

基于结构特征的二进制代码安全缺陷分析模型

许团, 屈蕾蕾, 石文昌

(中国人民大学信息学院, 北京 100872)

摘要: 针对现有方法检测复杂结构二进制代码安全缺陷的不足, 提出新的分析模型, 并给出其应用方法。首先以缺陷的源代码元素集合生成特征元素集合, 抽取代码结构信息, 构建分析模型。然后依据各类中间表示 (IR, intermediate representation) 语句的统计概率计算分析模型, 查找满足特征模型的 IR 代码组, 通过 IR 代码与二进制代码的转换关系, 实现对二进制程序中代码安全缺陷的有效检测。分析模型可应用于二进制单线程程序和并行程序。实验结果表明, 相对于现有方法, 应用该分析模型能够更全面深入地检测出各类结构复杂的二进制代码安全缺陷, 且准确率更高。

关键词: 二进制分析; 分析模型; 软件安全缺陷检测; 缺陷代码识别

中图分类号: TP309.5

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2017.00200

Analysis model of binary code security flaws based on structure characteristics

XU Tuan, QU Lei-lei, SHI Wen-chang

(School of Information, Renmin University of China, Beijing 100872, China)

Abstract: Aiming at the shortcomings of the existing methods to detect the security flaws that have complex structures, a new analysis model and its application method was proposed. First, analysis models based on key information of code structures extracted from path subsets of characteristic element sets that are generated by source code element sets of code security flaws were constructed. Then the analysis model according to the statistical probability of each kind of IR statement was calculated, and the IR code group which matched the feature model was found. Finally, through the translating relation between binary codes and IR codes, various code security flaws of binary program were found out. The analysis models can be applied to both common single-process binary programs and binary parallel programs. Experimental results show that compared with the existing methods, the application of the analysis model can be more comprehensive and in-depth in detecting various types of complex binary code security flaws with higher accuracy.

Key words: binary analysis, analysis model, software security detection, flaw code recognition

1 引言

二进制程序中的安全缺陷是危害软件系统的安全性和可靠性的重要原因, 目前二进制程序的

安全性检测成为研究热点。根据有代表性的研究成果^[1~3]可知, 多数类型的软件安全缺陷是在程序代码的实现过程中产生。本文把程序代码实现过程中产生的各种类型软件安全缺陷称为代码安全

收稿日期: 2017-07-24; 修回日期: 2017-08-13。通信作者: 屈蕾蕾, daisyqvruc@163.com

基金项目: 国家自然科学基金资助项目 (No.61472429); 北京市自然科学基金资助项目 (No.4122041)

Foundation Items: The National Natural Science Foundation of China (No.61472429), The Natural Science Foundation of Beijing (No.4122041)

缺陷, 并把二进制程序中代码安全缺陷称为二进制代码安全缺陷。

近年来, 各种结构复杂的二进制代码安全缺陷日益繁多。例如, 后门、竞争条件及鉴权绕过等逻辑错误类型的代码安全缺陷。它们不仅存在于二进制单线程程序, 而且分布于各种二进制并行程序。而符号执行^[4,5]、污点分析^[6,7]及模糊测试^[8,9]等现有方法难以发现结构复杂的代码安全缺陷, 也不能有效地应用于二进制并行程序, 使软件系统面临安全性风险。分析现有方法的检测过程可知, 这些方法均未深入解析二进制代码安全缺陷代码结构, 不能有效地识别其结构特征, 从而产生上述不足。

本文提出对二进制代码安全缺陷的分析模型。应用该模型, 可以有效地解决现有方法的不足。本文创新点如下。

1) 给出对二进制代码安全缺陷的分析原理, 基于该分析原理提出二进制代码安全缺陷分析模型的构建方法。该方法首先基于代码安全缺陷的源代码元素集合构建中间表示代码元素集合, 以校验样本测试 IR 代码元素集合, 获取缺陷 IR 代码的关键特征, 建立特征元素集合。然后以 IR 语句的执行关系划分特征元素集合, 并提取特征元素包含的结构信息生成分析模型。分析模型完整包含了代码安全缺陷的结构特征信息, 能够为准确检测二进制程序中各类代码安全缺陷提供充分的支持及有效的判定标准。

2) 给出对二进制代码安全缺陷分析模型的应用原理, 基于该应用原理提出分析模型的应用方法, 依据分析模型中结构特征信息实施对二进制程序中代码安全缺陷的有效检测。首先基于 IR 语句的统计概率计算分析模型中各路径单元的权重, 利用工具 VTS 获取二进制程序的 IR 执行路径集合。然后在 IR 执行路径中检测满足分析模型的 IR 代码组, 并通过 IR 语句与机器指令之间的对应信息, 检测出被测试二进制程序中代码安全缺陷。该方法既能够准确地查找出现有方法难以发现的各类结构复杂的二进制代码安全缺陷, 也可以有效地应用于现有方法难以检测的二进制并行程序, 并且其检测过程无需触发代码安全缺陷, 能够有效地避免检测结果的误报和漏报。

2 模型构建

为了阐述二进制代码安全缺陷分析模型的构建方法, 有必要给出以下若干定义。

定义 1 (程序执行路径) 在二进制程序 B 的一次执行过程中, 每一个线程执行的机器指令序列都称为一条指令序列。对 B 执行过程产生的任意一条指令序列, 若移除其中不包含于 B 的机器指令, 则得到的机器指令序列称为 B 的一条指令执行路径。若程序分析工具把 B 的一条指令执行路径转换为一条 IR 语句序列, 则该 IR 语句序列称为 B 的一条 IR 执行路径。IR 执行路径与指令执行路径统称为程序执行路径。

