(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 114936369 B
(45)授权公告日 2024.04.19

(21)申请号 202210438587.9

(22)申请日 2022.04.25

(65)同一申请的已公布的文献号
申请公布号 CN 114936369 A

(43)申请公布日 2022.08.23

(73)专利权人 中山大学
地址 510175 广东省广州市海珠区新港西路135号
专利权人 国网江苏省电力有限公司

(72)发明人 张涛 潘梦源 黄海东

(74)专利代理机构 广州市华学知识产权代理有限公司 44245
专利代理师 林梅繁

(51)Int.Cl.
G06F 21/56(2013.01)
G06F 21/55(2013.01)
G06F 16/22(2019.01)
G06F 16/242(2019.01)

(56)对比文件
CN 106355094 A,2017.01.25
CN 108712448 A,2018.10.26
CN 108875366 A,2018.11.23
US 2007130620 A1,2007.06.07
张慧琳;丁羽;张利华;段镭;张超;韦韬;李冠成;韩心慧.基于敏感字符的SQL注入攻击防御方法.计算机研究与发展.2016,(第10期),全文.

审查员 王亚辉

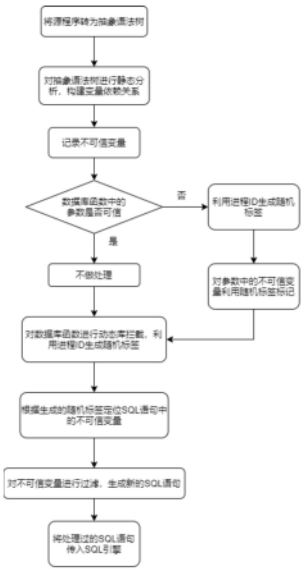
权利要求书2页 说明书6页 附图2页

(54)发明名称

基于标记的SQL注入攻击主动防御方法、系统及存储介质

(57)摘要

本发明涉及网络安全技术领域,为基于标记的SQL注入攻击主动防御方法、系统及存储介质。其方法包括:将源代码转换为抽象语法树,定义不可信变量;对抽象语法树中的变量自上至下逐行进行分析,分析所有变量的赋值引用关系,构建变量依赖关系,定位源代码中的所有不可信变量;对抽象语法树进行分析,判断数据库参数是否可信,判断SQL语句是否含有不可信变量;若SQL语句含有不可信变量,则对源代码进行修改,标记不可信变量;在数据库函数中加入对不可信变量过滤处理的代码;运行修改后的源代码,对数据库函数进行动态库拦截,识别出不可信输入的字符串部分,对不可信输入的字符串部分的检查和过滤,实现SQL注入攻击语句的运行失效。



CN 114936369 B

1.基于标记的SQL注入攻击主动防御方法,其特征在于,包括以下步骤:

将源代码转换为抽象语法树,并定义一系列不可信变量,构建不可信变量列表;

对抽象语法树中的变量自上至下逐行进行分析,分析抽象语法树中所有变量的赋值引用关系,构建变量依赖关系,从而定位源代码中的所有不可信变量,并将不可信变量记录在不可信变量列表中;

对不可信变量分析完毕后,再次对抽象语法树进行分析,寻找其中的数据库函数,对数据库函数中的参数进行分析,判断参数是否可信,进而判断SQL语句是否含有不可信变量;

若SQL语句含有不可信变量,则对源代码进行修改,标记数据库函数参数中的不可信变量;

对源代码修改完毕后,重新编写数据库函数,在数据库函数中加入对不可信变量过滤处理的代码;

当运行修改后的源代码时,对数据库函数进行动态库拦截,根据标记识别出SQL语句中来自不可信输入的字符串部分,并根据设定的过滤策略对不可信输入的字符串部分的检查和过滤,实现SQL注入攻击语句的运行失效。

2.如权利要求1所述的SQL注入攻击主动防御方法,其特征在于,在不可信变量的定位过程中,从数据库访问函数开始搜索,找出构成SQL语句的字符串中引用的不可信变量并对其添加标签,实现对SQL语句的字符串中所有不可信输入的位置标记。

3.如权利要求1所述的SQL注入攻击主动防御方法,其特征在于,过滤处理的过程包括:

利用当前进程ID生成标签;

