研 究 生 毕 业 论 文

（申请硕士专业学位）

论 文 题 目 结合污点分析与机器学习的隐私数据泄露检测系统设计与实现

作 者 姓 名 _____

专 业 名 称 _____

研 究 方 向 _____

指 导 教 师 _____

2021 年 4 月 29 日

学　　号：

论文答辩日期：**2021 年 5 月 x 日**

指 导 教 师：　　**（签字）**

Design and Implementation of Privacy Data Leakage Detection System Based on Taint Analysis and Machine Learning

By

(Author Name)

Supervised by

(Supervisor's position)

A Thesis
Submitted to the XXX Department
and the Graduate School
of XXX University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Engineering

May 2021

研究生毕业论文中文摘要首页用纸

毕业论文题目：结合污点分析与机器学习的隐私数据泄露检测系统设计与实现

专业 级硕士生姓名：

指导教师（姓名、职称）：

摘　要

在大数据时代，用户的个人隐私数据成为了一种有着巨大经济价值的商品。个人隐私数据的收集愈加广泛，同时也使得数据安全问题更加严重，各国组织纷纷出台相应的个人隐私保护政策。为了不违反隐私法规，企业需要进行数据合规的工作。数据合规最首要的任务便是对软件程序中隐私数据的定位以及泄露风险分析，使得自动化的隐私数据泄漏检测技术成为企业组织当前关心的重点。当前隐私泄露风险检测技术的实现多重点关注于泄露位置的检测，但一般不聚焦于识别隐私数据是否已经经过匿名处理，所以部分已经经过匿名化处理的脱敏数据本应不算做数据泄露，但因为其确实经过了危险操作而被判定存在风险从而造成误报，导致增加了数据合规的工作量，甚至增加了维护的成本。

本文旨在提供准确完整的隐私数据分布结果，提高匿名化隐私数据的检测能力，降低传统方法对隐私数据泄露风险的误报率。本文利用污点分析、程序图表示技术和机器学习分类模型为开发或安全工程师提供更准确的程序隐私数据泄露风险检测服务。本文由此提出了一种隐私数据路径定位的方法 Private Data Trace Positioning（PDTP），该方法首先基于污点分析技术得到外部输入的用户隐私相关数据变量在项目程序中的传播路径；其次通过隐私数据变量传播路径中的函数调用提取出可能为加密函数的候选函数集合；随后针对候选函数，利用程序表示学习技术提取函数特征并结合图核函数生成函数的特征向量；其后利用机器学习分类模型对候选函数进行分类识别，以提升对隐私数据泄露风险评估的准确性，生成最终的扫描检测结果。

本文针对所提方法设计并实现了一个面向 Java Web 项目的隐私数据泄露风险检测系统。本文系统主要分为可视化模块、业务处理模块、候选加密函数生成模块、程序特征提取模块以及分类识别，并使用 SpringBoot、Django、Docker 等技术与架构完成系统的实现。实验表明，本系统在牺牲较少召回率的情况下，其准确率可以达到 88% 以上，与传统隐私风险检测工具 Find Security Bug 相比，本系统将误报率减少 20%。综上，本文提出的方法能够提供全面的隐私数据分布并给出更准确的风险检测结果，从而节约维护成本，提高数据合规工作效率。

关键词：程序分析，污点分析，机器学习，程序表示学习，隐私数据

研究生毕业论文英文摘要首页用纸

THESIS: Design and Implementation of Privacy Data Leakage Detection System Based on Taint Analysis and Machine Learning

SPECIALIZATION: Software Engineering

POSTGRADUATE: (Author Name)

MENTOR: (Supervisor's position)

Abstract

In the era of big data, users' personal privacy data has become a commodity with great economic value. The collection of personal privacy data has become more extensive, and at the same time, data security problems have become more serious. Organizations in various countries have introduced corresponding personal privacy protection policies. In order not to violate privacy regulations, companies need to carry out data compliance work. The most important task of data compliance is to locate the private data in the software program and analyze the leakage risk, making the automated privacy data leakage detection technology the current focus of enterprise organizations. The current implementation of privacy leak risk detection technology focuses more on the detection of the location of the leak, but generally does not focus on identifying whether the private data has been anonymized, so some desensitized data that has been anonymized should not be counted as a data leak. But because it did undergo dangerous operations and was judged to be at risk, it caused false alarms, which increased the workload of data compliance and even increased the cost of maintenance.

This article aims to provide an accurate and complete view of private data, improve the detection capabilities of anonymized private data, and reduce the false positive rate of the risk of private data leakage by traditional methods. This article uses taint analysis, program graph representation technology and machine learning classification model to provide developers or security engineers with more accurate program privacy data leakage risk detection services, thereby reducing the workload of data compliance and ensuring the software system's comprehensive monitoring of private data And protection. This paper proposes a private data path positioning method, Private Data Trace Positioning (PDTP), which firstly obtains the propagation path of externally input user

privacy-related data variables in the project program based on taint analysis technology; secondly, through private data variables The function calls in the propagation path extract a set of candidate functions that may be encrypted functions; then for candidate functions, use program representation learning technology to extract function features and combine graph kernel functions to generate function feature vectors; then use machine learning classification models to compare candidate functions The function performs classification and identification to improve the accuracy of the risk assessment of privacy data leakage and generate the final scan detection result.

This paper designs and implements a privacy data leakage risk detection system for Java Web projects based on the proposed method. The system in this paper is mainly divided into visualization module, business processing module, candidate encryption function generation module, program feature extraction module and classification recognition, and uses SpringBoot, Django, Docker and other technologies and architectures to complete the system implementation. Experiments show that the accuracy of this system can reach more than 88% at the expense of less recall rate. Compared with the traditional privacy risk detection tool Find Security Bug, this system reduces the false alarm rate by 20%. In summary, the method proposed in this paper can provide a comprehensive view of private data and give more accurate risk detection results, thereby saving maintenance costs and improving data compliance efficiency.

Keywords: Program Analysis, Taint Analysis, Machine Learning,Program Understanding, Private Data

目录

表 目 录

图 目 录

第一章　引言

1.1　项目背景及意义

随着云计算、互联网 + 以及大数据分析等新一代技术的创新与发展, web 2.0 以及新一代移动智慧终端的应用与普及, 互互联网活动中涉及到大量的用户个人隐私数据。在大数据时代，用户的个人数据成为了一种有着巨大经济价值的商品，网络服务提供商可以通过对信息的收集和利用谋取相当大的利益。个人数据的收集愈加广泛，同时也使得个人数据隐私安全问题更加严重。例如主营打车业务的优步公司 2016 年 11 月被一个黑客攻击了其云存储服务器[1],16 个重要的大型数据文件被攻击者泄露, 其中一个涉及收集到来自全世界 3500 万的手机用户以及 370 万名汽车司机的诸多个人信息, 包含了大量用户居住地址、用户日常出行目的地以及用户出行过程历史等详细资料。这些被恶意泄露的涉及用户隐私的具体数据信息能让不法分子请以定位到用户的日常路线、工作单位甚至家庭住址，带来了严重的安全威胁。

为了准确有效地保护网络中个人隐私和网络信息的安全，各国安全组织分别制定出台了许多与此相关的安全政策和法律法规，如欧盟的 GDPR[2]（通用数据保护条例）、美国加州的 CCPA[3]（加州消费者隐私保护法案），我国的网络安全法等。以上的法规都涉及到对企业规定在对用户的数据收集、存储、保护和使用时的标准。为了不违反以上法规，企业需要进行数据合规的工作，包括但不限于对数据的定位、监控、保护以及最小化。其中对个人隐私数据的监控和定位尤其重要，只有对隐私数据的位置以及处理路径有完整清晰的认识才能对其进行全方位的保护。由于一般的产品代码中涉及的个人数据类别繁多，且对数据的处理各式各样，因此在对隐私合规进行检查时需要牵涉大量的人力工作，时间和经济成本都极高，所以应用自动化的扫描工具是企业组织当前关心的重点。

当前已有众多程序风险检测相关的应用和研究，然而目前程序风险扫描检测工具仍存在或多或少的问题。首先，这类工具一般希望不漏掉真实的产生隐私泄露的情况——不产生漏报，部分已经脱敏的数据在进入危险代码区域时也会判定为有漏洞，但实际上不存在真实的泄露风险，从而产生了误报，直接增加了安全工程师检查的成本。其次是现有的隐私安全扫描工具关注重点在于标记

[1]https://www.sohu.com/a/278266182_114835

[2]https://gdpr-info.eu/

[3]https://oag.ca.gov/privacy/ccpa

可能发生数据泄露的程序的位置而非数据在程序中的传播路径。程序分析一般的黑盒属性也让隐私数据的源头难以追溯，开发人员仅根据工具的提示难以定位隐私变量在程序中的传播路径，对该变量是否经过匿名处理也难以识别。

本文利用经典的程序分析技术——污点分析获得隐私数据在程序执行时的传播路径，以提供程序系统中隐私数据的完整分布情况；并在获得隐私数据传播路径的基础之上，结合机器学习的分类算法对传播路径中的函数进行识别，以判断其是否经过匿名处理，降低隐私泄露风险检测的误报率，从而缓解上文提到的当前研究所存在的问题。本文就此提出了一种结合污点分析以及机器学习的隐私数据程序检测的方法，通过污点分析、程序表示方法、图核函数技术以及机器学习分类算法，提供准确完备的隐私数据扫描检测服务。本文方法以污点分析为基础，以所有外部输入数据作为源点进行分析，并根据数据变量的命名进行筛选，将关键点标记嵌入分析流程中，记录隐私数据变量在程序执行中经过的所有程序语句和函数调用，得到隐私数据污点传播路径图。由于污点分析本身只关注数据的传播而不对实际经过的函数功能做判断，该结果存在误报（已经经过有效处理的数据也报出数据泄露风险）；所以接下来根据污点传播图进行分解，提取路径中隐私数据流经的函数，基于程序图表示方法以及图核函数生成函数的向量表示，并用机器学习分类算法模型识别该函数是否是有效的匿名清洁函数，从而给出一个准确的隐私数据扫描检测结果，以此让安全工程师或软件开发人员将更多注意力放在更可能存在隐私泄露的数据传播链路上而不用浪费精力在已经经过有效处理的隐私数据上。本文基于上述方法实现了一个针对 Java Web 项目的原型系统。具体来说，系统需要用户输入已经经过编译的包含待扫描工程的类文件文件夹的路径，系统通过上文所述方法对程序进行分析后会给用户提供程序隐私数据扫描检测结果。综上所述，本文方法一定程度上缓解了当前隐私泄露风险监测技术的输出结果难以直接利用以及产生误报多的问题，并提供了实现的原型系统作为参考。

