一种基于程序切片相似度匹配的脆弱性发现方法

刘 强[1,2] 况晓辉[1,3] 陈 华[1] 李 响[1] 李广轲[3]
(军事科学院系统工程研究院信息系统安全技术国家级重点实验室 北京 100101)[1]
(清华大学计算机科学与技术系 北京 100084)[2] (北京邮电大学 北京 100876)[3]

摘 要 基于脆弱性代码的相似度匹配是静态发现脆弱性的有效方法之一,在不降低漏报率的情况下如何降低误报率和提升分析效率是该方法优化的主要目标。针对这一挑战,提出了基于代码切片相似度匹配的脆弱性发现框架。文中研究了基于关键点的代码切片、特征抽取和向量化的方法,主要思想是以脆弱性代码的脆弱性语义上下文切片作为参照物,通过计算被测代码的切片与脆弱性样本切片的相似性来判断存在脆弱性的可能性。文中实现了该方法,并以已知脆弱性的开源项目为分析对象进行了验证。与已有研究的对比实验表明,切片相似度能更准确地刻画脆弱性上下文,通过切片技术优化了基于相似度匹配的脆弱性发现方法,有效降低了脆弱性发现的误报率和漏报率,验证了所提框架和方法的有效性。

关键词 脆弱性分析,程序切片,相似度匹配
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.07.020

Vulnerability Discovery Approach Based on Similarity Matching of Program Slicing

LIU Qiang[1,2] KUANG Xiao-hui[1,3] CHEN Hua[1] LI Xiang[1] LI Guang-ke[3]
(National Key Laboratory of Science and Technology on Information System Security, Institute of System and Engineering, Academy of Military Science, Beijing 100101, China)[1]
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)[2]
(Beijing University of Posts and Telecommunications, Beijing 100876, China)[3]

Abstract Vulnerability analysis method based on similarity matching is one of the most effective vulnerability analysis methods. How to reduce the false positive rate without reducing the false negative rate, and increase the efficiency, are the main goals to optimize the method. Aiming at these challenges, this paper proposed an optimization vulnerability analysis framework based on similarity matching of program slices. In the framework, the methods of code slice, feature extraction and vectorization based on vulnerable key points were studied. The core ideal of the framework is taking vulnerability semantic context slice of the vulnerability code as a reference to calculate the similarity between the slice of the tested code and the vulnerable sample slice, and determining the likelihood of a vulnerability. This paper implemented this framework and validateed it with open source projects with known vulnerabilities. Compared with the existing research, the vulnerability slice similarity framework has the ability to close describe the vulnerability context, and the vulnerability discovery method based on similarity matching is optimized by the slice technique. The proposed framework and method are verified to effectively reduce the false positive rate and false negative rate of vulnerability discovery.

Keywords Vulnerability analysis, Programing slicing, Similarity matching

1 引言

软件脆弱性是指软件存在的内在缺陷,其可能被利用从而引发违背或破坏系统安全状态,造成可用性、授权访问、数据的机密性等安全需求的受损[1]。为了提高软件系统的安全性,研究者和相关组织提出了一系列安全编程和最佳实践规

到稿日期:2018-06-09 返修日期:2018-11-26

刘 强(1979-),男,博士生,工程师,CCF学生会员,主要研究方向为网络与信息安全,E-mail:liuqiang_xjtu@163.com;况晓辉(1975-),男,教授,硕士生导师,CCF高级会员,主要研究方向为网络与信息安全,E-mail:xiaohui_kuang@163.com(通信作者);陈 华(1962-),女,研究员,硕士生导师,主要研究方向为信息安全;李 响(1983-),女,工程师,主要研究方向为网络与信息安全;李广轲(1995-),男,硕士生,主要研究方向为网络与信息安全。

范,然而由于人们对信息系统的高度依赖和软件的复杂性,软件脆弱性的出现越来越频繁,使得脆弱性分析和挖掘成为学术界和工业界研究的热点领域[1-3]。

基于相似度匹配的脆弱性分析方法通过提取脆弱性代码的特征,然后在待分析程序中匹配特征,来寻找与已知脆弱性相似的代码,将其作为疑似脆弱性代码进行报告或供进一步的分析。相似性比对方法原本应用于程序源代码抄袭,将其应用于脆弱性发现则开辟了一条在研究思路上有别于经典的基于规则的静态和动态脆弱性方法,值得进一步研究。

针对现有基于相似度匹配的脆弱性分析方法在特征提取精度、脆弱性发现准确性等方面存在的不足,本文提出了基于程序切片相似度匹配的脆弱性分析方法。该方法以典型脆弱性特征为热点对代码进行切片,在切片代码的向量空间上进行相似度比较,与脆弱性代码切片具有较高相似度的被测代码切片有较大概率存在脆弱性。通过切片方法减少了与脆弱性无关代码的干扰,提高了脆弱性发现的准确性。