定义 2 (代码组) 任意一个源程序 P 中语义相关的 m ($m \geq 1$) 个代码段构成一个源代码组。在 P 编译为二进制程序 B 的过程中, 若 P 中一个源代码组 β 编译生成 n ($n \geq 1$) 个机器指令段, 则该 n 个机器指令段构成一个二进制代码组 χ , 称 χ 为 β 的机器码, 称 β 为 χ 的源码。当程序分析工具把 B 的指令执行路径转换为 IR 执行路径时, 若 χ 包含的机器指令被转换为 n 个 IR 语句段, 则该 n 个 IR 语句段构成一个 IR 代码组 ξ , 称 ξ 为 χ 和 β 的 IR 码, 称 β 为 ξ 的源码, 称 χ 为 ξ 的机器码。源代码组、二进制代码组及 IR 代码组统称为代码组。

定义 3 (缺陷 IR 代码) 若 Ψ 类型软件安全的缺陷是在程序代码的实现过程中产生, 则称其为 Ψ 类型代码安全缺陷。若源程序中一个 Ψ 类型代码安全缺陷包含 m ($m \geq 1$) 个代码段, 则该 m 个代码段构成的一个源代码组 β , 称为 Ψ 类型缺陷源代码, 并称 β 的机器码及 IR 码分别为 Ψ 类型缺陷二进制代码及 Ψ 类型缺陷 IR 代码。

定义 4 (IR 语句的匹配) 设 u 为一条程序语句, 对 u 中任意一项数据 d , 称 d 为 u 的数据, 记为 $d \in u$ 。设 u_i 、 u_j 为两条 IR 语句, 若它们之间满足以下条件: ① u_i 和 u_j 的语句类型相同; ② 对任意数据 $d_s \in u_i$, 存在数据 $d_t \in u_j$, u_i 中 d_s 的位置与 u_j 中 d_t 的位置相同, 且 d_s 与 d_t 具有相同的类型。则称 u_i 与 u_j 匹配, 记为 $u_i \leftrightarrow u_j$ 。

定义 5 (IR 代码组的匹配) 设 ξ 为一个 IR 代码组, 对 ξ 中任意一条 IR 语句 u , 称 u 为 ξ 的语

句, 记为 $u \in \xi$ 。设 ξ_m 、 ξ_n 为 2 个 IR 代码组, 若 ξ_m 中 IR 语句与 ξ_n 中 IR 语句之间具有一一对应的关系, 满足以下条件: ①对任意 IR 语句 $u_i \in \xi_m$, 若 u_i 与 ξ_n 中 IR 语句 u_s 对应, 则 $u_i \leftrightarrow u_s$; ②对于任意 IR 语句 $u_i, u_j \in \xi_m$, 存在 IR 语句 $u_s, u_t \in \xi_n$, u_s 、 u_t 分别与 u_i 、 u_j 对应, 且 u_s 、 u_t 之间的关系与 u_i 、 u_j 之间的关系相同。则称 ξ_m 与 ξ_n 匹配, 记为 $\xi_m \leftrightarrow \xi_n$ 。

基于上述概念可知, 缺陷源代码、缺陷二进制代码及缺陷 IR 代码都是代码安全缺陷的具体存在形式, 它们之间可以相互转换, 对它们的代码结构进行有效的分析, 是检测代码安全缺陷的基础和前提。本文对二进制代码安全缺陷的分析原理如图 1 所示。

首先把缺陷源代码转换为缺陷 IR 代码, 基于校验样本提取出其代码特征。其次利用程序语句之间的执行关系, 划分代码特征的组成部分。然后对各组成部分进行形式化抽象, 获取其中代码结构的完整信息。由于 IR 语句与机器指令之间一一对应, 且语义相同, 所以上述过程获取的结构特征信息准确地描述了二进制代码安全缺陷的具体存在形式, 能够作为检测二进制程序中代码安全缺陷的有效依据。

根据上述原理, 给出构建任意 Ψ 类型二进制代码安全缺陷的分析模型的 6 个主要步骤。

步骤 1 建立源代码元素集合。首先获取具有 Ψ 类型代码安全缺陷的源程序, 选择出有代表性的源程序 P , 标记其中 Ψ 类型缺陷源代码 β 包含的程序语句。其次基于程序的控制流和数据流检测 β 中各程序语句之间的关系, 并把它们表示为向量形式。对 β 中任意两条程序语句 c_i 、 c_j , 它们之间的关系表示为: $\theta = (r_1, \dots, r_n)$, 其中 r_i ($1 \leq i \leq n$) 是 c_i 和 c_j 之间基于控制流和数据流的

一项关系, 记为 $r_i \in \theta$ 。然后以 β 中任意两条程序语句及它们之间关系组成三元组, 并放入集合 C , 表示为: $C = \{(c_i, \theta, c_j) \mid \theta \vdash (c_i, c_j), c_i, c_j \in \beta, i \leq j\}$ 。 C 中三元组 (c_i, θ, c_j) 称为源代码元素, $\theta \vdash (c_i, c_j)$ 表示程序语句 c_i 、 c_j 之间具有向量 θ 中各种关系, $c_i, c_j \in \beta$ 表示程序语句 c_i 、 c_j 包含于缺陷源代码 β 。

步骤 2 把源程序 P 编译为二进制程序 B , 应用工具 VTS 获取 B 的 IR 执行路径。VTS 是基于开源软件 Valgrind^[10,11]和 Taintgrind^[12]实现的程序分析工具, 能够监控 B 的动态执行过程。当 B 的动态执行过程结束, VTS 不仅输出 B 的 IR 执行路径, 而且输出 IR 语句、机器指令及源程序语句之间的对应信息, 据此能够定位每一条源程序语句所对应的机器指令组和 IR 语句组。图 2 中示例为 Linux C 语句 “if(t)” 对应的机器指令组和 IR 语句组。