利用生成的标签定位数据库函数中SQL语句的不可信变量;

对不可信变量按照设定的过滤策略进行过滤,除去其中可能的恶意代码;

利用过滤后的不可信变量构成新的SQL语句;

重新调用系统的数据库函数,将SQL语句传给SQL引擎,以执行相应数据库操作。

4.如权利要求3所述的SQL注入攻击主动防御方法,其特征在于,设定的过滤策略为:

通过对互联网中存在的常见SQL注入攻击代码进行总结,得到通用的SQL注入攻击的语句构成模式;基于所述SQL注入攻击的语句构成模式,利用决策树对SQL语句的输入长度、特殊符号信息进行检查,实现对不可信变量的过滤。

5.如权利要求1所述的SQL注入攻击主动防御方法,其特征在于,当对数据库函数进行动态库拦截时,每次调用数据库函数时调用重新编写的数据库函数而非系统的数据库函数,使得SQL语句每次都得到过滤处理。

6.基于标记的SQL注入攻击主动防御系统,其特征在于,包括:

列表构建模块,用于将源代码转换为抽象语法树,并定义一系列不可信变量,构建不可信变量列表;

变量分析模块,用于对抽象语法树中的变量自上至下逐行进行分析,分析抽象语法树中所有变量的赋值引用关系,构建变量依赖关系,从而定位源代码中的所有不可信变量,并将不可信变量记录在不可信变量列表中;

参数分析判断模块,用于对不可信变量分析完毕后,再次对抽象语法树进行分析,寻找其中的数据库函数,对数据库函数中的参数进行分析,判断参数是否可信,进而判断SQL语句是否含有不可信变量;

变量标记模块,用于当SQL语句含有不可信变量时,对源代码进行修改,标记数据库函数参数中的不可信变量;

代码修改模块,用于对源代码修改完毕后,重新编写数据库函数,在数据库函数中加入对不可信变量过滤处理的代码;

动态库拦截模块,用于当运行修改后的源代码时,对数据库函数进行动态库拦截,根据标记识别出SQL语句中来自不可信输入的字符串部分,并根据设定的过滤策略对不可信输入的字符串部分的检查和过滤,实现SQL注入攻击语句的运行失效。

7.如权利要求6所示的SQL注入攻击主动防御系统,其特征在于,变量分析模块在不可信变量的定位过程中,从数据库访问函数开始搜索,找出构成SQL语句的字符串中引用的不可信变量并对其添加标签,实现对SQL语句的字符串中所有不可信输入的位置标记。

8.如权利要求6所示的SQL注入攻击主动防御系统,其特征在于,代码修改模块在数据库函数中加入的代码对过滤处理的过程包括:

利用当前进程ID生成标签;

利用生成的标签定位数据库函数中SQL语句的不可信变量;

对不可信变量按照设定的过滤策略进行过滤,除去其中可能的恶意代码;

利用过滤后的不可信变量构成新的SQL语句;

重新调用系统的数据库函数,将SQL语句传给SQL引擎,以执行相应数据库操作。

9.如权利要求8所示的SQL注入攻击主动防御系统,其特征在于,设定的过滤策略为:

通过对互联网中存在的常见SQL注入攻击代码进行总结,得到通用的SQL注入攻击的语句构成模式;基于所述SQL注入攻击的语句构成模式,利用决策树对SQL语句的输入长度、特殊符号信息进行检查,实现对不可信变量的过滤。

10.存储介质,其上存储有计算机指令,其特征在于,所述计算机指令被处理器执行时,实现权利要求1-5中任一项所述SQL注入攻击主动防御方法的步骤。

基于标记的SQL注入攻击主动防御方法、系统及存储介质

技术领域

[0001] 本发明涉及网络安全技术领域,为基于标记的SQL注入攻击主动防御方法、系统及存储介质。

背景技术

[0002] 结构化查询语言(StructuredQueryLanguage)简称SQL,是一种特殊目的的编程语言,是一种数据库查询和程序设计语言,用于存取数据以及查询、更新和管理关系数据库系统,被广泛应用于类似MicrosoftAccess、DB2、Informix、MicrosoftSQLServer、Oracle、Sybase以及其他的数据库系统中。

