开源软件安全检测关键技术

靳鑫，杜林
（安徽继远检验检测技术有限公司，安徽 合肥 231200）

摘　要：随着计算机技术的不断发展，开源软件在软件研发方面得到了十分广泛的使用。然而，开源软件中存在大量的安全隐患，急需通过检测手段发现并控制这一类风险。为了有效地实现开源软件的安全检测，需要重点研究两个方面的关键技术。在已有技术成果的基础上，对开源软件安全检测技术的关键技术进行了论述，主要包括基于机器学习的开源代码片段分析技术和基于二进制代码分析的开源软件安全检测技术。针对这两种关键技术方案的研究，为开源软件安全检测系统的实现打好了坚实的基础。

关键词：开源软件；安全检测；机器学习；代码克隆；二进制分析

1　引言

随着互联网技术的兴起，开源软件已逐步成为主流。近些年，著名开源项目的范围涵盖移动操作系统 Android、数据库软件 MySQL、大数据计算平台 Spark、深度学习框架 TensorFlow 等。在当代的软件开发过程中，工程师们越来越多地选择使用开源软件来减少重复的工作。然而开源代码滥用会带来大量代码缺陷和安全漏洞，导致代码质量和性能大幅下降。目前我国对开源项目的风险识别意识也已初步形成，但在实际管控体系、检测手段等方面有待进一步提高，且尚未形成成熟的开源安全类产品。

国外虽然已经开展了开源软件安全技术的研究，但国外的相关技术方案也仅支持在有源码情况下的检测服务，由于开源软件自身的应用特点，大量的开源组件仍然属于“部分开源”状态，针对这种组件的检测未进行相关研究。开源软件的应用面很广，而在软件开发和运行过程中，可能随时引入新的“开源成分”，很多研发环境都是内网，很难实现特征库的及时更新，因此很难保证较高的检测成功率。结合当前开源软件安全研究的不足，需要重点研究两个方面：基于机器学习的开源代码片段分析技术、基于二进制代码分析的开源软件安全检测技术，并提出相关技术方案。通过对开源软件安全检测技术进行研究，不但给解决当前开源软件安全管控存在的问题指出了方向，而且为后续的检验检测工具的研发也提供了思路，针对电力信息系统外的信息系统也有一定的应用价值，以此为基础可形成针对开源软件安全检测的产品，未来可形成可观收益。

本文研究开源代码片段分析技术，通过提取开源代码片段特征，运用机器学习技术实现对开源代码的快速识别，同时结合恶意代码分析技术实现代码的安全分析；基于二进制代码分析的开源软件安全检测技术，通过提取二进制代码多维度特征，结合分布式安全检测技术，实现对二进制开源代码的安全检测。结合这两个技术方案，形成开源软件安全检测技术方案，为进一步的系统设计提供技术支撑。技术路线如图 1 所示。

2　基于机器学习技术的开源代码片段

为了实现基于机器学习技术的开源代码片段分析技术，首先要对开源代码进行片段特征提取，建立开源代码片段特征库，并基于特征库建立分布式索引库。通过机器学习算法对源代码成分分析模块进行训练，提高源代码成分分析模块的速度和准确度。本技术的技术方案如图 2 所示。

基于机器学习技术的开源代码片段分析技术主要由词法分析器、代码片段特征提取模块、Spark 计算平台、机器学习引擎、开源代码片段特征库、开源代码分布式索引库和源代码成分分析模块组成。源代码经过词法分析器处理后，形成预处理文件，然后利用代码片段特征提取模块对原始代码文件和预处理文件进行代码片段特征提取，建立开源代码片段特征库。