步骤 3 建立 IR 代码元素集合。首先基于 VTS 输出的信息获取缺陷源代码 β 转换的缺陷 IR 代码 ξ , 并标记 ξ 中 IR 语句。其次基于集合 C 中源代码元素, 在 B 的 IR 执行路径中检测 ξ 的各条语句之间关系, 并建立 ξ 中任意两条 IR 语句之间的关系向量。对 ξ 中任意两条 IR 语句 u_s 、 u_t , 它们之间的关系向量表示为: $\delta = (r_1, \dots, r_k)$, 其中 r_i ($1 \leq i \leq k$) 是 u_s 与 u_t 之间基于控制流和数据流的一项关系, 记为 $r_i \in \delta$ 。例如, 若一个数据 d 同时包含于 u_s 和 u_t , 则该项关系称为数据共享关系, 记为 $ID(a_{si}, a_{tj}) \in \delta$, 其中字符串 a_{si} 、 a_{tj} 分别标记了 u_s 和 u_t 中 d 的数据信息。然后建立集合: $U = \{(u_s, \delta, u_t) \mid \delta \vdash (u_s, u_t), u_s, u_t \in \xi, s \leq t\}$, 其中三元组 (u_s, δ, u_t) 称为 IR 代码元素, IR 语句 u_s 、 u_t 分别称为集合 U 的语句, $u_s, u_t \in \xi$ 表示 u_s 、 u_t 包含于缺陷 IR 代码 ξ , $\delta \vdash (u_s, u_t)$ 表示 IR 语句 u_s 、 u_t 之间具有向量 δ 中各种关系。

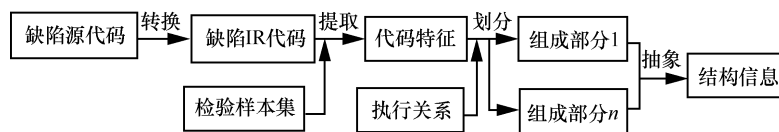


图1 二进制代码安全缺陷分析原理

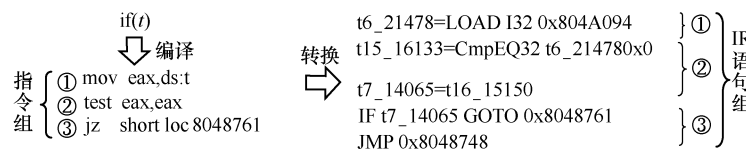


图2 if(t)对应的机器指令组及 IR 语句组

步骤4 构建校验样本集合。首先收集候选校验样本, 建立候选校验样本集合。对任意一个IR代码组 λ , 若其满足如下条件: ① λ 不是缺陷IR代码; ② λ 中IR语句与 ξ 中IR语句一一对应, 且每一对对应的IR语句之间都具有匹配关系。则选择 λ 作为候选校验样本, 把 λ 放入候选校验样本集合 J 。然后选择 J 的一个子集 F , 使以下条件成立: ①对任意IR代码组 $\lambda_s, \lambda_t \in F$, λ_s 与 λ_t 不匹配; ②对任意IR代码组 $\lambda_i \in J$, 存在IR代码组 $\lambda_j \in F$, 满足: $\lambda_j \leftrightarrow \lambda_i$ 。 F 称为校验样本集合, 其中IR代码组称为校验样本。

步骤5 构建特征元素集合。以IR代码元素集合 U 和校验样本集合 F 为输入, 执行特征元素集合生成算法, 得到集合 T , 其中元素描述了缺陷IR代码 ξ 的关键特征, 称为特征元素。

算法1 特征元素集合生成算法 Characteristic-ElementSetGenerate

输入 IR代码元素集合 U , 校验样本集合 F

输出 特征元素集合 T

```

1)  $T \leftarrow U$ 
2) for each  $(u_s, \delta, u_t) \in T$  do
3)   if PreOpt  $((u_s, \delta, u_t), T, F) == \text{False}$  then
4)     RelationVectorOpt  $((u_s, \delta, u_t), T, F)$ 
5) for each  $(u_s, \delta, u_t) \in T$  do
6)   if FurOpt  $((u_s, \delta, u_t), T, F) == \text{False}$  then
7)     RelationVectorOpt  $((u_s, \delta, u_t), T, F)$ 
8) for each  $(u_s, \delta, u_t) \in T$  do
9)   if ER  $((u_s, \delta, u_t), T, F) == \text{False}$  then
10)    RelationVectorOpt  $((u_s, \delta, u_t), T, F)$ 
11) return  $T$ 

```

算法1中各函数分别执行不同的检测过程, 下面分别阐述。

1) 函数 PreOpt $((u_s, \delta, u_t), T, F)$ 检测向量 δ 中语句关系。若 δ 包含数据共享关系, 则返回True。否则, 返回函数 ElementRedundancy $((u_s, \delta, u_t), T, F)$ 。

2) 函数 FurOpt $((u_s, \delta, u_t), T, F)$ 在向量 δ 中查找数据依赖关系。若 δ 中包含数据依赖关系, 则返回True。否则, 返回函数 ER $((u_s, \delta, u_t), T, F)$ 。

3) 函数 ER $((u_s, \delta, u_t), T, F)$ 去除 T 中 (u_s, δ, u_t) , 并调用 SampleMatching (T, F) , 若其返回True, 则把 (u_s, δ, u_t) 加入集合 T , 返回False。否则, 返回True。

4) 函数 RelationVectorOpt $((u_s, \delta, u_t), T, F)$ 检

测向量 δ 中冗余的语句关系, 过程如下。

① 查找 δ 中未检测的语句关系。若 δ 中不存在未检测的语句关系, 则检测过程结束。否则, 从 δ 中去除任意一项未检测的语句关系, 设其为 r_i 。然后调用函数 SampleMatching (T, F) 。

② 若函数 SampleMatching 返回True, 则把 r_i 加入 δ , 并标记其为已检测。执行步骤①。

5) 函数 SampleMatching (T, F) 基于 F 中校验样本检测集合 T 。若存在校验样本 $\lambda \in F$, λ 的语句与 T 的语句之间具有一种一一对应关系, 成立如下条件。

