



(12) 发明专利申请

(10) 申请公布号 CN 104850493 A

(43) 申请公布日 2015. 08. 19

(21) 申请号 201510202537. 0

(22) 申请日 2015. 04. 24

(71) 申请人 百度在线网络技术(北京)有限公司

地址 100085 北京市海淀区上地十街 10 号

百度大厦

申请人 南开大学

(72) 发明人 过辰楷 李鸣雷 许静 周向荣

王丹 曾莹 白梦莹

(74) 专利代理机构 北京汉昊知识产权代理事务

所(普通合伙) 11370

代理人 罗朋 励向南

(51) Int. Cl.

G06F 11/36(2006. 01)

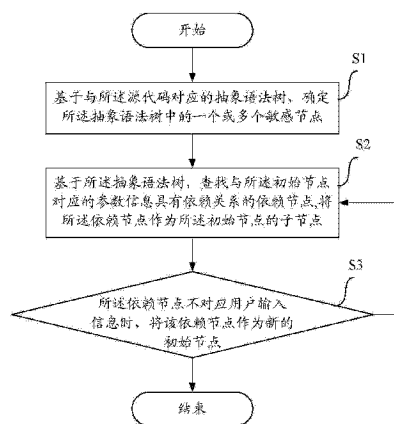
权利要求书2页 说明书11页 附图1页

(54) 发明名称

一种检测源代码漏洞的方法和装置

(57) 摘要

本发明的目的是提供一种检测源代码漏洞的方法和装置。根据本发明的方法包括:基于与所述源代码对应的抽象语法树,确定所述抽象语法树中的一个或多个敏感节点;基于所述抽象语法树,查找与所述初始节点对应的参数信息具有依赖关系的依赖节点,以将所述依赖节点作为所述初始节点的子节点;当所述依赖节点不对应用户输入信息时,将该依赖节点作为新的初始节点,并重复前述步骤。根据本发明的方案的优点在于:通过基于赋值操作等方式的分析,来建立基于数据的依赖关系的敏感路径信息,从而能够降低对于漏洞的误报率,并且能够提升所获得的敏感路径的完整度,提高漏洞检测的全面性。



1. 一种检测源代码漏洞的方法, 其中, 所述方法包括以下步骤:

x 基于与所述源代码对应的抽象语法树, 确定所述抽象语法树中的一个或多个敏感节点, 以将敏感节点作为初始节点, 其中, 所述敏感节点对应于所述源代码中的敏感函数的参数信息;

其中, 所述方法还包括以下步骤:

a 基于所述抽象语法树, 查找与所述初始节点对应的参数信息具有依赖关系的依赖节点, 以将所述依赖节点作为所述初始节点的子节点;

b 当所述依赖节点不对应用户输入信息时, 将该依赖节点作为新的初始节点, 并重复前述步骤 a 和 b。

2. 根据权利要求 1 所述的方法, 其中, 所述步骤 a 进一步包括以下步骤:

- 基于所述抽象语法树, 查找用于对所述初始节点对应的参数信息进行赋值的赋值信息, 以将所述赋值信息对应的节点作为依赖节点。

3. 根据权利要求 1 或 2 所述的方法, 其中, 所述方法还包括以下步骤:

- 所述依赖节点对应于用户输入信息时, 记录该依赖节点到其对应的敏感节点的敏感路径信息。

4. 根据权利要求 1 至 3 中任一项所述的方法, 其中, 对于所述抽象语法树中的分支语句, 所述方法进一步包括:

- 对分支语句的每一分支, 分别执行步骤 a 至 b。

5. 根据权利要求 1 至 4 中任一项所述的方法, 其中, 对于所述抽象语法树中的循环语句, 所述方法还包括:

- 基于当前的初始节点, 在当前循环条件下执行步骤 a 至 b。

6. 根据权利要求 1 至 5 中任一项所述的方法, 其中, 所述步骤 x 进一步包括以下步骤:

- 基于预定的危险函数列表, 与抽象语法树中的各个节点进行正则运算, 以确定所述抽象语法树中各个节点是否为敏感节点。

7. 根据权利要求 3 至 5 中任一项所述的方法, 其中, 所述方法还包括以下步骤:

- 对于一敏感路径信息, 获取与该路径信息对应的敏感函数;

- 基于预定安全规则来对所述敏感函数进行验证;

- 当所述敏感函数符合所述安全规则时, 确定所述敏感路径信息非漏洞。

8. 根据权利要求 1 至 7 中任一项所述的方法, 其中, 将敏感函数作为初始函数, 所述方法还包括以下步骤:

m 基于源代码中的函数调用关系, 确定调用所述初始函数的一个或多个待选函数;

n 对所述一个或多个待选函数, 分别判断所述一个或多个待选函数中的各个待选函数的参数信息与所述初始函数的参数信息是否存在依赖关系;

o 当待选函数的参数信息与初始函数的参数信息存在依赖关系时, 保存所述待选函数, 并将该待选函数作为初始函数, 重复执行步骤 m 至 o。

9. 根据权利要求 8 所述的方法, 其中, 所述方法还包括以下步骤:

- 基于所保存的待选函数更新危险函数列表。

10. 一种检测源代码漏洞的检测装置, 其中, 所述检测装置包括:

第一确定装置, 用于基于与所述源代码对应的抽象语法树, 确定所述抽象语法树中的

一个或多个敏感节点,以将敏感节点作为初始节点,其中,所述敏感节点对应于所述源代码中的敏感函数的参数信息;

其中,所述检测装置还包括:

查找装置,用于基于所述抽象语法树,查找与所述初始节点对应的参数信息具有依赖关系的依赖节点,并将所述依赖节点作为所述初始节点的子节点;

第一控制装置,用于当所述依赖节点不对应用户输入信息时,将所述子节点作为新的初始节点,并重复查找装置和第一控制装置的操作。

11. 根据权利要求 10 所述的检测装置,其中,所述查找装置进一步 用于:

