第41卷 第4期 西 安 交 通 大 学 学 报 Vol.41 №4
2007年4月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Apr. 2007

一种基于证明树反演的安全漏洞定位方法

王清[1,2]，郑庆华[1,2]，管晓宏[1,2,3]，张哲菲[1,2]
（1.西安交通大学智能网络与网络安全教育部重点实验室，710049，西安；2.西安交通大学机械制造系统工程国家重点实验室，710049，西安；3.清华大学信息科学与技术国家实验室，100084，北京）

摘要：针对Web应用中普遍存在SQL注入漏洞等问题，提出了一种基于证明树反演的漏洞定位方法.该方法可对攻击产生的根源直接进行源代码的漏洞挖掘，其核心思想是：通过在脚本中反向追踪数据库操作所涉及的变量来检验其是否受到外界的影响，从而达到控制数据库操作的安全隐患.实验结果表明，所提方法能够准确验证53.8%的数据库操作的安全性，对于不能做出自动判定的风险操作，则能给出变量的依赖关系.该方法适用于以各种脚本语言和数据库系统搭建的Web应用平台.

关键词：注入漏洞；数据库操作；反演
中图分类号：TP393 **文献标识码**：A **文章编号**：0253-987X(2007)04-0439-05

New Method for Detecting Code Security Vulnerability Based on Reverse Deduction with Proof-Tree

Wang Qing[1,2], Zheng Qinghua[1,2], Guan Xiaohong[1,2,3], Zhang Zhefei[1,2]
(1. MOE Key Laboratory for Intelligent and Network Security, Xi′an Jiaotong University, Xi′an 710049, China; 2. State Key Laboratory for Manufacturing Systems Engineering, Xi′an Jiaotong University, Xi′an 710049, China; 3. State Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China)

Abstract: Based on reverse deduction with proof-tree, a method for detecting SQL injection vulnerabilities which are ubiquitous in Web applications is presented. Differing from traditional real time security strategies such as IDS and firewall, the origin of producing attack can be found by directly mining source codes vulnerabilities. The essentials are to track reversely the variables that are related to database (DB) script operations and check whether they are influenced from outside so as to control the hidden damage existing in the DB operations. The experimental results show that the proposed method can accurately verify the security of 53.8% DB operations, and it is applicable for any type of Web application platforms configured with different script languages and database systems.

Keywords: injection vulnerability; database operation; reverse deduction

系统漏洞的挖掘和防范一直是网络安全讨论的热点话题，对于SQL注入式漏洞攻击的防范，学术界已有研究[1-5]，主要分为代码审计和系统运行防护.本文在吸收了前人优秀思想的基础上，提出了证明树的概念，并通过在脚本中反向追踪数据库操作所涉及的变量来检验其是否受到外界的影响，从而达到证明数据库操作是否存在安全隐患的目的.

为了便于讨论，本文在阐述过程中所用到的代码片断及函数均基于PHP脚本和MY SQL数据库.

收稿日期：2006-09-01. 作者简介：王清(1982～)，男，硕士生；郑庆华(联系人)，男，教授，博士生导师. 基金项目：国家自然科学基金资助项目(60243001)；国家高技术研究发展计划资助项目(2003AA142060).

1 SQL注入式攻击原理

在Web应用的CGI脚本源代码中，不难见到如图1所示的语句.

```
$result = mysql_db_query('sample_db',"select*
from table_name where user = '$user' and
psw = '$psw' ");
```

图1 登录验证的脚本代码片断

如果有攻击者在客户端的统一资源定位(URL)请求中将变量$user的值赋为admin′ or 1=1 #，经变量值的代入，脚本执行的代码则如图2所示.

```
$result = mysql_db_query('sample_db',"select* from
admin where user = 'admin' or 1 = 1#' and psw = '$psw' ");
```

图2 经变量赋值后的脚本代码片断

在图2中，行注释符号#后的SQL语句在执行时将被数据库解释器去掉，所以最终数据库执行的语句如图3所示.

```
select* from admin where user = 'admin' or 1 = 1;
```

图3 数据库执行语句

可见，由于程序员的疏忽，即没有对数据库操作的输入变量加以限制，所以口令验证机制可被攻击者轻易突破.