本文在回顾相关研究工作的基础上,首先提出了基于切片相似度的脆弱性发现方法框架;接着从切片算法、脆弱性特征提取、切片向量化、脆弱性算法等方面展开方法讨论;最后,基于切片和相似度计算实现了基于切片相似性匹配的脆弱性分析原型系统,并针对实际脆弱性项目开展实验,验证了所提方法的有效性。

2　相关工作

2.1　程序切片技术

Weiser 在 1979 年首次引入了程序切片的概念,这是一种从与特定计算相关的程序语句中提取代码的静态分解技术[4],该技术以代码中选取的兴趣点为切片标准计算程序的后向静态切片。Korel 等在 1988 年提出了动态切片的概念[5]。动态切片是基于程序执行的,在给定一些特定的输入下实现,是一种适用于软件调试的技术。Lyle 等提出了面向程序错误分析的切块方法[6],其核心思想是观察给定变量输入不正确值和正确值时的静态切片的差异来发现代码错误。这种切块方法减少了错误报告的数量且能有效定位软件中的故障。Mark 等提出了一种传统程序切片的变体,称为无定形切片[7],其核心思想是不把语句删除作为简化的唯一手段,并且将一些语义属性作为兴趣点。这种方法得到的切片比基于语法的切片更小。此外,其他程序切片概念还包括准静态切片[8]、约束切片[9]、条件切片[10]、相关切片[11]等,限于篇幅不一一展开。

在代码切片的方法上,围绕代码的不同图形化中间表达形式及其优化衍生出了不同方法。Weiser 首次提出了基于程序控制流图(CFG)的静态代码切片方法[4]。Karl 等首先提出了程序依赖图(PDG)来计算程序切片[12],该程序切片方法的数学基础是图可达性。Danicic 等提出了并行处理方法的网络表示,并提出了一种并行分割算法[13]。Bergeretti 等[14]提出了基于信息流的程序切片。张新杰等以传统的数据流方程和图的可达性理论为基础,提出了基于基本块的切片和基于路径图的程序切片[15]。更多优化方法可以参考程序切片方法的相关综述[16-18]。

切片工具方面,有很多基于经典理论的成果,代表性工具包括 Wisconsin,CodeSurfer,Frama-C 等。Wisconsin 切片工具是较早的一款免费的切片工具,该工具基于程序依赖图和系统依赖图实现了前向和后向切片能力,目前已经停止更新[19]。CodeSurfer 是一款商业工具,是在 X86 平台上实现的基于数据依赖、控制依赖等分析方法的代码分析商业平台,提供代码切片功能[20]。Frama-C 是一款专为源代码的验证而设计的程序分析平台,基于流分析、指向分析等技术实现的切片是该平台的核心基础功能,它基于切片还可以通过插件方式实现功能的扩展[21]。这些基于经典程序分析理论的切片工具大多在流敏感、上下文敏感、路径敏感方面尽可能地提升精度,从而获得更加准确的切片以利于后续分析和优化,但缺点是分析复杂度和代价越来越高。另一些代表性切片工具则朝着技术融合和轻量级方向发展。VERMEER 把程序切片技术和 SMT 求解器结合起来,用于发现程序中的错误,大幅提升了发现缺陷的效率[22]。开源切片工具 srcSlice 是一个轻量级的前向切片工具,该工具以 srcML 工具生成的抽象语法树 XML 中间表示为输入,在分析的工程中不会预先构造程序依赖图,而是在对每个变量生成切片时快速地分析其依赖关系,在线生成代码关于变量的系列切片[23-24]。李润青等针对代码复用和代码搜索面临的查找精度低的问题,把切片算法融入到搜索引擎工具中,通过实验证明了该工具和基于自然语言的搜索工具在代码查询上的改进效果[25]。