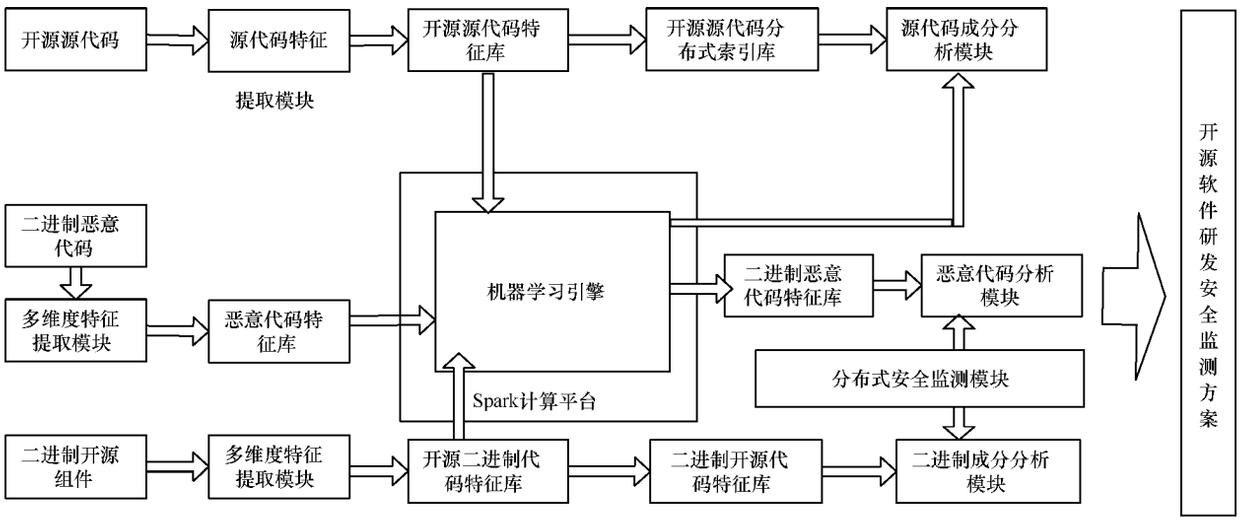

图 1　开源软件安全检测技术路线

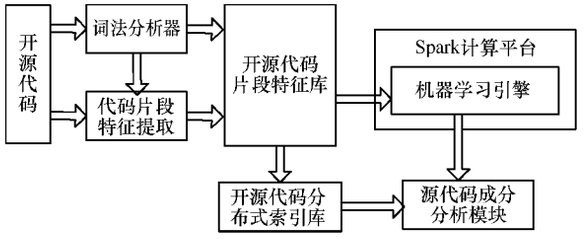

图 2　基于机器学习技术的开源代码片段分析技术的技术方案

基于开源代码片段特征库建立开源代码分布式索引库，最后利用源代码成分分析模块进行成分分析。主要技术模块的技术方案如下。

（1）利用词法分析器的预处理

词法分析器对代码的无关部分进行去噪操作，过滤注释、头文件、多余的无意义字符，获得标准化的词法单元（token）之后，再进一步获取固定长度的索引单元（chunk），实现源代码的预处理。

本项目中的词法分析器主要过滤的词法单元有以下 5 种。

- 注释：包括一行、多行注释以及 doc 注释。
- 预处理命令：如 import、include、package、define 等。
- 可见度修饰符：如 static、public、private 等。
- 命名空间限定符：如 global 等。
- this 调用。

另外，词法分析器还对源代码中所有的标识符、类型、常量进行统一编号，处理规则如下。

- 标识符采用 id+编号来替换，编号由 0 递增，如果前面已经出现过，则采用第一次的编号。
- 字符串统一替换为“”空串。
- 数据类型为整形变换为 0。
- 数据类型为浮点形变换为 00.0。
- 布尔值变换为 true。

如图 3 所示，词法分析器针对每个输入的词法单元，按照上述 5 条规则以及 5 个 token 过滤标准，得到最终的标准化 token 集合。其中，函数名 getTest 也会作为标识符，被转化为 id+编号的形式；而小括号都会被忽略。