2 证明树的生成

注入漏洞的根源是直接或间接暴露了数据库操作的变量.在代码中验证一个数据库操作是否存在这种漏洞，本文采用了反向变量追踪方法.变量追踪的证明过程实际上是一棵树的生长过程，称其为证明树.由于证明树的搜索方向与代码的编译方向相反，它相当于编译技术中自下而上的语法树生长，也类似于语法归约的逆运算，故将证明过程称之为反演.

2.1 验证点

验证点是系统为脚本语言提供的数据库操作函数，也是引起注入攻击的根本因素.

就PHP脚本和MY SQL数据库而言，代码片断中的第78行、第80行的函数mysql_db_query()就是典型的验证点，如图4所示.每个验证点都对应一棵证明树，验证点的参数则是证明树的树根.

```
<?php
 ⋮
40: $col_1 = 'name';
41: $col_2 = 'time';
42: $first_name = 'wang';
43: $name = $first_name . $last_name;
44: $level = 'admin';
45: $id = filter($id);
46: $table_name = 'sample_table';
 ⋮
76: mysql_connect('dbhost','dbuser','test') or die
    ("数据库连接失败");
77: $sql = "select * from $table_name where name = $name";
78: $result_1 = mysql_db_query ('sample_db', $sql);
79: $sql = "select $col_1 , $col_2 from $table_name
    where level = $level ";
80: $result_2 = mysql_db_query('sample_db', $sql);
81: $sql = "select * from $table_name where id = $id";
82: $result_3 = mysql_db_query('sample_db', $sql);
 ⋮
?>
```

图4 PHP脚本和MY SQL数据库代码片断

2.2 脚本代码的域划分

脚本的代码可分为如下2种区域.

(1)动态区.脚本中的条件分支、循环等代码块被定义为动态区，在执行过程中，这些代码区是否执行应根据用户的交互行为来决定，所以这种代码不可作为证明的证据.

另外，本文规定用户自定义函数体的实现部分也是动态区.

(2)静态区.除动态区以外，脚本的其余可见代码区都是静态区.

2.3 反演证据

从证明树当前位置反向逐行搜索，若证明树叶子节点中的变量第一次出现在赋值表达式的左边，这样的表达式称作证据.证据分为以下2种.

(1)可靠证据.证据表达式位于静态区，当赋值运算符右边只有变量、常量和字符串等运算符时，即作为可靠证据.可靠证据可以替换证明树上的叶子节点.

(2)不定证据.当证据位于动态代码区，或证据赋值表达式右边出现函数运算，称为不定证据.由不定证据扩充出的叶子节点不能被再次扩充.

2.4 证明树的生成算法

综上所述，可以给出证明树反演算法，其步骤描述如下.

步骤1：扫描脚本文件，生成验证点集合

$$V=\{v_1,v_2,\cdots,v_m\}$$

同时设置初值 $i=1$.

步骤 2：判断 i 值，若 $i>m$，下转步骤 11.

步骤 3：从 V 中取出验证点 v_i，将其参数作为证明树的根节点，并设置当前证明位置 p 为验证点所在的行号.

步骤 4：从左向右扫描根结点，并用常量和变量初始化叶子节点集合 L，用变量初始化可扩展叶子节点集合 L_E.

步骤 5：若 $p>0$，赋 $p=p-1$，否则下转步骤 9.

步骤 6：判断当前行是否是证据，若不是证据，上转步骤 5；若是可靠证据，下转步骤 7；若是潜在证据，下转步骤 8.

步骤 7：扫描可靠证据赋值号的右端，并用常量和变量替换集合 L 中的相应元素，用变量替换集合 L_E 中的相应元素，同时判断 L_E 是否为空，若 L_E 为空则下转步骤 9，否则上转步骤 5.