2.2　基于相似性匹配的脆弱性发现方法

基于相似性的分析方法是一种建立在参照物比对基础上的方法。从代码角度基于相似性匹配来开展脆弱性发现是一种基于相似性预期来判断疑似脆弱性的静态方法。该方法不依赖于脆弱性规则,通过相似性值来量化地筛选或者判定脆弱性,实现快速收缩脆弱性发现的范围或者达到脆弱性发现的目的。这种判断的策略包括如下两种:1)和已知存在脆弱性的代码具有高度相似性;2)和大量类似正常代码存在相似性上的巨大差异。基于这种思想,研究者提出了各种脆弱性的相似性比对方法,以提升脆弱性分析的准确性。代表性的研究和成果如下:Qian 等针对 IoT 设备二进制代码的脆弱性分析挑战,提出了基于异常分析的跨平台代码脆弱性发现方法[26],该方法以条件判断缺失或无效为该类型脆弱性的主要特征,对脆弱性实例进行特征抽取,用于判断其他平台下的二进制代码是否存在类似的脆弱性。Yamaguchi 围绕基于相似度匹配的脆弱性发现开展了多项工作[27-30],其通用的思想就是对样本进行基于特征的表达并进行向量化,然后利用与已知脆弱性的相似性来度量函数存在类似脆弱性的可能性。基于不同层次的特征采集思路,Yamaguchi 取得了系列研究成果:2011 年,基于 API 符号相似度匹配的脆弱性分析成果中,函数是使用所包含的所有 API 调用名称来表示的[27];2012

年,其使用了基于抽象语法树的嵌入来表达函数,与 2011 年的工作相比增加了语法结构关系,提升了脆弱性发现的准确能力[28];2013 年,针对污染传播未检查类型的脆弱性,其提出了基于异常检测的脆弱性分析方法,并开发了原型系统 Chucky,它基于待测样本和大多数相似函数的污染检查特征向量的偏离程度来判定脆弱性[29];2015 年,其针对污染传播未清洗的脆弱性,进一步提出了融合抽象语法树、控制流、数据流等代码属性图来表征风险函数的上下文,通过聚类分析生成了相应的脆弱性模式,进一步提升了脆弱性发现的能力和准确性[30]。2017 年,Xu 等针对目前大规模条件下相似度检测速度慢且不准确的问题,提出了一种基于神经网络的控制流图嵌入方法,大幅度提升了跨平台的二进制代码函数的相似性匹配效率和准确性,实现了跨平台固件脆弱性发现的原型系统 Gemini[31]。甘水滔等提出了一种基于特征矩阵的脆弱性代码克隆检测方法[32],该方法从不同克隆代码类型的语法分析树中提取特征,并用特征矩阵表示代码,通过对代码进行基于多种克隆类型的聚类计算,达到了从被测软件代码中检测脆弱代码的目的。限于篇幅,仅列举了相关研究工作中的部分典型工作。

2.3 现状分析与研究动机

基于软件切片的朴素思想,伴随着软件技术的发展,切片理论和方法也不断形成了不同分支,被广泛应用于软件度量、程序测试、代码分析领域,在提升程序的可靠性和安全性等方面奠定了理论和方法基础。

相似性判定方法符合人工根据经验判断脆弱性的逻辑,这种方法尽管不能避免误报率,甚至还存在高漏报率,但可以基于已有的脆弱性样本实现可以量化的脆弱性预测,从而极高效地发现脆弱性或者为脆弱性指明方向。

因此,本文将两种方法结合起来,提出一种面向脆弱性发现的切片相似性方法,以提升基于相似性的脆弱性发现的准确性。但是从分析研究现状可以得知,在脆弱性发现的应用背景下已有研究还有以下挑战:

1)在代码切片方面,现有的切片方法是以代码分解为本质的,而面向脆弱性发现的切片则希望反映脆弱性从输入到缺陷触发的上下文,因此,需要建立适合的表达脆弱性的切片模型和算法。

2)在相似性判定方面,一方面由于被检测代码数量巨大,面临着如何高效地生成被测代码切片以及快速高效比对的问题,另一方面,需要结合脆弱性机理设计合理的相似性判定算法以获得较好的判定准确度。

本文把脆弱性特征作为切片和相似度的结合关键点,围绕脆弱性相关特征的抽取和表达方法展开,使得脆弱性样本的表达既满足准确性又具有一定的泛化能力。因此,样本脆弱性上下文提取和表达是提升基于相似度匹配发现脆弱性的核心。本文方法的思想是获取脆弱性相关的切片,进行特征抽取,在此基础上进行相似性比对,以提升基于相似性的脆弱性发现的准确性。

3 框架和方法

3.1 基于切片相似度匹配的脆弱性分析框架

图 1 是基于切片相似度匹配的脆弱性分析框架,该方法分两个部分:1)在脆弱性切片准备部分,首先对脆弱性样本进行程序切片,获取与脆弱性密切相关的程序片段,然后通过特征提取算法获得脆弱性代码的特征,并将其映射到向量空间;2)在脆弱性检测应用部分,待检测代码采取和脆弱性样本切片相同的程序切片准则、特征抽取和向量空间映射算法,然后计算其与脆弱性切片的向量距离,向量距离越近则相似程度越高,该程序越有可能存在脆弱性。