[0003] 由于一些应用程序对用户输入数据的合法性没有判断或过滤不严,攻击者可以在应用程序中事先定义好的查询语句的结尾上添加额外的SQL语句,在管理员不知情的情况下实现非法操作,以此来实现欺骗数据库服务器执行非授权的任意查询,从而进一步得到相应的数据信息。

[0004] 目前,代码注入攻击的安全防护方法主要源于程序分析和输入规则匹配两种技术思路,比如识别SQL语句中的危险符号并进行过滤;对程序中SQL语句中的关键字进行随机化处理,以达到和用户输入相区别的方法等。这些方法通常需要对利用方式、攻击行为进行分析,并采取对应的防护措施,但在面对新型的、未知的攻击行为或是当攻击者有防护系统的先验知识时存在着固有缺陷。

发明内容

[0005] 为了能够更好地应对未知攻击行为,实现主动防御,本发明提出了基于标记的SQL注入攻击主动防御方法、系统及存储介质,适用于多语言环境,利用污点分析的原理对服务器源代码进行分析,对不可信变量进行标记处理。

[0006] 本发明防御方法所采取的技术方案为:基于标记的SQL注入攻击主动防御方法,包括以下步骤:

[0007] 将源代码转换为抽象语法树,并定义一系列不可信变量,构建不可信变量列表;

[0008] 对抽象语法树中的变量自上至下逐行进行分析,分析抽象语法树中所有变量的赋值引用关系,构建变量依赖关系,从而定位源代码中的所有不可信变量,并将不可信变量记录在不可信变量列表中;

[0009] 对不可信变量分析完毕后,再次对抽象语法树进行分析,寻找其中的数据库函数,对数据库函数中的参数进行分析,判断参数是否可信,进而判断SQL语句是否含有不可信变量;

[0010] 若SQL语句含有不可信变量,则对源代码进行修改,标记数据库函数参数中的不可信变量;

[0011] 对源代码修改完毕后,重新编写数据库函数,在数据库函数中加入对不可信变量过滤处理的代码;

[0012] 当运行修改后的源代码时,对数据库函数进行动态库拦截,根据标记识别出SQL语句中来自不可信输入的字符串部分,并根据设定的过滤策略对不可信输入的字符串部分的检查和过滤,实现SQL注入攻击语句的运行失效。

[0013] 本发明防御系统所采取的技术方案为:基于标记的SQL注入攻击主动防御系统,包括:

[0014] 列表构建模块,用于将源代码转换为抽象语法树,并定义一系列不可信变量,构建不可信变量列表;

[0015] 变量分析模块,用于对抽象语法树中的变量自上至下逐行进行分析,分析抽象语法树中所有变量的赋值引用关系,构建变量依赖关系,从而定位源代码中的所有不可信变量,并将不可信变量记录在不可信变量列表中;

[0016] 参数分析判断模块,用于对不可信变量分析完毕后,再次对抽象语法树进行分析,寻找其中的数据库函数,对数据库函数中的参数进行分析,判断参数是否可信,进而判断SQL语句是否含有不可信变量;

[0017] 变量标记模块,用于当SQL语句含有不可信变量时,对源代码进行修改,标记数据库函数参数中的不可信变量;

[0018] 代码修改模块,用于对源代码修改完毕后,重新编写数据库函数,在数据库函数中加入对不可信变量过滤处理的代码;

[0019] 动态库拦截模块,用于当运行修改后的源代码时,对数据库函数进行动态库拦截,根据标记识别出SQL语句中来自不可信输入的字符串部分,并根据设定的过滤策略对不可信输入的字符串部分的检查和过滤,实现SQL注入攻击语句的运行失效。

[0020] 本发明的存储介质,其上存储有计算机指令,当计算机指令被处理器执行时,实现本发明SQL注入攻击主动防御方法的各步骤。

[0021] 本发明与现有技术相比,其取得的技术效果包括:

[0022] 1、本发明采取了静态分析方法对源码进行分析,对SQL语句中的不可信变量进行标记,从而可以准确定位SQL语句中的不可信变量;再根据标记定位不可信变量并动态过滤有害输入,防止SQL注入;避免了对可信变量的错误过滤,并且能更好地针对不可信变量进行过滤处理,提高了准确性。

