锦富,卢炎生,谢晓东. 软件错误注入测试技术研究[J]. 软件学报,2009,20(6):1425-1443.

[8] JENSEN M, GRUSCHKA N, HERKENHONER R, *et al*. SOA and Web services new technologies, new standards, new attacks[C]// Proc of the 5th European Conference on Web Services. New York: IEEE Computer Society,2007:35-44.

[9] 陈锦富,卢炎生,谢晓东. 一种采用接口错误注入的构件安全性测试方法[J]. 小型微型计算机系统,2010,31(6):1090-1096.

[10] 蒋嶷川,田盛丰. 入侵检测中对系统日志审计信息进行数据挖掘的研究[J]. 计算机工程,2002,28(1): 159-161.

[11] MASSEGLIA F, PONCELET P, TEISSEIRE M. Web usage mining: extracting unexpected periods from Web logs [J]. Data Min Knowledge Discovery,2008,16(1):39-65.

[12] HAN Jia-wei, MICHELINE K. 数据挖掘概念与技术[M]. 2 版. 范明,孟小峰,译. 北京:机械工业出版社,2008.