- 基于所述抽象语法树,查找用于对所述初始节点对应的参数信息进行赋值的赋值信息,以将所述赋值信息对应的节点作为依赖节点。

12. 根据权利要求 10 或 11 所述的检测装置,其中,所述检测装置还包括:

记录装置,用于所述依赖节点对应于用户输入信息时,记录该依赖节点到其对应的敏感节点的敏感路径信息。

13. 根据权利要求 10 至 12 中任一项所述的检测装置,其中,对于所述抽象语法树中的分支语句,所述检测装置进一步用于:

- 对分支语句的每一分支,分别执行查找装置和第一控制装置的操作。

14. 根据权利要求 10 至 13 中任一项所述的检测装置,其中,对于所述抽象语法树中的循环语句,所述检测装置还包括:

- 基于当前的初始节点,在当前循环条件下执行查找装置和第一控制装置的操作。

15. 根据权利要求 10 至 14 中任一项所述的检测装置,其中,所述第一确定装置进一步包括用于:

- 基于预定的危险函数列表,与抽象语法树中的各个节点进行正则运算,以确定所述抽象语法树中各个节点是否为敏感节点。

16. 根据权利要求 12 至 15 中任一项所述的检测装置,其中,所述检测装置还包括:

获取装置,用于对于一敏感路径信息,获取与该路径信息对应的敏感函数;

验证装置,用于基于预定安全规则来对所述敏感函数进行验证;

第二确定装置,用于当所述敏感函数符合所述安全规则时,确定所述敏感路径信息非漏洞。

17. 根据权利要求 10 至 16 中任一项所述的检测装置,其中,将敏感函数作为初始函数,所述检测装置还包括:

第三确定装置,用于基于源代码中的函数调用关系,确定调用所述 初始函数的一个或多个待选函数;

判断装置,用于对所述一个或多个待选函数,分别判断所述一个或多个待选函数中的各个待选函数的参数信息与所述初始函数的参数信息是否存在依赖关系;

第二控制装置,用于当待选函数的参数信息与初始函数的参数信息存在依赖关系时,确定改待选函数为所述初始函数的依赖函数,保存所述待选函数,并将该待选函数作为初始函数,重复执行第三确定装置、判断装置以及第二控制装置的操作。

18. 根据权利要求 17 所述的检测装置,其中,所述检测装置还包括:

更新装置,用于基于所保存的待选函数更新危险函数列表。

一种检测源代码漏洞的方法和装置

技术领域

[0001] 本发明涉及计算机技术领域,尤其涉及一种检测源代码漏洞的方法和装置。

背景技术

[0002] 互联网的发展给人们带来了方便而丰富的体验,然而随之而来的安全问题也越来越显著。Web 网站的漏洞检测是预防安全问题的重要手段,其主要分为动态和静态分析两种方法,动态检测虽然具有准确率高、误报低的优点,然而由于其构造模拟攻击的局限,必然有着高漏报的风险。

[0003] 静态分析是通过剖析代码而发现网页中所隐藏的安全漏洞,虽然不能保证所检测的漏洞的误报率,但是会大大降低漏报。虽然当前静态分析已作为一项有力的工具在网站漏洞检测方面发挥作用,但是现有技术的静态分析方法仍然存在误报率高、无法处理过程间调用等问题。

[0004] 现有技术中所采用的静态分析主要包括两种方式:一种是运用单纯的正则匹配的方法,匹配程序中的用户可控输入点或者危险函数点,从而得出漏洞所在位置;该方式由于无法追踪用户所传变量的路径,误报极高。另一种是先用编译的手段将代码进行词法、语法分析,继而匹配危险函数,再回溯程序中的敏感变量,一直到遇见用户可控输入点。而这种方式由于没有进行相关的数据依赖处理,导致该方法的回溯过程仅仅是对变量的简单识别,而没有对其是否是赋值、条件、循环或者函数调用进行具体的分析,导致较高的误报率。

发明内容

[0005] 本发明的目的是提供一种检测源代码漏洞的方法和装置。

[0006] 根据本发明的一个方面,提供了一种检测源代码漏洞的方法,其中,所述方法包括以下步骤:

[0007] x 基于与所述源代码对应的抽象语法树,确定所述抽象语法树中的一个或多个敏感节点,其中,所述敏感节点对应于所述源代码中的敏感函数的参数信息;

[0008] 其中,将敏感节点作为初始节点,所述方法还包括以下步骤:

[0009] a 基于所述抽象语法树,查找与所述初始节点对应的参数信息具有依赖关系的依赖节点,以将所述依赖节点作为所述初始节点的子节点;

[0010] b 当所述依赖节点不对应用户输入信息时,将该依赖节点作为新的初始节点,并重复前述步骤 a 和 b。

[0011] 根据本发明的一个方面,提供了一种检测源代码漏洞的检测装置,其中,所述检测装置包括:

[0012] 第一确定装置,用于基于与所述源代码对应的抽象语法树,确定所述抽象语法树中的一个或多个敏感节点,其中,所述敏感节点对应于所述源代码中的敏感函数的参数信息;

[0013] 其中,将敏感节点作为初始节点,所述检测装置还包括:

[0014] 查找装置,用于基于所述抽象语法树,查找与所述初始节点对应的参数信息具有依赖关系的依赖节点,并将所述依赖节点作为所述初始节点的子节点;

[0015] 第一控制装置,用于当所述依赖节点不对应用户输入信息时,将所述子节点作为新的初始节点,并重复查找装置和第一控制装置的操作。

[0016] 与现有技术相比,本发明具有以下优点:1) 根据本发明的方法,通过基于赋值操作等方式的分析,来建立基于数据的依赖关系的敏感路径信息,从而能够降低对于漏洞的误报率,并且能够提升所获得的敏感路径的完整度,提高漏洞检测的全面性;2) 通过对于分支、循环等语句的处理,能够尽可能全面地获得参数在各个语句中的依赖关系,进而记录更加完整的敏感路径信息;3) 基于该种方案,通过追踪函数间、类之间的敏感变量传递调用,进一步提升了所获取的危险函数信息的完整性,从而降低了漏报率,使得漏洞检测更加全面准确。