① 对任意IR语句 $u_i \in \lambda$, 若 T 的语句 u_s 与 u_i 对应, 则 $u_i \leftrightarrow u_s$;

② 对任意IR代码元素 $(u_s, \delta, u_t) \in T$, 若其中 u_s, u_t 分别与 λ 的语句 u_i, u_j 对应, 则 u_i 与 u_j 之间具有向量 δ 中全部关系。

则函数返回True。否则, 函数返回False。

步骤6 生成二进制代码安全缺陷的分析模型。首先划分集合 T 中元素, 主要过程为: ①把集合 T 中不包含于任意一条IR执行路径的元素放入集合 R_1 ; ②把集合 T 中不包含于 R_1 的元素分别放入集合 $L_1 \sim L_k$, 使 L_i ($1 \leq i \leq k$) 中元素包含于同一条IR执行路径; ③把集合 $L_1 \sim L_k$ 分别作为元素加入集合 R_2 , 得到 $R_2 = \{L_1, \dots, L_k\}$ 。 $L_1 \sim L_k$ 及 R_1 称为 T 的路径子集。

然后以集合序偶 $\langle R_1, R_2 \rangle$ 为输入, 执行分析模型生成算法, 得到分析模型 $\langle M_1, M_2 \rangle$, M_1 中元素 D_i ($1 \leq i \leq k$) 称为路径单元, M_2 称为组合单元, M_2 和 D_i 中元素称为模型元素。

算法2 分析模型生成算法 AnalysisModel-Generate

输入 集合序偶 $\langle R_1, R_2 \rangle$

输出 分析模型 $\langle M_1, M_2 \rangle$

```

1) Set  $M_1 \leftarrow \emptyset, M_2 \leftarrow \emptyset$ 
2) for each  $L_i \in R_2$  do
3)   Unit  $D_i \leftarrow \emptyset$ 
4)   for each  $((u_s, \delta, u_t) \in L_i$  do
5)     Element  $(\eta_s, \delta, \eta_t) \leftarrow \text{Transform}(u_s, \delta, u_t)$ 
6)      $D_i \leftarrow D_i \cup \{(\eta_s, \delta, \eta_t)\}$ 
7)    $M_1 \leftarrow M_1 \cup \{D_i\}$ 
8) for each  $(u_s, \delta, u_t) \in R_1$  do
9)   Element  $(\eta_s, \delta, \eta_t) \leftarrow \text{Transform}(u_s, \delta, u_t)$ 
10)   $M_2 \leftarrow M_2 \cup \{(\eta_s, \delta, \eta_t)\}$ 

```

11) return $\langle M_1, M_2 \rangle$

算法 2 中, 函数 $\text{Transform}(u_s, \delta, u_t)$ 以字符串标记 IR 语句 u_s 、 u_t 的关键信息, 并把各字符串分别组成向量 $\eta_s = (o_s, a_{s1}, \dots, a_{sn})$ 、 $\eta_t = (o_t, a_{t1}, \dots, a_{tm})$ 。 η_s 、 η_t 称为语句项, 其中字符串 o_s 、 o_t 分别标记了 u_s 、 u_t 的语句类型, 称为语句类型字符串, 字符串 a_{si} ($1 \leq i \leq n$)、 a_{tj} ($1 \leq j \leq m$) 分别标记了 u_s 、 u_t 中单项数据的类型及位置等数据信息, 称为数据字符串。该函数返回以语句项 η_s 、 η_t 及关系向量 δ 构成的模型元素 (η_s, δ, η_t) 。

若 Ψ 类型二进制代码安全缺陷具有多种分析模型, 则每一种分析模型的构建过程都需要经过上述步骤。上述建模过程在两个层次上对二进制代码安全缺陷的结构进行分析: 不仅在程序语句的层次把其结构分解为模型元素集合, 而且在程序执行路径的层次把其结构分解为路径单元和组合单元。这使分析模型完整包含了二进制代码安全缺陷的结构特征信息, 不仅其包含的 n ($n \geq 1$) 个路径单元可以准确地描述二进制代码安全缺陷的 n 个不同组成部分, 并且其包含的组合单元给出了该 n 个组成部分应满足的必要条件。因此, 分析模型能够为准确地检测二进制程序中代码安全缺陷提供充分支持及有效的判定标准。

3 模型应用

应用分析模型可以实现对二进制代码安全缺陷的有效检测, 分析模型的应用原理如下。

1) 依据分析模型中信息建立二进制代码安全缺陷的判定标准。以分析模型包含的 n ($n \geq 1$) 个路径单元分别作为二进制代码安全缺陷中 n 个不同组成部分的识别标准, 并把分析模型中组合单元作为该 n 个组成部分之间的匹配标准, 由 n 项识别标准与 1 项匹配标准共同构成二进制代码安全缺陷的判定标准。

2) 基于判定标准把检测内容划分为 n 项查找任务和 1 项组合任务。其中 n 项查找任务分别在被测试程序代码中查找满足不同识别标准的代码组, 组合任务是在查找得到的代码组中选出满足匹配标准的 $t \times n$ ($t \geq 1$) 个代码组, 并把它们组成 t 个二进制代码安全缺陷。

3) 依据各路径单元分别描述的结构特征确定检测过程中各项任务的执行次序, 从而使检测过程具有最高的效率。

依据上述原理把分析模型应用于二进制代码安全缺陷的检测过程, 得到分析模型的应用方法, 其简称为模型应用方法。该方法依据分析模型中结构特征信息实施对二进制代码安全缺陷的有效检测。下面给出该方法的主要内容。基于分析模型, 定义以下若干关系。