1.2　国内外研究现状

本文结合当前研究现状，将传统的隐私泄露风险检测方法，即程序分析技术与当前的热门研究话题，机器学习技术相结合，对程序系统进行隐私泄露的风险检测，从而保护用户隐私数据的安全性，故本节将分别从基于程序分析以及机器学习两方面的隐私泄露风险检测相关工作描述当前该领域的研究现状。

1.2.1　基于程序分析的隐私泄露风险检测技术研究现状

在现代计算机科学中，程序分析 [1] 是指自动化分析计算机程序正确性，鲁棒性，安全性和健壮性等各种状态属性的方法。程序分析按程序执行与否一般

分为静态程序分析和动态程序分析；静态程序分析在不需要执行程序的情况下分析其程序实际执行的过程；另一方面，动态程序分析通过执行检测程序并生成某种形式的桩插入程序内来分析程序。静态分析技术较动态程序分析更加快速直观且没有程序执行的成本，因此仍然是分析计算机程序的最流行选择，即也是程序风险分析的主流选择。

程序风险漏洞检测面向的代码类型包括开发人员常用的源代码以及经过编译后的二进制代码 [2]。虽然面向二进制代码的相关研究具有适用性广泛，不受语言类型约束，漏洞检测的准确度较高的优点，但该类研究工作仍相对较少，原因在于二进制代码缺乏高级语言的结构和类型信息，分析难度较大。相对而言，源代码拥有原始的结构以及语义信息，具有比编译后的二进制代码更多的可供分析元素，对漏洞的定位更加友好，因此在程序风险漏洞检测领域有更多的研究工作集中在对源代码的分析上，以下就对该类研究进行简要介绍。

在开发实践的过程中，研究人员发现相同的漏洞经常出现在相似的代码中，由此基于代码相似性的漏洞检测方法成为该领域研究的热点。Chowdhury 等 [3] 基于复杂性、内聚以及耦合 3 个维度检测漏洞。SecureSync[4] 的研究旨在检测共享库中的再现类型漏洞，其使用来抽象风险缺陷程序片段的基础是图的 API(xGRUM)。Neuhaus 等 [5] 研究发现可以通过分析两段代码是否包含相似的导入和函数调用集合来判断其中一段代码是否具有与另一段代码相同的漏洞，以此提出了一种预测风险漏洞的方法。上述基于代码相似性的漏洞检测方法大多局限于检测代码相同或者几乎相同的代码复制类型 I，II[6]，检测能力有限。

基于符号执行的漏洞检测方法在中间语言表示上对程序进行分析，结合符号执行和约束求解。Liang[7] 等人结合符号执行和 Z3 SMT 解释器来检测程序缺陷，开发了一个基于 LLVM 中间表征的静态分析工具，可以分析除零错误、指针溢出和死代码等 3 种类型的程序漏洞。Cassez 等 [8] 通过分析 LLVM 中间表征，分析中间表征中的风险区域是否能狗被执行到以预测程序缺陷是否存在，并基于此开发了一款静态分析工具。Thome[9] 采用基于蚁群优化元启发式算法的混合约束求解过程，提出了一种搜索驱动的约束求解技术，可以作为任何现有字符串求解器提供的复杂字符串操作的补充。符号执行的分析精度受限于约束求解器无法求解所有形式的约束；同时由于进行程序实际执行的成本高，该方法往往无法扩展到大规模程序。

基于数据流分析的漏洞检测方法通常在词法语法解析基础上，对源代码建模，并基于专家针对各类漏洞人工分析生成漏洞规则，由规则进行相应的数据流分析。常见的基于数据流分析的程序风险扫描检测检测工具包括开源工具

Flawfinder[10]、RATS[11]、ITS4[12]，业界产品 Checkmarx[4]、Fortify[5]、LGTM[6]等。上述开源工具由于基于的漏洞规则和解析器的实现较为简单，所以存在误报漏报高的问题；与之相对应的，商业产品由于更加复杂的专家规则定义，检测效果一般优于开源工具。但基于数据流的漏洞检测方法中，对风险漏洞的发现规则依靠人工经验，难以覆盖所有漏洞缺陷的形成情况，也无法对程序的功能语义进行完全理解，导致基于数据流的风险检测方法也存在较高的漏报和误报。

污点分析是一种常用的数据流分析方法，在对程序进行污点分析时，该方法将程序执行中的部分数据变量（大多数是由系统用户进行的数据输入）标记成为污点源，并将污点随着程序语句的执行传播，从而达到追踪程序数据流的目的，与本系统对隐私数据路径进行扫描的场景相契合。本系统考虑应用数据流分析中的污点分析技术进行隐私数据的扫描，而针对污点分析作为数据流分析技术所带来的较高的误报和漏报的缺陷则通过结合机器学习技术进行解决。

1.2.2　基于机器学习的隐私泄露风险检测技术研究现状

当前各个应用领域都在与机器学习技术相关联，也证明了机器学习在各种领域任务上的有效性。的 [13]。在应用程序风险控制的领域机器学习也在发光发热，如垃圾邮件过滤 [14][15] 和入侵检测系统 [16][17] 等。由于以上众多研究的进展，近年来研究人员也纷纷采用机器学习技术试图突破程序风险检测领域的瓶颈，从大量的风险缺陷相关样本中提取特征，并训练生成模型对未识别的样本进行分类预测，以提高当前风险监测方法的准确率和效率。

在软件漏洞风险分析和发现与机器学习结合的领域，已经发表了许多研究成果，本文将这些研究分为三个主要类别进行介绍，包括：基于软件度量的脆弱性预测、基于相似度的漏洞识别以及易受攻击代码模式识别。

基于软件度量的脆弱性预测是指利用通用的软件度量值作为特征的预测模型，评估软件模块（源代码、面向对象的类、二进制组件等）的脆弱性状态。Perl 等人 [18] 提出了使用代码存储库中包含的元数据（CVEs 到 GitHub 提交的映射）以及代码指标来识别造成漏洞的提交的影响，并利用 SVM 分类器来标记可疑的提交的方法。基于该方法的工具 VCCFinder 与对照的传统工具 FlawFinder[19] 相比，在相同的召回水平下，该方法将错误警报的数量减少了 99% 以上。

基于相似度识别漏洞也是当前基于机器学习识别程序风险行为的一个重要研究领域，有众多学者进行了相关研究 [20, 21]。该方法的主要思想是如果两个

[4]https://www.checkmarx.com/

[5]https://www.microfocus.com/en-us/products/static-code-analysis-sast/overview

[6]https://lgtm.com/

代码片段具有高相似度且其中一个片段为漏洞代码，则另一个代码片段也极有可能同为漏洞代码，反之则可能存在误报。研究学者在度量相似性时通常先将程序代码抽象转化为词向量 [21]、树和图，提取程序不同方面的特征，以此比较相似性。李珍等学者开发的 VulPecker[20] 从不同的特征维度定义并设计相似度算法，针对各类型漏洞制定不同的分类方法。该类工作的优势在于不需要提前兑大量数据进行标记，但其识别精度较低，未得到广泛应用。

易受攻击代码模式识别与基于相似性的漏洞识别类似，但不同于后者是对于代码内容的匹配，该方法核心在于提取代码样本的模式，利用模式匹配技术识别程序中相同模式的漏洞代码。山口等 [22][23] 通过引入“漏洞外推法”的概念，提出了一种用于辅助检测和发现安全漏洞的技术。漏洞外推的原因是基于安全分析师每天的日常工作周期内经常在其余的代码库中对最近发现的各个漏洞进行搜索类似实例 [24]。为此，文章提出了四步计算过程：第一步是通过基于岛语法的鲁棒解析器为程序中的每个函数生成抽象语法树；第二步，利用词袋技术将函数的抽象语法树嵌入向量空间中；第三步，提取程序中与代码库中的抽象语法树中常共同出现的子树相应的结构模式；第四部，使用距离函数比较上一步输出矩阵的行以运行脆弱性外推。文章基于以上方法，使用测试项目中最新报告中的某些漏洞作为推断的引子，从而发现了多个零日漏洞。

1.3　本文的主要工作

为解决现有隐私安全扫描工具产生误报多，扫描结果难以直接追溯到隐私数据源头及路径的问题，本文提出了一种隐私数据程序扫描检测方法，并根据以上方法开发了原型系统，本文主要工作如下：

在方法层面，本文将经典的数据流分析方法污点分析与智能化的机器学习相关技术相结合，提出了一种隐私数据传播路径定位方法（（Private Data Trace Positioning，PDTP）。该方法框架分为三个部分：（1）隐私数据变量污点传播路径分析，该部分使用污点分析技术分析外部输入的隐私变量在程序系统中的传播路径，并生成污点传播图；（2）候选加密函数特征提取以及向量化，该部分特征提取基于程序表示技术，结合抽象语法树以及控制流图提取程序特征，并利用图核函数技术对程序图表示进行向量化；（3）基于机器学习的候选加密函数分类识别，该部分针对上一步骤的生成的函数向量，利用机器学习分类算法进行分类，识别该函数是否伪匿名化函数。

在系统层面本文基于上述方法实现了一个原型系统工具，该系统面向 Java Web 应用，使用 Spring Boot 以及 Django 框架开发，并向外提供 RESTful 接口，简单标准易扩展，供详细直观的隐私数据程序扫描检测结果报告，为安全工程

师进行数据合规工作提供有效的帮助。本文系统分为以下三个核心模块：（1）污点分析模块，本系统以项目工程文件夹为输入，基于 Java 分析优化框架 Soot 实现了针对隐私数据变量的污点分析。系统首先根据用户配置以及系统默认配置对外部输入的数据变量进行筛选，初步定位隐私数据变量源，并基于过程间分析的经典 IFDS 算法进行上下文敏感的污点分析的实现，以达到对敏感数据传播路径的精确覆盖。模块的输出为污点传播图，其中点包括源点、流经的程序语句以及可能的泄漏点，边则代表点与点之间的变量传播路径，该图也是后续所有分析的基础。（2）程序特征提取模块，首先对上个模块污点传播图进行处理，筛选出需要分析的函数。针对待分析的函数，系统基于 JavaParser 生成函数的控制流图以及数据流图，并进行结合产生控制数据流图（CDFG），以 JSON 的形式存储。针对 CDFG，系统应用 javalang 对每个语句进行 token 的生成并将其作为点连接到 CDFG 中，生成最终的程序图表示。（3）分类识别模块，系统应用机器学习分类算法对程序图进行分类识别，判断其是否为加密函数。