附图说明

[0017] 通过阅读参照以下附图所作的对非限制性实施例所作的详细描述,本发明的其它特征、目的和优点将会变得更明显:

[0018] 图 1 示意出了根据本发明的一种检测源代码漏洞的方法流程图;

[0019] 图 2 示意出了根据本发明的一种用于检测源代码漏洞的检测装置的结构示意图;

[0020] 附图中相同或相似的附图标记代表相同或相似的部件。

具体实施方式

[0021] 下面结合附图对本发明作进一步详细描述。

[0022] 图 1 示意出了根据本发明的一种用于检测源代码漏洞的方法流程图。根据本发明的方法包括步骤 S1、步骤 S2 和步骤 S3。

[0023] 其中,根据本发明的方法通过包含于计算机设备中的检测装置来实现。优选地,所述源代码为网站源代码。

[0024] 所述计算机设备包括一种能够按照事先设定或存储的指令,自动进行数值计算和/或信息处理的电子设备,其硬件包括但不限于微处理器、专用集成电路(ASIC)、可编程门阵列(FPGA)、数字处理器(DSP)、嵌入式设备等。

[0025] 其中,所述计算机设备包括网络设备和用户设备。

[0026] 所述网络设备包括但不限于单个网络服务器、多个网络服务器组成的服务器组或基于云计算(Cloud Computing)的由大量主机或网络服务器构成的云,其中,云计算是分布式计算的一种,由一群松散耦合的计算机集组成的一个超级虚拟计算机。

[0027] 所述用户设备包括但不限于任何一种可与用户通过键盘、鼠标、遥控器、触摸板、或声控设备等方式进行人机交互的电子产品。例如,手提电脑、平板电脑、智能手机、PDA、掌上游戏机以及智能导航设备等。

[0028] 其中,所述计算机设备可处于网络中,该计算机设备所处的网络包括但不限于互联网、广域网、城域网、局域网、VPN 网络等。

[0029] 需要说明的是,所述计算机设备以及网络仅为举例,其他现有的或今后可能出现的计算机设备以及网络如可适用于本发明,也应包含在本发明保护范围以内,并以引用方

式包含于此。

[0030] 在步骤 S1 中,检测装置基于与所述源代码对应的抽象语法树,确定所述抽象语法树中的一个或多个敏感节点。

[0031] 其中,所述敏感节点对应于所述源代码中的敏感函数的参数信息。其中,所述敏感函数为包含于预定的危险函数列表中的函数;或者,与所述危险函数列表中的函数相似或相近的函数。

[0032] 具体地,检测装置基于预定的危险函数列表,以及所述抽象语法树中的各个节点,来确定该抽象语法树中的各个节点是否为与所述危险函数列表中的危险函数对应的敏感函数的节点信息。

[0033] 更优选地,检测装置基于该预定的危险函数列表的函数信息,与抽象语法树中的各个节点进行正则运算,以确定所述抽象语法树中各个节点是否为敏感函数的敏感节点,将所确定的敏感节点作为初始节点。

[0034] 其中,本领域技术人员应能了解基于源代码建立抽象语法树的过程,此处不再赘述。

[0035] 在步骤 S2 中,检测装置基于所述抽象语法树,查找与所述初始节点对应的参数信息具有依赖关系的依赖节点,以将所述依赖节点作为所述初始节点的子节点。

[0036] 其中,当一个参数的需要依赖于另一个参数来确定时,说明该两个参数之间具有依赖关系。

[0037] 例如,当参数 A 需要通过将参数 B 的计算结果后对 A 进行赋值得到,则可认为参数 A 与 B 之间具有依赖关系。

[0038] 优选地,检测装置基于所述抽象语法树,查找用于对所述初始节点对应的参数信息进行赋值的赋值信息,以将所述赋值信息对应的节点作为依赖节点,并将该依赖节点作为所述初始节点的子节点。

[0039] 接着,在步骤 S3 中,当所述依赖节点不对应用户输入信息时,检测装置将该依赖节点作为新的初始节点,并重复步骤 S2 和步骤 S3 的操作。

[0040] 根据本发明的第一示例,检测装置在步骤 S1 中,通过正则匹配,确定参数 var_1 为敏感函数的参数,进而确定参数 var_1 对应的节点 node_var_1 为敏感节点,则检测装置将该节点 node_var_1 作为初始节点,接着,在步骤 S2 中,检测装置沿着所述抽象语法树,将获得的用于对该参数 var_1 进行赋值的赋值信息参数 var_2 对应的节点 node_var_2 作为依赖节点,并将该依赖节点 node_var_2 作为初始节点 node_var_1 的子节点。

[0041] 并且,检测装置判断该依赖节点 node_var_2 对应的参数信息 var_2 并非用户输入信息,则在步骤 S3 中,检测装置将该依赖节点 node_var_2 作为新的初始节点,并重新执行步骤 S2,沿着抽象语法树,查找到用于对参数信息 var_2 进行赋值的参数 var_3,并将 var_3 对应的节点 node_var_3 作为当前的初始节点 node_var_2 的子节点。接着,检测装置根据参数 var_3 是通过语句 \$_GET 来获得的,从而判断 var_3 为用户输入信息,进而停止继续查找该依赖节点的子节点。

[0042] 优选地,根据本发明的方法还包括步骤 S4(图未示)。

[0043] 在步骤 S4 中,当所述依赖节点对应于用户输入信息时,检测装置记录该依赖节点到其对应的敏感节点的敏感路径信息。

[0044] 继续对前述第一示例进行说明,当依赖节点 node_var_3 对应于用户输入信息时,检测装置记录 node_var_3 至敏感节点 node_var_1 对应的敏感路径信息:node_var_1 → node_var_2 → node_var_3。