（2）代码片段特征提取

源代码经过词法分析器处理后，形成新的预处理代码。通过对原始代码和预处理代码进行特征提取，即可建立代码片段特征库。以预处理后代码为例，代码片段特征的提取步骤如下：

步骤 1　预处理代码中的 token，按照固定数目进行结合，形成一个代码片段；

步骤 2　对该代码片段进行 Hash 值计算，即得到了该代码片段的特征；

步骤 3　偏移固定的 token 数目，安全固定书面进行结合，形成一个新的代码片段，并计算新片段的 Hash 值；

步骤 4　重复步骤 2 和步骤 3，直至文件结束。

（3）建立分布式索引库

索引单元里面不含源代码的文本信息，含有代码片段的特征值、token 数量等信息。其结构见表 1。

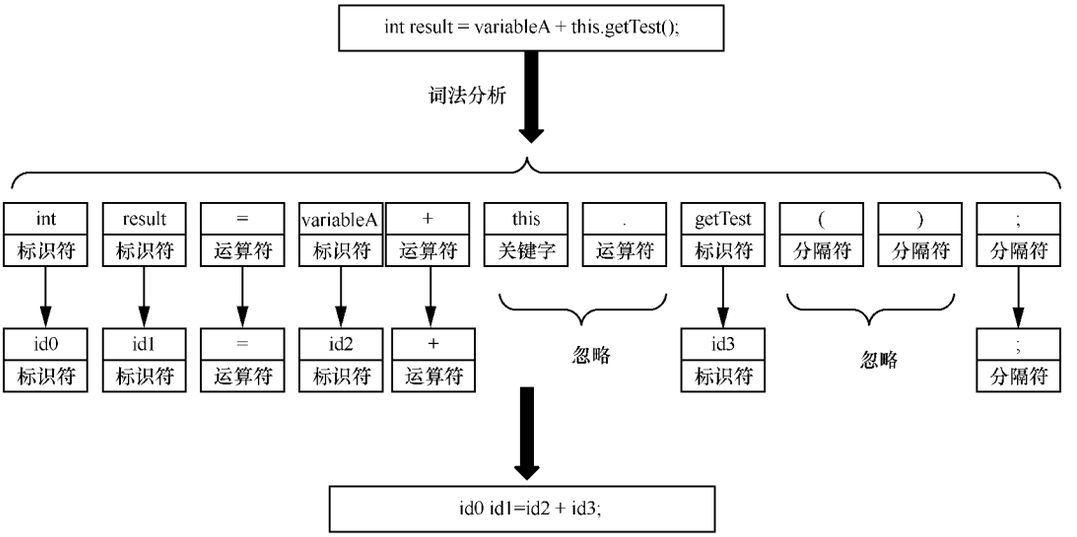

图 3　词法分析器预处理过程

表 1　基本索引单元

属性	说明
chunkHash	该段代码文本的 MD5 Hash 值
elementUnits	索引单元中包含的词法 token 数目
firstRawLineNumber	对应源文件的开始行号
firstUnitIndex	所含 token 中，第一个 token 的编号
lastRawLineNumber	对应源代码文件的结束行号
originId	该索引胆原所在的文件唯一路径
rawEndOffset	对源代码文件中结束行中的偏移量
rawStartOffset	对应源代码文件中开始行中的偏移量

索引库采用分布式数据库 HBase，因为 HBase 的数据是按列存储的。当存储索引数据 Chunk 时，特定列由 originId 加上#firstUnitIndex 唯一标识。例如，elementUnits 列是散列值为 hashA 的块，存储在具有 hashA 的行中，其 elementUnits 列的列键是 elementUnits：originId＃firstUnitIndex，存储的值或 elementUnits 的值。图 4 显示了创建索引库的完整过程。

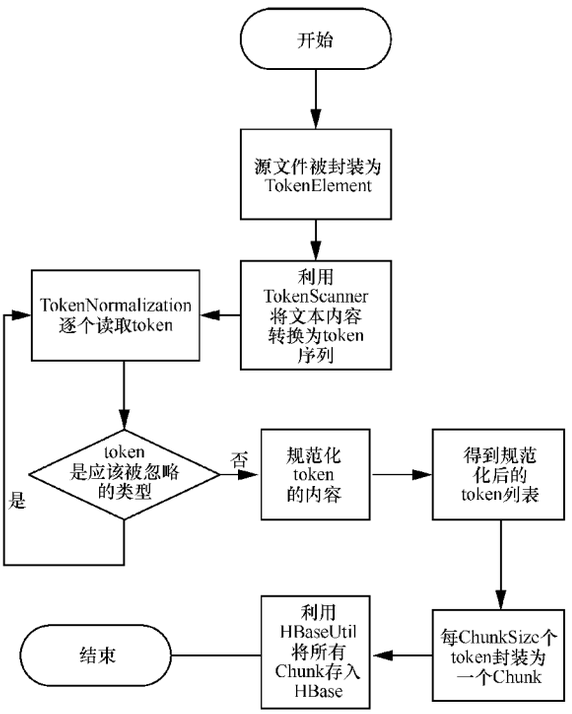

图 4　建立索引库完整流程

（4）建立基于 Spark 的机器学习应用框架

本项目建立了基于 Spark 的机器学习应用框架，通过对框架接口的调用，实现了高效的代码片段分析技术。该机器学习应用框架的原理如下。

- 相关机器学习任务是通过调用接口进行访问实现的。
- 根据获得的参数和接口情况，向算法优化器提交任务，实现算法优化。同时将从代码特征库中加载到特征数据，并将相关数据提交到算法优化器，由算法优化器进行处理。
- 机器学习算法将指导算法优化器完成算法的预处理优化。