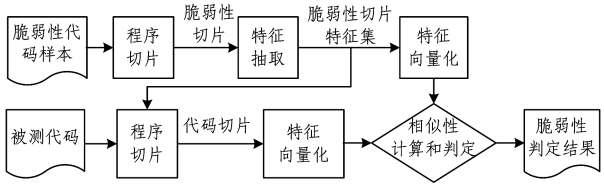

图 1 基于切片相似性匹配的脆弱性分析框架

Fig. 1 Vulnerability discovery framework based on similarity matching of program slicing

3.2 基于切片相似度比对的脆弱性发现算法

3.2.1 脆弱性上下文切片

在脆弱性分析应用中,程序的复杂性导致脆弱性分析面临挑战。研究脆弱性发生上下文有助于降低待分析程序的规模,引入切片则使得分析更加关注于那些感兴趣的脆弱性上下文代码。

设计一种切片方法,需要为切片定义切片方向和准则。按照 Weiser 的经典定义,如果语句和判定表达式可能会影响在程序的某个位置 P 上的变量 V 的值,那么这些语句和表达式的集合为程序的后向切片;如果语句和判定表达式可能被在程序的某个位置 P 上的变量 V 的值影响,这些语句和表达式的集合为程序的前向切片。因此,将(V,P)称为切片准则(slicing criterion),程序切片是按照切片准则从程序中抽取的部分语句和判定表达式组成的集合。程序的图形化表示是切片的基础,基于程序依赖图或者系统依赖图的切片和基于数据流图的切片是其中两种最主要的通用方法。

由于脆弱性上下文在代码中具有的局部特性,切片首先需要确定脆弱性切片的头部和尾部,这对已经披露的 CVE 脆弱性通常是已知的。同时,在仅仅标注了头部和尾部的情况下,基于图的可达性算法会生成很多切片,而其中大部分切片与脆弱性无关。为了解决这个问题,建立在对已知脆弱性的描述信息和人对脆弱性实例分析的基础上,通过人工标注或者建立的一些基于语法和语义的判定规则自动化地标注一些代码位置,这些位置的集合称为关键点集合 K。通过指定关键点集合 K,可以大幅缩减产生的脆弱性切片的数量,降低了分析中不关心或者与脆弱性无关的程序代码干扰,从而提高了相似性比对中的特征和脆弱性的相关性。

基于上述思想,本文针对以下两种脆弱性模型,提出了基

于头-关键点(集)-尾模型的切片方法,以实现刻画脆弱性上下文表达的目的。这两类脆弱性模型如下:

1)〈source,sink〉脆弱性模型。该模型代表风险可以被外部触发利用类的高危漏洞,其中 sink 代表风险触发。通常采用后向切片以聚焦分析 sink 点上的参数是否来自外部 source 且没有经过清洗;

2)〈sensitive,output〉脆弱性模型。该模型代表敏感信息被泄露类的高危漏洞,其中 sensitive 代表敏感数据。通常采用前向切片以聚焦分析 sensitive 点上的数据是否流向了 output 且没有经过认证。

在实际的脆弱性分析中,程序员可以向前跟踪变量和程序状态的变化,也可以从崩溃点出发向后分析可能的触发条件,按照需求实现双向的切片过程。

在脆弱性上下文描述的基础上,对脆弱性样本切片和待测样本的切片建模如下。

定义 1(基于头-关键点-尾的脆弱性样本切片)　CVE 脆弱性代码实例可用其 CVE 信息属性进行描述,如 CVE_Sample〈CVE_ID,CWE_Type,Project_Name,File_Name,Funtion_Name〉。脆弱性代码实例的切片集合是给定头尾节点且包含给定若干关键点的可能路径集合,其脆弱性语义是脆弱性从输入到触发的可能过程。切片的起点 e 代表头,s 代表尾,切片是以 e 开头以 s 为终点的集合。切片形如〈$Le,L_1,L_2,\cdots,L_i,\cdots,L_n,O_1,O_2,\cdots,O_m,Ls$〉,$Le$,$Ls$ 和 L_i 分别代表 e,s 和关键点 ki 所在的语句,〈$e,s,k_1,k_2,\cdots,k_n$〉为该脆弱性实例的切片准则。

按照定义,脆弱性样本实例切片算法如算法 1 所示。

算法 1　脆弱性样本实例切片算法

输入:脆弱性样本实例,头、尾和关键点集合

输出:脆弱性上下文切片(集)