[0045] 优选地,检测装置基于所记录的敏感路径信息,来提示所述源代码的漏洞信息。

[0046] 继续对前述第一示例进行说明,检测装置根据记录的敏感路径信息:node_var_1 → node_var_2 → node_var_3,提示程序员:发现漏洞信息 node_var_3,其对应的路径信息包括以敏感节点作为根节点的路径信息 node_var_1 → node_var_2 → node_var_3。

[0047] 更优选地,检测装置还可提示与该敏感路径信息对应的其他信息,例如,各个节点分别所属的函数名称等。

[0048] 根据本发明的方法,通过基于赋值操作等方式的分析,来建立基于数据的依赖关系的敏感路径信息,从而能够降低对于漏洞的误报率,并且能够提升所获得的敏感路径的完整度,提高漏洞检测的全面性。

[0049] 根据本发明的一个优选方案,对于所述抽象语法树中与源代码的分支语句,检测装置分别基于该分支语句的每一分支执行步骤 a 至 b。

[0050] 其中,所述分支语句包括至少包括两个并行分支的语句,例如,if else 条件语句,又例如,switch 语句等。

[0051] 根据本发明的第二示例,检测装置已经通过重复执行步骤 S2 和步骤 S3,查找并记录到从敏感节点 node_var_4 到当前的初始节点 node_var_5 的这一路径信息 node_var_4 → node_var_5,接着,检测装置再次执行步骤 S2,沿着抽象语法树查找到用于对参数 var_4 进行赋值的参数 var_6,并将该参数 var_6 对应的节点 node_var_6 作为当前的初始节点 node_var_5 的子节点,并记录从 node_var_6 到敏感节点 node_var_4 的这一路径信息 node_var_4 → node_var_5 → node_var_6。并且,检测装置确定 var_6 为以下分支语句中的一个 if 语句的分支:

[0052] if(term a){var_5 = var_6;}

[0053] else{var_5 = var_7;}

[0054] 则检测装置获取抽象语法树中,与 if 语句分支对应的 else 语句中的参数 var_7 对应的节点 node_var_7,作为与当前初始节点 node_var_5 的另一子节点,并记录从 node_var_7 到敏感节点 node_var_4 的这一路径信息 node_var_4 → node_var_5 → node_var_7。

[0055] 接着,检测装置对于 if 语句分支,以节点 node_var_6 作为新的初始节点重复执行步骤 S2 和步骤 S3,对于 else 语句分支,以节点 node_var_7,作为新的初始节点,重复执行步骤 S2 和步骤 S3。

[0056] 根据本发明的又一个优选方案,对于所述抽象语法树中的循环语句,检测装置基于当前的初始节点,在当前循环条件下执行步骤 S2 至步骤 S3。

[0057] 根据本发明的第三示例,当前的初始节点为与参数 var_8 对应的节点 node_var_8,接着,检测装置在步骤 S2 中,从抽象语法树中查找到与下述 while 循环的赋值语句中参数 var_9 对应的节点 node_var_9,并将该节点 node_var_9 作为初始节点 node_var_8 的子节点,接着,检测装置继续查找抽象语法树中其他语句对应的节点中与参数 var_9 具有依赖关系的参数,而无需通过控制 while 语句的循环变量 a 的改变进而重复获取该 while 语句中的参数 var_9。

[0058]

```
while(a!=0)
{
    var_8=var_9;
    a--;
}
```

[0059] 通过上述优选方案,通过对于分支、循环等语句的处理,检测装置能够尽可能全面地获得参数在各个语句中的依赖关系,进而记录更加完整的敏感路径信息。从而能够对网站的源代码进行更加全面的漏洞检测,并能降低误报率。

[0060] 根据本发明的一个优选实施,根据本实施例的方法还包括步骤 S5(图未示)、步骤 S6(图未示)以及步骤 S7(图未示)。

[0061] 在步骤 S5 中,检测装置对于一敏感路径信息,获取与该路径信息对应的敏感函数。

[0062] 具体地,检测装置基于所记录的敏感路径信息,获取与该敏感路径信息的根节点对应的敏感函数。

[0063] 继续对前述第一示例进行说明,检测装置基于所记录的敏感路径信息 node_var_1 → node_var_2 → node_var_3,获取与根节点 node_var_1 对应的参数 var_1,并进而确定与该参数 var_1 对应的敏感函数 func_1。

[0064] 接着,在步骤 S6 中,检测装置基于预定安全规则来对所述敏感函数进行验证。

[0065] 具体地,检测装置判断当前的安全规则所采用的验证语句是否足以验证所述敏感函数。

[0066] 优选地,检测装置在建立抽象语法树时,保存与各个函数对应的安全验证代码,并当函数被确定为敏感函数时,将所保存的安全验证代码与当前的安全规则进行匹配,当所述安全验证代码能够与所述安全规则进行匹配时,确定所属敏感函数符合所述安全规则。

[0067] 更优选地,检测装置基于敏感函数的代码信息进行分类,以确定其可能对应的漏洞类别,并基于当前的安全规则与漏洞类型的对应关系,来确定其对应的安全验证代码所需要匹配的安全规则,并当所述安全验证代码能够与所述安全规则进行匹配时,确定所述敏感函数符合所述安全规则。

[0068] 例如,当基于敏感函数本身的代码,确定该敏感函数为命令注入类型的脚本,则检测装置从当前的安全规则中获得与命令注入类型脚本对应的安全规则包括如下两个函数:

[0069] escapeshellarg();以及,

[0070] safe_mode_exec_dir();