- 数据处理完成后，算法优化器开始优化工作，主要的工作内容包括：数据提取、向框架提交优化子任务、将子任务信息注册添加到信息统计器中。
- 框架将各个任务分配至不同的计算节点，开始优化工作。
- 最后优化器将所有节点完成的结果进行有效整合，最终得到的机器学习模型。

基于 Spark 的机器学习应用框架如图 5 所示。

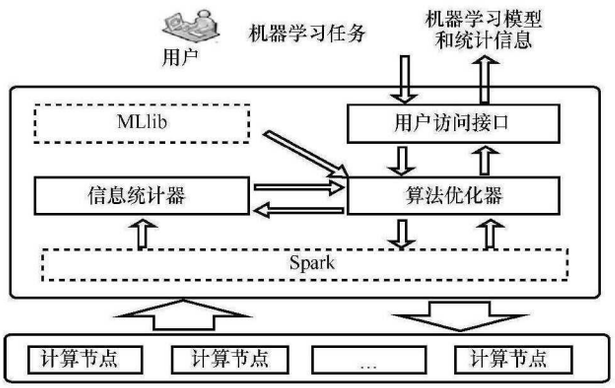

图 5　基于 Spark 的机器学习应用框架

（5）利用代码片段特征实现开源软件成分分析

通过建立的分布式索引和机器学习模型，实现基于代码特征的开源软件成分分析模块。主要过程如下：

- 构建待检测源代码索引，利用索引到 HBase 中进行检索；
- 利用检索生成的二维邻接表结构，按照待检测的代码的索引序列作为行头，每行存储与行头元素相同的散列值作为索引数据，每行的元素按照文件路径进行排序，行头则根据待检测文件中的索引号排序；
- 通过索引与开源软件分布式索引库，分析识别开源软件成分；
- 结合恶意代码特征库和分析技术，实现开源成分安全的识别。

3　基于二进制分析的开源软件安全检测技术

为了实现对二进制代码的安全检测，及时发现二进制开源代码中的漏洞和恶意代码成分，需要先对二进制代码进行多维度特征提取，建立二进制开源软件特征库和恶意代码特征库；然后利用机器学习技术，实现对二进制开源成分和恶意代码的分析识别；在此基础上，进一步设计实现分布式安全检测技术，最终实现基于二进制分析的开源软件安全检测技术。技术方案如图 6 所示。

3.1　提取二进制代码多维度特征，建立二进制代码特征库

针对二进制开源代码，本项目主要提取二进制代码的静态特征，包括文件属性的属性、文件结构层的二进制结构特征、字节层的字节序列特征、指令层的指令序列特征、语义层的关键 API（application programming interface，应用程序编程接口）调用序列特征以及特征提取。过程如图 7 所示。