[0023] 2、相对于传统的静态分析方法对所有SQL语句进行处理,本发明只对包含不可信变量的SQL语句进行处理,一定程度上减小了过滤处理的运行时开销。

[0024] 3、本发明利用抽象语法树进行静态分析,该分析方法可以适用于多种语言。

[0025] 4、本发明采用了动态库拦截的方式对数据库函数进行拦截,可以更好地适用于多语言复杂使用场景,拥有更强的普适性。

附图说明

[0026] 图1为本发明实施例中防御方法的流程图;

[0027] 图2为本发明实施例中不可信变量过滤模式的示意图;

[0028] 图3为本发明实施例中防御系统的结构示意图。

具体实施方式

[0029] 总的来说,本发明利用污点分析的原理对服务器源代码进行分析,对不可信变量进行标记处理。污点分析是一种跟踪并分析污点信息在程序中流动的技术。在漏洞分析中,使用污点分析技术将所感兴趣的数据标记为污点数据,然后通过跟踪和污点数据相关的信息的流向,可以知道它们是否会影响某些关键的程序操作,进而挖掘程序漏洞。

[0030] 本发明的主要过程为:首先将数据库应用的源代码转化为抽象语法树(Abstractsyntaxtree,AST),通过对AST进行静态分析和源代码修改,标记出SQL语句中使用的不可信输入变量;之后对程序中的数据库访问函数进行运行时动态库拦截(Hook),根据通用规则对SQL语句中标记出来的不可信输入字符串进行有效性判定和过滤,实现SQL注入攻击语句的运行失效。

[0031] 下面结合实施例和附图对本发明的实施方式做进一步的说明,但本发明的实施方式不限于此。

[0032] 实施例1

[0033] 本实施例提供的是基于标记的SQL注入攻击主动防御方法,如图1所示,包括以下步骤:

[0034] S1、将数据库应用的源代码转换为抽象语法树,并定义一系列污点源,如$_GET、$_POST等用户输入变量,将其设置为不可信变量,构建不可信变量列表。

[0035] S2、对抽象语法树中的变量自上至下逐行进行分析,分析抽象语法树中所有变量的赋值引用关系,构建变量依赖关系,也即跟踪污点数据的流动传递过程,从而定位源代码中所有不可信变量,并将不可信变量记录在不可信变量列表中。

[0036] 在不可信变量的定位过程中,可以从数据库访问函数开始搜索,找出构成SQL语句的字符串中引用的不可信变量并对其添加标签,实现对SQL语句的字符串中所有不可信输入的位置标记。

[0037] 如果一个变量由另一个不可信变量赋值而来,则该变量同样可视为不可信变量。例如“$name=$_GET[‘name’]”,$name变量由$_GET变量赋值而来,因此将其加入不可信变量列表。

[0038] S3、对不可信变量分析完毕后,再次对抽象语法树进行分析,寻找其中的数据库函数,如mysqli_query函数,对数据库函数中的参数进行分析,判断参数是否可信,进而判断SQL语句是否含有不可信变量。

[0039] 数据库函数是SQL注入攻击的攻击点,恶意注入代码往往通过数据库函数传向数据库并执行相应操作,而数据库函数中的SQL语句通常由用户输入构成,是引发SQL注入攻击的根源。

[0040] 对数据库函数中的SQL语句进行分析,如果为不可信变量,则说明该SQL语句依赖用户输入,为潜在攻击点。例如$sql=“select*fromtablewherename=‘$name’”,$sql为SQL语句,其由两部分构成,字符串常量select*fromtablewherename=‘’和不可信变量$name构成,因此该SQL语句含有不可信变量;如果SQL语句为可信变量,则说明该SQL语句为程序内置变量或常量构成,不会成为攻击点。

[0041] S4、若SQL语句含有不可信变量,则对源代码进行修改,标记数据库函数参数中的不可信变量。

[0042] 如果SQL语句含有不可信变量,则需要对SQL语句进行处理,对程序(即源代码)进行修改,利用进程ID作标签,标记其中的用户输入部分,构成新的SQL语句,例如上述变量$sql,对其进行处理后可以得到$sql＝“select*fromtablewherename＝‘<$pid>$name<$pid>’”,其中<$pid><$pid>为标签,$pid为利用进程ID生成的变量。当运行代码时,会生成新进程,进程ID每次都是随机生成的,因此可以有效避免攻击者利用先验知识对防御系统进行绕过欺骗。