定义 6 (语句项与 IR 语句的匹配) 设 η_s 、 u_s 分别为语句项及 IR 语句。对 η_s 中任意字符串 a_{si} , 记为 $a_{si} \in \eta_s$ 。若 u_s 中数据 d 的信息与字符串 a_{si} 标记的数据信息相一致, 则称 d 符合 a_{si} 。若 η_s 与 u_s 之间成立以下条件: ① η_s 中字符串标记的语句类型与 u_s 的语句类型相同; ② 对于任意数据字符串 $a_{si} \in \eta_s$, 存在数据 $d \in u_s$, d 符合 a_{si} 。则称 u_s 为 η_s 的像, 并称 η_s 与 u_s 匹配, 记为 $\eta_s \leftrightarrow u_s$ 。

定义 7 (路径单元与 IR 代码组的匹配) 设 D 、 λ 分别为路径单元及 IR 代码组。对任意模型元素 $(\eta_s, \delta, \eta_t) \in D$, 其中 η_s 、 η_t 称为 D 的语句项, 记为 $\eta_s \subset D$ 、 $\eta_t \subset D$ 。若 D 的语句项与 λ 的语句具有一一对应关系, 成立以下条件: ① 对任意 $\eta_s \subset D$ 及 $u_s \subset \lambda$, 若 η_s 与 u_s 对应, 则 η_s 与 u_s 匹配; ② 对任意 $(\eta_s, \delta, \eta_t) \in D$, 存在 $u_s, u_t \subset \lambda$, u_s 、 u_t 分别与 η_s 、 η_t 对应, 且 u_s 与 u_t 之间具有向量 δ 中的全部关系。则称 λ 为 D 的像, 并称 D 与 λ 匹配, 记为 $D \leftrightarrow \lambda$ 。

定义 8 (路径单元相关) 设 $\langle M_1, M_2 \rangle$ 是代码安全缺陷特征模型, D_i 、 D_j 是 M_1 中 2 个路径单元。若成立如下任意一项条件:

- ① 存在语句项 $\eta_s \subset D_i$, 满足 $\eta_s \subset D_j$;
- ② 存在模型元素 $(\eta_s, \delta, \eta_t) \in M_2$, 满足 $\eta_s \subset D_i$, $\eta_t \subset D_j$, $ID(a_{si}, a_{tj}) \in \delta$, $a_{si} \in \eta_s$, $a_{tj} \in \eta_t$ 。

则称路径单元 D_i 与 D_j 相关, 记为 $D_i \cap D_j$ 。若条件①成立, 则语句项 η_s 称为 D_i 与 D_j 的相关语句项。若条件②成立, 则模型元素 (η_s, δ, η_t) 称为 D_i 与 D_j 的相关元素。

定义 9 (路径单元与 IR 代码组的相关匹配) 设 $D_i \cap D_j$, $D_i \leftrightarrow \lambda_i$, $D_j \leftrightarrow \lambda_j$, λ_i 与 λ_j 为不同的 IR 代码组。若成立以下条件:

- ① 若 D_i 与 D_j 具有相关语句项 η_s , 则存在 λ_i 和 λ_j 的共同语句 u_s , 满足: $\eta_s \leftrightarrow u_s$;
- ② 若 D_i 与 D_j 具有相关元素 (η_s, δ, η_t) , 则存在 IR 语句 $u_s \subset \lambda_i$ 及 $u_t \subset \lambda_j$, 满足 u_s 、 u_t 分别为 η_s 、 η_t 的像, 且它们之间具有向量 δ 中全部关系。

则称路径单元 D_i 、 D_j 与 IR 代码组 λ_i 、 λ_j 相关匹配。

基于上述关系,下面给出对任意 Ψ 类型二进制代码安全缺陷的 5 个主要检测步骤。

步骤 1 建立分析模型。全面选择各种不同结构的 Ψ 类型缺陷源代码,然后应用它们分别建立分析模型。设共得到 m ($m \geq 1$) 种 Ψ 类型二进制代码安全缺陷的分析模型。

步骤 2 建立分析模型的权重集合。主要过程如下。

① 度量各种类型 IR 语句的相对比率。首先,随机选择 x 个具有 Ψ 类型代码安全缺陷的有代表性的二进制程序,应用 VTS 获取它们各自 y 次不同执行过程产生的 IR 执行路径: $q_1, \dots, q_n$ 。然后,分别统计 $q_1 \sim q_n$ 中不同类型 IR 语句所占的比率。该步骤中 x 和 y 的取值与特征模型的复杂度成正相关。

② 分别建立 m 种特征模型的权重集合。对任意一种特征模型 $\langle M_1, M_2 \rangle$, 首先计算 M_1 中各路径单元的权重。设 M_1 中任意路径单元 D_r 的语句项集合为 $\{\eta_1, \dots, \eta_k\}$, η_i ($1 \leq i \leq k$) 中字符串标记的语句类型记为 o_i , 则 D_r 的权重为:

$$v_r = \prod_{i=1}^k \left(\frac{\sigma_i}{n} \sum_{j=1}^n \omega_{ij} \right)^{-1}, \text{ 其中 } \omega_{ij} \text{ 是 IR 执行路径 } q_j$$

($1 \leq j \leq n$) 中 o_i 类型 IR 语句所占的比率, σ_i 是权重系数。然后以 M_1 中各路径单元的权重信息为元素建立 $\langle M_1, M_2 \rangle$ 的权重集合 W 。对于任意路径单元 $D_r \in M_1$, 若其权重为 v_r , 把权重信息 $-D_r - v_r$ 作为元素加入权重集合 W 。

步骤 3 获取被测试二进制程序的 IR 执行路径集合。首先根据被测试程序的类型和功能,生成测试数据集作为程序输入。然后应用 VTS 监控该二进制程序的不同动态执行过程,获得 IR 执行路径集合 Q 。

