(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102185930 A
(43)申请公布日 2011.09.14

(21)申请号 201110153505.8

(22)申请日 2011.06.09

(71)申请人 北京理工大学
地址 100081 北京市海淀区中关村南大街5号

(72)发明人 金福生 宋挺 戴银涛 牛振东 韩翔宇

(51)Int.Cl.
*H04L 29/08*(2006.01)
*G06F 17/30*(2006.01)
*H04L 29/06*(2006.01)

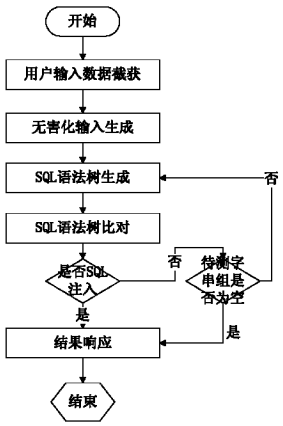
权利要求书 2 页 说明书 8 页 附图 6 页

(54)发明名称

一种SQL注入漏洞检测方法

(57)摘要

本发明涉及一种SQL注入漏洞检测方法，包括以下步骤：一、用户输入数据截获；二、无害化输入生成；三、进行SQL词法分析和语法分析，生成SQL语法树，分别是基于用户输入字串的语法树以及基于无害化字串的语法树；四、SQL语法树比对，如果相同认为该组测试字串通过本轮测试；五、结果响应：如果发现了用户有SQL注入的企图，则阻止该HTTP包，否则将该HTTP包放行。本发明分析对象均直接或间接来源于用户输入，这样可以最大限度还原用户本意，降低了误报率。同时基于SQL语法树分析，能够从根本上阻断进行SQL注入的可能，从而提高检测的准确率。




CN 102185930 A

1. 一种 SQL 注入漏洞检测方法，包括以下步骤：

一、用户输入数据截获：

(1) 获取用户向应用程序提交的 HTTP 包；

(2) 将用户提交的 GET、POST 数据按 URL、COOKIE、表单分类，并按类型提取用户数据包中提交的所有参数值；

(3) 将获得的多组参数值按照 URL 编码和其他 HTTP 包指定编码方式解码参数值；

二、无害化输入生成：

(1) 将步骤一中得到的 n 组参数值记为 Q1，Q2……Qn，同时生成等量的空白字串 Q’1，Q’2……Q’n；

(2) 按照无害化规则将 Qi 字串转化为无害字串拷贝至字串 Q’i，Qi 仍保留原内容 (i ＝ 1，2，……n)；

(3) 将 n 组用户输入的原始字串 Qi 与 n 组由用户输入生成的无害化字串 Q’i 归为 n 组待测试字串组 Si(i ＝ 1，2，3……n)；

三、SQL 语法树生成：

(1) 预设 SQL 语句的注入点模板，将待测试字串组 Si(i ＝ 1，2，3，……n) 按序同 SQL 语句注入点模板组合，生成包含用户输入的 SQL 语句和包含无害字串的 SQL 语句，分别输入 SQL 词法分析器；

(2) 将词法分析结果输入 SQL 语法分析器；

(3) 将语法分析结果生成两棵语法树，分别是基于用户输入字串 Qi 的语法树 Ti 以及基于无害化字串 Q’i 的语法树 T’i；

四、SQL 语法树比对：

(1) 将两棵语法树 Ti 及 T’i 通过孩子－兄弟表示法转为等价的二叉树形式 BTi 及 BT’i；

(2) 对 BTi 及 BT’i 进行前序遍历，得到前序序列 Fi，F’i，通过字符串比较算法对 Fi 与 F’i 进行比较，如发现不相同，则判定用户在进行 SQL 注入，直接转入步骤五；

(3) 对 BTi 和 BT’i 进行中序遍历，得到中序序列 Mi，M’i，通过字符串比较算法对 Mi 与 M’i 进行比较，如发现不相同，则判定用户在进行 SQL 注入，直接转入步骤五，否则认为该组测试字串通过本轮测试；

(4) 更换组合的 SQL 语句模板，如已经组合了全部模板，则认为该组测试字串通过本次检测，否则转到步骤三继续测试；

(5) 将 i 值加 1，如 i ＜＝ n 转入步骤三继续测试，否则转入步骤五；

五、结果响应：

(1) 如果有任意一组测试发现了用户有 SQL 注入的企图，则阻止该 HTTP 包，并产生一个警告，按系统配置显示在本地或远程屏幕上，同时记录进入日志文件；

(2) 如果所有测试均没有发现用户有 SQL 注入的企图，则将该 HTTP 包放行。

2. 根据权利要求 1 所述的一种 SQL 注入漏洞检测方法，其特征在于，步骤二中所述的无害化规则为将数字转为等长度的数字 3，将字符信息转为等长度的 x，对空格予以保留。

3. 根据权利要求 1 或 2 所述的一种 SQL 注入漏洞检测方法，其特征在于，步骤三种 SQL 语句的注入点模板为：选择涵盖所有注入点类型的 SQL 语句，有：SELECT CREATE DROP

ALTER INSERT UPDATE DELETE GRANT REVOKE 这九种类型；对每种类型可能插入用户输入数据的地方进行标注，用户输入或者无害化输入只需要填充到有同样标注的地方即可构成相同类型的两句完整的 SQL 语句。

4. 根据权利要求 1 至 3 任一项所述的一种 SQL 注入漏洞检测方法，其特征在于，步骤三中所述词法分析器为以 SQL99 为标准建立的对标准 SQL 语句进行词法分析的词法分析器，及语法分析器，语法分析器为以 SQL99 为标准建立的对标准 SQL 语句进行语法分析的语法分析器。

5. 根据权利要求 1 至 4 任一项所述的一种 SQL 注入漏洞检测方法，其特征在于，在语法树中加入一种类型的语法节点 ERRSTR，ERRSTR 表示在语法分析中由用户输入部分引发了不能识别的关键字及未封闭的引号之后的一串字符串类型错误的节点，在进行语法树比较时 ERRSTR 被当做空节点。

6. 根据权利要求 1 至 5 任一项所述的一种 SQL 注入漏洞检测方法，其特征在于，获取用户输入数据的方法包括：

(1) 通过 WEB 服务器提供的接口即 WEB 服务器的核心组件获取用户向服务器提交的所有参数；

(2) 对 WEB 应用进行语法分析和预编译，在其调用与数据库交互的 API 处插入一段“交流程序”，这段程序的任务是在将参数提交数据库前先使用 SQL 注入漏洞检测方法进行检测，并根据检测程序的检测结果判断是否该继续向数据库提交这段参数。