1.4　本文的组织结构

本文总共分为六个章节，组织结构如下：

第一章，引言部分，该部分是对本文内容的整体概括，首先是介绍了当前大数据环境下个人隐私数据保护的重要性，即本文的研究背景意义；其次重点分析了当前隐私数据泄露风险检测领域的国内外相关研究以及当前存在的不足；最后介绍了本文的主要研究工作内容和本文的组织结构。

第二章，技术综述，根据本文的背景和目的提出并介绍了本文工作相关的概念以及技术框架。对本文方法使用的核心技术，包括污点分析、程序表示技术、图核函数进行介绍，对本文系统的开发技术选型也做出简要介绍。

第三章，技术研究，该章节详细介绍了本文提出的隐私数据路径定位研究方法 PDTP，依次从方法简介、方法流程、主要组成部分进行了详细的描述。

第四章，需求分析与概要设计，首先对本文隐私数据泄露风险检测系统的功能以及非功能特点需求进行一个整体性的概述，其次是对系统总体设计的描述，最后对系统各模块以类图流程图等形式进行模块设计的介绍。

第五章，详细设计与实现，在第四章的基础上，对于系统各个模块结合代码以及系统截图进行详细说明。

第六章，系统测试与实验评估，首先介绍系统的测试环境，然后对系统进行相关的功能测试和性能测试，对系统的检测效果进行分析评估。

第七章，总结与展望，该章节是对本文研究以及开发工作的成果以及不足的分析，并就未来的论文研究相关工作进展情况进行了长期展望。

第二章　技术综述

本章主要介绍与本文方法相关的技术以及与本文系统开发相关的技术。本文方法有关的技术包括提取隐私数据传播路径的污点分析技术、提取候选加密函数特征的程序表示技术、向量化程序图表示的图核函数技术以及对候选加密函数进行分类识别的机器学习分类技术。本文系统有关的技术包括后台系统所使用的的 Spring Boot 框架、Django 框架、后台模块之间进行通信的微服务技术以及进行服务部署的 Docker 技术。本章节将就上述各个技术作简要的介绍。

2.1　污点分析

污点分析是数据流分析的一种实践技术，分析程序中由外部引入的数据是否会传播到包含危险操作的程序位置来推测程序是否存在安全性漏洞。因此被广泛地应用在程序安全漏洞发掘以及程序隐私数据泄露检测等实际领域，被众多程序扫描检测工具所使用。

2.1.1　污点分析原理

污点分析主要有三个组成要素：污点信息产生点、污点信息汇聚点和污点信息清洁点，这三个组成要素的代表含义如下：

污点源（source）：污点源代表从外部直接引入程序系统的危险数据或者不受信任的数据，即攻击者可以用来操控的数据或者用户的敏感数据，比如 Web 应用中读取 URL 参数的函数。

汇聚点（sink）：污点汇聚点代表直接产生违反数据完整性的安全敏感操作或者违反数据保密性的泄露隐私数据操作，攻击者可以通过这些操作获取隐私数据，如控制台输出语句、数据库查询语句以及日志输出语句。

清洁点（sanitizer）：清洁点即无害处理，代表通过对数据进行加密、移除危害操作、过于外部数据等手段使数据传播不再对软件系统的信息安全产生危害。清洁点决定了污点是否能继续传播，所以对清洁的识别是重点问题，不能有效处理将引发过污染问题，但当前污点分析本身并不能解决。

如图2.1所示，污点分析主要关注 3 个部分 [25]:

(1) 污点源。对于外部输入的变量，若该变量为不受信任的数据，则将该变量关联污点标记，成为污点变量，如图中的污点变量 1,2。

(2) 污点传播。污点变量在程序中的传播过程中可能与其他变量由于函数或

表达式等途径进行结合，结合后的变量也为污点变量，如图中污点变量 3 与变量 4 结合后成为污点变量 5；同时污点变量也有可能由于无害化处理（清洁函数）变成普通变量，如图中的污点变量 7 经由无害处理最终输出普通变量 9.

(3) 污点汇聚点。若在污点汇聚点之前污点变量尚未经过无害处理，如图中的污点变量 10，则此处产生了安全敏感性操作。

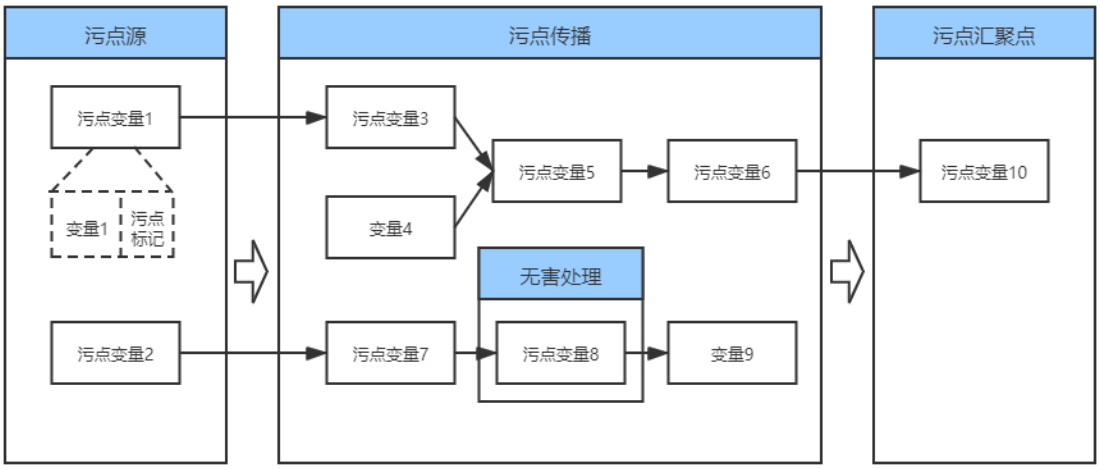

图 2.1: 污点传播过程

下面以图2.2的程序为例来说明污点分析的具体过程。分析过程从左侧函数开始，因为其找到了一处污点源，即获取请求参数值的函数 request.getParameter(”psw”)，于是将污点传递到变量 psw，接着调用函数 func(psw)；接着对函数 func() 做过程内分析，得到其函数内部的传播路径并标记返回值为污点变量；于是回到调用者的函数内，变量 psw_new 被标记为污点变量，又因为第三行存在一处汇聚点，即变量写入日志的操作 logger.logInfo(psw_new), 并且参数 psw_new 为污点变量，可以判断此处有数据泄露风险。

污点分析能够一定程度上理解程序上下文，分析出某个程序位置是否存在风险漏洞，因此该方法已被很多工业界、学术界的安全静态扫描工具所使用[26, 27, 27–29]，本文研究方法以及原型系统也将使用该方法对隐私数据传播路径进行定位。然而，污点分析并不能得出高准确率的分析结果，其对容器类型数据不能很好处理以及对清洁函数功能难以识别都使得其存在产生误报的缺陷，针对这些不足，本文将在下文引入机器学习分类模型对程序函数进行分类识别来降低隐私数据泄露风险误报率。

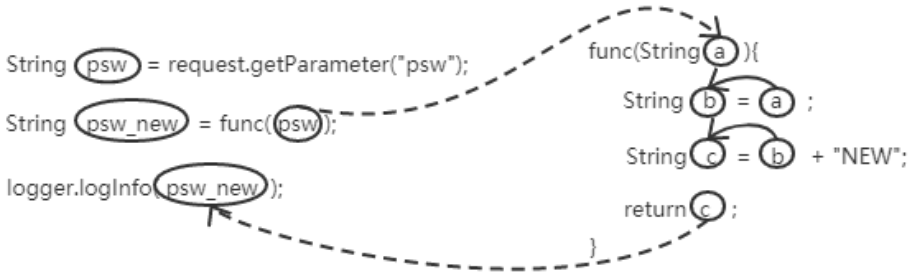

图 2.2: 污点分析程序示例

2.2　机器学习分类

分类是将给定数据集分类为多个族的过程，通常可以结构化或非结构化数据上执行。该过程从预测给定数据点的类别开始，这些类通常称为目标，标签或类别。分类预测建模是将映射函数从输入变量近似为离散输出变量的任务，主要目标是确定新数据将属于哪个类别。

机器学习分类存在以下几个核心的概念：（1）分类器：用于将输入数据映射到特定类别的算法。（2）分类模型：该模型预测或得出给定训练数据的结论，它将预测数据的类别或类别。（3）特征：特征是观测样本得到的单个可测量属性。（4）二元分类：这是一种具有两种结果的分类，例如正确或错误。（5）多元分类：具有两个以上类的分类，在多分类中，每个样本都分配给一个且仅一个标签或目标。（6）训练分类器：通过对训练样本进行不断地拟合来调整模型参数，以达到模型的最佳效果。

分类器可以分为两种类型，被动分类器和主动分类器[1]：被动分类器仅存储训练数据，使用存储的训练数据中最相关的数据完成分类。与主动分类器相比，该种分类器有更多的预测时间。例如，k 近邻，基于案例的推理；主动分类器在获取预测数据之前，根据给定的训练数据构建分类模型。它必须能够对整个空间都适用的单个假设作出承诺。因此，他们花费大量时间进行训练，而花更少的时间进行预测。例如决策树，朴素贝叶斯，人工神经网络等。

本文选用的机器学习分类算法为支持向量机，该方法是一种监督的机器学

[1]https://www.edureka.co/blog/classification-in-machine-learning/classification

习算法，可用于分类或回归任务，其中分类是主要应用方向。该方法的原理是将每个数据项绘制为 n 维空间中的一个点（其中 n 是特征数量），每个特征的值是特定坐标的值，通过找到能很好地区分两个类的超平面进行分类。SVM 算法的优势计算代价不高，易于理解和实现，适用于数值型和标称型数据，本文的分类识别是二分类问题，判断函数是否为匿名化函数，故选择支持向量机作为使用的具体算法。

2.3　程序表示

2.3.1　基于序列的表示

程序是由有序的符号序列组成的，由此可以将程序形式化地表示为字符 (character) 或词素 (token) 序列，除此之外, 代码中的应用程序接口 (API) 序列包含了程序的功能信息, 因此 API 序列也成为了许多研究工作的研究对象 [30]。

基于字符序列的程序表示方法当前的研究存在例如 Cummins 等人 [31] 提出了基于 LSTM 的字符级别的程序学习模型。基于应用程序接口 (API) 序列的程序表示方法的研究对象是程序中包含的 API 序列，对程序中包含的 API 序列进行分析建模可以进行 API 序列检索、代码检索等任务，例如 Gu 等人 [32] 提出了基于深度学习的 API 检索方法。