[0043] S5、对源代码修改完毕后,表示静态分析部分已完成;然后重新编写数据库函数,在数据库函数中加入对不可信变量过滤处理的代码。

[0044] 其中,过滤处理的过程包括:

[0045] 利用当前进程ID生成标签;

[0046] 利用生成的标签定位数据库函数中SQL语句的不可信变量;

[0047] 对不可信变量按照设定的过滤策略进行过滤,除去其中可能的恶意代码;

[0048] 其中,设定的过滤策略为:通过对互联网中存在的常见SQL注入攻击代码进行总结,得到一套通用的SQL注入攻击的语句构成模式,比如一个攻击语句往往都会包含注释符、SQL关键字(例如INSERT、DELETE等)等;基于所述SQL注入攻击的语句构成模式,利用决策树对SQL语句的输入长度、特殊符号等信息进行检查,实现对不可信变量的过滤,从而除去其中可能的恶意代码;一个简单的过滤模式如图2所示,用户可以根据程序生成的过滤日志对过滤情况进行检查,并适当的修改过滤策略,以减小误报率和漏报率;

[0049] 利用过滤后的不可信变量构成新的SQL语句;

[0050] 重新调用系统的数据库函数,将SQL语句传给SQL引擎,以执行相应数据库操作。

[0051] S6、当运行修改后的源代码时,实现数据库访问函数的运行时拦截,即对数据库函数进行动态库拦截,根据标记识别出SQL语句中来自不可信输入的字符串部分,并根据设定的过滤策略对不可信输入字符串部分的检查和过滤,实现SQL注入攻击语句的运行失效。

[0052] 当对数据库函数进行动态库拦截时,可以使每次调用数据库函数时调用重新编写的数据库函数而非系统的数据库函数,使得SQL语句每次都可以得到过滤处理,从而可以较好地避免SQL注入攻击。

[0053] 本步骤通过动态库拦截的方式,一方面可以在不对系统数据库函数进行修改的条件下对输入进行过滤,避免用户修改系统数据库函数造成未知影响,使得系统稳定性、便捷性得到提升。另一方面,由于大部分脚本语言如Python、PHP等为了提高运行速度,会调用C语言动态库进行处理,因此动态库拦截可以在只修改拦截C语言动态库的情况下,实现同时对多种语言的处理,因此能够适用于更加复杂的使用场景,提升了本发明技术方案的普适性。

[0054] 例如用户输入一段恶意代码,使得用户输入变量$name＝“ or1＝1#”,如果不对该SQL语句进行修改过滤,传入数据库函数中的SQL语句为select*fromtablewherename＝‘ or1＝1#’,该SQL语句为重言式,执行该语句可以获取table表中的所有数据而不需要输入正确的参数,从而完成了一次SQL攻击。而进行代码修改之后传入数据库函数中的SQL语句为select*fromtablewherename＝‘<$pid>’or1＝1#<$pid>’,对数据库函数进行动态库拦截,并根据标签对内部内容进行过滤,可以得到新的SQL语句select*fromtablewherename＝‘’,完成了对SQL语句中的恶意代码的过滤,从而避免了SQL攻击的发生。

[0055] 实施例2

[0056] 本实施例与实施例1基于相同的发明构思,提供的是基于标记的SQL注入攻击主动防御系统,包括以下模块:

[0057] 列表构建模块,用于将源代码转换为抽象语法树,并定义一系列不可信变量,构建不可信变量列表;

[0058] 变量分析模块,用于对抽象语法树中的变量自上至下逐行进行分析,分析抽象语法树中所有变量的赋值引用关系,构建变量依赖关系,从而定位源代码中的所有不可信变量,并将不可信变量记录在不可信变量列表中;

[0059] 参数分析判断模块,用于对不可信变量分析完毕后,再次对抽象语法树进行分析,寻找其中的数据库函数,对数据库函数中的参数进行分析,判断参数是否可信,进而判断SQL语句是否含有不可信变量;

[0060] 变量标记模块,用于当SQL语句含有不可信变量时,对源代码进行修改,标记数据库函数参数中的不可信变量;