7. 根据权利要求 1 至 6 任一项所述的一种 SQL 注入漏洞检测方法，其特征在于，步骤四中字符串比较算法为改进 KMP 算法即改进的克努特——莫里斯——普拉特操作。

# 一种 SQL 注入漏洞检测方法

## 技术领域

[0001] 本发明涉及一种针对 SQL 注入威胁的可以用于 web 防护及入侵检测的检测方法，属于网络信息安全领域。

## 背景技术

[0002] 数据库与 WEB 之间的关联已经越来越密切，而 web 方便快捷，面向用户群体广泛的特点使 B/S 开发的热度逐年增加，可以说 web 大大扩展了数据库的用户群，使其真正影响到了个人的生活方式。但是 web 在给数据库带来这些优点的同时也为数据库带来了许多安全隐患。其中危害最大，攻击手段最多，最难防范的便是 SQL 注入 (SQL Injection) 攻击。

[0003] 在 WEB 应用常见的前台语言，如 ASP.NET，PHP 或 JSP 中，一个典型的处理登录的 SQL 语句可以写作：

[0004] Query ＝“SELECT * FROM user WHERE user ＝”+“‘$username’”+“AND pass ＝”+“‘$password’”；

[0005] 通常我们通过获取用户提交的变量 $username、$password 来处理该次登录请求，将其提交给数据库并查看返回的 Query 值，从而决定本次登录是否成功。这条查询 SQL 语句在大多数情况下均能很好的工作。然而不幸的是，当攻击者输入一些精心构造的语句时，我们提交给数据库的 SQL 语句的执行结果就会偏离编写者的本意。如攻击者输入的 $username 为 :admin’ OR ‘1’ ＝‘1’ --。此时整条查询语句变成了：

[0006] Query ＝“SELECT * FROM user WHERE user ＝”+“‘admin’OR‘1’＝‘1’--’”+“AND pass ＝”+“‘$password’”；

[0007] 这条 SQL 语句由于存在了 OR ‘1’ ＝‘1’ 这个恒真子查询，所以不管攻击者提交的用户名密码是否正确，攻击者输入得到的 Query 返回值总是真，也就是攻击者通过这种 SQL 注入攻击行为，绕过了登录检测，从而可以以任何用户名登录系统。

[0008] SQL 注入漏洞更为可怕的一点在于，他为攻击者提供了一条可以随意操作数据库的最大权限的通道，从而可以随意对 WEB 应用程序的后台数据库进行查询，增加，修改，删除等操作。

[0009] 然而 SQL 注入漏洞在理论上存在于所有应用程序与数据库交互之处，一个中等规模的应用即可能有上百处，且由于整条 SQL 语句是由用户输入与程序提供的原语句拼合而成，对于用户可能的复杂输入，分析防范非常困难。

[0010] 通过分析，我们可以看到 SQL 注入具有危害大，漏洞存在离散，分析防御困难的特点。且漏洞覆盖所有支持标准 SQL 的数据库，SQL Server、MySql、Oracle、DB2、Sybase 等数据库均不能幸免。故而其被 OWASP 评定为 2007-2010 年最大网络安全威胁。

[0011] SQL 注入漏洞的危害是如此之大，从常理上讲存在这一漏洞的网站应该非常少才对。不幸的是，通过使用 Google 对中国地区网站搜索关键字为“.asp ？”，“.php ？”，“.jsp ？”的网址链接，进而采用 SQL 注入扫描工具进行检测，我们惊人的发现 13％的 asp 链接，8％的 php 链接以及 3％的 jsp 链接存在着一种或多种类型的 SQL 注入漏洞，这也就意

味着十分之一左右的网页存在着潜在风险，浏览这样的网页很可能被植入的木马攻击。

[0012] 目前，对 SQL 注入攻击的防御方式主要有以下几种：

[0013] 1. 人工管理：手工添加参数过滤语句，对用户输入进行严格过滤。主要不足在于复杂的应用程序需要处理大量的用户输入，手工检测不仅极大地增加了程序员的负担，也使程序变的难以维护。同时也因为 SQL 注入攻击的多样性使得该方法难以防御所有类型的 SQL 注入攻击。

[0014] 2. 关键词过滤：制定一个关键词集合（主要为 SQL 语言的关键字及符号，如 AND、单引号），对于所有的用户输入均与该集合中的每一个关键词匹配，如果用户的输入中存在关键字，则判定为非法输入。主要不足在于，SQL 语句是千变万化的，一种语句的执行结果可以等价转换为成其他多句语句执行结果，利用编码和利用数据库命令动态构造结构字符串都可绕过这类防范。且由于关键词过滤，导致一些含关键词的正常输入（如：YOU AND ME）也被影响，这是关键词过滤的最大弊病。

[0015] 3.API 及存储过程：编写专用的 API 或者采用存储过程的方式避免使用 SQL 语句的拼接，从而屏蔽 SQL 注入攻击。主要不足在于，对于 WEB 应用所有与数据库交互的地方均使用 API 或存储过程成本高昂，开发不便。且所编写的一套 API 在内部执行过程中依旧采用关键字过滤技术，换汤不换药。一般只有企业级应用才采用，虽然付出许多额外代价可以做到比较安全，但是依旧不能从根本上完全防止 SQL 注入攻击。

[0016] 总体来看，因为其攻击的隐蔽性与多样性，同时有些关键字也常常为普通用户使用，对 SQL 注入的检测难点在于难以保证不漏报且不错报。

## 发明内容

[0017] 本发明的目的是针对 SQL 注入攻击的检测与防御，提出一种全新、高效的检测模式与防御思路，从根本上阻断 SQL 注入途径，不漏报非法用户的 SQL 注入行为同时不错报合法用户的正常行为。

[0018] SQL 是一种结构化的查询语言，对于固定的查询他有着固定的语法结构，而在进行 SQL 注入攻击时，由于攻击者必须改变 SQL 语句的语义，而语义的改变也必将影响到其提交查询的语法树形结构。于是，根据无害的标准输入及可疑输入所构成的语法树形结构的比对结果，即可判断用户输入是否含有恶意。本发明就是基于该思想实现的。