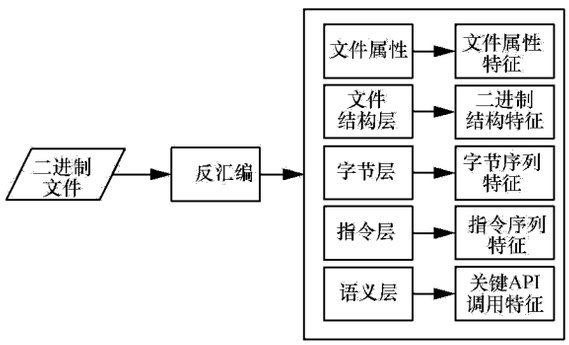

图 7　二进制多维度特征提取

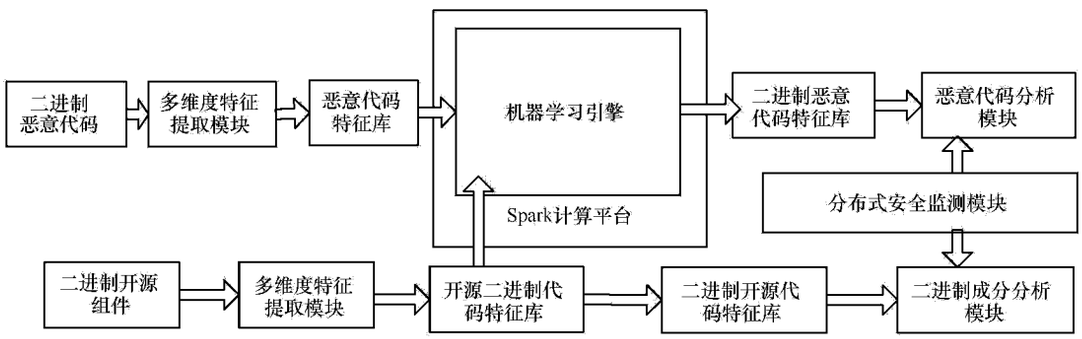

图 6　基于二进制分析的开源软件安全检测技术方案

（1）提取文件属性特征

二进制文件的文本可以用来在开源或二进制组件中进行初步的识别，直接识别二进制组件的版本，可以根据其属性涉及的特性来指以文件名、容量、基本属性（如以安全风险分析初步特点）为代表的散列值，结合以下特点，进一步对二进制代码进行多维特征分析、识别安全风险，如恶意代码。

（2）提取文件结构层的特征

文件结构层的结构与文件的静态结构信息有关。恶意代码通常会修改二进制文件的结构，以实现其重定位、文件搜索、感染和破坏等功能，目的在于查杀结构，并防止杀毒软件将修改入口点转为非恶意行为、标准件、修改件名称、修改导入表等。这些结构特征与普通文件不同，因此二进制文件的结构可以作为二进制文件的特征之一。

（3）提取字节层特征

字节层特征提取方便，但其最大的缺点是容易造成代码混淆，字节层特征仍然可以代表二进制编码特征的维度。字节层的特点是首先将 Hexview 等相应工具转换为十六进制序列，然后用 *n*-gram 窗口获取特征，并使用 *n*-gram 窗口滑动特征提取。

（4）提取指令层特征

在获得操作码序列之后，使用 *n* 元组来获得期望的特征集。具体步骤如下：首先对样本进行分解，然后从反汇编结果中提取常用指令，形成初始特征集。最后，*n*-gram 滑动窗口技术得到最终的期望特征集，*n* 取 2 或 3。

（5）提取语义层特征

通常，语义信息可以由 API 函数调用信息表示。一个或多个 API 函数序列表示相关的操作或行为。因此，就字节或指令序列而言，语义层的特点具有良好的对抗性。语义特征的提取如下：

- 建立一个关键的系统调用库，它包含了公共关键系统调用 API 函数；
- 根据反汇编的结果构建关键系统调用关系流程图；
- 使用深度优先遍历算法遍历调用流程图，得到关键系统调用的调用顺序。

利用 *n*-gram 窗口滑动调用序列获取所需特征，如图 8 所示。

通过对二进制开源软件和二进制恶意代码进行多维度特征提取，建立二进制开源软件特征库和二进制恶意代码特征库。

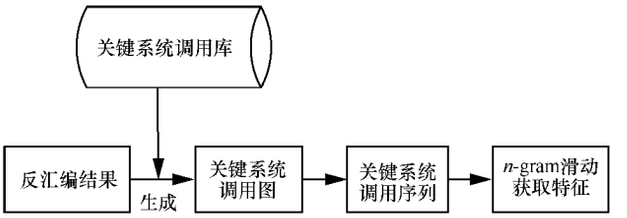

图 8　语义特征提取方法

3.2　识别二进制文件中的开源成分

二进制文件中的开源代码成分的识别基于二进制开源软件特征库，通过对被检测二进制文件进行多维度特征提取和匹配，实现二进制文件中的开源成分识别。本项目通过如下方案实现二进制文件中的开源成分识别。

（1）基于基本块代码序列的特征匹配方案

通过使用生物 DNA 序列相似度矩阵映射方法和评分矩阵法和思路，利用二维阵列点矩阵进行项目开发模拟，实现编译后的可执行二进制码序列相似性度量。对需要比较的二进制汇编代码序列提取汇编指令操作符，分别表示为：

$$\text{sequenceA}=\{A_1,A_2,A_3,\cdots,A_i,\cdots,A_{\text{lengthA}}\} \tag{1}$$

$$\text{sequenceB}=\{B_1,B_2,B_3,\cdots,B_i,\cdots,B_{\text{lengthB}}\} \tag{2}$$