步骤 8：用不定证据更新 L 中的相关元素，并标记新加入的元素为不定证据派生的特殊元素，删除 L_E 中的相应元素，同时判断 L_E 是否为空，若 L_E 空下转步骤 9，否则上转步骤 5.

步骤 9：v_i 证明结束，按顺序输出 L 中的元素. 若 L 含有不定证据派生的特殊元素，v_i 存在潜在注入漏洞，则 v_i 是一个受限访问点；若 L 不含变量，且不存在特殊元素，则 v_i 不存在注入漏洞，是一个静态访问点；若 L 含有变量，且不存在特殊元素，则 v_i 存在注入漏洞，是一个高危访问点.

步骤 10：赋 $i=i+1$，上转步骤 2.

步骤 11：结束.

2.5　证明树与代码安全态势

对于无法给出断言的验证点，证明树能够给出相关变量的位置和依赖关系，还可以在反演过程中提供相关信息，并对代码整体的安全性给出一种态势评价.

证明树的深度直接反映了 SQL 语句生成逻辑的复杂程度，即证明树的深度越大，SQL 语句生成的步骤就越繁杂. 在代码中，证明树的宽度反映了数据库操作所涉及的变量数，也反映了数据库操作语句本身的逻辑复杂程度. 利用证明树的形状信息，可以对网站代码作出整体的安全态势评估，本文给出了评估模型.

变量定义如下：

O 表示高危操作点数目；

P 表示受限操作点数目；

p_w 表示受限操作点证明树的宽度（叶子节点个数）；

p_h 表示受限操作点证明树的高度；

p_{pn} 表示受限操作点在证明树叶子节点中由不定证据派生出的节点数目.

根据上述的变量定义，可以获得反映代码安全性的指标，即

$$\left.\begin{aligned} S&=O && O>0\\ S&=\frac{\sum_{i=1}^{P}p_{wi}}{\sum_{i=1}^{P}p_{hi}+\sum_{i=1}^{P}p_{pni}} && O=0 \end{aligned}\right\}$$

当 $O>0$ 时，S 表示高危验证点的个数，这时的网站一定存在漏洞；当 $O=0$ 时，S 表示网站存在入侵风险.

下面分析证明树的产生过程.

扁平状的证明树意味着数据库操作语句本身的逻辑非常复杂，涉及很多的变量，这些变量也没有通过复杂的逻辑判断就进入了验证点，这样安全隐患就比较大. 相反，证明树瘦长，意味着数据库操作所涉及的变量比较少，且经过了复杂的逻辑判断才进入操作点，这样安全性比较高.

若数据库操作的复杂程度为 $\sum_{i=1}^{P}p_{wi}$，数据库操作越复杂，引用的变量越多，安全性越差；若 SQL 字符串生成逻辑复杂度为 $\sum_{i=1}^{P}p_{hi}$，证明树越高，变量传递进入验证点的困度就越大；若不定证据派生的叶子失去生长的能力为 $\sum_{i=1}^{P}p_{pni}$，这种叶子将隐藏证明树的部分树高.

考虑了上述因素可以认为，S 值越大，安全性越差，当 $S>1$ 时，网站代码的安全性较差；当 $S<1$ 时，网站代码的安全性较高.

实际上 S 反映了所有证明树的平均形状属性，所以可以用它评估代码的整体安全性能.

3　反演举例

3.1　反演结果分类

树的节点分为 3 种：常量、可派生变量和不可派生变量. 树的根节点是验证点，验证点中的变量是可派生变量.

派生是由根结点开始反向搜索证据，若证据左部与当前树叶子结点的变量相同，则将证据右部挂

在该变量对应节点之下.由可靠证据派生出的变量可以继续派生,由不定证据派生出的变量只可作常量,其为不可派生变量.