[0019] 本发明提供了一种 SQL 注入漏洞检测方法，包括以下步骤：

[0020] 一、用户输入数据截获：

[0021] (1) 获取用户向应用程序提交的 HTTP 包；

[0022] (2) 将用户提交的 GET、POST 数据按 URL、COOKIE、表单分类，并按类型提取用户数据包中提交的所有参数值；

[0023] (3) 将获得的多组参数值按照 URL 编码和其他 HTTP 包指定编码方式解码参数值；

[0024] 二、无害化输入生成：

[0025] (1) 将步骤一中得到的 n 组参数值记为 Q1，Q2……Qn，同时生成等量的空白字串 Q’1，Q’2……Q’n；

[0026] (2) 按照无害化规则将 Qi 字串转化为无害字串拷贝至字串 Q’i，Qi 仍保留原内容 (i ＝ 1,2, ……n)；

[0027] (3) 将 n 组用户输入的原始字串 Qi 与 n 组由用户输入生成的无害化字串 Q’i 归为 n 组待测试字串组 Si(i ＝ 1,2,3……n)；

[0028] 三、SQL 语法树生成：

[0029] (1) 预设 SQL 语句的注入点模板，将待测试字串组 Si(i ＝ 1,2,3, ……n) 按序同 SQL 语句注入点模板组合，生成包含用户输入的 SQL 语句和包含无害字串的 SQL 语句，分别输入 SQL 词法分析器；

[0030] (2) 将词法分析结果输入 SQL 语法分析器；

[0031] (3) 将语法分析结果生成两棵语法树，分别是基于用户输入字串 Qi 的语法树 Ti 以及基于无害化字串 Q’i 的语法树 T’i；

[0032] 四、SQL 语法树比对：

[0033] (1) 将两棵语法树 Ti 及 T’i 通过孩子－兄弟表示法转为等价的二叉树形式 BTi 及 BT’i；

[0034] (2) 对 BTi 及 BT’i 进行前序遍历，得到前序序列 Fi,F’i，通过字符串比较算法对 Fi 与 F’i 进行比较，如发现不相同，则判定用户在进行 SQL 注入，直接转入步骤五；

[0035] (3) 对 BTi 和 BT’i 进行中序遍历，得到中序序列 Mi,M’i，通过字符串比较算法对 Mi 与 M’i 进行比较，如发现不相同，则判定用户在进行 SQL 注入，直接转入步骤五，否则认为该组测试字串通过本轮测试；

[0036] (4) 更换组合的 SQL 语句模板，如已经组合了全部模板，则认为该组测试字串通过本次检测，否则转到步骤三继续测试；

[0037] (5) 将 i 值加 1，如 i ＜＝ n 转入步骤三继续测试，否则转入步骤五；

[0038] 五、结果响应：

[0039] (1) 如果有任意一组测试发现了用户有 SQL 注入的企图，则阻止该 HTTP 包，并产生一个警告，按系统配置显示在本地或远程屏幕上，同时记录进入日志文件；

[0040] (2) 如果所有测试均没有发现用户有 SQL 注入的企图，则将该 HTTP 包放行。

[0041] 有益效果

[0042] 本发明分析对象均直接或间接来源于用户输入，这样可以最大限度还原用户本意，降低了误报率。同时基于 SQL 语法树分析，能够从根本上阻断进行 SQL 注入的可能，从而提高检测的准确率。

## 附图说明

[0043] 图 1 为本发明的五个主要步骤；

[0044] 图 2 为本发明主要步骤的流程图；

[0045] 图 3 为用户输入为 admin’ OR ‘1’ ＝ ‘1’ -- 时的 SQL 语法树；

[0046] 图 4 为用户输入为 admin’ OR ‘1’ ＝ ‘1’ -- 时的无害化输入的 SQL 语法树；

[0047] 图 5 为语法树比对的流程图；

[0048] 图 6 为用户输入为 and’ AND 时的 SQL 语法树；

[0049] 图 7 为用户输入为 and’ AND 时的无害化输入的 SQL 语法树。

## 具体实施方式

[0050] 下面结合附图，具体说明本发明的优选实施方式。

[0051] 本实施例具体实现了本发明所述的一种 SQL 注入漏洞检测方法，包括以下步骤：

[0052] 一、用户输入数据截获：

[0053] 在用户输入数据截获步骤中，如何在完全获取到用户可能向应用程序提交参数的同时忽略与数据库无关的数据，是进行后续检测工作的关键。

[0054] 在本实施例中，提供了两种方法获取输入数据：

[0055] 1、标准的 web 应用总是要通过服务器来使其可以被访问（或者该应用本身就是服务器），而最常见的流行的 WEB 服务器总是为我们提供一组接口来对用户提交的数据进行再加工，即 WEB 服务器的核心组件，可以理解为 WEB 服务器的内核程序。

[0056] IIS/APACHE 均提供这样的接口，例如 ISAPI 以及 Apache Module，本实施例正是使用 WEB 服务器提供的接口来获取用户向服务器提交的所有参数。

[0057] 2、本实施例同时还采用了另一种方法，即对 WEB 应用进行语法分析和预编译，在其调用与数据库交互的 API 处插入一段“交流程序”，这段程序的任务是在将参数提交数据库前先使用本实施例提供的检测方法进行检测，并根据检测程序的检测结果判断是否该继续向数据库提交这段参数。本质上就是一段对被保护程序的 hook 程序。

[0058] 以上两种获取用户输入参数的方法各有侧重，通过服务器核心组件方式简单高效适应性强，而通过 Hook 程序则准确、全面；

[0059] 获取用户输入的过程为：

[0060] (1) 通过数据过滤程序获取用户向应用程序提交的 HTTP 包；

[0061] (2) 将用户提交的 GET、POST 数据按 URL、COOKIE、表单分类，并按类型提取用户数据包中提交的所有参数值；

[0062] (3) 将获得的多组参数值按照 URL 编码和其他 HTTP 包指定编码方式解码参数值；

[0063] 二、无害化输入生成：

[0064] (1) 将步骤一中得到的 n 组参数值记为 Q1，Q2……Qn，同时生成等量的空白字串 Q’1，Q’2……Q’n；

[0065] (2) 按照无害化规则将 Qi 字串转化为无害字串拷贝至字串 Q’i，Qi 仍保留原内容 (i ＝ 1，2，……n)；

[0066] 无害化转换是计算机领域研究人员的一种常见技术手段，研究人员需要根据解决的问题制定无害化规则，然后进行转换。由于 SQL 语言中并没有全部由 x 构成的关键字，故我们将输入的字符替换成等长度的 x 字串以达到还原用户输入并做无害化处理的目的。同理，用户输入的数字被替换为等长度的数字 3。本实施例中采用的无害化规则为：将数字转为等长度的数字 3，将字符信息转为等长度的 x，对空格予以保留。比如，对于输入 admin’OR ‘1’＝‘1’-- 这样一个输入，替换后的无害输入为 xxxxxx xx x3xxx3xxx。可以看到，由于字符及数字均被替换，该输入已经变为无害输入，但是又因为无害输入是由用户输入转换而来的，所以很好的还原了用户本意，因此本发明可以更容易区分攻击者与普通用户。