步骤 4 缺陷 IR 代码检测。分别输入 m 种分析模型,应用缺陷 IR 代码检测算法,在集合 Q 包含的 IR 执行路径中检测 Ψ 类型缺陷 IR 代码,如算法 3 所示。

算法 3 缺陷 IR 代码检测算法 FlawIRCode-Detection

输入 分析模型 $\langle M_1, M_2 \rangle$ 及其权重集合 W , IR 执行路径集合 Q

输出 缺陷 IR 代码集合 IS

1) Set $IS \leftarrow \emptyset, IG \leftarrow \emptyset, E_1 \leftarrow \emptyset, CT \leftarrow \emptyset, RL \leftarrow \emptyset, M'_1 \leftarrow M_1$

2) while $M_1 \neq \emptyset$ do

3) if $E_1 \neq \emptyset$ then

4) $E_2 \leftarrow \emptyset$

5) for each $D_i \in M_1$ do

6) $RL \leftarrow \text{CorrelationUnit}(D_i, E_1, M_2)$

7) if $RL \neq \emptyset$ then

8) if $SI(Q, D_i, RL, CT, IG) = \emptyset$ then

9) return IS

10) $E_2 \leftarrow E_2 \cup \{D_i\}$

11) $M_1 \leftarrow M_1 - \{D_i\}$

12) $E_1 \leftarrow E_2$

13) else

14) if $MW(W, M_1, Q, E, CT, IG) = \emptyset$ then

15) return IS

16) for each $(\eta_s, \delta, \eta_t) \in M_2$ do

17) if $\text{AssembleCodeGroup}(Q, (\eta_s, \delta, \eta_t), M'_1, CT, IG) = \emptyset$ then

18) return IS

19) return $IS \leftarrow \text{GenerateInstance}(CT, IG)$

算法 3 中应用了多个函数,分别阐述各函数的功能如下。

1) 函数 $\text{CorrelationUnit}(D_i, E_1, M_2)$ 在集合 E_1 中查找与 D_i 相关的路径单元,依据查找结果生成识别条件。该函数返回识别条件集合 RL 。

2) 函数 $SI(Q, D_i, RL, CT, IG)$ 遍历集合 Q 中执行路径查找路径单元 D_i 的像,以 RL 中识别条件对 D_i 的像进行有效性验证,并根据有效的验证结果更新集合 CT 和 IG 。对于查找得到的任意一个 D_i 的像 λ_i ,若 λ_i 满足 RL 中所有识别条件,则把 λ_i 放入 D_i 的像集 $G_i \in IG$,并把验证过程得到的各条相关匹配信息加入集合 CT 。该函数最后返回 G_i 。

3) 函数 $MW(W, M_1, Q, E_1, CT, IG)$ 首先基于集合 W 中权重信息选择 M_1 中权重最大的路径单元。设得到 M_1 中权重最大的路径单元 D_j 。然后遍历 Q 中 IR 执行路径查找 D_j 的像,把查找得到的各 IR 代码组放入 D_j 的像集 G_j 。若检测结束后 G_j 不为空集,则把 D_j 、 G_j 分别放入集合 E_1 、 IG ,并从 M_1 中去除 D_j 。该函数最后返回集合 G_j 。

4) 函数 $GP(Q, (\eta_s, \delta, \eta_t), M'_1, CT, IG)$ 首先在 M'_1 中查找包含 η_s 、 η_t 的路径单元,设得到 $\eta_s \subset D_i$ 、 $\eta_t \subset D_j$ 。然后在 IG 中选择 D_i 、 D_j 的像集 G_i 、 G_j ,基于 (η_s, δ, η_t) 检测 G_i 、 G_j 中冗余的 IR 代码组,并

更新集合 CT 中信息。对 G_i 中任意一个 IR 代码组 λ_i ，若 G_j 中存在 IR 代码组 λ_j ，满足 λ_i 与 λ_j 不包含于同一条 IR 执行路径，且 η_s, η_t 的像 $u_s \subset \lambda_i, u_t \subset \lambda_j$ 之间具有向量 δ 中全部关系，则把 D_i, D_j 与 λ_i, λ_j 的相关匹配信息作为元素加入集合 CT 。否则，从 CT 中去除同时包含 D_i 和 λ_i 信息的元素。该函数最后返回 CT 中同时包含 D_i 和 D_j 信息的元素构成的子集。

5) 函数 $\text{GenerateInstance}(CT, IG)$ 基于集合 CT 中的信息，把 IG 中各集合包含的 IR 代码组组合为缺陷 IR 代码。若 IG 中集合不包含缺陷 IR 代码，则该函数返回空集。否则，该函数返回集合 $IS = \{\xi_1, \dots, \xi_t\}$ ，其中 ξ_i ($1 \leq i \leq t$) 为检测得到的一个 Ψ 类型缺陷 IR 代码。

步骤 5 获取 Ψ 类型缺陷二进制代码。基于集合 IS 中缺陷 IR 代码，应用 VTS 输出的 IR 语句与机器指令之间的对应信息，查找出被测试二进制程序中 t 个 Ψ 类型缺陷二进制代码，它们构成了该程序中 t 个 Ψ 类型代码安全缺陷。

基于以上检测过程可知，上述模型应用方法具有如下功能和优点：①对任意类型代码安全缺陷的检测功能；②对结构复杂的代码安全缺陷的检测功能；③能够准确地识别代码安全缺陷，从而有效地避免检测结果的误报和漏报；④检测过程无需触发二进制代码安全缺陷。

4 实验验证

实验目的为验证二进制代码安全缺陷分析模型的实用性和有效性。为此，实验中对模型应用方法进行验证，实验内容包括：①方法的功能和优点验证；②方法的有效性验证。根据该方法实施的检测过程可知，上述实验内容能够准确地验证分析模型的实用性和有效性。