滑动窗口值设为 H，打分规则为：

$$P\{A_i,B_i)=\begin{cases}1, A_i=B_i\\ 0, A_i\neq B_i\end{cases} \tag{3}$$

sequenceA 和 sequenceB 比对得出打分矩阵 ScoreMatrix，扫描 ScoreMatrix 得到打分矩阵非重叠平行斜线的最长组合的序列长度：

$$M[m]=\sum_{i+m=j}\text{ScoreMatrix}_{ij}, m\in[0,\text{lengthA}-1], \quad 0\leqslant i<\text{lengthA}, 0\leqslant j<\text{lengthB} \tag{4}$$

$$N[n]=\sum_{i=j+n}\text{ScoreMatrix}_{ij}, m\in[0,\text{lengthA}-1], \quad 0\leqslant i<\text{lengthA}, 0\leqslant j<\text{lengthB} \tag{5}$$

sequenceA 和 sequenceB 中出现的相同序列长度为 SC。该矩阵算法能检测出连续相同指令序列，并且能有效消除指令重排的影响，准确地得出两个指令序列的相似度。

（2）基于基本块子函数名特征的特征匹配方案

在二进制代码中，调用指令通常会泄漏一定的字符串，其中一些是由代码作者命名的，一些是系统提供的 API 函数的名称。通过编译链接在同一位置调用的系统

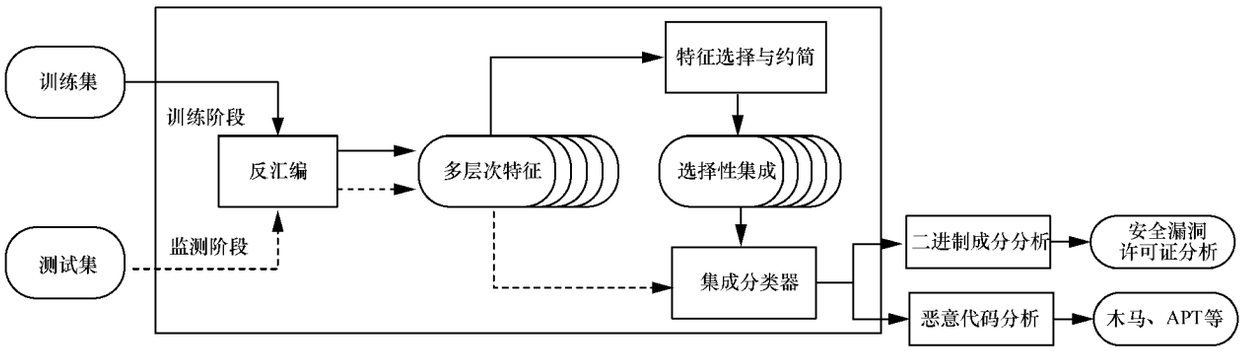

图 9　基于机器学习的二进制安全风险分析

函数是相同的。该项目从调用指令的操作数中提取有效的子功能名称作为另一个重要的匹配功能。这个算法的原理是计算两个字符串之间相同的两个字符串。每个步骤用于修改、添加或删除一个字符。 这个最小的步骤是编辑距离 ED。

一个基本块和基本块 B，分别扫描 A 和 B 基本块所有调用指令，提取出调用指令操作数是 A 的子程序名，过滤掉一个数为 8 的函数十六进制，得到 StrA 和 StrB 两个字符串，并且 $m>n$。

$$\mathrm{StrA}=(m,\mathrm{StrA}_1,\mathrm{StrA}_2,\cdots,\mathrm{StrA}_m) \tag{6}$$

$$\mathrm{StrB}=(n,\mathrm{StrB}_1,\mathrm{StrB}_2,\cdots,\mathrm{StrB}_n) \tag{7}$$

两个字符串集的相似度为：

$$F_{\mathrm{Similarity}}(\mathrm{StrA},\mathrm{StrB})=\frac{\sum_{j=1}^{n}\left(1-\frac{\mathrm{Min}(\mathrm{ED}(\mathrm{StrA}_i,\mathrm{StrB}_j))}{\mathrm{MaxLengthOf}(\mathrm{StrA}_i,\mathrm{StrB}_j)}\right)}{\mathrm{Min}(m,n)} \tag{8}$$

（3）基于二进制代码常量特征的特征匹配方案

代码中的常量可分为字符串常量和数字常量。二进制代码中的常量特征与硬件平台和编译器无关。因此，可以通过提取二进制代码中的常量特征实现与硬件平台和编译器无关的特征匹配方法。通过提取二进制开源软件中的常量特征，建立开源软件特征库。然后，使用基于倒排索引的特征匹配方法，实现二进制代码中的开源成分分析。