[0067] (3) 将 n 组用户输入的原始字串 Qi 与 n 组由用户输入生成的无害化字串 Q’i 归为 n 组待测试字串组 Si(i ＝ 1，2，3……n)；

[0068] 三、SQL 语法树生成：

[0069] 不同的SQL注入点采用的SQL注入手法是不同的，为了不漏掉任何一种情况，本实施方式预设了所有类型SQL语句的注入点模板，即：对下表中关键字的合法SQL语句表示形式中，可以插入用户输入数据的位置都视为注入点，并按编号标记。

[0070]

| SQL注入可能关键字 | |
|---|---|
| 查询 | SELECT |
| 定义 | CREATE DROP ALTER |
| 操纵 | INSERT UPDATE DELETE |
| 控制 | GRANT REVOKE |
| SQL注入典型模板 | |

[0071]

[0072]

| 编号 | 典型关键字 | |
|---|---|---|
| 1 | SELECT，UPDATE，DELETE，DROP | SELECT * FROM user WHERE userinput=① |
| 2 | SELECT，UPDATE，DELETE，DROP | SELECT * FROM user WHERE userinput='①' |
| 3 | INSERT，CREATE | INSERT INTO user(set1) VALUES('①') |
| 4 | INSERT，CREATE | INSERT INTO user(set1) VALUES(①) |
| 5 | UPDATE | UPDATE user SET set1=① WHERE set2=1 |
| 6 | UPDATE | UPDATE user SET set1='①' WHERE set2='1' |

[0073] 表中，标号①为可能注入点的标记。

[0074] 由于不同类型的SQL语句其注入点不同，故选择涵盖所有注入点类型的SQL语句，有：SELECT CREATE DROP ALTER INSERT UPDATE DELETE GRANT REVOKE这九种类型；对每种类型可能插入用户输入数据的地方进行标注，用户输入或者无害化输入只需要填充到有同样标注的地方即可构成相同类型的两句完整的SQL语句，确保不漏过任何注入点。

[0075] 每次选择一个SQL语句模板，将待测试字串组Si(i = 1,2,3, ……n)按序同SQL语句模板组合，在同一标记处填充用户输入字串与无害化字串，生成两条SQL语句，以备下一步的SQL语法树生成使用。

[0076] 如上文所述例子中，用户输入了admin'OR '1'='1'--，无害化字串为xxxxxx xx x3xxx3xxx。我们选取SELECT语句作为模板：

[0077] 用户输入：

[0078]

```
SELECT * from user WHERE username='admin' OR '1'='1'--' AND password='password'
```

[0079] 无害化输入：

[0080]

```
SELECT * from user WHERE username='xxxxxx xx x3xxx3xxx' AND password='password'
```

[0081] 接下来为这两组输入构建两棵语法树。本实施例以 SQL99 为标准建立对标准 SQL 语句进行词法分析和语法分析的词法及语法分析器，同时新加入一种类型的语法节点 ERRSTR，ERRSTR 表示在语法分析中由用户输入部分引发了不能识别的关键字及未封闭的引号之后的一串字符串类型错误的节点，在进行语法树比较时 ERRSTR 可以被当做空节点。

[0082] 通过建立的词法及语法分析器，可以生成对应的用户输入的 SQL 语法树以及无害化输入的 SQL 语法树。例如，对于以上用户输入，生成的语法树如图 3 所示，对应的无害化输入的语法树如图 4 所示。

[0083] 四、对已经得到的两棵语法树树形进行比对，其流程如图 5 所示。由已经得到的两棵 SQL 语法树树形，通过下面所述方法对树形进行比对。如果用户在进行注入攻击，那么必然会改变 SQL 语句的语法树结构，这也就导致在进行两棵树比较时马上被识别出来。如果检测的结果二者树形匹配，则更换 SQL 语句模板继续测试。如果所有模板都通过了测试，则进行下一组输入的检测。

[0084] 比对方法如下：

[0085] (1) 将两棵语法树 Ti 及 T'i 通过孩子 - 兄弟表示法转为等价的二叉树形式 BTi 及 BT'i，左枝代表兄弟右枝代表孩子。在转换过程中，原语法树中如果某节点有多个孩子则以从左至右的转换顺序为标准，这样即可生成等价的唯一二叉树。

[0086] (2) 对 BTi 及 BT'i 进行前序遍历，得到前序序列 Fi，F'i，通过字符串比较算法对 Fi 与 F'i 进行比较，如发现不相同，则判定用户在进行 SQL 注入，直接转入步骤五；本实施例中采用的字符串比较算法具体为改进 KMP 算法即改进的克努特——莫里斯——普拉特操作，该方法可以高效的进行字符串比对，从而提升程序效率。字串中 ERRSTR 节点可以视为空节点，不参与比较。

[0087] (3) 对 BTi 和 BT'i 进行中序遍历，得到中序序列 Mi，M'i，通过字符串比较算法对 Mi 与 M'i 进行比较，如发现不相同，则判定用户在进行 SQL 注入，直接转入步骤五，否则认为该组测试字串通过本轮测试；

[0088] (4) 更换组合的 SQL 语句模板，如已经组合了全部模板，则认为该组测试字串通过本次检测，否则转到步骤三继续测试；

[0089] (5) 将 i 值加 1，如 i ＜＝ n 转入步骤三继续测试，否则转入步骤五；

[0090] 很明显，步骤三所生成的两棵语法树不一致，所以可以判定存在 SQL 注入攻击。

[0091] 五、结果响应

[0092] 通过之前的检测结果，如果有任意一组测试发现了用户有 SQL 注入的企图，则阻断该 HTTP 包的继续传递，同时产生一个警告，按系统配置显示在本地或远程屏幕上，并记录进入日志文件，。如果所有测试均发现用户没有 SQL 注入的企图，则将该 HTTP 包放行。