[0071] 并且,检测装置基于所获得的与敏感函数对应的安全验证代码,与该两个函数进行匹配,当两者均匹配时,确定改敏感函数已经经过验证并符合安全规则。

[0072] 我们理解,此处的漏洞类型与检测所采用的安全规则是本领域技术人员可根据情况来确定/获得的,因此不再赘述。

[0073] 在步骤 S7 中,当所述敏感函数符合所述安全规则时,检测装置确定所述敏感路径信息非漏洞。

[0074] 其中,当检测装置确定该敏感路径信息不为漏洞时,不再提示该漏洞。

[0075] 根据本发明的方案,可以先对源代码的抽象语法树执行步骤 S1 至步骤 S3,并重复多次执行步骤 S2 至步骤 S3,以获得多个敏感路径信息后,再通过执行步骤 S5 至步骤 S7,以分别确定每个敏感路径信息是否为漏洞,并仅提示为漏洞的敏感路径信息;或者,可以每次获得一个敏感路径信息后,即执行步骤 S5 至步骤 S7,以确定所获得的敏感路径信息是否为漏洞,并仅提示为漏洞的敏感路径信息。

[0076] 根据本发明的又一优选实施,将敏感函数作为初始函数,根据本实施例的方法还包括步骤 S8(图未示)、步骤 S9(图未示)以及步骤 S10(图未示)。

[0077] 在步骤 S8 中,检测装置基于源代码中的函数调用关系,确定调用所述初始函数的一个或多个待选函数。

[0078] 接着,检测装置对所述一个或多个待选函数,分别判断所述一个或多个待选函数中的各个待选函数的参数信息与所述初始函数的参数信息是否存在依赖关系。

[0079] 具体地,检测装置可基于与所述源代码对应的抽象语法树,来确定各个参数信息之间是否具有依赖关系。

[0080] 接着,在步骤 S10 中,当待选函数的参数信息与初始函数的参数信息存在依赖关系时,确定该待选函数为所述初始函数的依赖函数,则检测装置保存所述待选函数,并将该待选函数作为初始函数,重复执行步骤 S8 至步骤 S10。

[0081] 其中,当待选函数中的任一参数信息与所述初始函数中的任一参数信息之间具有依赖关系时,即认为该待选函数为所述初始函数的依赖函数。

[0082] 例如,初始函数为 `func_2(var_11, var_12)`,而调用初始函数 `func_2` 的待选函数 `func_3` 的参数 `var_13` 基于 `var_11` 和 `var_12` 的计算结果来获得,则检测装置确定该待选函数 `func_3` 为初始函数 `func_2` 的依赖函数。则检测装置保存该待选函数 `func_3`,并重复执行步骤 S8 至步骤 S10 来进一步获取该待选函数 `func_3` 的依赖函数。

[0083] 优选地,检测装置可采用一个专门的数据结构来保存所述待选函数。例如,检测装置可维护一个 `temp` 数据结构,该结构用以存放依赖函数的所有内容,包括依赖函数的类名、对象名、函数名、参数、返回值等。更优选地,当依赖函数与初始函数为类之间的函数调用时,类名和对象名可以为空。

[0084] 根据本实施例的优选方案,所述方法还包括步骤 S11(图未示)。

[0085] 在步骤 S11 中,检测装置基于所保存的待选函数更新危险函数列表。

[0086] 具体地,检测装置基于预定的危险函数列表,与抽象语法树中的各个节点进行正则匹配并确定源代码中的初始敏感函数后,执行步骤 S8 至步骤 S10,来确定并记录各个初始敏感函数的依赖函数,接着,基于所确定的依赖函数来更新所述危险函数列表,并采用更新后的危险函数列表,再次执行步骤 S1 至步骤 S3,来获取敏感路径信息。

[0087] 基于该种方式,通过追踪函数间、类之间的敏感变量传递调用,进一步提升了所获取的危险函数信息的完整性,从而降低了漏报率,使得漏洞检测更加全面准确。

[0088] 图 2 示意出了根据本发明的一种用于检测源代码漏洞的检测装置的结构示意图。根据本发明的检测装置包括第一确定装置 1、查询装置 2 和第一控制装置 3。

[0089] 第一确定装置 1 基于与所述源代码对应的抽象语法树,确定所述抽象语法树中的一个或多个敏感节点。

[0090] 其中,所述敏感节点对应于所述源代码中的敏感函数的参数信息。其中,所述敏感函数为包含于预定的危险函数列表中的函数;或者,与所述危险函数列表中的函数相似或相近的函数。

[0091] 具体地,第一确定装置 1 基于预定的危险函数列表,以及所述抽象语法树中的各个节点,来确定该抽象语法树中的各个节点是否为与所述危险函数列表中的危险函数对应的敏感函数的节点信息。

[0092] 更优选地,第一确定装置 1 基于该预定的危险函数列表的函数信息,与抽象语法树中的各个节点进行正则运算,以确定所述抽象语法树中各个节点是否为敏感函数的敏感节点,将所确定的敏感节点作为初始节点。

[0093] 查询装置 2 基于所述抽象语法树,查找与所述初始节点对应的参数信息具有依赖关系的依赖节点,以将所述依赖节点作为所述初始节点的子节点。

[0094] 其中,当一个参数的需要依赖于另一个参数来确定时,说明该两个参数之间具有依赖关系。

[0095] 例如,当参数 A 需要通过将参数 B 的计算结果后对 A 进行赋值得到,则可认为参数 A 与 B 之间具有依赖关系。

[0096] 优选地,查询装置 2 基于所述抽象语法树,查找用于对所述初始节点对应的参数信息进行赋值的赋值信息,以将所述赋值信息对应的节点作为依赖节点,并将该依赖节点作为所述初始节点的子节点。

[0097] 接着,当所述依赖节点不对应用户输入信息时,第一控制装置 3 将该依赖节点作为新的初始节点,并重复查询装置 2 和第一控制装置 3 的操作。

[0098] 根据本发明的第一示例,第一确定装置 1 通过正则匹配,确定参数 var_1 为敏感函数的参数,进而确定参数 var_1 对应的节点 node_var_1 为敏感节点,则第一确定装置 1 将该节点 node_var_1 作为初始节点,接着,查询装置 2 沿着所述抽象语法树,将获得的用于对该参数 var_1 进行的赋值的赋值信息参数 var_2 对应的节点 node_var_2 作为依赖节点,并将该依赖节点 node_var_2 作为初始节点 node_var_1 的子节点。