实验中被测试代码安全缺陷的选择标准如下：①其类型具有代表性，该类型的代码安全缺陷广泛存在于各类软件系统及二进制程序，且目前对其尚无有效的检测方法；②具有复杂的结构；③具有较大的识别难度；④具有较大的检测难度；⑤包含于结构较复杂的二进制程序。

基于以上选择标准，实验中选择二进制多线程程序中竞争条件 (race condition) 类型代码安全缺陷作为检测对象。基准测试程序如表 1 所示，

$B_1 \sim B_3$ 分别以算法 1~3 作为线程同步算法，其中算法 1、2 能够产生竞争条件。算法 1~3 是语义较复杂、有代表性的同步算法。Holzmann 在其被引用 4 680 多次的论文 “The Model Checker SPIN”^[13] 中给出算法 1、2，其中算法 2 是对算法 1 的修正算法。Bang 等在论文 “Comments on ‘The Model Checker SPIN’”^[14] 中以实时进程代数 ACSR^[15] 检测出算法 2 并未消除竞争条件，于是给出算法 1 的正确修正算法——算法 3。

表 1	基准测试程序	
	二进制程序	同步算法
		算法缺陷
	B_1	算法 1
	B_2	算法 2
	B_3	算法 3
		竞争条件
		竞争条件
		无

通过对 $B_1 \sim B_3$ 的测试，不仅可以有效地验证模型应用方法的功能和优点，而且能够准确地验证该方法的有效性，依据如下：①算法 1~3 是语义较复杂的同步算法，使 $B_1 \sim B_3$ 的二进制代码结构较复杂，具有较大的检测难度；② B_1 、 B_2 中竞争条件类型代码安全缺陷包含复杂的代码结构，使它们具有较大的识别难度和检测难度；③ $B_1 \sim B_3$ 的代码结构相似，使 B_3 中存在多个二进制代码组与 B_1 、 B_2 中缺陷二进制代码具有相似的代码结构，所以 B_3 中这些代码组能够有效地检验模型应用方法对代码安全缺陷识别能力，从而验证该方法是否能够有效地避免检测结果的误报和漏报；④ $B_1 \sim B_3$ 的动态执行过程被实时监控，若产生竞争条件可以被及时发现和记录。

实验中，计算机硬件配置为 Intel Core 2 Duo (CPU 2.4 GHz)、2 GB 内存及 4 MB 二级缓存，操作系统为 Ubuntu Linux 14.04 (4.4 HWE kernel)，处理器架构为 X86/Linux。程序分析工具 VTS 中 Valgrind 版本为 3.12.0。Linux C 源代码应用 GCC 5.3 编译为二进制程序。

4.1 方法功能和优点验证

首先以模型应用方法测试 $B_1 \sim B_3$ 中竞争条件类型代码安全缺陷。测试结果如下： B_1 和 B_2 中分别存在竞争条件类型代码安全缺陷 χ_1, χ_2 ，如表 2、表 3 所示。根据 $B_1 \sim B_3$ 的动态执行过程可知，该测试过程没有使 $B_1 \sim B_3$ 中线程之间产生竞争条件。

表2 χ_1 中机器指令

地址	指令
0804872B	mov ds:r_want,l
08048761	mov eax, state
08048766	cmp eax,l
08048769	jz shortloc_8048777
08048813	mov ds:r_want,0
08048826	mov state,l
08048735	mov state, 0
080487FE	move ax, ds:r_want
08048803	test eax, eax
08048805	jz shortloc_8048848
0804881D	mov eax, state
08048822	test eax, eax
08048824	jnz shortloc_804883C

表3 χ_2 中机器指令

地址	指令
0804872B	mov ds:r_want,l
08048761	mov eax, state
08048766	cmp eax,l
08048769	jz shortloc_8048777
08048813	mov ds:r_want,0
0804883E	mov state,l
08048735	mov state, 0
080487FE	move ax, ds:r_want
08048803	test eax, eax
08048805	jz shortloc_8048860
08048835	mov eax, state
0804883A	test eax, eax
0804883C	jnz shortloc_8048854

然后以文献[13,14]中的正确结论与上述测试结果进行对比分析,得到模型应用方法的测试结果完全正确,不存在误报和漏报。因此,该方法的功能和优点得到全面验证。

4.2 方法有效性验证

Helgrind^[16]和DRD^[17]是两款检测线程同步错误的成熟工具,能够有效地检测出多种类型的线程同步错误。由于Helgrind和DRD分别应用了国际上现有的多种实用检测方法,所以根据它们对B₁~B₃的检测结果可以准确地评估模型应用方法的有效性。

实验中以2017年6月最新发布的Helgrind和DRD对B₁~B₃检测,它们在完成检测后分别给出对多个数据变量的警告。其中Helgrind对B₁中r_want的警告如图3所示,DRD对B₃中state

(0x0804a048)的警告如图4所示。经过检验,Helgrind和DRD的检测结果均漏报了B₁和B₂中竞争条件类型代码安全缺陷,且它们给出的各条警告均为误报。根据Helgrind和DRD的检测结果可以得到评估结论:相比较国际上现有检测方法,模型应用方法可以有效地应用于二进制并行程序,能够检测出现有方法难以发现的结构复杂的代码安全缺陷,具有更加全面和深入的检测能力,并且其检测结果具有更高的准确率。由此模型应用方法的有效性得到验证。

```
Possible data race during write of size 4 at 0x804a054 by thread #32
Locks held: 1, at address 0x804a05c
  at 0x804872B: client (in /home/success/Binary/Multithread)
  by 0x402f88D: mythread_wrapper (hg_intercepts.c:389)
  by 0x4058f6f: start_thread (pthread_create.c:312)
  by 0x415e70d: clone (clone.S:129)

This conflicts with a previous read of size 4 by thread #33
Locks held: none
  at 0x80487FE: server (in /home/success/Binary/Multithread)
  by 0x402f88D: mythread_wrapper (hg_intercepts.c:389)
  by 0x4058f6f: start_thread (pthread_create.c:312)
  by 0x415e70d: clone (clone.S:129)
Address 0x804a054 is 0 bytes inside data symbol "r_want"
```