[0061] 代码修改模块,用于对源代码修改完毕后,重新编写数据库函数,在数据库函数中加入对不可信变量过滤处理的代码;

[0062] 动态库拦截模块,用于当运行修改后的源代码时,对数据库函数进行动态库拦截,根据标记识别出SQL语句中来自不可信输入的字符串部分,并根据设定的过滤策略对不可信输入的字符串部分的检查和过滤,实现SQL注入攻击语句的运行失效。

[0063] 其中,变量分析模块在不可信变量的定位过程中,从数据库访问函数开始搜索,找出构成SQL语句的字符串中引用的不可信变量并对其添加标签,实现对SQL语句的字符串中所有不可信输入的位置标记。

[0064] 而代码修改模块在数据库函数中加入的代码对过滤处理的过程包括:

[0065] 利用当前进程ID生成标签;

[0066] 利用生成的标签定位数据库函数中SQL语句的不可信变量;

[0067] 对不可信变量按照设定的过滤策略进行过滤,除去其中可能的恶意代码;

[0068] 利用过滤后的不可信变量构成新的SQL语句;

[0069] 重新调用系统的数据库函数,将SQL语句传给SQL引擎,以执行相应数据库操作。

[0070] 代码修改模块中,设定的过滤策略为:

[0071] 通过对互联网中存在的常见SQL注入攻击代码进行总结,得到通用的SQL注入攻击的语句构成模式;基于所述SQL注入攻击的语句构成模式,利用决策树对SQL语句的输入长度、特殊符号信息进行检查,实现对不可信变量的过滤。

[0072] 如图3所示,本实施例的防御系统在实际应用中,变量分析模块、参数分析判断模块和变量标记模块可呈现为源代码静态分析与标记工具,实现数据库应用的原始源代码的分析与不可信变量的标记,得到数据库应用标记后源代码;代码修改模块对标记后源代码进行编译执行或解释执行;动态库拦截模块生成包含过滤规则的数据库函数动态库,并使其调用优先级高于系统动态库,使得应用程序在调用数据库函数时优先调用自己编写的动态库,从而利用编写的数据库函数实现对输入的过滤。

[0073] 本实施例还提供存储介质,其上存储有计算机指令,当计算机指令被处理器执行时,实现实施例1中SQL注入攻击主动防御方法的各步骤。

[0074] 以上所述是本发明的优选实施方式,应当指出,对于本技术领域的普通技术人来说,在不脱离本发明所述原理的前提下,还可以做出若干改进和润饰,这些改进和润饰应视为本发明的保护范围。