[0099] 并且,第一控制装置 3 判断该依赖节点 node_var_2 对应的参数信息 var_2 并非用户输入信息,则第一控制装置 3 将该依赖节点 node_var_2 作为新的初始节点,并重新执行查询装置 2 的操作,沿着抽象语法树来查找到用于对参数信息 var_2 进行赋值的参数 var_3,并将 var_3 对应的节点 node_var_3 作为当前的初始节点 node_var_2 的子节点。接着,第一控制装置 3 根据参数 var_3 为 PHP 中用于获取用户输入信息的 \$_GET 语句,从而判断 var_3 为用户输入信息,进而停止继续查找该依赖节点的子节点。

[0100] 优选地,根据本发明的检测装置还包括记录装置(图未示)。

[0101] 当所述依赖节点对应于用户输入信息时,记录装置记录该依赖节点到其对应的敏感节点的敏感路径信息。

[0102] 继续对前述第一示例进行说明,当依赖节点 node_var_3 对应于用户输入信息时,记录装置记录 node_var_3 至敏感节点 node_var_1 对应的敏感路径信息:node_var_1 → node_var_2 → node_var_3。

[0103] 优选地,检测装置基于记录装置所记录的敏感路径信息,来提示所述源代码的漏洞信息。

[0104] 继续对前述第一示例进行说明,检测装置根据记录装置所记录的敏感路径信息: node_var_1 → node_var_2 → node_var_3,提示程序员:发现漏洞信息 node_var_3,其对应的路径信息包括以敏感节点作为根节点的路径信息 node_var_1 → node_var_2 → node_var_3。

[0105] 更优选地,检测装置还可提示与该敏感路径信息对应的其他信息,例如,各个节点分别所属的函数名称或者类名称等等。

[0106] 根据本发明的方案,通过基于赋值操作等方式的分析,来建立基于数据的依赖关系的敏感路径信息,从而能够降低对于漏洞的误报率,并且能够提升所获得的敏感路径的完整度,提高漏洞检测的全面性。

[0107] 根据本发明的一个优选方案,对于所述抽象语法树中与源代码的分支语句,检测装置分别基于该分支语句的每一分支由查找装置和第一控制装置的执行操作。

[0108] 其中,所述分支语句包括至少包括两个并行分支的语句,例如,if else 条件语句,又例如,switch 语句等。

[0109] 根据本发明的第二示例,检测装置已经通过重复执行查找装置 2 和第一控制装置 3,查找并记录到从敏感节点 node_var_4 到当前的初始节点 node_var_5 的这一路径信息 node_var_4 → node_var_5,接着,查找装置 2 再次执行操作,沿着抽象语法树查找到用于对参数 var_4 进行赋值的参数 var_6,并将该参数 var_6 对应的节点 node_var_6 作为当前的初始节点 node_var_5 的子节点,并由记录装置记录从 node_var_6 到敏感节点 node_var_4 的这一路径信息 node_var_4 → node_var_5 → node_var_6。并且,检测装置确定 var_6 为以下分支语句中的一个 if 语句的分支:

[0110] if(term a){var_5 = var_6;}

[0111] else{var_5 = var_7;}

[0112] 则查找装置 2 获取抽象语法树中,与 if 语句分支对应的 else 语句中的参数 var_7 对应的节点 node_var_7,作为与当前初始节点 node_var_5 的另一子节点,并由记录装置记录从 node_var_7 到敏感节点 node_var_4 的这一路径信息 node_var_4 → node_var_5 → node_var_7。

[0113] 接着,检测装置对于 if 语句分支,以节点 node_var_6 作为新的初始节点重复执行查找装置和第一控制装置的操作,对于 else 语句分支,以节点 node_var_7,作为新的初始节点,重复执行查找装置和第一控制装置的操作。

[0114] 根据本发明的又一个优选方案,对于所述抽象语法树中的循环语句,检测装置基于当前的初始节点,在当前循环条件下执行查找装置和第一控制装置的操作。

[0115] 根据本发明的第三示例,当前的初始节点为与参数 var_8 对应的节点 node_var_8,接着,查找装置 2 和第一控制装置 3 从抽象语法树中查找到与下述 while 循环的赋值语句中参数 var_9 对应的节点 node_var_9,并将该节点 node_var_9 作为初始节点 node_var_8 的子节点,接着,查找装置 2 继续查找抽象语法树中其他语句对应的节点中与参数 var_9 具有依赖关系的参数,而无需通过控制 while 语句的循环变量“a”的改变进而重复获取该 while 语句中的参数 var_9。

[0116]

```
while(a!=0)
{
var_8=var_9;
a--;
}
```

[0117] 通过上述优选方案,检测装置能够尽可能全面地获得参数在各个语句中的依赖关系,进而记录更加完整的敏感路径信息。从而能够对网站的源代码进行更加全面的漏洞检测,并能降低误报率。

[0118] 根据本发明的一个优选实施,根据本实施例的检测装置还包括获取装置(图未示)、验证装置(图未示)以及第二确定装置(图未示)。

[0119] 获取装置对于一敏感路径信息,获取与该路径信息对应的敏感函数。

[0120] 具体地,获取装置基于所记录的敏感路径信息,获取与该敏感路径信息的根节点对应的敏感函数。

[0121] 继续对前述第一示例进行说明,获取装置基于所记录的敏感路径信息 node_var_1 → node_var_2 → node_var_3,获取与根节点 node_var_1 对应的参数 var_1,并进而确定与该参数 var_1 对应的敏感函数 func_1。