1. 以头、尾点为切片准则切片,得到头到尾的候选切片集;
2. 对每一条切片,按照匹配关键点的程度进行打分;
3. 找出最匹配的切片,输出。

不失方法的一般性,一个已知脆弱性的样本选择输出一个脆弱性切片,必要时可以依赖人工的知识和经验进行辅助和调整。

定义 2(基于脆弱性切片的待测代码切片)　在基于切片相似度比对的脆弱性发现中,已知给定脆弱性实例 CVE_Sample 的切片准则中的起点和终点分别为 e 和 s。对被测样本 Sample 根据〈e,s〉切片准则进行切片,每一个获得的切片形如〈$Ne,N_1,N_2,\cdots,N_i,\cdots,N_p,Ns$〉,$Ne$,$Ns$ 和 N_i 分别代表 e 和 s 所在语句和途径的所有语句,即切片是以 e 为开头以 s 为终点的语句集合。切片算法的含义是查找所有已起点 e 开始并以 s 为终点的切片集合 Sample_Slices。

按照定义,待测代码的切片算法如算法 2 所示。

算法 2　待测代码的切片算法

输入:待测代码实例,指定脆弱性样本切片

输出:待检测脆弱性切片集

1. 从指定脆弱性样本切片中取出该指定脆弱性样本切片的头和尾,在被测代码实例上进行标记;
2. 以头、尾点为准则进行切片,得到头到尾的候选切片集;
3. 输出候选切片集合。

不失方法的一般性,一个被测代码为了与一个指定样本切片进行比较,会生成 0 到多个候选待比较的脆弱性切片。

3.2.2　基于规则的切片代码特征抽取

基于上述切片算法得到的切片是语句的集合,但语句并不适合作为相似性匹配的基本单位。例如,简单对已知脆弱性样本的脆弱性上下文中的某变量进行重命名,就会导致相关语句的文本变化,但是脆弱性依然是存在的。为此,在基于切片相似度匹配的脆弱性发现框架中引入了切片代码的特征抽取,以提升方法的准确性。

本文使用 3 条规则来实现代码中的自动化特征抽取:

1)将与切片时指定的关键点定义(或规则)匹配的部分作为特征;

2)将切片中的变量类型、API 调用名称、API 参数类型作为特征;

3)将切片中的条件语句包含的谓词判断作为特征。

其中,规则 1)是用户指定的兴趣点,与脆弱性语义相关;规则 2)以语法分析为基础,体现了脆弱性代码中相对比较固定的语法特征和模式;规则 3)特别选择了条件语句和谓词,考虑了条件判断对脆弱性发生与否可能相关,也是〈source-sink〉模型和〈sensitive-output〉模型中基于流分析的焦点,与脆弱性语义密切相关。

按照规则,特征提取算法如算法 3 所示。

算法 3　特征提取算法

输入:脆弱性样本实例的切片,头、尾和关键点集合;

输出:该脆弱性切片的实例化特征序列

1. 初始化空的特征序列;
2. 对切片中的每一个语句:
   2.1. 对语句进行抽象语法解析;
   2.2. 对解析获得的叶节点序列,依次判断节点是否满足特征规则:
      2.2.1. 满足,则将该叶节点内容作为特征加入特征序列;
      2.2.2. 不满足,返回步骤 2.2
3. 输出特征序列。

基于上述特征提取规则,对于给定的脆弱性样本切片,得到了该脆弱性样本切片的实例化特征序列 $feature\_list\langle f\rangle$,按照生成规则该序列不可能为空。

3.2.3　基于特征嵌入的向量化

在特征抽取以后,基于特征做切片的向量化表示是便于计算机进行基于相似性匹配的一个数学步骤。在基于切片相似度匹配的脆弱性发现框架中,由于脆弱性代码样本切片和被测代码样本切片是一对一进行相似度匹配的,可以基于词频统计技术对结构化特征进行嵌入化,实现切片的向量化表示。切片可以表示为特征集合向向量空间的映射,公式如下:

$$\varphi(feature\_list):\alpha\rightarrow\beta$$

其中,α 代表一个语句集合组成切片所代表的空间,β 代表经过 φ 规则映射得到的向量化空间。映射规则为:

$$\varphi(s,f)=\begin{cases}1, & \text{if } s.hasFeature(f)\\ 0, & \text{otherwise}\end{cases}$$

其中，f 属于实例化特征序列 $feature\_list(f)$，对于切片中的每一条语句应用该映射规则，判断其是否满足特征 f，以生成向量化结果。

3.2.4 基于向量空间的相似性匹配算法

基于上述向量化方法，对于给定脆弱性样本切片和从中提取的特征序列，所有被测样本切片按照特征序列进行向量化，完成了相似性匹配的数据准备工作。在此基础上，本文应用了基于向量空间的相似性度量来计算被检测样本切片和脆弱性样本切片的相似度，越高的相似度证明被检测样本存在类似脆弱性的可能性越高，以被检测切片为目标进行人工复检以确认脆弱性。