程序词素 (token) 序列较前两种序列表示保留了更多的程序特性。基于对已知程序的分析，模型中给定了部分程序的序列，模型中可以预测之后最有可能会出现的 token。因此，该模型被广泛应用于对程序的代码进行补全以及对程序的注释产生等任务 [33–37]。但程序一般具有丰富的分支循环等结构，将程序当做顺序的序列进行表示会丢失程序重要的结构信息导致相关任务效果难以提升。

2.3.2　基于结构的表示

基于结构的表示主要分为两类：基于抽象语法树的程序表示以及基于图的程序表示。抽象语法树 (AST) 是基于源代码的一种抽象代码语法和语言结构的表现形式，树中各个节点都具有可以与其对应的源代码中的一种抽象代码的语法结构，树中各节点都对应源代码一种代码结构，示例如图2.3所示，对应下方的代码片段。之所以说 AST 是“抽象”的，原因在于树结构中的节点不会表示真是语法中出现的所有细节而是一种高度抽象。抽象语法树能有效地表示程序的语法及其结构，被广泛应用于程序理解相关任务中 [38–41]。

程序中的语句之间通常存在控制或依赖的关系，抽象语法树中只能体现程序单元的无指向性的固定结构而不能体现出具有方向性的依赖关系；而图结构

可以对程序的各程序指令之间的数据流动以及控制和依赖关系进行建模，具体的表现为控制流图以及数据依赖图，示例如图2.4所示，图中实线为数据依赖而虚线代表控制依赖。近年来许多程序分析的研究工作基于程序的图表示展开相关工作 [42–45]。本文方法结合两种结构式程序表示方法，融合程序控制流图以及 AST 生成混合式的程序图结构表示。

```
1
2  public static void sort(Comparable[] a) {
3          int n = a.length;
4          for (int i = 0; i < n; i++) {
5              for (int j = i; j > 0 && less(a[j], a[j-1]); j--) {
6                  exch(a, j, j-1);
7              }
8          }
9      }
```

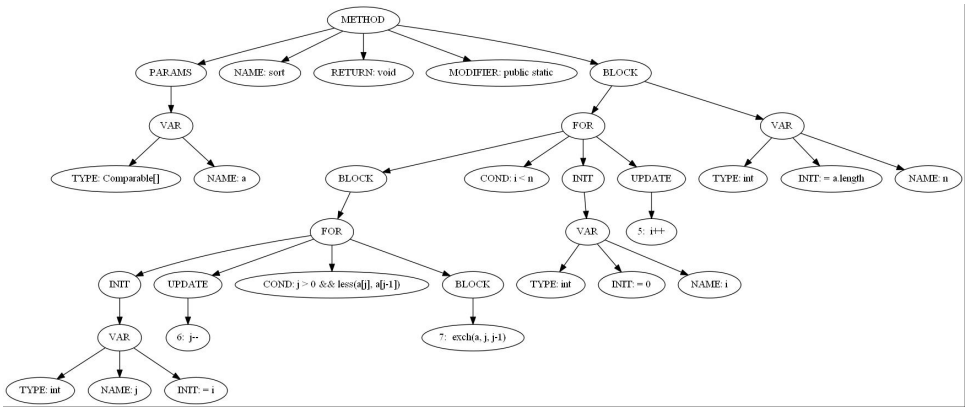

图 2.3: 抽象语法树

2.4　图核函数

准确衡量图之间相似性的问题是各个学科领域的核心，图核函数最近成为解决该问题的一种极有前景的方法。图核函数属于核函数方法，一个图核是图集 $\mathcal{G}$ 的一个对称的半正定函数。一旦在集 $\mathcal{G}$ 上定义了这样的函数 $k : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$，就知道存在一个映射 $\phi : \mathcal{G} \to \mathcal{H}$ 在希尔伯特空间 $\mathcal{H}$，使得：

$$k(G_i, G_j = < \phi(G_i), \phi(G_j) >_{\mathcal{H}}$$

其中对于所有 $G_i, G_j \in \mathcal{G}$ 其中 $< ... >_{\mathcal{H}}$ 是 $\mathcal{H}$ 中的内积。总的来说图核是一个用于度量两个图的相似性的函数，其允许如支持向量机之类的内核化学习算法直接在图上工作，不需要另外进行特征提取即可将图结构转换为固定长度的实值特征向量。

在图核函数算法中,Weisfeiler-Lehman(威斯费勒-莱曼) 核是一种经典的图核算法 [46]，本文采用该图核函数算法进行函数图表示的向量化。该方法基于快速特征提取算法将原始图结构映射成一个图的序列，通过这个序列中每个图中的顶点特征属性提取出图源的标志信息和拓扑结构，并且使用了一族核函数的方式来精确地计算两个图像之间的相似程度。核方法主要可以分为三大类：基于游走与路径 (walk and path) 的核方法、基于子树 (subtree) 的核方法以及基于有限子图（limited size subgraph）的核方法。

本文在图核函数技术部分借助 GraKeL 库进行实现。GraKeL 是一个提供多种完善的图内核实现的库，并将其统一为一个通用框架。该库是基于 scikit-learn 框架编写的，利用 GraKeL，可以轻松地为诸如图形分类和聚类等任务构建完整的机器学习通道。在 GraKeL 中，所有图内核都必须继承 scikit-learn 的 TransformerMixin 类，并实现以下四种方法：(1) fit：从输入图集合中提取依赖于内核的特征。(2) $fit_transform$：拟合并计算输入图集合的核矩阵。(3) $transform$：计算新图集与作为输入给定的图集之间的核矩阵。(4) $diagonal$：如果已调用此方法，则返回作为输入给出的所有图的自身内核值，以适合与作为变换输入的给定值。此方法用于归一化内核矩阵。所有内核都统一在名为 *kernels* 的子模块下。它们全部包装在一个称为 *GraphKernel* 的通用类中。除了提供统一的接口外，GrakeL 更具优势的是具有多种支持的内核框架，包括本文选用的 Weisfeiler Lehman 图核以及其他主流算法如 Graphlet 图核、Lovasz-theta 图核、GraphHopper 图核等近 20 种图核算法。

2.5　系统开发技术栈

2.5.1　Spring Boot 框架

Spring Boot 是当前最为流行的基于 Java 的开源框架，由 Pivotal 团队开发[2]，用于构建独立的和可生产的 Spring 应用程序。Springboot 可以帮助开发者创建独立的、基于 Spring 的产品级应用程序。大多数 Spring Boot 应用程序只需要很少的 Spring 配置，节省了开发者在配置上的麻烦。通过用 SpringBoot 创建的 Java 应用程序可以通过使用 jar 包或更传统的 war 包部署来启动。基于以上 Spring Boot 在开发上的优势，本文的业务处理后台部分采用该框架进行开发。

2.5.2　Django 框架

Django 是当前流行的高级 Python Web 开发开源框架[3]，由经验丰富的开发人员构建，拥有活跃的社区和详尽的文档，具有拥有强大的数据库访问组件、灵

[2]https://spring.io/projects/spring-bootoverview

[3]://www.djangoproject.com/

活的 URL 映射、丰富的 Template 模板语言等优势。本系统后台分类识别模块主要对机器学习模型操作，因此自然选择 Python 语言进行开发，本系统可以利用 Django 的 URL 分派系统接受客户端传来的预测、训练、标记等请求并对其处理，以完成对系统后台的快速开发。

2.5.3　Docker 技术

Docker 是一款开源软件，同时也是一个开放平台，用于开发、交付（shipping）以及运行应用。Docker 允许用户将基础设施（Infrastructure）中的应用单独分割出来，以更小的粒度存在（容器），从而提高交付软件的速度[4]，基于上述优点，本文依托 Docker 进行系统发布。

2.6　本章小结

本章主要介绍本文提出计数方法使用的技术、工具和框架及其原因。在技术角度，本章依次了介绍污点分析技术、程序表示技术和图核函数并对其优缺点以及使用原因进行总结；在工具角度实现角度，介绍了本系统开发过程中涉及的核心技术栈，主要包括 Spring Boot 框架、Docker 容器技术、微服务架构以及 Django 框架等。