[0122] 接着,验证装置基于预定安全规则来对所述敏感函数进行验证。

[0123] 具体地,验证装置判断当前的安全规则所采用的验证语句是否足以验证所述敏感函数。

[0124] 优选地,验证装置在建立抽象语法树时,保存与各个函数对应的安全验证代码,并当函数被确定为敏感函数时,将所保存的安全验证代码与当前的安全规则进行匹配,当所述安全验证代码能够与所述安全规则进行匹配时,确定所属敏感函数符合所述安全规则。

[0125] 更优选地,验证装置基于敏感函数的代码信息进行分类,以确定其可能对应的漏洞类别,并基于当前的安全规则与漏洞类型的对应关系,来确定其对应的安全验证代码所需要匹配的安全规则,并当所述安全验证代码能够与所述安全规则进行匹配时,确定所述敏感函数符合所述安全规则。

[0126] 例如,当基于敏感函数本身的代码,确定该敏感函数为命令注入类型的脚本,则检测装置从当前的安全规则中获得与命令注入类型脚本对应的安全规则包括如下两个函数:

[0127] escapeshellarg();以及,

[0128] safe_mode_exec_dir();

[0129] 并且,验证装置基于所获得的与敏感函数对应的安全验证代码,与该两个函数进行匹配,当两者均匹配时,确定该敏感函数已经经过验证并符合安全规则。

[0130] 我们理解,此处的漏洞类型与检测所采用的安全规则是本领域技术人员可根据情况来确定/获得的,因此不再赘述。当所述敏感函数符合所述安全规则时,第二确定装置确

定所述敏感路径信息非漏洞。

[0131] 其中,当第二确定装置确定该敏感路径信息不为漏洞时,检测装置不再提示该漏洞。

[0132] 根据本发明的方案,可以先由第一确定装置 1 对源代码的抽象语法树执行操作,并重复多次执行查找装置 2 和第一控制装置 3 的操作,以获得多个敏感路径信息后,再通过执行获取装置、验证装置以及第二确定装置的操作,以分别确定每个敏感路径信息是否为漏洞,并仅提示为漏洞的敏感路径信息;或者,可以每次获得一个敏感路径信息后,即执行获取装置、验证装置以及第二确定装置的操作,以确定所获得的敏感路径信息是否为漏洞,并仅提示为漏洞的敏感路径信息。

[0133] 根据本发明的又一优选实施,将敏感函数作为初始函数,根据本实施例的验证装置还包括第三确定装置(图未示)、判断装置(图未示)以及第二控制装置(图未示)。

[0134] 第三确定装置基于源代码中的函数调用关系,确定调用所述初始函数的一个或多个待选函数。

[0135] 接着,判断装置对所述一个或多个待选函数,分别判断所述一个或多个待选函数中的各个待选函数的参数信息与所述初始函数的参数信息是否存在依赖关系。

[0136] 具体地,判断装置可基于与所述源代码对应的抽象语法树,来确定各个参数信息之间是否具有依赖关系。

[0137] 其中,当待选函数中的任一参数信息与所述初始函数中的任一参数信息之间具有依赖关系时,判断装置即认为该待选函数为所述初始函数的依赖函数。

[0138] 接着,当待选函数的参数信息与初始函数的参数信息存在依赖关系时,第二控制装置确定该待选函数为所述初始函数的依赖函数,则第二控制装置保存所述待选函数,并将该待选函数作为初始函数,重复执行第三确定装置、判断装置以及第二控制装置的操作。

[0139] 例如,初始函数为 func_2(var_11, var_12),而调用初始函数 func_2 的待选函数 func_3 的参数 var_13 基于 var_11 和 var_12 的计算结果来获得,则判断装置确定该待选函数 func_3 为初始函数 func_2 的依赖函数。则第二控制装置保存该待选函数 func_3,并重复执行步骤 S8 至步骤 S10 来进一步获取该待选函数 func_3 的依赖函数。

[0140] 优选地,第二控制装置可采用一个专门的数据结构来保存所述待选函数。例如,第二控制装置可维护一个 temp 数据结构,该结构用以存放依赖函数的所有内容,包括依赖函数的类名、对象名、函数名、参数、返回值等。更优选地,当依赖函数与初始函数为类之间的函数调用时,类名和对象名可以为空。

[0141] 根据本实施例的优选方案,所述检测装置还包括更新装置(图未示)。

[0142] 更新装置基于所保存的待选函数更新危险函数列表。

[0143] 具体地,更新装置基于预定的危险函数列表,与抽象语法树中的各个节点进行正则匹配并确定源代码中的初始敏感函数后,执行第三确定装置、判断装置以及第二控制装置的操作,来确定并记录各个初始敏感函数的依赖函数,接着,更新装置基于所确定的依赖函数来更新所述危险函数列表,并且,采用更新后的危险函数列表,再次执行第一确定装置 1、查找装置 2 以及第一控制装置 3 的操作,来获取敏感路径信息。

[0144] 基于该种方案,通过追踪函数间、类之间的敏感变量传递调用,进一步提升了所获取的危险函数信息的完整性,从而降低了漏报率,使得漏洞检测更加全面准确。

[0145] 其中,根据本发明的方法通过包含于计算机设备中的装置来实现。所述计算机设备包括一种能够按照事先设定或存储的指令,自动进行数值计算和 / 或信息处理的电子设备,其硬件包括但不限于微处理器、专用集成电路 (ASIC)、可编程门阵列 (FPGA)、数字处理器 (DSP)、嵌入式设备等。所述计算机设备包括网络设备和 / 或用户设备。其中,所述网络设备包括但不限于单个网络服务器、多个网络服务器组成的服务器组或基于云计算 (Cloud Computing) 的由大量主机或网络服务器构成的云,其中,云计算是分布式计算的一种,由一群松散耦合的计算机集组成的一个超级虚拟计算机。所述用户设备包括但不限于任何一种可与用户通过键盘、鼠标、遥控器、触摸板、或声控设备等方式进行人机交互的电子产品,例如,个人计算机、平板电脑、智能手机、PDA、游戏机、或 IPTV 等。其中,所述用户设备及网络设备所处的网络包括但不限于互联网、广域网、城域网、局域网、VPN 网络等。