图3 Helgrind对B₁中“r_want”的警告

```
Conflicting store by thread 3 at 0x0804a048 size 4
  at 0x804882D: server (in /home/success/Binary/Multithread)
  by 0x402fEE5: vgDrD_thread_wrapper (drd_pthread_intercepts.c:444)
  by 0x4063f6f: start_thread (pthread_create.c:312)
  by 0x416670d: clone (clone.S:129)
Allocation context: Data section of /home/success/Binary/Multithread
```

图4 DRD对B₃中“state”的警告

基于以上两项实验内容的结论可知,分析模型可以应用于任意类型二进制程序的代码安全缺陷检测过程,其包含的信息可以为识别与检测二进制代码安全缺陷提供充分的支持及有效的判定标准,使任意类型结构复杂的二进制代码安全缺陷能够被准确和高效地检测出来。所以二进制代码安全缺陷分析模型的实用性和有效性得到验证。

5 结束语

为实现对结构复杂的二进制代码安全缺陷的有效检测,本文给出对二进制代码安全缺陷的分析原理,提出二进制代码安全缺陷的分析模型,并依据分析模型的应用原理,给出分析模型的应用方法。首先提取缺陷IR代码的关键特征,抽象其中结构信息,生成二进制代码安全缺陷的分析模型。然后用工具VTS获取二进制程序的IR执行路径集合,基于各类语句的统计概率计算路径单元的权重,依据缺陷IR代码检测算法遍历IR执行路径,查找满足特征模型的IR代码组,并通过IR代码与二进制代码的转换信息,检测出二进

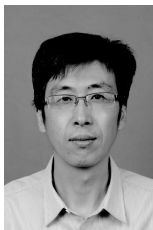
制程序中代码安全缺陷。

实验结果表明, 分析模型兼具实用性和有效性, 应用分析模型可以检测出现有方法难以发现的各类结构复杂的二进制代码安全缺陷, 分析模型的应用方法无需源代码, 能够直接检测二进制应用程序, 极大提高了检测结果准确率, 具有简单、方便的优点。分析模型可以应用于软件开发维护、漏洞挖掘与利用及程序分析与测试等多种领域。同时, 分析模型能够有效地应用于现有方法难以检测的二进制并行程序, 所以随着网络和并行系统的发展, 其具有广泛的应用前景。

参考文献:

- [1] LANDWEHR C E, BULL A R, MCDERMOTT J P, et al. A taxonomy of computer program security flaws[J]. Computing Surveys, 1994, 26(3): 211-254.
- [2] WEBER S, KARGER P A, PARADKAR A. A software flaw taxonomy: aiming tools at security[C]//The Workshop on Software Engineering for Secure Systems—Building Trustworthy Applications. 2005: 1-7.
- [3] HUI Z, HUANG S, REN Z, et al. Review of software security defects taxonomy[C]//The 5th International Conference on Rough Set and Knowledge Technology, Lecture Notes in Computer Science. 2010: 310-321.
- [4] ZHANG B, FENG C, WU B, et al. Detecting integer overflow in windows binary executables based on symbolic execution[C]//The 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. 2016:385-390.
- [5] SIDIROGLOU-DOUSKOS S, LAHTINEN ERIC, RITTENHOUSE N, et al. Targeted automatic integer overflow discovery using goal-directed conditional branch enforcement[C]//The Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems.2015: 473-486.
- [6] YADEGARI B, STEPHENS J, DEBRAY S. Analysis of exception-based control transfers[C]//The Seventh ACM on Conference on Data and Application Security and Privacy.2017: 205-216.
- [7] MING J, WU D, WANG J, et al. StraightTaint: decoupled offline symbolic taint analysis[C]//The 31st IEEE/ACM International Conference on Automated Software Engineering.2016: 308-319.
- [8] CHA S K, WOO M, BRUMLEY D. Program-adaptive mutational fuzzing[C]//The IEEE Symposium on Security and Privacy.2015: 725-741.
- [9] PHAM V, BÖHME M, ROYCHOUDHURY A. Model-based whi-
- tebox fuzzing for program binaries[C]//The 31st IEEE/ACM International Conference on Automated Software Engineering. 2016: 543-553.
- [10] The valgrind developers[EB/OL]. <http://valgrind.org>.
- [11] NETHERCOTE N, SEWARD J. Valgrind: a framework for heavyweight dynamic binary instrumentation[J]. ACM SIGPLAN Notices, 2007, 42(6):89-100.
- [12] [EB/OL]. <https://github.com/wmkhoo/taintgrind>.
- [13] HOLZMANN G J. The model checker SPIN[J]. IEEE Transactions on Software Engineering, 1997, 23(5): 279-295.
- [14] BANG K, CHOI J, YOO C. Comments on “the model checker SPIN”[J]. IEEE Transactions on Software Engineering, 2001,27(6): 573-576.
- [15] BRÉMOND-GRÉGOIRE P, CHOI J, LEE I. A complete axiomatization of finite-state ACSR processes[J]. Information and Computation, 1997,138 (2): 124-159.
- [16] The valgrind developers.[EB/OL]. <http://valgrind.org/docs/manual/hg-manual.html>.
- [17] The valgrind developers[EB/OL]. <http://valgrind.org/docs/manual/drd-manual.html>.

作者简介:



许团(1973-), 男, 黑龙江鹤岗人, 中国人民大学博士生, 主要研究方向为信息安全。



屈蕾蕾(1995-), 女, 新疆乌鲁木齐人, 中国人民大学博士生, 主要研究方向为可信计算、云安全。



石文昌(1964-), 男, 广西浦北人, 中国人民大学教授、博士生导师, 主要研究方向为系统安全、可信计算与数字取证。