3.3　分析二进制文件安全风险

本项目通过如下手段二进制代码中的安全风险进行分析。

通过匹配二进制代码的多维度特征，分析得到被检测的二进制代码中的开源软件成分及版本，通过全球级开源软件信息知识库，得到该版本开源软件的安全漏洞风险情况，从而实现安全风险分析。

基于全球级开源软件风险特征库中的恶意代码特征，利用机器学习进行学习训练，得到二进制文件安全风险分析模型和规则库，通过该模型和规则库，实现对二进制文件的安全风险分析。

原理结构如图 9 所示。

3.4　设计分布式安全检测方案，实现开源软件安全检测

为了对二进制开源软件进行安全检测，本项目对开源软件安全检测进行任务分解，按照任务分解后的功能要求，设计多种功能角色的 Agent 部署在安全检测目标系统中。主要设计检测 Agent、数据传输 Agent、特征提取 Agent 和决策 Agent。检测 Agent 用于检测目标系统中被检测的二进制文件的状态。数据传输 Agent 用于对需传输的数据进行压缩、分包发送到指定目标。特征提取 Agent 用于提取二进制文件的部分简单特征，对于复杂特征的提取，需要通过数据传输 Agent 将二进制文件进行压缩分包传输到专门的计算集群中进行计算和分析。决策 Agent 根据其他 Agent 的状态和输出，对其他 Agent 的工作状态进行管理，同时具有对二进制文件的安全风险做出分析决策的能力。二进制文件安全检测 Agent 功能模型如图 10 所示。

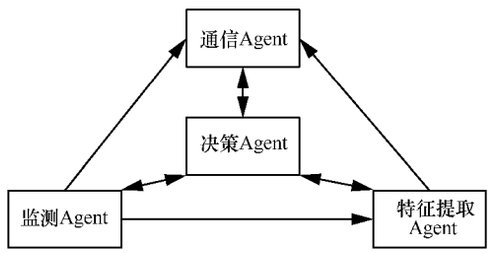

图 10　二进制文件安全检测 Agent 功能模型

检测 Agent 是二进制安全检测的数据来源，是 Agent 感知能力的基本体现。其主要结构如图 11 所示。

图 11　检测 Agent 结构

数据传输 Agent 主要用于在必要时将检测文件传输到指定目标中，如安全检测服务器集群。其主要结构如图 12 所示。

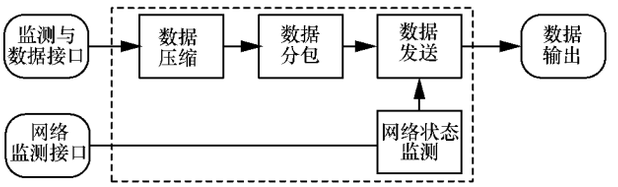

图 12　数据传输 Agent 结构

特征提取 Agent 主要对目标二进制代码进行多维度特征提取。由于不同的目标系统计算能力差异很大，为了简化部署在目标系统的特征提取 Agent 的设计，特征提取 Agent 只进行一些简单的特征提取，如文件属性特征等。其结构如图 13 所示。

图 13　特征提取 Agent 结构

部署于检测目标系统的决策 Agent 可以对检测 Agent、数据传输 Agent、特征提取 Agent 的工作状态进行管理，并根据各个 Agent 的状态和数据输出，对二进制文件的安全性做出决策。其结构如图 14 所示。

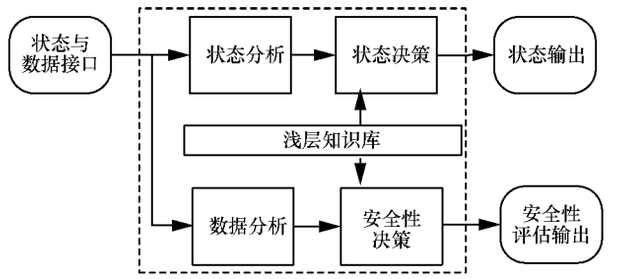

图 14　决策 Agent 结构

结合两种技术成果和基于 Multi-Agent 的二进制代码多维度特征的分布式数据采集与检测技术，在开源软件信息知识库和风险特征库的数据支持下，通过提取和匹配被检测的二进制代码的多维度特征和风险特征扫描，实现二进制代码中的开源软件成分识别和漏洞分析以及被检测二进制代码的其他安全风险情况，如是否被植入恶意代码、木马、APT（advanced persistent threat，高级持续性威胁）等安全风险，对出现的安全风险进行警告，并提供相应的修复建议。最终实现基于二进制代码分析的开源软件安全检测技术。