[0146] 需要说明的是,所述用户设备、网络设备以及网络仅为举例,其他现有的或今后可能出现的用户设备、网络设备以及网络如可适用于本发明,也应包含在本发明保护范围以内,并以引用方式包含于此。

[0147] 本发明的软件程序可以通过处理器执行以实现上文所述步骤或功能。同样地,本发明的软件程序 (包括相关的数据结构) 可以被存储到计算机可读记录介质中,例如, RAM 存储器,磁或光驱动器或软磁盘及类似设备。另外,本发明的一些步骤或功能可采用硬件来实现,例如,作为与处理器配合从而执行各个功能或步骤的电路。

[0148] 另外,本发明的一部分可被应用为计算机程序产品,例如计算机程序指令,当其被计算机执行时,通过该计算机的操作,可以调用或提供根据本发明的方法和 / 或技术方案。而调用本发明的方法的程序指令,可能被存储在固定的或可移动的记录介质中,和 / 或通过广播或其他信号承载媒体中的数据流而被传输,和 / 或被存储在根据所述程序指令运行的计算机设备的工作存储器中。在此,根据本发明的一个实施例包括一个装置,该装置包括用于存储计算机程序指令的存储器和用于执行程序指令的处理器,其中,当该计算机程序指令被该处理器执行时,触发该装置运行基于前述根据本发明的多个实施例的方法和 / 或技术方案。

[0149] 对于本领域技术人员而言,显然本发明不限于上述示范性实施例的细节,而且在不背离本发明的精神或基本特征的情况下,能够以其他的具体形式实现本发明。因此,无论从哪一点来看,均应将实施例看作是示范性的,而且是非限制性的,本发明的范围由所附权利要求而不是上述说明限定,因此旨在将落在权利要求的等同要件的含义和范围内的所有变化涵括在本发明内。不应将权利要求中的任何附图标记视为限制所涉及的权利要求。此外,显然“包括”一词不排除其他单元或步骤,单数不排除复数。系统权利要求中陈述的多个单元或装置也可以由一个单元或装置通过软件或者硬件来实现。第一,第二等词语用来表示名称,而并不表示任何特定的顺序。

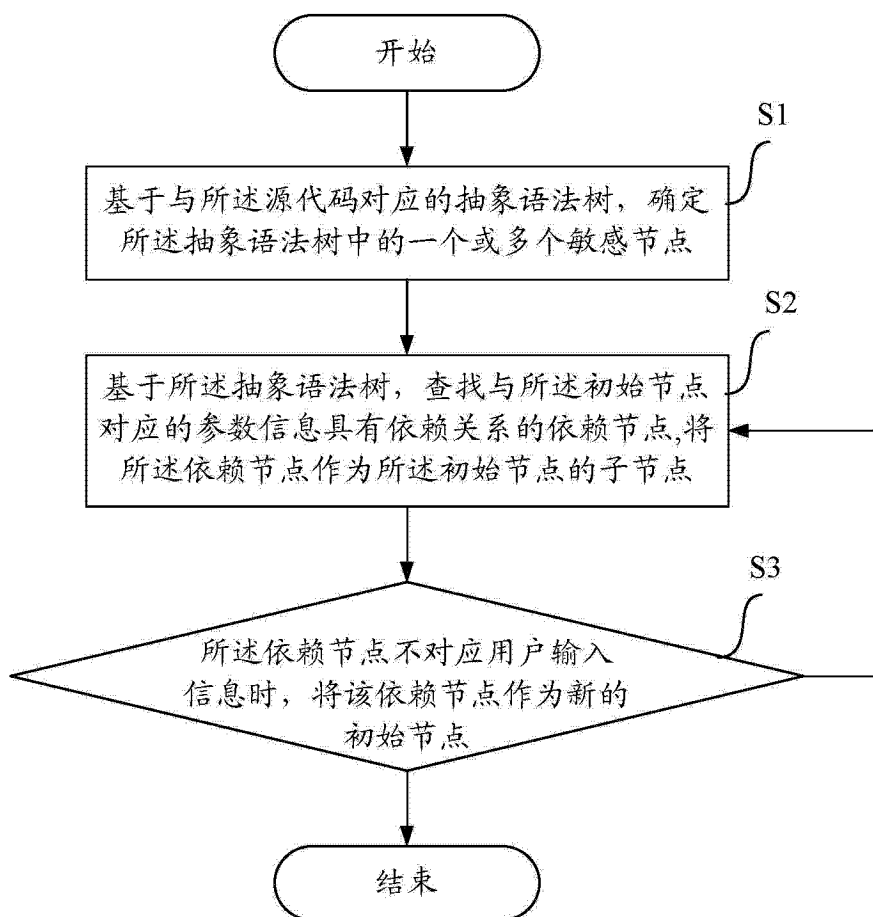


图 1

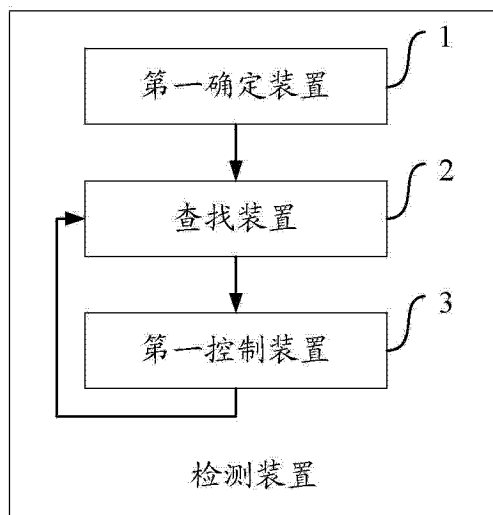


图 2