[4]https://docs.docker.com/get-docker/

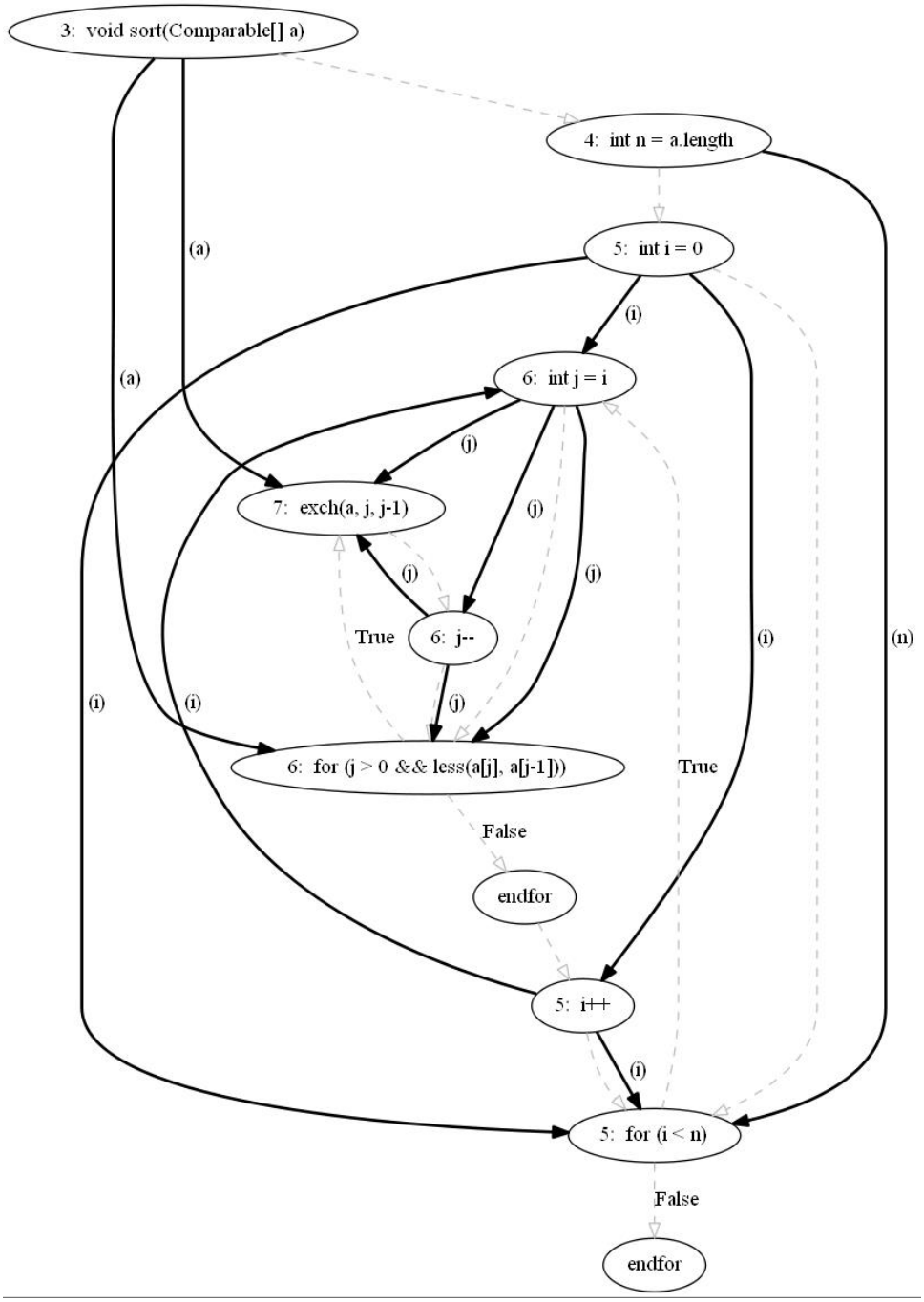

图 2.4: 控制流/数据依赖图

第三章　技术研究

本章主要介绍本文应用的基于污点分析技术以及程序表示技术的隐私数据传播路径定位（Private Data Trace Positioning，PDTP）的整体方法概述。首先是对 PDTP 方法的整体框架进行介绍，包括 PDTP 的简介、整体流程、主要组成部分等；其次是对 PDTP 的三个主要组成部分分别进行介绍，包括：

（1）隐私数据变量污点传播路径分析；

（2）候选加密函数特征提取以及向量化；

（3）基于机器学习的候选加密函数分类。

3.1　PDTP 整体框架

图 3.1 展示了 PDTP 方法的整体流程，包括以下六个主要步骤，其中步骤一-三对应图中的第一部分，隐私数据路径分析；步骤四-五对应图中的第二部分，候选加密函数特征向量生成；步骤六对应图中第三部分，候选加密函数分类识别：

步骤一：对于源程序 P，提取 P 的所有外部输入变量或参数集合 OV；

步骤二：根据系统内部预设的规范变量名配置以及用户配置的规范变量名定义对 OV 进行筛选，得到规范命名的隐私数据变量集合 NPV；

步骤三：以 NPV 作为污点源进行污点分析，得到隐私数据变量 NPV 的污点传播路径图 NP-TTrace-Graph，并从图中节点筛选出可能为加密函数的函数，作为候选加密函数集合 CDD-SET，在后续步骤进行分析；

步骤四：对候选加密函数集合进行程序特征的提取，生成融合程序控制流（控制流图，CFG）、结构（抽象语法树，AST）和文本信息（Token）的程序图表示，并对其各个节点进行变量以及方法名等数据的泛化处理；

步骤五：基于图核函数对函数表示图进行向量化处理，此处使用在许多图分类任务中表现出竞争优势的 Weisfeiler-Lehman 图内核函数进行向量的生成，得到函数向量 FUNC-VECTOR；

步骤六：以函数向量作为输入，基于机器学习分类算法模型进行函数分类，通过对数据集训练得到最佳的模型参数，并使用该训练好的模型进行函数分类预测。

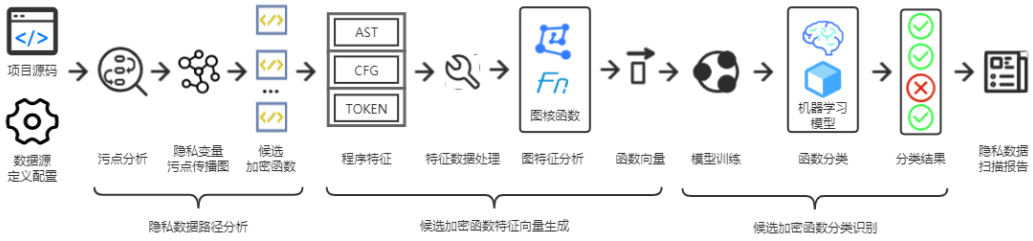

图 3.1: PDTP 流程图

3.2　隐私数据变量污点传播路径分析

PDTP 的隐私数据变量污点传播路径分析部分的流程有以下几步，首先输入被扫描的 Jar 包，以及关键隐私变量集合 KeyValSet；接着其调用分析器对每一个关键隐私变量 v 进行分析，依次对第 i 个关键隐私变量进行污点分析。该部分污点分析的实现主要关注三种元素：输入、规则以及输出。

首先对输入进行说明，此处的输入即污点源（source，产生污点数据的函数，如获取 HTTP 请求参数的函数等）以及污点汇聚点（违反安全规则的函数，如将数据输出到日志等）。其中识别处理 source 的依据如表3.1所示，其中 l 是指当前分析的程序语句行，图中的语句为常见的函数调用，x 为被调用函数所在对象，k 为被调用函数，$a_1, ..., a_n$ 是函数参数。规则一栏的表示采用推导形式，横线上面是条件，横线下面是结论，横线上方第一行表示程序语句 l 调用的方法 m 属于当前的 CG（调用图），第二行表示方法 m 属于会产生污点源的函数调用 $Sources$，由以上两行就可以推导出语句左侧变量 r 指向的对象可以更新为标记成污点变量值的 t_l，至此完成对污点源的定位处理。

表 3.1: 处理污点源（生成污点数据）

语句类型	语句表示	规则
函数调用（Call）	$l : r = x.k(a_1, ..., a_n)$	$l \rightarrow \text{m} \in \text{CG}$ $m \in Sources$ ——————— $t_l \in pt(r)$

与对污点源的处理类似，下图3.2展示了处理污点汇聚点并生成污点传播流

的规则，与上述不同处在于规则的第三行，代表当调用方法为有危险操作的函数，且存在方法参数 a_i 为污点数据时，可以推断得到二元组 $< t_j, m >$ 可以加入污点传播流的集合。在输入定位好以后需要对污点数据传播进行跟踪，传播的规则跟上述的输入输出识别规则类似，如表3.3所示，在此不作详述。

表 3.2: 处理污点汇聚点（生成污点传播流信息）

语句类型	语句表示	规则
函数调用（Call）	$l : r = x.k(a_1, ..., a_n)$	$l \rightarrow \text{m} \in \text{CG}$ $m \in Sinks$ $\exists i, 1 \leq i \leq n : t_j \in pt(a_i)$ ——————— $< t_j, m > \in TaintFlows$

表 3.3: 传播规则

语句类型	语句表示	规则
初始化（New）	$l : r = x.k(a_1, ..., a_n)$	$l \rightarrow \text{m} \in \text{CG}$ $m \in Sinks$ $\exists i, 1 \leq i \leq n : t_j \in pt(a_i)$ ——————— $< t_j, m > \in TaintFlows$
赋值（Assign）	$i : x = newT()$	$o_i \in pt(y)$ ——————— $o_i \in pt(x)$
存储（Store）	$x.f = y$	$o_j \in pt(x), o_i \in pt(y)$ ——————— $o_j \in pt(o_i, f)$
加载（Load）	$l : x.f = y$	$o_i \in pt(x), o_j \in pt(o_i, f)$ ——————— $o_j \in pt(y)$

通过上述分析获得的污点传播流集合可以合成系统整体的污点传播图，针

对图的函数调用节点，首先判其是否为已知的常用加密工具类，可直接判断为加密函数，若不是则判断其是否为系统内部函数，若为系统内部函数则将其视为候选加密函数，加入候选集合中，留待后续步骤进行分析。

3.3　候选加密函数特征提取以及向量化

针对候选加密函数，需要提取函数的程序特征并生成函数的特征向量，这一部分首先依赖常用的 java 分析工具 Java 框架进行特征图的提取，其次基于图核函数进行向量化的实现，，具体步骤如下：

步骤一: 生成程序控制流图 CFG；

步骤二: 生成抽象语法树 AST；

步骤三: 将 CFG 的各节点的程序语句链接到对应的 AST 节点，使两图相结合，生成融合图 ACFG；

步骤四: 遍历 ACFG 的各个节点，生成对应的 Token 串进行替换，生成最终的程序图表示；

步骤五: 基于图核函数对程序图表示进行向量生成，最终得到候选函数的特征向量。

最后一个步骤中选用的是第二章中介绍的 Weifeiler-Lehman 图核函数。在本文的实现中，将采用该算法来提取同在第二章中介绍的程序代码抽象语法树以及数据流图的融合图中的子结构作为机器学习的数据特征。

3.4　基于机器学习的候选加密函数分类

由于前一部分已经得到函数的特征向量，本部分的分类识别可以直接着眼于当前流行的机器学习分类算法进行分类，本文使用的是 SVM 支持向量机算法，具体步骤如下：

步骤一: 加载数据集到当前工作环境。

步骤二: 对数据集进行划分，分为训练集和测试集；

步骤三: 初始化 SVM 模型，进行模型的参数配置，包括核函数以及损失函数等参数的选取。

步骤四: 对模型进行迭代训练，此处可以借助开源的机器学习框架，如 scikit-learn、pytorch、tensorflow 等。

步骤五: 训练模型直到模型分类效果不再有提升，得到训练好的 SVM 模型；

步骤六: 用得到的 SVM 模型进行测试评估，若结果不理想则调整初始参数直到模型具有较好的分类性能，并以此模型进行分类预测。

第四章　需求分析与概要设计

4.1　系统整体概述

本文实现的原型系统旨在提供针对 Java Web 项目代码的隐私数据泄露风险检测扫描功能，选择该类项目作为目标是因为 Java 语言通常用于进行 Web 服务的开发，大量的用户数据通过程序接口传入到 Java Web 系统中进行分析处理。相对于传统的隐私漏洞检测工具，本系统将经典的污点分析技术与程序表示技术、机器学习技术相结合，从污点分析得到的污点传播路径以及路径中途径的函数进行分析处理，得到隐私数据的类别，隐私数据是否经过加密处理等精确的分析结果，提高数据合规工作的效率。系统总体框架如 4.3.1 节的图4.2所示，系统采取 CS（Client/Server）架构，主要分为客户端和服务端两大部分，客户端包括可视化模块，服务端包括业务处理模块、候选加密函数生成模块、程序特征提取模块以及分类识别模块，其中服务端与数据库进行交互。