[0093] 以上实施例中给出了一个涉嫌注入输入的例子，下面给出一个正常输入的例子与一个含有 SQL 关键字但不是 SQL 注入攻击的例子。

[0094] 假如用户在一个登录界面用户名字段输入了 abcde，而密码字段输入了 and' AND，则对应的 HTTP 包为：

[0095]

```
POST /user/login.jsp HTTP/1.1\r\n
Accept: image/gif, image/jpg\r\n
……
\r\n
username=abcde&password=and%27AND\r\n
\r\n
```

[0096] 通过用户输入数据截取步骤我们可以得到用户的输入数据为 {username = abcde} 以及 {password = and' AND} ;

[0097] 通过无害化输入生成步骤我们生成了两组待测字串,分别为 {username = abcde, username = xxxxx} 以及 {password = and' AND, password = xxxxxxx} ;

[0098] 首先选取第一组,我们不妨采用 SELECT 模板,生成了下面两句 SQL 语句 :

[0099] 用户输入 :

[0100]

```
SELECT * from user WHERE username = 'abcde'
```

[0101] 无害化输入 :

[0102]

```
SELECT * from user WHERE username = 'xxxxx'
```

[0103] 通过本发明的语法分析器生成 SQL 语法树,上面两组 SQL 语句的语法树完全相同,则可以认为用户的这组输入没有进行 SQL 注入。

[0104] 接着选取第二组,同样采用 SELECT 模板,生成了下面两句 SQL 语句 :

[0105] 用户输入 :

[0106]

```
SELECT * from user WHERE username = 'and' AND'
```

[0107] 无害化输入 :

[0108]

```
SELECT * from user WHERE username = 'xxxxxxx'
```

[0109] 通过本发明的语法分析器,生成的两棵 SQL 语法树如图 6 和图 7 所示。图 6 为无害化输入生成的语法树,图 7 为用户输入生成的语法树,根据比较规则,可以看出虽然两棵语法树不完全相同,但是图 7 的语法树只是比图 6 多了一个 ERRSTR 的节点,而根据 ERRSTR 可以被当做空节点的规则来看,这两棵语法树在 SQL 语法树比较模块中的结果就完全相同

了。所以同样认定本次用户输入没有进行 SQL 注入。

[0110] 而通过用户输入的 and' AND 我们也能知道，虽然用户输入了 SQL 语法的关键字但是用户并没有 SQL 注入的企图，而往往通常的 SQL 注入检测程序就会认定含有 SQL 语法关键字的输入在进行 SQL 注入，那么就容易造成错误的判断。

[0111] 另一方面，如果用户输入了一些不常用的关键字，或者用某些关键字替换了另外的关键字，如用 OR 代替 AND 的效果，往往造成一些 SQL 注入检测程序的漏判，而本发明从 SQL 语法语义入手，从根本上杜绝了用户进行 SQL 注入的可能。

[0112] 为了说明本发明的内容及实施方法，给出了一个具体实施例。在实施例中引入细节的目的不是限制权利要求书的范围，而是帮助理解本发明所述方法。本领域的技术人员应理解：在不脱离本发明及其所附权利要求的精神和范围内，对最佳实施例步骤的各种修改、变化或替换都是可能的。因此，本发明不应局限于最佳实施例及附图所公开的内容。