当反向搜索至程序入口点时,树生长完毕,这时的树称为验证点的证明树.证明树的最终形态可分为3类:①叶子节点存在变量,即外界输入可以直接影响数据库操作,这种验证点为高危访问点;②叶子节点均为常量,即在程序内数据库操作是封闭的,不受外界影响,这种验证点为静态访问点;③证明树在生长时被不定证据中止而未检验至程序的起始位置,该验证点为受限访问点,这种情况多是程序员已对数据库的输入操作进行了限制.

3.2 高危访问点

在图4中,代码片断第78行的验证点首先将第77行的表达式作为可靠证据,并扩展根结点$sql,然后在第46行找到第2个证据,以用于扩展叶子节点$table_name,最后一个证据出现在第43行.这个验证点证明树的生长过程如图5所示.

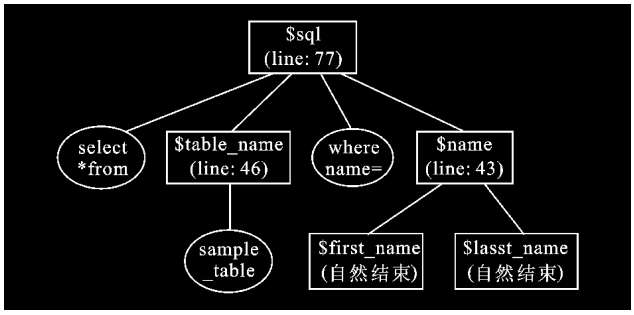

图5 存在注入漏洞的证明树

当反向搜索证据的过程进行到程序起始点时,证明树停止生长.这时,按照从左到右的顺序遍历证明树的叶子节点,并还原代码中数据库操作的全部信息,如图6所示.

```
select* from sample_table where
name = $first_name.$last_name;
```

图6 还原后数据库操作的全部信息

从图6可以看到,$first_name和$last_name这2个变量没有经过任何限制就用于生成数据库操作命令,这就给攻击者构造巧妙的SQL语句进行恶意的数据库操作带来可乘之机.如果数据库权限配置不当,甚至可以利用这种漏洞执行任意代码.

3.3 静态访问点

图4第80行的验证点证明树的生成过程如图7所示,遍历证明树的叶子节点还原后的安全数据库操作如图8所示.

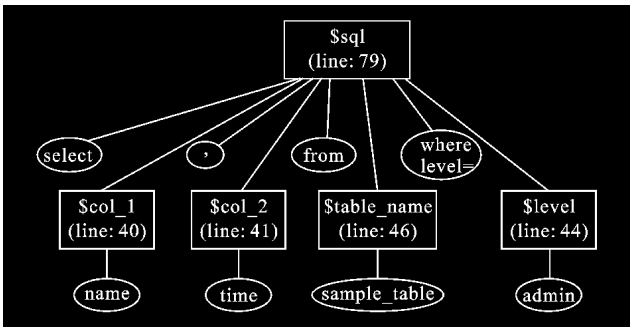

图7 图4第80行的验证点证明树的生长过程

```
select name , time from sample_table
where level = 'admin';
```

图8 还原后的安全数据库操作

这时,SQL命令字符串独立于外界输入,而在代码空间里是封闭的,所以第80行的验证点的数据库操作是安全的.

3.4 受限访问点

如果在遍历证明树叶子节点时,所有的节点是静态常量的或是由不定证据派生而来的,则无法断言该验证点是否存在漏洞.

就脚本中数据库查询串的生成过程而言,产生不定证据的最主要原因就是程序员已经对用户输入的变量内容进行了限制,所以这种验证点的数据库操作称为受限访问.典型的例子就是图4中第82行的验证点,经过第45行的不定证据扩展后,证明树最终还原出的SQL命令,如图9所示.

```
select* from sample_table where id = filter($id);
```

图9 证明树还原出SQL命令

图9中的函数filter()是程序员编写的,是对输入变量内容进行过滤的函数,但其是否能够真正、彻底地防止注入攻击,证明树却无法断言,所以该函数仍然保持着注入点属性.