考虑到不同特征对脆弱性发现的指示意义不同，本文设计了加权的相似度匹配方法，为 3 个规则提取的特征设置了不同权值，公式如下：

$$Similarity=\sum_{rule=1}^{3} Weight_{rule} * Similarity_{rule}$$

在计算相似度时，常用的向量间距离计算方法包括欧几里得距离、曼哈顿距离、余弦相似度、皮尔森相关系数等。在基于切片相似度匹配的脆弱性发现框架下，本文对上述算法进行了应用，从脆弱性发现的准确度和效率的角度，选择了余弦相似度算法。限于篇幅不对其进行展开说明。

4 实现和实验验证

4.1 原型系统搭建

本文的原型系统是在 srcSlice 和 Joern 两个开源工具的基础上集成实现的。其中，srcSlice 提供基于变量的高效大规模前向切片能力，Joern 是 Yamaguchi 提出的基于模式的脆弱性分析的原型系统[1)-2)]。本文的原型实现了基于切片相似度匹配的脆弱性发现，改进效果是通过实验与 Joern 进行对比得到的。

在原型系统实现中，一些技术的实现细节如下。

1)脆弱性代码的切片准则和特征提取。在基于头-关键点(集)-尾模型的切片中，头、尾、关键点的选择对脆弱性切片具有重要作用。切片头和尾是关于该脆弱性的强语义特征点，具有指示性和筛选作用。本文分析了 NIST 的 SAMATE 项目的 SARD 子项目提供的 Juliet 的 C/C++测试集[33]，从例子的已有语义注释信息中提取了近 200 个不同 CWE 类型的近千个 API 库函数名，作为了头、尾和关键点的特征集合。

2)脆弱性样本切片。考虑到通常对脆弱性的利用是通过控制流实现的，因此原型采用了静态控制流切片。基于 Joern 生成的控制流结果，原型系统实现了一个静态控制流切片。脆弱性样本的切片过程分为 3 个步骤：首先，遍历获取所有从头到尾的控制流路径，遇到循环时则展开一次并标记其为环即可；其次，按照路径上包含关键点的数量对路径进行排序；最后，对排序路径进行人工分析，确认脆弱性路径。

3)被测样本的快速切片。对于被测代码的切片，通过遍历控制流图获取所有从头到尾的控制流路径即可得到。为了降低切片过程的时空开销，在切片过程中首先使用了词法层的特征筛查，不具备指定脆弱性样本关键特征的函数可以被快速过滤掉，省略了这些函数的程序分析开销。其具体流程如图 2 所示。

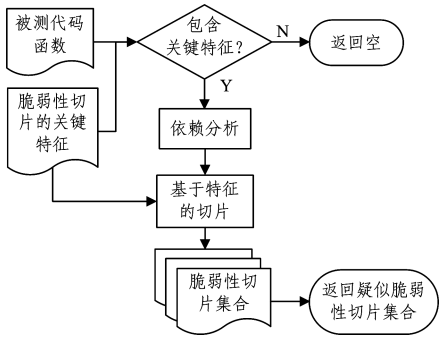

图 2 基于特征的快速切片生成流程

Fig. 2 Work flow of feature based program slicing

4)相似度匹配。原型实现了基于特征嵌入的向量化，通过 Cos 相似度算法计算和已知脆弱性函数的相似度来给出存在脆弱性可能的排名。

综上，基于所提框架的脆弱性匹配的流程中，脆弱性切片起到模板作用，被测样本以脆弱性切片的特征集为切片准则进行切片。当脆弱性代码切片和特征提取完成以后，按照所提框架进行切片，切片向量化后进行快速匹配。本文中，相似性匹配公式中的权值是人工指定的，考虑到与脆弱性语义的相关程度，权值参数分别设为 0.5，0.1 和 0.4。

4.2 实验验证

为了说明所提方法的有效性，本文选择了与 Joern 同样的 CVE 脆弱性样本和待检测对象。脆弱性样本选择了 Pidgin 的 CVE-2006-3459、LibTiff 的 CVE-2011-4601、FFmpeg 的 CVE-2010-3429 对应的脆弱性代码。待测对象包括 Pidgin 2.10.0 版本、LibTiff3.8.1 版本和 FFmpeg0.6 版本。在对比评估方法上，对比原型系统和 Joern 发现脆弱性的开销和准确度，以评估方法的有效性。

4.2.1 代码空间对比分析

如图 1 所示，部署在前端的切片用以降低被分析代码空间，以提升分析效率。与 Joern 把被测代码的所有函数都送入系统中进行符号向量化并进行相似性比对相比，本文方法只分析那些有可能存在类似脆弱性的函数中的高度存疑切片。被测代码在经过本文系统切片前端以后的代码比例在 Pidgin，LibTiff，FFmpeg 中分别占比 1.5%，2.8%和 5.9%，平均为 3.4%，大大收缩了相似性比对的代码空间，验证了所提方法的有效性。

1) http://ediss.uni-goettingen.de/handle/11858/00-1735-0000-0023-9682-0.