候选加密函数生成模块基于污点分析技术以及 Soot 框架进行开发，并在分析转换函数中插入标记操作，使其能输出污点传播路径中所有语句及函数信息，为隐私数据类型的确定提供基础，并在此基础上进行数据加密候选函数的提取；业务处理模块整合候选加密函数生成模块以及分类识别模块的输出结果生成呈现给用户的完整报告，包括系统中各个隐私数据以及其相关的传播路径、泄露位置、是否经过加密处理等；程序特征提取模块对从污点传播路径中提取的函数进行程序特征的提取，基于 JavaParser 生成程序的控制/数据依赖图 (CDFG)，并对程序中的变量名进行匿名处理，以免影响后续分类，然后基于 CDFG 上的各个节点，即程序语句，应用 JavaLang 生成 token 序列并将其替代原语句连接到 CDFG 图中，即聚合程序的文本、控制流、结构特征，形成最终的程序图表示；分类识别模块基于 GraKel 应用图核函数对程序特征提取模块输出的程序图表示进行图的向量化，并应用神经网络对其进行分类识别，判断该程序是否为加密函数。本系统前端界面采用 JavaScript 开发，应用了 React 框架，通过 Axios 与后端接口进行交互；客户端采用 Java 开发，应用了 SpringBoot 框架；服务端采用 Python 开发，应用了 Django 框架便于与客户端交互。

4.2　系统需求分析

4.2.1　功能需求

本系统的功能性需求分析将按照系统模块进行逐一介绍，即分为可视化模块、候选加密函数生成模块、业务处理模块、程序特征提取模块以及分类识别模块五个方面进行。

可视化模块负责处理用户有关于项目扫描任务的配置工作，收集用户上传到系统的项目代码信息，包含项目代码本地路径，项目名称等信息，并且接受用户上传的数据命名配置文件、用户标记报告结果信息等，并将这些信息按类别传输到其他模块进行处理。其设计得具体功能性需求如表4.1所示。

表 4.1: 可视化模块功能性需求列表

需求编号	需求名称	需求内容	优先级
R1	上传项目	用户新建检测任务，并将检测的有关信息（项目路径、名称等）输入到系统中	高
R2	上传数据命名配置文件	用户可以上传数据命名配置文件到用户个人空间，并可以在检测任务中选择是否应用	中
R3	查看扫描报告	扫描完成后，用户可以查看扫描报告，并可以文件形式导出	高
R4	标记检测报告正确性	用户对于报告中的每个条目可以标记检测结果信息是否正确，包括该数据是否为隐私数据以及给出的加密函数是否为真实的加密函数等	中

候选加密函数生成模块对用户上传的项目文件进行扫描，得出外部输入数据在项目中的污点传播路径，该模块所涉及的功能性需求如表4.2所示。该模块通过对系统代码进行污点分析得到隐私数据的传播路径图，并基于该路径图提取候选加密函数。所以该模块包含三个部分的需求，首先是污点分析，该模块需要实现污点分析的全过程；其次是基于污点分析的路径得出路径中的系统内部函数调用，将其作为候选加密函数；最后是对污点分析结果的保存，以便后续报告的生成。

表 4.2: 候选加密函数生成模块功能性需求列表

需求编号	需求名称	需求内容	优先级
R5	污点传播路径分析	对用户上传的项目代码进行过程内及过程间的污点分析	高
R6	候选加密函数提取	从污点传播路径中提取候选的可能为加密函数的程序段	高
R7	保存分析结果	将污点分析结果转换成易存储转换的格式，存储到数据库中，以便后续报告生成及查看	高

业务处理模块负责连接客户端与服务端，获取候选加密函数生成模块的结果后调用服务端进行下一步分析，分析完成后整合客户端候选加密函数生成模块的中间结果以及服务端返回的分类识别结果生成最终的扫描检测报告。该模块需要给用户提供最终的扫描报告进行查看并接受可视化模块对报告的标记信息传送到服务端进行处理并更新报告结果。该模块的功能性需求如表4.3所示。

表 4.3: 业务处理模块功能性需求列表

需求编号	需求名称	需求内容	优先级
R8	项目预处理	对用户导入的项目进行预处理，包括文件的导入、调用图的生成等	高
R9	扫描报告生成	将污点分析结果以及服务端分类识别结果进行整合，生成整合后的最终扫描检测报告，并存储到数据库中	高
R10	导出报告	分析结果整合完成后，用户可以随时导出查看该项目的扫描检测报告	高
R11	处理用户报告标记	接收可视化模块传入的用户对于报告的标记信息，更新报告内容，并将正确的标记分类信息传入服务端进行持久化数据的更新	中

程序特征提取模块对程序代码进行特征提取并转换为可以应用于分类模型的向量形式，该模块所涉及的功能需求如表4.4所示。该模块的功能主要是对于

程序代码进行控制流图以及数据流图的生成，其次对生成的图进行泛化，保证该程序图不会受到特定程序变量命名的影响，最后是向量化的需求，程序图表示不能直接应用于分类模型，需将其转化为数值型的向量表示，此处利用图核函数进行程序图的向量的生成，以便作为后续分类模型的输入。

表 4.4: 程序特征提取模块功能性需求列表

需求编号	需求名称	需求内容	优先级
R12	生成控制/数据流图	对于输入的函数体，利用 JavaParser 等工具进行程序控制流图以及数据流图的生成，以提取程序的该部分语义特征	高
R13	泛化	对于程序图表示中的每一个节点进行泛化处理，将程序中具体的字符串内容、变量以及方法名称、函数以及引用包的名称以统一形式的抽象符号代替	高
R14	向量化	对于已经经过上述处理的程序图表示，基于图核函数算法生成函数的数值型向量表示	高

分类识别模块是保障分析结果准确性的核心模块，该模块所涉及的功能性需求如表4.5所示，该模块的需求主要为对给候选加密函数实例进行分类预测，判断其是否为真实的匿名化加密函数，标记预测结果的正误以及开启模型训练。

表 4.5: 分类识别模块功能性需求列表

需求编号	需求名称	需求内容	优先级
R15	预测函数类型	对于输入的函数向量，预测该函数是否为加密函数的类型	高
R16	训练分类模型	管理员可通过系统直接发起立即重新训练模型的任务，或者设置定期对模型基于当前最新的数据进行重新训练	高
R17	保存分类模型	管理员可以在 Web 页面发起训练模型任务，或是用户可以保存分类预测结果	低

4.2.2　非功能需求

本文系统旨在为开发者和安全工程师提供 Java Web 项目的隐私数据检测扫描功能，主要的非功能需求如表4.6所示。本文系统要求系统界面简洁，操作流程直观易懂，用户可以在经过短时间学习后了解本系统的用户操作流程。且由于本系统功能为对于项目代码进行隐私数据泄露风险扫描检测，项目一般出在实时更替维护的状态，对于风险的检测也需要时常进行，因此检测过程要保证一定的效率和健壮性，若检测时间过长会影响开发维护的进程。其次，由于本系统设计用户隐私数据相关，则系统本身也很可能面临被恶意入侵的危机存在，因此需要对安全性方面进行重视。最后由于当前业界使用的程序框架随着开发社区的进步也不断在更新迭代，系统需要保证一定的扩展性，以保证后期能以较低的成本进行功能的扩展。

表 4.6: 非功能性需求列表

需求编号	需求名称	需求内容	优先级
R18	易用性	系统应当界面简洁美观，操作给予充分的提示语，使用户能短时间内掌握系统使用	高
R19	可用性	系统需保证在较长时间内能稳定运行，每次停机时间小于 5 分钟	高
R20	可扩展性	系统各功能应当便于扩展，为将来可能增加的功能预留号接口	中
R21	健壮性	系统在分析过程中崩溃应有机制使系统尽快重启恢复服务并恢复数据	中
R22	性能	系统的分析时间对于小型项目不超过 3 分钟，对于大型项目不超过 15 分钟	低

4.2.3　用例分析

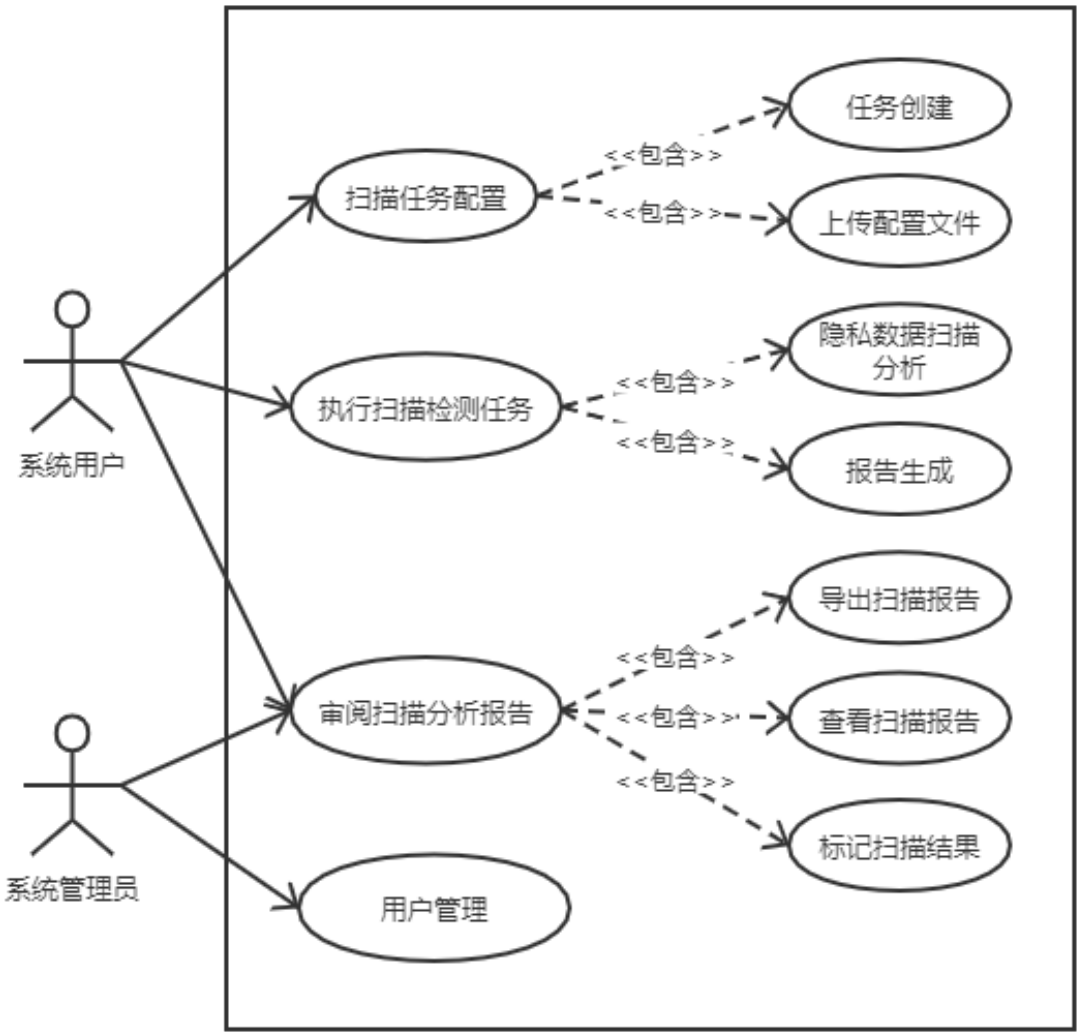

图 4.1: 系统用例图

本文系统角色包含系统用户和系统管理员，根据对系统需求的分析，得到系统用例图如图4.1所示。本文系统 4 个主要的用例分别为扫描检测任务配置、执行扫描检测任务、审阅扫描分析报告、用户管理，前两个为系统用户所属的用例，第三个为系统用户和系统管理员均有的用例，最后一个为系统管理员所属的用例。其中扫描任务配置包含扫描任务创建、配置文件上传，执行扫描检测任务包含隐私数据扫描分析、报告生成，扫描分析报告审阅包含扫描分析报告查