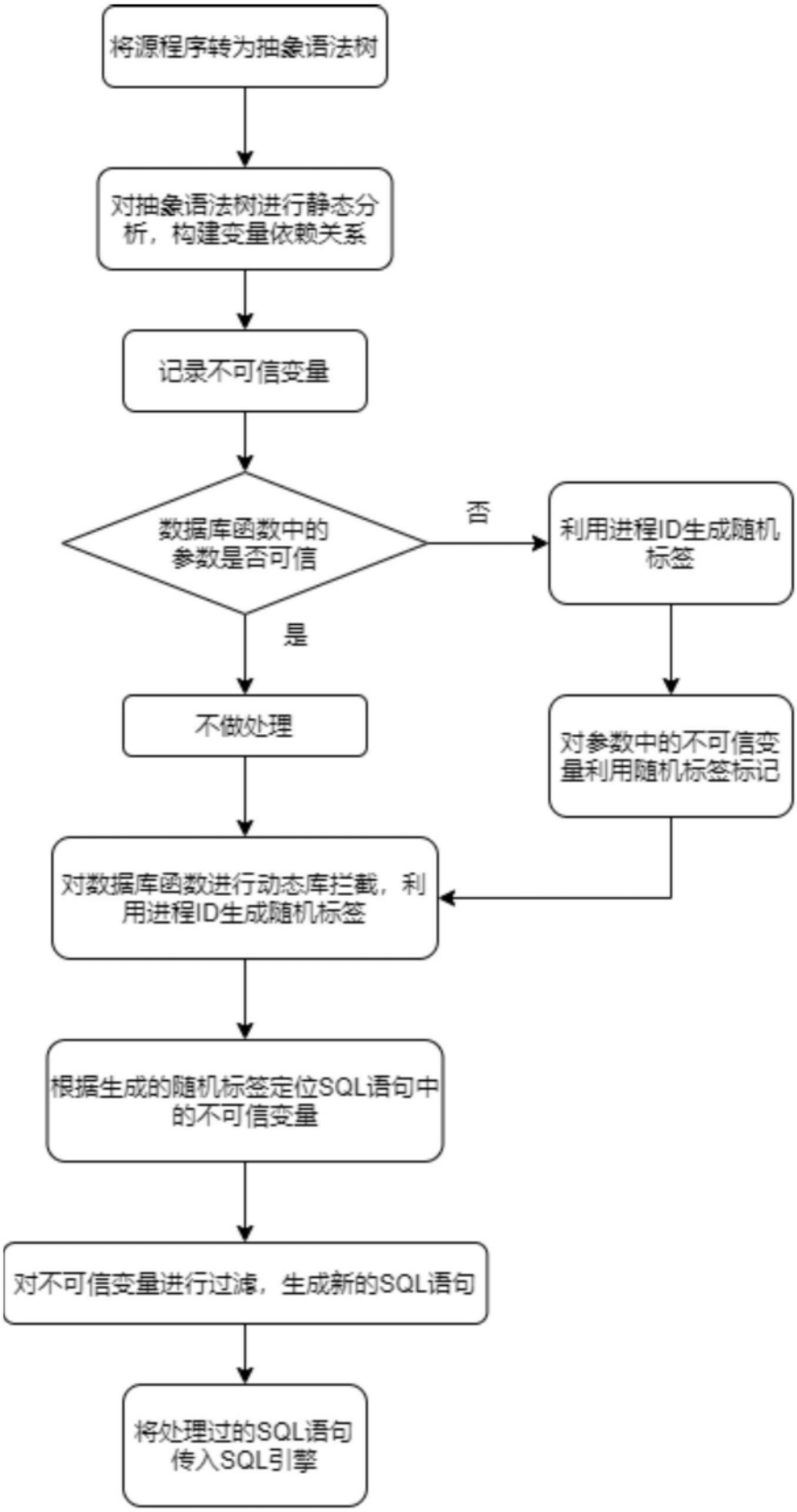

将源程序转为抽象语法树
对抽象语法树进行静态分析，构建变量依赖关系
记录不可信变量
数据库函数中的参数是否可信
否
是
利用进程ID生成随机标签
不做处理
对参数中的不可信变量利用随机标签标记
对数据库函数进行动态库拦截，利用进程ID生成随机标签
根据生成的随机标签定位SQL语句中的不可信变量
对不可信变量进行过滤，生成新的SQL语句
将处理过的SQL语句传入SQL引擎