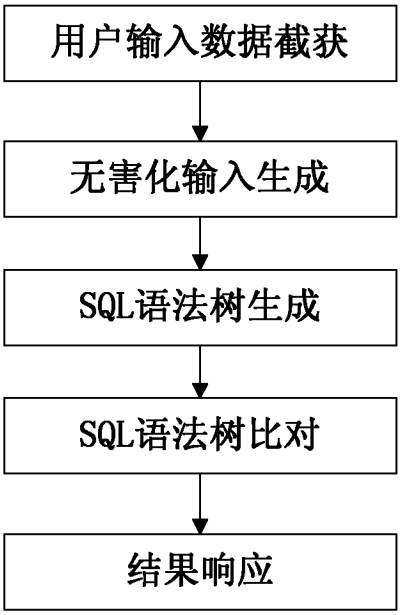

用户输入数据截获
无害化输入生成
SQL语法树生成
SQL语法树比对
结果响应

图 1

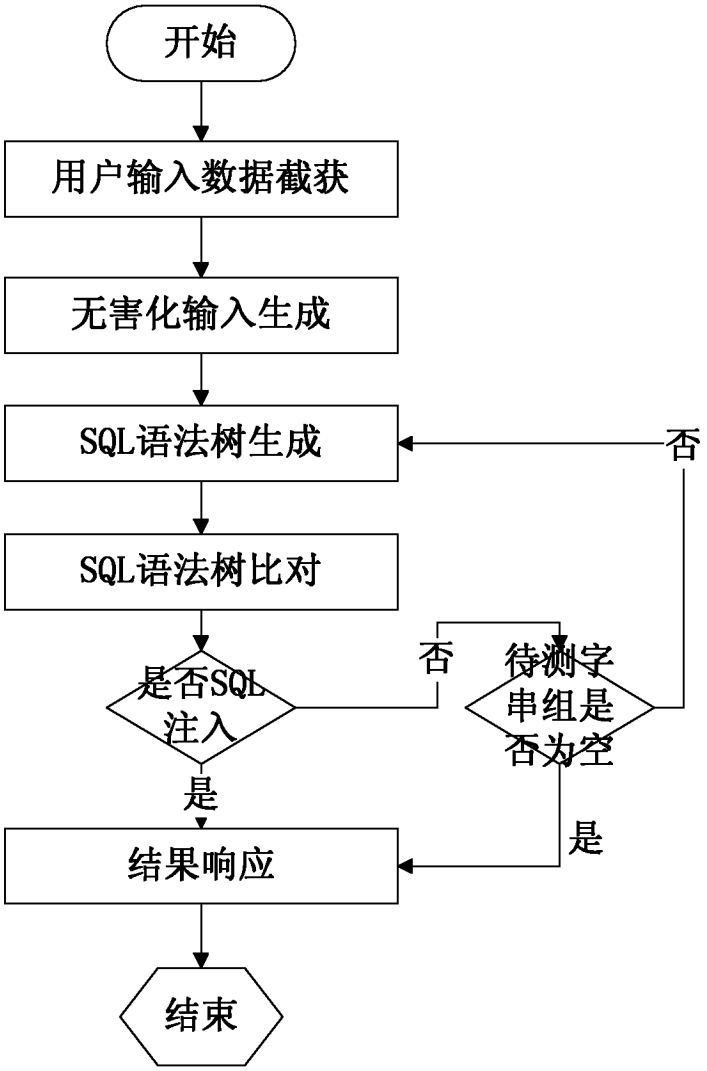

开始
用户输入数据截获
无害化输入生成
SQL语法树生成
SQL语法树比对
是否SQL注入
否
是
待测字串组是否为空
否
是
结果响应
结束

图 2

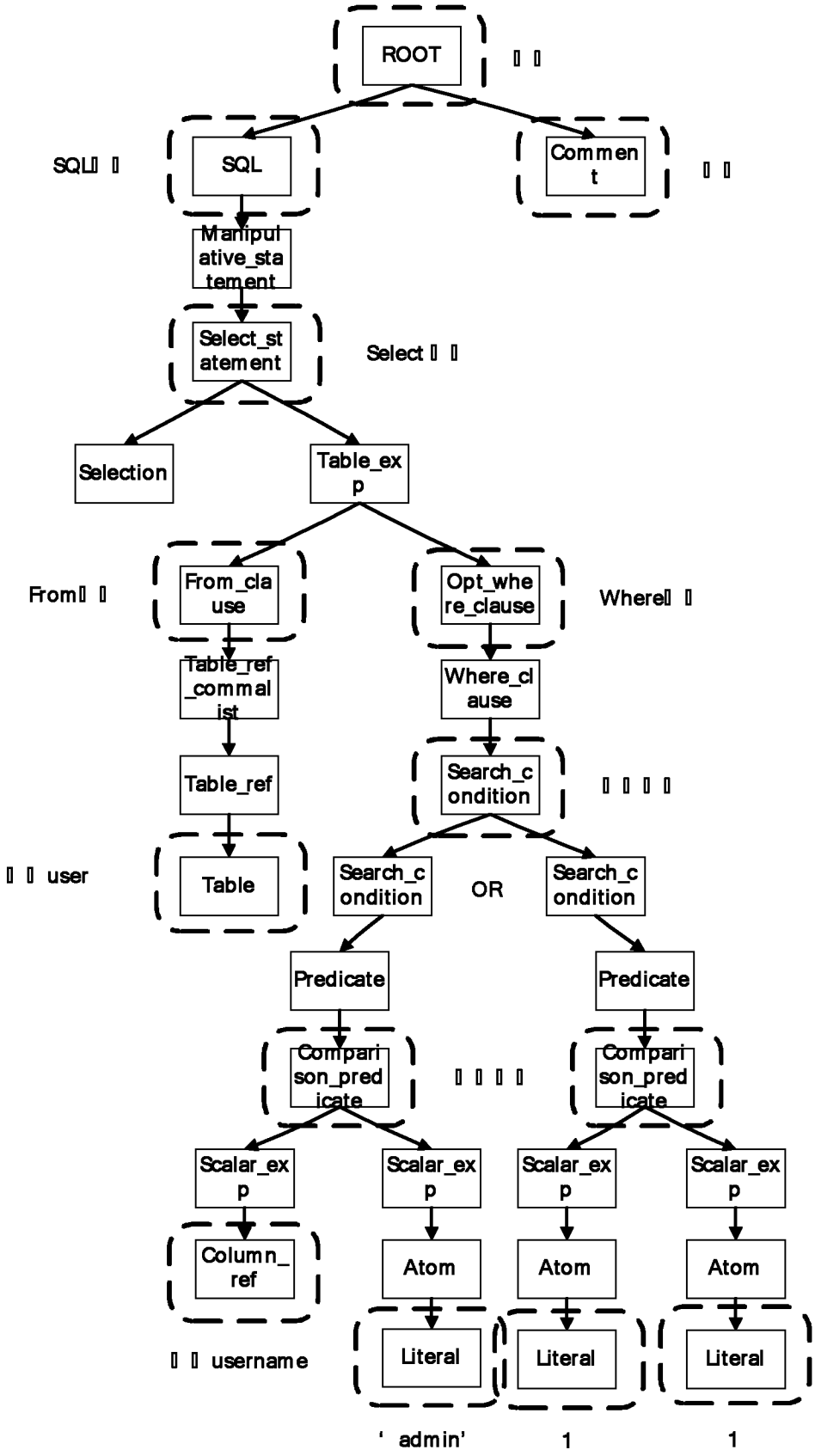

ROOT
SQL
Comment
Manipulative_statement
Select_statement
Selection
Table_exp
From_clause
Opt_where_clause
Table_ref_commalist
Where_clause
Table_ref
Search_condition
Table
Search_condition
OR
Search_condition
Predicate
Predicate
Comparison_predicate
Comparison_predicate
Scalar_exp
Scalar_exp
Scalar_exp
Scalar_exp
Column_ref
Atom
Atom
Atom
Literal
Literal
Literal
' admin'
1
1