看、扫描报告导出、扫描结果标记。本小节将对上述 4 个主要用例所进行详细阐述并给出相关用例描述。

扫描检测任务配置用例如表4.7所示，用户创建扫描任务，上传包含项目程序文件以及编译后 class 文件的压缩包，填写项目基本信息，并进行项目的数据命名配置，最后完成任务的创建和提交上传，等待用户后续对任务的操作。

表 4.7: 扫描检测任务配置用例描述

描述项	说明
用例 ID	UC1
用例名称	扫描任务配置
参与者	系统用户
用例说明	系统用户进行扫描检测任务的信息输入并新建任务，同时可以上传数据配置文件
前置条件	系统用户成功登录系统
基本事件流	1. 系统用户点击主页创建检测任务的图形按钮，系统弹出创建任务的输入框，输入框中显示待输入任务信息的表格，表格各项有对用户的引导； 2. 系统用户点击弹出框中上传项目的图形按钮，并选择本地文件库中需进行扫描的项目文件进行上传，文件格式为压缩包； 3. 系统用户填写任务的基本信息，包括任务名称、项目描述等； 4. 系统用户进行数据命名个性化配置，可点击“上传数据配置文件”上传数据命名配置文件； 5. 用户点击弹出框最下方确认的图形按钮，确认创建任务，系统显示“任务创建成功，请您在任务管理模块进行后续操作”。
其他事件流	2a. 用户在上传待检测项目时可选择当前已上传至系统服务器的项目。 4b. 用户可选择配置管理模块中已上传的的历史上传配置文件作为当前任务的配置文件。
后置条件	任务上传至系统服务器，等待用户后续的操作

执行扫描检测任务用例如表4.8所示，用户可以在主页的任务列表中选择扫描任务点击按钮进行执行，系统将会对项目进行隐私数据扫描，扫描完成后系统将进行扫描报告的生成，包括项目中各个隐私数据的信息以及相关分析。

表 4.8: 执行扫描检测任务用例描述

描述项	说明
用例 ID	UC2
用例名称	执行扫描检测任务
参与者	系统用户
用例说明	系统用户选择进行扫描任务的执行，系统进行隐私数据泄露风险检测扫描，并在检测完成后提供给用户检测扫描报告的查看
前置条件	系统用户已经创建了扫描检测任务
基本事件流	1. 系统用户点击左侧的“任务管理”一栏，系统显示任务管理界面，该界面显示当前用户的扫描检测任务列表，包含用户所有已经创建的扫描任务； 2. 系统用户选择某一任务，点击“执行”的图形按钮，系统开始执行该任务，且当前任务条目的状态变为“检测中”； 3. 系统进行隐私数据泄露风险检测、报告生成等流程； 4. 扫描完成后，系统更新当前任务条目的状态为“已完成”；
其他事件流	3a. 系统用户可在扫描未完成的期间选择停止该扫描任务
异常事件流	4a. 项目扫描失败，系统提示错误原因。
后置条件	系统提供扫描报告

审阅扫描分析报告用例如表4.9所示，系统用户/管理员在扫描任务完成后可进行扫描报告的审阅，包括报告的查看，报告结果的标记，系统用户/管理员可以对报告内容的正误进行标记，并上传至系统服务端，最后用户可进行报告的导出下载。

表 4.9: 审阅扫描分析报告用例描述

描述项	说明
用例 ID	UC3
用例名称	审阅扫描分析报告
参与者	系统用户/系统管理员
用例说明	系统用户进行任务的创建以及配置文件的上传
前置条件	系统用户成功登录
基本事件流	1. 系统用户在主页点击左侧的“任务管理”条目，系统跳转至任务管理界面，扫描完成的任务可以点击显示报告相关选项； 2. 系统用户点击“查看报告”，系统显示弹出框，以表格显示扫描检测报告详情，用户可进行查看； 3. 用户点击任务条目“下载”的图形按钮，系统导出检测报告并自动进行浏览器的文件下载；
异常事件流	3b. 任务未完成时用户点击报告按钮，系统提示检测尚未完成。 4b. 报告下载失败，系统提示错误原因。
后置条件	无

用户管理用例如表4.10所示，系统管理员可以在系统的用户管理界面进行用户的管理，包括对用户的创建、用户信息的编辑以及用户的删除等操作。

表 4.10: 用户管理用例描述

描述项	说明
用例 ID	UC5
用例名称	用户管理
参与者	系统管理员
用例说明	系统管理员进行用户的管理
前置条件	系统管理员已经登录
基本事件流	1. 系统管理员点击“用户管理”系统跳转至用户管理界面； 2. 系统显示用户管理界面，展示系统所有用户的列表； 3. 系统管理员对某条用户信息进行编辑或删除的操作； 4. 系统管理员可点击“新增用户”进行系统用户的创建；
其他事件流	无
异常事件流	无
后置条件	无

4.3　系统总体设计

本节将对本文的隐私数据泄露风险检测系统的总体设计进行详细描述，该详细设计建立在上一节对系统的需求分析以及用例分析的结果之上。首先，本节给出了系统的整体框架图，并基于该图对系统的层次结构划分进行解析，介绍了系统的各个模块、模块的功能和模块之间的联系。其次，通过 4+1 视图中除架构图外的四个视图对本文系统的总体设计进行多角度的描述。最后，介绍本系统的数据存储方案以及数据库表的具体设计。

4.3.1　系统架构

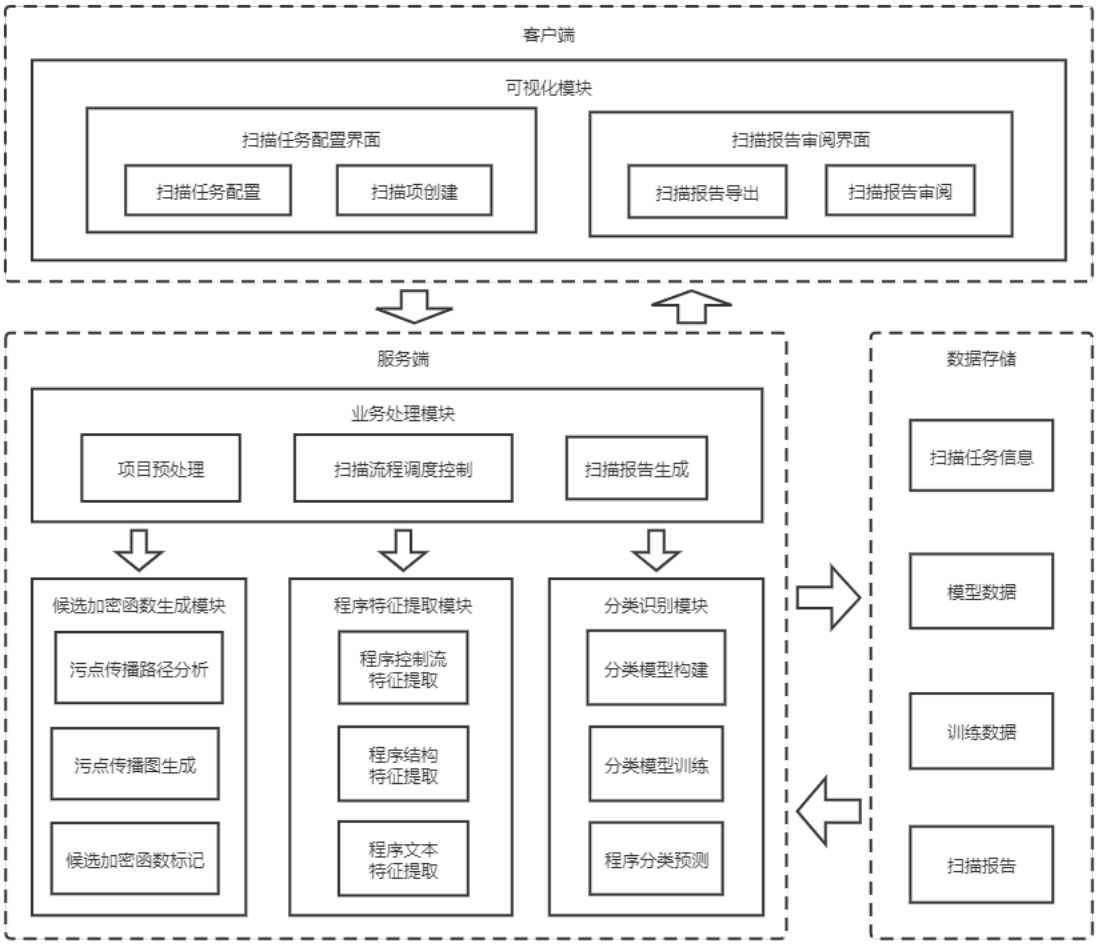

图 4.2: 系统模块图

图4.2给出了本文系统的架构图，系统整体分为三层：客户端、服务端和数据存储端，且由于本文系统以 Web 的形式存在，所以选择了基于 C/S（Client/Server）架构进行设计，其中开发的具体设计模式为 MVC（Model View Controller）。系统具体分为以下几部分：客户端是系统直接与用户交互的窗口，包含了可视化模块、服务端主要是系统功能实现的核心，包含业务处理模块、候选加密函数生成模块、程序特征提取模块以及分类识别模块，数据存储端则为上述所有部分提供数据支撑。