2) https://github.com/octopus-platform/joern.

4.2.2　分析准确度对比

为了基于相似度判定代码是否是脆弱性,基于相似度的度量方法需要给定相似度阈值或者指定 TOP-N 或者被测函数的固定比例来选取相似度高的函数代码集合送给人工分析确认。Top-N 命中方法是按照从大到小选择 N 个脆弱性切片来判定脆弱性的方法。在这种方法中,不同的 N 可以计算得到不同的漏报率和误报率。在 CVE-2010-3429 的实验中,被测对象 FFmpeg0.6 版本有 4 个同类型脆弱性(含脆弱性样本),使用本文方法和 Joern 的对比结果如表 1 所列,可以看出在 N 较小的情况下,本文方法的误报率和漏报率均优于 Joern 方法。证明了采用本文框架在较少的相似性筛选集合中即可取得较好的准确性,验证了方法的有效性。

表 1　实验结果的准确性对比

Table 1　Comparison of accuracy of experiment results

(单位:%)

Top-N 的阈值 N	本方法		Joern 方法	
	误报率	漏报率	误报率	漏报率
5	0.6	0.4	0.4	0.2
10	0.7	0.1	0.6	0
15	0.73	0	0.73	0
30	0.87	0	0.87	0

为了解释引入切片带来了脆弱性发现准确度的提升,需要从原理上简单对比本方法和 Joern 方法。两者相似性比对的特征对象序列选取方法是不同的:Joern 方法是直接提取了脆弱性样本函数的参数类型、变量类型和 API 调用序列作为特征;而本方法则提取了反映脆弱性上下文的切片中的 API 调用、谓词判断、类型信息等作为特征。因此,本方法借助切片技术并结合词法、语法和语义等多层次分析注释提升了脆弱性样本实例表征的准确度,进而提升了脆弱性发现精度。本方法的缺点是引入了切片带来了额外的开销,但是考虑到发现准确度的提升能大大节约人工对代码脆弱性审查的时间,因此我们认为这种开销是值得的。

结束语　本文为了提升基于相似度匹配的脆弱性发现方法的准确性和分析效率,提出了基于程序切片相似度匹配的脆弱性发现框架,并详细讨论了框架下的切片、特征抽取、向量化和相似性计算等方法,通过原型系统的对比实验,验证了所提框架和方法的有效性。

目前在框架实现和实验时,基于头-关键点-尾的脆弱性上下文切片方法的切片准则还需要依赖于语义标注,后续希望通过语义标注的半监督学习加以改进。另外,在基于加权的相似性算法中,权值参数的选择采用了实验和人工指定的方法,后续会考虑加权公式的合理性,同时引入机器学习方法来进行优化。

参 考 文 献

[1] WU S Z,GUO T,DONG G W,et al. Software vulnerability analyses:A road map [J]. Journal of Tsinghua University (Science and Technology),2012,52(10):1309-1319. (in Chinese)
吴世忠,郭涛,董国伟,等. 软件漏洞分析技术进展[J]. 清华大学学报(自然科学版),2012,52(10):1309-1319.

[2] WIJAYASEKARA D,MANIC M,WRIGHT J,et al. Mining bug databases for unidentified software vulnerabilities[C]// 2012 5th International Conference on Human System Interactions (HSI). IEEE,2012.

[3] NEUHAUS S,ZIMMERMANN T,HOLLER C,et al. Predicting vulnerable software components[C]// Proceedings of the 14th ACM Conference on Computer and Communications Security. ACM,2007.

[4] WEISER M. Program slicing[C]// Proceedings of the 5th International Conference on Software Engineering. IEEE Press,1981:439-449.

[5] KOREL B,LASKI J. Dynamic program slicing[J]. Information processing letters,1988,29(3):155-163.

[6] GALLAGHER K B,LYLE J R. Using program slicing in software maintenance[J]. IEEE Transactions on Software Engineering,1991,17(8):751-761.

[7] HIERONS R,HARMAN M,DANICIC S. Using program slicing to assist in the detection of equivalent mutants[J]. Software Testing Verification and Reliability,1999,9(4):233-262.

[8] VENKATESH G A. The semantic approach to program slicing [C]// ACM SIGPLAN Notices. ACM,1991,26(6):107-119.

[9] FIELD J,RAMALINGAM G,TIP F. Parametric program slicing[C]// Proceedings of the 22nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages. ACM,1995:379-392.

[10] CANFORA G,CIMITILE A,DE LUCIA A. Conditioned program slicing[J]. Information and Software Technology,1998,40(11-12):595-607.