4　结束语

本文在已有技术成果的基础上对于开源软件安全检测技术的关键技术进行了论述，主要包括基于机器学习的开源代码片段分析技术和基于二进制代码分析的开源软件安全检测技术。从两种技术的关系上来说，开源代码片段分析技术为开源软件安全检测技术提供了有力支撑，保证了开源成分分析的有效性。开源代码片段分析技术主要是利用代码克隆分析技术以及机器学习分析引擎实现对代码片段成分的准确分析；基于二进制代码分析的开源软件安全检测技术则是有效利用了数据平台技术、恶意代码识别技术、二进制特征匹配以及代码片段分析的结果，实现了二进制代码和有源软件风险的识别。针对这两种关键技术方案的研究，为开源软件安全检测系统的实现打好了坚实的基础。

尽管通过本文的研究已经完成了既定的论证目标，但是依然存在着一些不足之处。首先因为本文研究的内容属于安全检测类的研究内容，所以涉及的技术内容很广，通过本文难以全面阐述清楚。其次，由于时间关系和技术研发本身所需的时间，本文涉及的技术方案在检测效果上可能还有缺陷，还需要进一步优化，具体的成果还需要通过大量的工作来实现。

参考文献：

[1] 金芝, 周明辉, 张宇霞. 开源软件与开源软件生态:现状与趋势[J]. 科技导报, 2016(14): 42-48.

[2] 林婵, 李俊杰, 饶飞, 等. 基于索引的分布式代码克隆检测[J]. 信息安全研究, 2016(3): 201-210.

[3] 舒翔. 基于索引和序列匹配的代码克隆检测技术研究[D]. 杭州: 杭州电子科技大学, 2015.

[4] 叶青青. 软件源代码中的代码克隆现象及其检测方法[J]. 计算机应用与软件, 2008(9): 147-149, 159.

[5] 黎文阳. 大数据处理模型 Apache Spark 研究[J]. 现代计算机(专业版), 2015(8): 55-60.

[6] 胡俊, 胡贤德, 程家兴. 基于 Spark 的大数据混合计算模型[J]. 计算机系统应用, 2015(4): 214-218.

[7] 温馨, 罗侃, 陈荣国. 基于 Shark/Spark 的分布式空间数据分析框架[J]. 地球信息科学学报, 2015(4): 401-407.

[8] 宋喆, 初广丽. 基于 Multi-Agent 的个性化信息检索模型结构体系[J]. 图书馆学研究, 2011(3): 62-66.

[9] 徐小龙, 程春玲, 熊婧夷. 基于 multi-agent 的云端计算融合模型的研究[J]. 通信学报, 2010(10): 203-211.
[10] 陈锦富, 赵小磊, 刘一松, 等. 基于数据挖掘的第三方构件安全性测试模型及其框架[J]. 计算机学报, 2017(1): 1-16.
[11] 陈良. 恶意代码检测中若干关键技术研究[D]. 扬州: 扬州大学, 2012.
[12] 唐振坤. 基于 Spark 的机器学习平台设计与实现[D]. 厦门: 厦门大学, 2014.
[13] 敬锐. 恶意代码检测系统的研究与实现[D]. 成都: 电子科技大学, 2010.
[14] 孙科. 基于 Spark 的机器学习应用框架研究与实现[D]. 上海: 上海交通大学, 2015.
[15] 邱景. 面向软件安全的二进制代码逆向分析关键技术研究[D]. 哈尔滨: 哈尔滨工业大学, 2015.
[16] 马金鑫, 李舟军, 忽朝俭, 等. 一种重构二进制代码中类型抽象的方法[J]. 计算机研究与发展, 2013(11): 2418-2428.
[17] 李舟军, 张俊贤, 廖湘科, 等. 软件安全漏洞检测技术[J]. 计算机学报, 2015(4): 717-732.
[18] 李朝君. 二进制代码安全性分析[D]. 合肥: 中国科学技术大学, 2010.
[19] 颜颖, 方勇, 刘亮, 等. 基于基本块指纹的二进制代码同源性分析[J]. 网络安全技术与应用, 2017(3): 67-69.