图 3

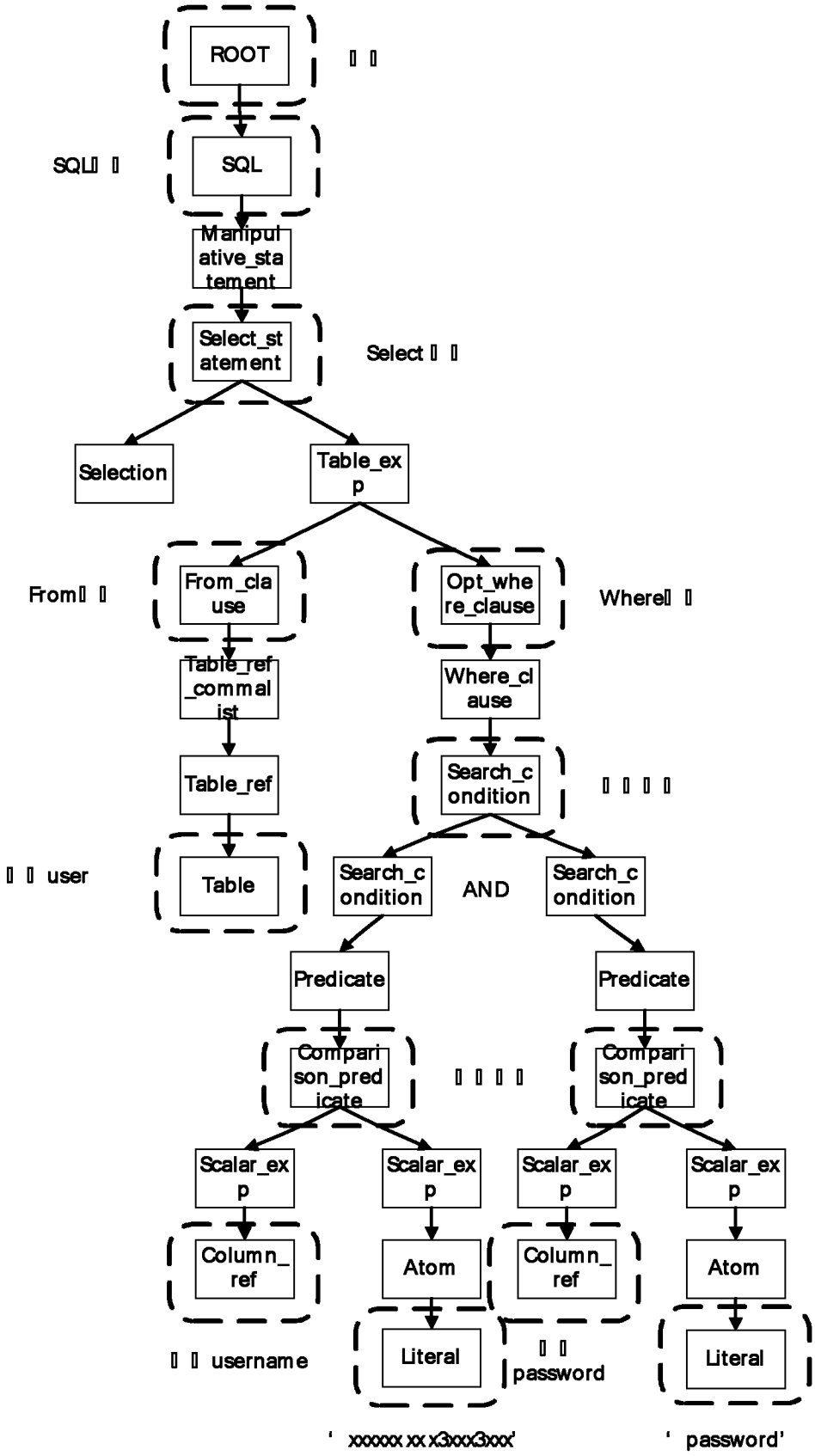

ROOT
SQL
Manipulative_statement
Select_statement
Selection
Table_exp
From_clause
Opt_where_clause
Table_ref_commalist
Where_clause
Table_ref
Search_condition
Table
Search_condition
AND
Search_condition
Predicate
Predicate
Comparison_predicate
Comparison_predicate
Scalar_exp
Scalar_exp
Scalar_exp
Scalar_exp
Column_ref
Atom
Column_ref
Atom
Literal
Literal
username
password
'xxxxxxxxx3xxx3xxx'
'password'