3.5 应用结果

SQL注入漏洞检测器(CVSID)是基于证明树反演理论而开发的一种代码安全漏洞挖掘软件.本文利用LEXY和YACC等编译工具对脚本语言进行词法和语法分析,在中间代码的基础上用证明树

的生成算法进行推理，得到最终的证明树.

在 CVSID 的测试过程中使用了大量的开源代码，例如用 CVSID 对文章发布网站的开源代码进行分析，该网站包含了 22 个 PHP 文件，4 088行代码. 经过 CVSID 的词法和语法分析，提取出的、可供证明树反演的只剩下 830 行代码，精简率达到了 80%. 再经反演后，CVSID 在 6 处精确定位出漏洞，经人工核查，这些均可造成 SQL 注入漏洞.

在实验中，根据代码质量、代码用途、代码书写风格，本文方法给出结论的数据库操作也有比较大的差异，其中最容易出现 SQL 注入漏洞的系统，用下载的开源代码作为输入，能够准确验证 53. 8%的数据库操作的安全性，且不会出现类似 IDS 系统中的误报现象. 对于不能作出自动判定的风险操作，CVSID 则给出变量依赖关系，以协助程序员进行人工审计.

4　结束语

证明树反演方法在检测 SQL 注入漏洞的实际应用中取得了比较满意的效果. 增强证明树的分析能力，缩小受限访问点的比率，减少人工处理，是下一步工作的重点和研究方向. 如果解决了程序中动态流程问题，本文方法就可以推广，如对 C 语言中缓存溢出漏洞的检测，或用数学的方法证明系统的安全性等.

参考文献：

[1] Pietraszek T, Berge C V. Defending against injection attacks through context-sensitive string evaluation [C]//8th International Symposium on Recent Advances in Intrusion Detection. Berlin: Springer-Verlag, 2006: 124-145.

[2] Buehrer G T, Bruce W. Using parse TreeValidation to prevent SQL injection attacks [C]// Proceedings of the 5th international Workshop on Software Engineering and Middleware. New York: ACM, 2005: 106-113.

[3] Fosdick L D, Osterweil L J. Data flow analysis in software reliability [J]. Computing Surveys, 1976, 8 (3): 305-330.

[4] Gustafsson J, Lisper B. A tool for automatic flow analysis of C-programs for WCET calculation [C]// 8th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems. Piscataway, USA: IEEE, 2003: 106-112.

[5] Walker D. A type system for expressive security policies [C]// Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. New York: ACM, 2000: 254-267.

（编辑　苗凌）

我校获 2006 年度国家科学技术奖 4 项

2006 年度国家科学技术奖励大会于 2 月 27 日在北京召开，胡锦涛、温家宝等国家领导人出席大会并给获奖人员颁奖，我校获奖代表蒋庄德教授、邢建东教授、邢子文教授、郑庆华教授参加大会. 此次我校获得国家科学技术奖 4 项，其中国家技术发明二等奖 2 项、国家科学技术进步二等奖 2 项，继去年之后再次取得好成绩. 获奖项目如下：

（1）“耐高温压力传感器设计、制造关键技术及系列产品开发”获技术发明二等奖，项目主要参加人有蒋庄德、赵玉龙、赵立波、高建忠、王文襄、刘秀娥；

（2）“高温抗磨材料制备技术及其应用”获技术发明二等奖，项目主要参加人有邢建东、符寒光、高义民、鲍崇高、张国赏、王恩泽；

（3）“螺杆压缩机设计理论、关键技术及系列产品开发”获科技进步二等奖，项目主要参加人有邢子文、刘元璋、吴华根、姜韶明、束鹏程、杨永才、彭学院、韩博飞、畅云峰、张永立；

（4）“天地网远程教育关键技术、系列产品及其应用”获科技进步二等奖，项目主要参加人有郑庆华、申瑞民、刘均、韩鹏、管晓宏、于德弘、胡云华、杨帆、吴茜媛、彭挺.

（科技处）