[11] AGRAWAL H,DEMILLO R A,SPAFFORD E H. Debugging with dynamic slicing and backtracking[J]. Software:Practice and Experience,1993,23(6):589-616.

[12] OTTENSTEIN K J,OTTENSTEIN L M. The program dependence graph in a software development environment[C]// ACM Sigplan Notices. ACM,1984,19(5):177-184.

[13] DANICIC S,HARMAN M,SIVAGURUNATHAN Y. A parallel algorithm for static program slicing[J]. Information Processing Letters,1995,56(6):307-313.

[14] BERGERETTI J F,CARRÉ B A. Information-flow and data-flow analysis of while-programs[J]. ACM Transactions on Programming Languages and Systems (TOPLAS),1985,7(1):37-61.

[15] ZHANG X J. Program slicing technology research and slicing scheme design [D]. Chengdu:University of Electronic Science and Technology,2017. (in Chinese)
张新杰. 程序切片技术研究及切片方案设计[D]. 成都:电子科技大学,2017.

[16] XU B,QIAN J,ZHANG X,et al. A brief survey of program slicing[J]. ACM SIGSOFT Software Engineering Notes,2005,30(2):1-36.

[17] TIP F. A survey of program slicing techniques[M]. Centrum

voor Wiskunde en Informatica,1994.

[18] 李必信.程序切片技术及其应用[M].北京:科学出版社,2006.

[19] Wisconsin Program-Slicing Project. The Wisconsin Program-Slicing Tool, Version 1. 0. 1[OL]. http://www. cs. wisc. edu/wpis/slicing_tool.

[20] BALAKRISHNAN G,GRUIAN R,REPS T,et al. CodeSurfer/x86-A platform for analyzing x86 executables[C]// International Conference on Compiler Construction. Springer,Berlin,Heidelberg,2005:250-254.

[21] CUOQ P,KIRCHNER F,KOSMATOV N,et al. Framac[C]// International Conference on Software Engineering and Formal Methods. Springer,Berlin,Heidelberg,2012:233-247.

[22] SCHWARTZ-NARBONNE D,OH C,SCHÄF M,et al. VERMEER:A tool for tracing and explaining faulty C programs[C]// 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE). IEEE,2015:737-740.

[23] ALOMARI H W,COLLARD M L,MALETIC J I,et al. srcSlice:very efficient and scalable forward static slicing[J]. Journal of Software:Evolution and Process,2014,26(11):931-961.

[24] NEWMAN C D,SAGE T,COLLARD M L,et al. srcSlice:a tool for efficient static forward slicing[C]// IEEE/ACM International Conference on Software Engineering Companion (ICSE-C). IEEE,2016:621-624.

[25] LI R Q,ZENG G B. Slice Abstract Extraction in Program Source Code and Its Application in Search[J]. Information Technology & Network Security,2018,37(3):122-125.(in Chinese)
李润青,曾国荪.程序源代码中的切片摘要提取及在搜索中的应用[J].信息技术与网络安全,2018,37(3):122-125.

[26] FENG Q,WANG M,ZHANG M,et al. Extracting conditional formulas for cross-platform bug search[C]// ASIACCS. 2017.

[27] YAMAGUCHI F,FELIX L,KONRAD R. Vulnerability extrapolation:Assisted discovery of vulnerabilities using machine learning[C]// Proceedings of the 5th USENIX Conference on Offensive Technologies. USENIX Association,2011.

[28] YAMAGUCHI F,MARKUS L,KONRAD R. Generalized vulnerability extrapolation using abstract syntax trees[C]// Proceedings of the 28th Annual Computer Security Applications Conference. ACM,2012.

[29] YAMAGUCHI F,et al. Chucky:exposing missing checks in source code for vulnerability discovery[C]// Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. ACM,2013.

[30] YAMAGUCHI F,MAIER A,GASCON H,et al. Automatic inference of search patterns for taint-style vulnerabilities[C]// 2015 IEEE Symposium on Security and Privacy (SP). IEEE,2015:797-812.

[31] XU X,LIU C,FENG Q,et al. Neural Network-based Graph Embedding for Cross-Platform Binary Code Similarity Detection[C]// Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM,2017:363-376.

[32] GAN S T,QIN X J,CHEN Z N,et al. Software vulnerability code clone detection method based on characteristic metrics[J]. Journal of Software,2015,26(2):348-363 (in Chinese).
甘水滔,秦晓军,陈左宁,等.一种基于特征矩阵的软件脆弱性代码克隆检测方法[J].软件学报,2015,26(2):348-363.

[33] https://samate. nist. gov/SRD/testsuites/juliet/Juliet_Test_Suite_v1. 3_for_C_Cpp. zip.