图 4

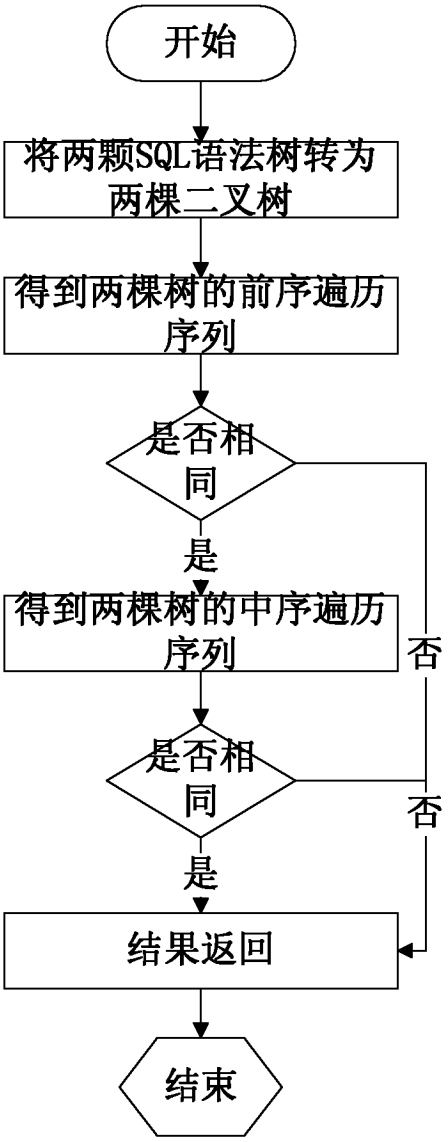

开始
将两颗SQL语法树转为两棵二叉树
得到两棵树的前序遍历序列
是否相同
是
得到两棵树的中序遍历序列
否
是否相同
否
是
结果返回
结束

图 5

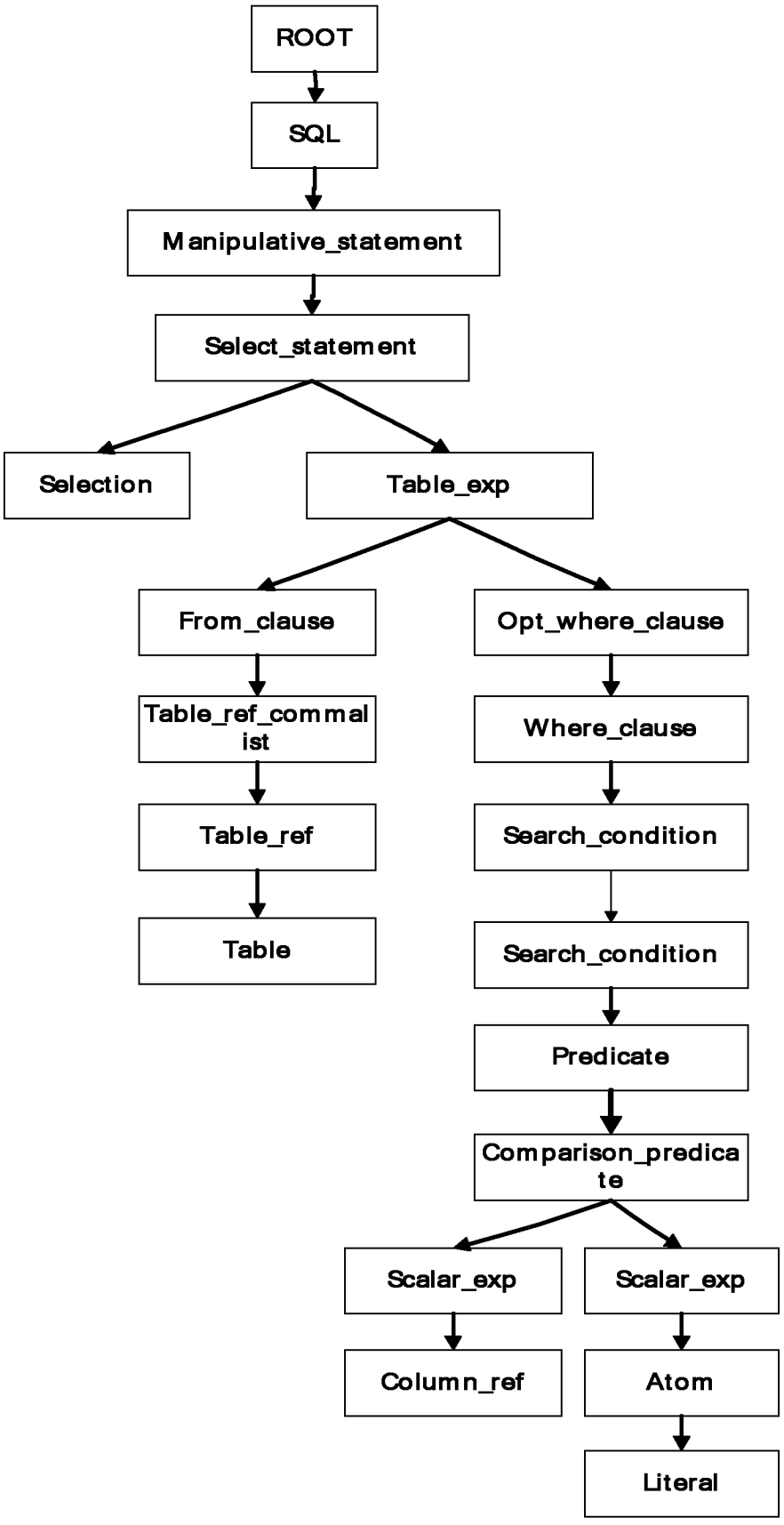

ROOT
SQL
Manipulative_statement
Select_statement
Selection
Table_exp
From_clause
Opt_where_clause
Table_ref_commalist
Where_clause
Table_ref
Search_condition
Table
Search_condition
Predicate
Comparison_predicate
Scalar_exp
Scalar_exp
Column_ref
Atom
Literal

图 6

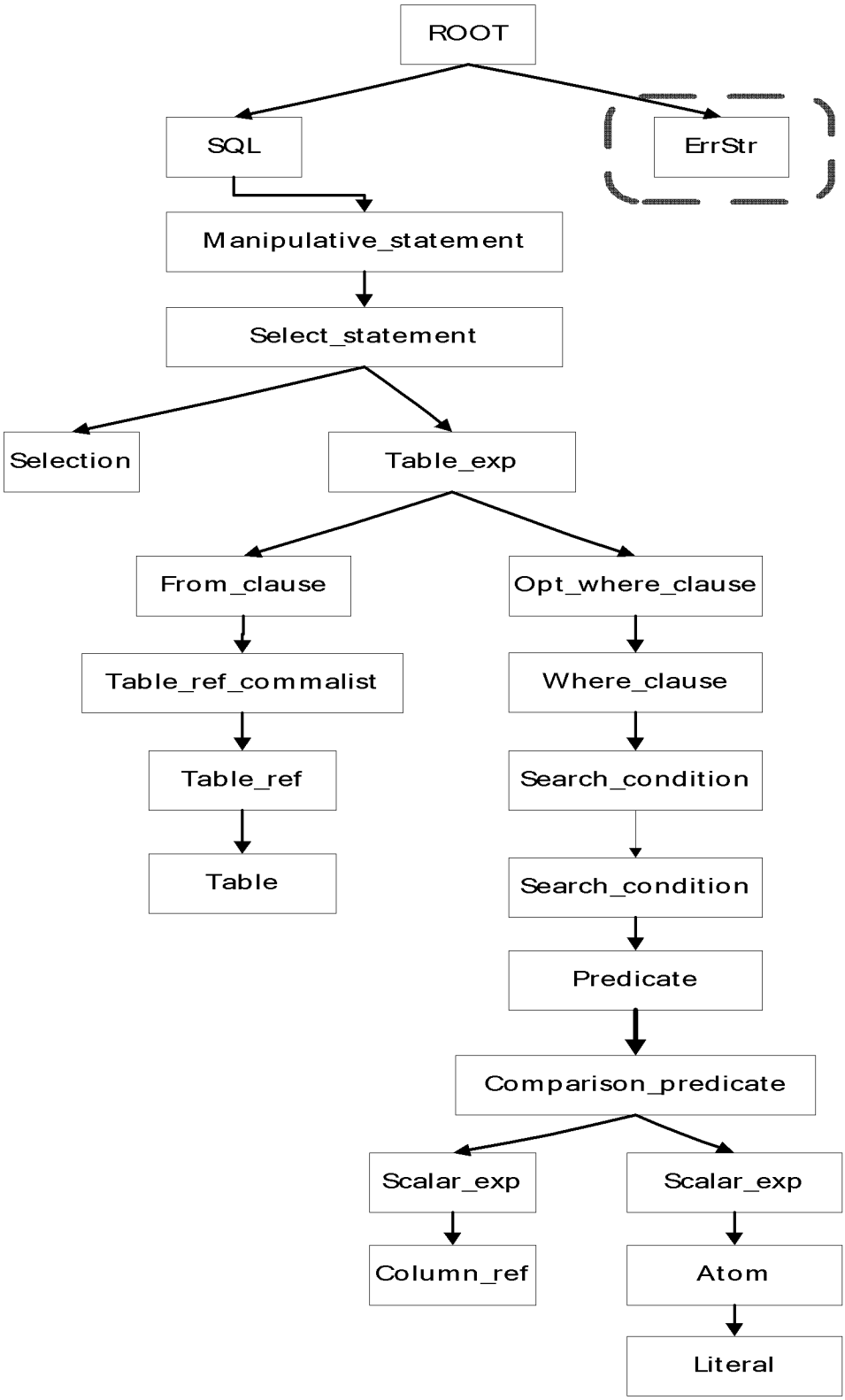

ROOT
SQL
ErrStr
Manipulative_statement
Select_statement
Selection
Table_exp
From_clause
Opt_where_clause
Table_ref_commalist
Where_clause
Table_ref
Search_condition
Table
Search_condition
Predicate
Comparison_predicate
Scalar_exp
Scalar_exp
Column_ref
Atom
Literal

图 7