图1

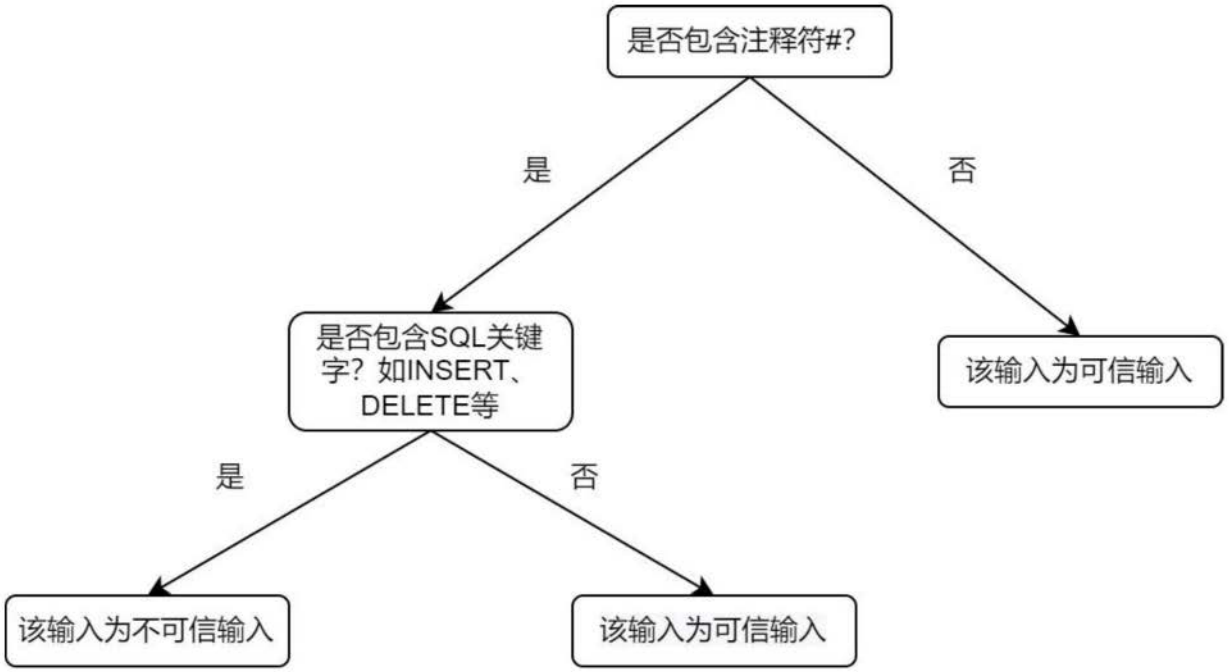

是否包含注释符#?
是
否
是否包含SQL关键字? 如INSERT、DELETE等
该输入为可信输入
是
否
该输入为不可信输入
该输入为可信输入

图2

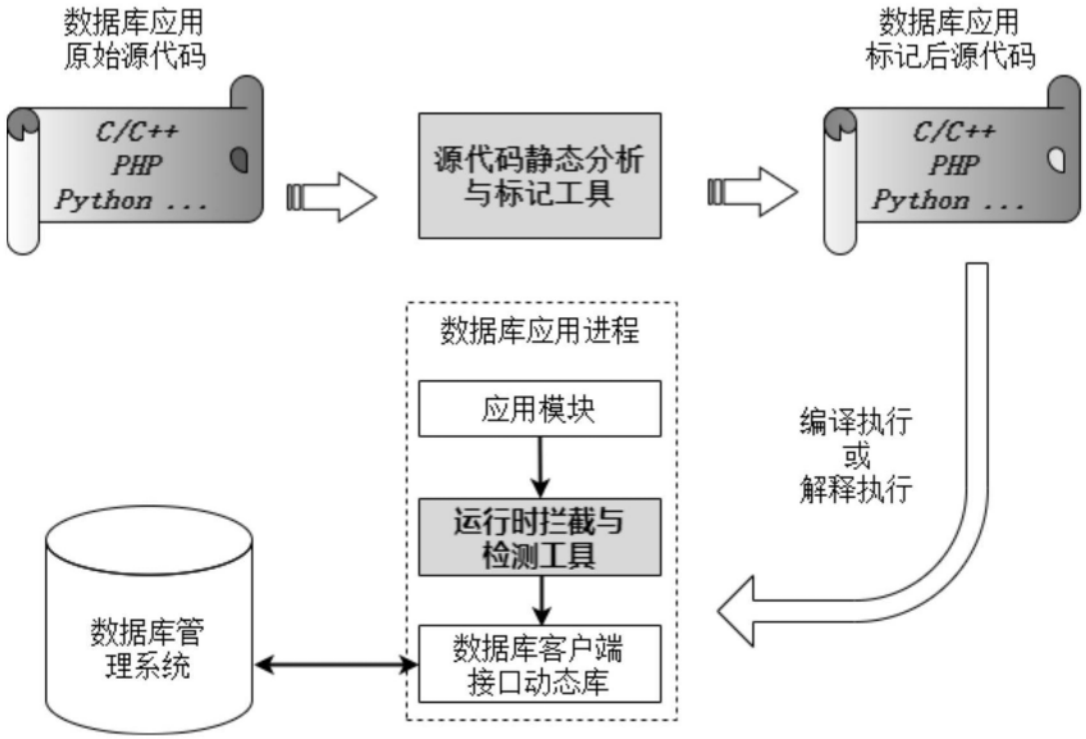

数据库应用
原始源代码
C/C++
PHP
Python ...
源代码静态分析与标记工具
数据库应用
标记后源代码
C/C++
PHP
Python ...
数据库应用进程
应用模块
运行时拦截与检测工具
数据库客户端接口动态库
数据库管理系统
编译执行
或
解释执行

图3