可视化模块中，用户进行隐私数据扫描任务的流程：上传待扫描的项目代码压缩包，并进行任务信息的配置；用户在创建好的任务可选择继续编辑或者

开始执行；任务执行完成后，系统将显示任务已完成的状态提示并提供扫描报告供给用户展示与下载，报告内容包括：隐私数据名称与类型、隐私数据所在位置信息、隐私数据相关代码上下文、隐私数据传播路径、隐私数据是否有泄露风险、隐私数据是否经过匿名处理等。

业务处理模块中，通过 HTTP 请求为可视化模块提供相应业务功能。业务处理模块负责对上传的项目进行预处理，调度控制扫描流程并通过 HTTP 以及 TCP 调用其他模块的服务，扫描流程调度完毕后实现扫描报告的生成。

候选加密函数生成模块，主要是对隐私数据传播图的生成以及其中的候选加密函数的提取。该模块利用污点分析技术，以外部输入变量作为起点进行隐私数据传播的分析，可以得到隐私数据经过的程序语句以及可能泄露的程序位置，但污点分析难以识别清洁函数，咋本系统将其先定位加密函数，故此模块将路径中调用的函数提取出来作为候选加密函数，以供其他模块进一步分析。

程序特征提取模块主要是提取候选加密函数的特征以便分类识别。该模块依靠 JavaParser、Soot 等程序分析框架生成程序的控制流图、抽象语法树以及 token 序列，以提取程序的控制流、结构以及词法特征，并进行聚合以及泛化，得到程序的图表示，其后利用图核函数对程序的图表示进行向量化，得到函数的特征向量。

分类识别模块主要是对函数进行分类识别。该模块利用及其学历的分类算法模型对函数进行分类，输出函数功能的分类结果。

数据存储层是对系统所有流程相关数据导入导出的功能模块的抽象，系统数据主要包括扫描检测任务信息、用户信息、模型数据、训练数据、扫描检测报告等。

从工程设计角度而言，本文的隐私数据泄露风险检测系统前后端分离，前端采用 React 框架及业界成熟的 Bootstrap 组件库，在开发便捷的基础上也保证了高效的渲染。系统组件服务端开发业务部分主要是基于利用当前 Java Web 开发最流行的 Spring Boot 框架进行系统的开发，该框架使得与第三方框架的整合更加友好，在开发时的配置工作也极为简单，具有强大的扩展性且易于上手，又可以能够为系统服务端用户提供 Restful API, 以供外部调用。

4.3.2　4+1 架构视图

4+1 架构视图 [47] 是目前软件架构设计领域的通用标准，从 5 个不同视角——逻辑视图、开发视图、进程视图、物理视图和场景视图来描述一个系统不同维度的结构体系。本小节也将沿用 4+1 视图对系统设计进行描述，其中场景视图针对用户场景，已于上文用例分析一节详细说明，所以本小节不再赘述。

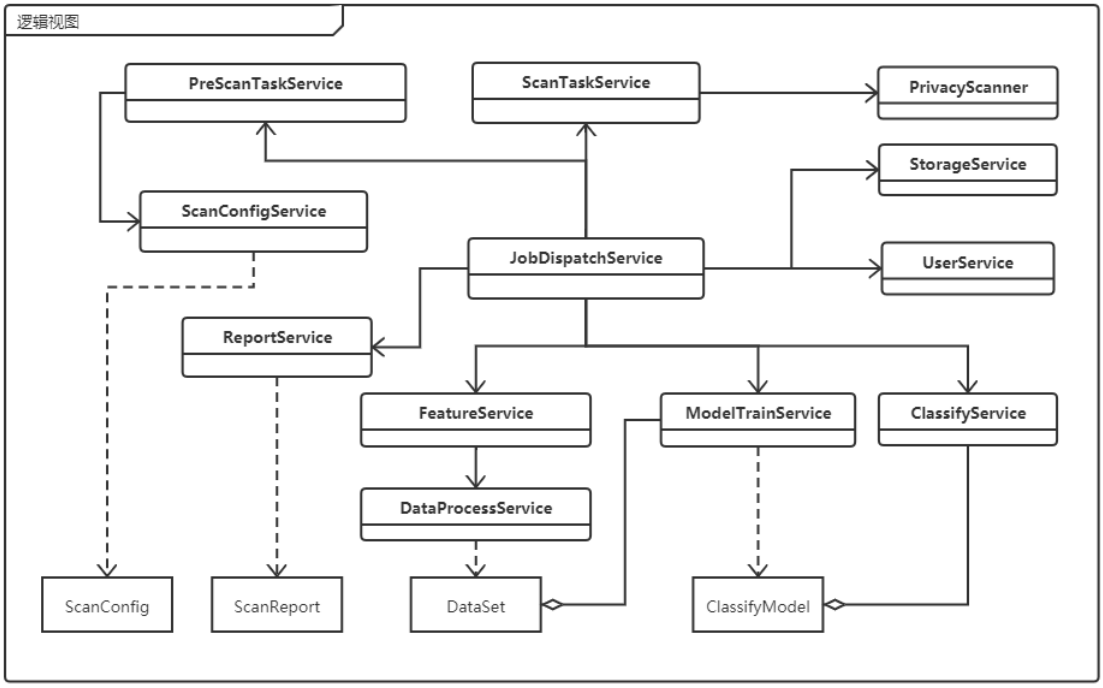

图 4.3: 系统逻辑视图

逻辑视图。为帮助理解系统设计的结构和组织，在系统的分析与设计工作流程中常使用称为逻辑视图的体系结构视图。系统只有一个逻辑视图，它说明了关键用例实现，子系统，程序包和类，这些包含了具有体系结构意义的行为。本系统的逻辑视图如图4.3所示。

JobDispatchService 是系统实现业务调度的重要服务，该服务主要负责隐私数据泄露风险检测系统的所有业务功能的调度、处理和分发的任务，该服务连接沟通了系统的其他模块，使得系统成为一个连通的整体。JobDispatchService 调度的其他模块包括：UserService 提供用户登录注册、身份权限认证等基本服务；PreScanTaskService 下载扫描项目包以及项目包预处理的服务；ScanTaskService 提供项目隐私数据路径扫描的服务；ScanConfigService 提供扫描任务配置的服务，FeatureService 提供对函数特征提取的服务，主要包括对程序图标是的生成，并调用 DataProcessService 进行泛化以及向量化；ModelTrainService 提供对函数分类识别模型进行迭代训练的服务，ClassifyService 提供对函数进行是否为加密函数的识别服务，ReportService 提供扫描报告的生成功能；StorageService 为系统的所有数据存储功能做支撑，提供与数据库以及文件系统交互的功能，负责数据的存储以及导出。

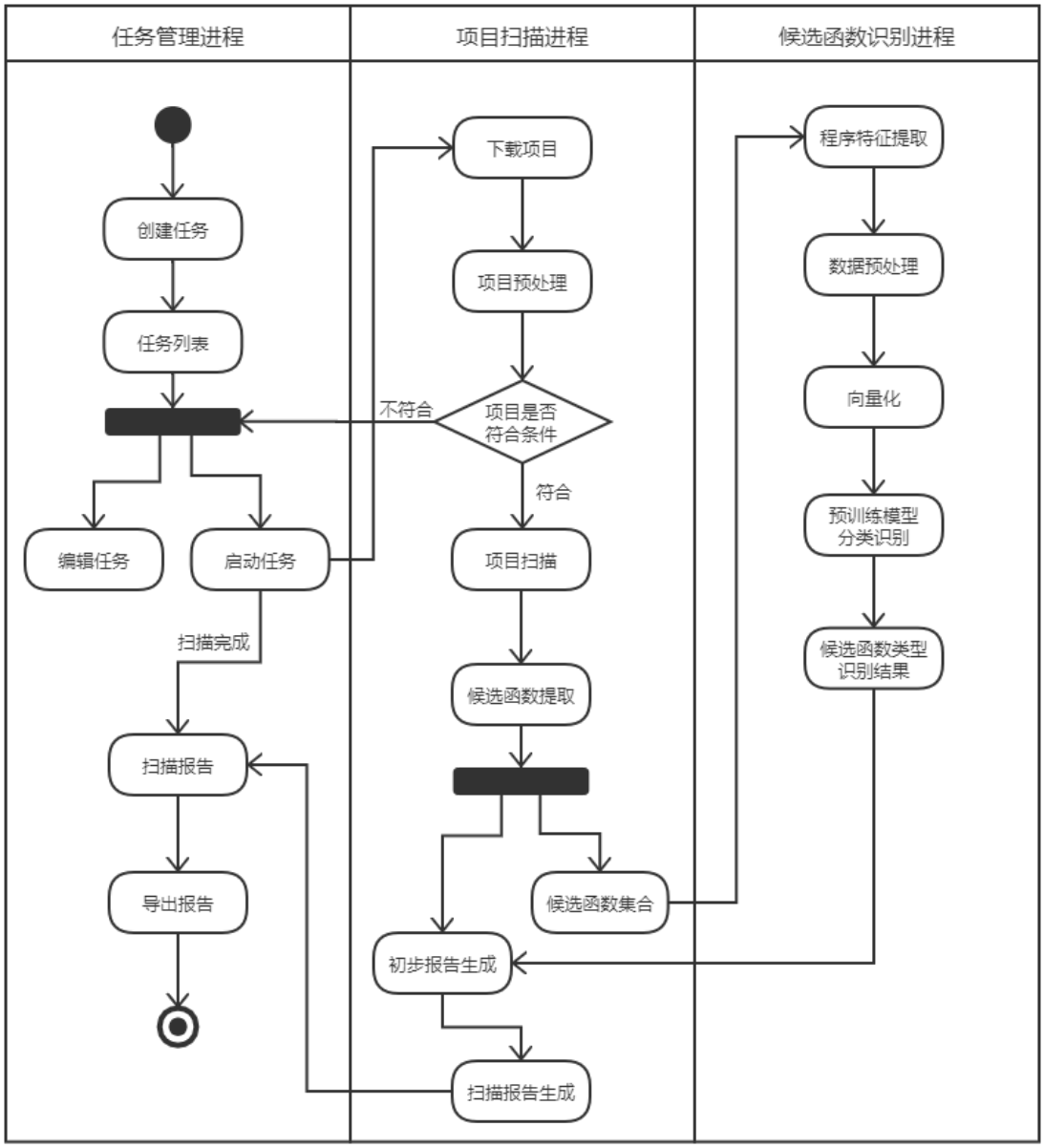
任务管理进程
项目扫描进程
候选函数识别进程
创建任务
任务列表
编辑任务
启动任务
扫描完成
扫描报告
导出报告
下载项目
项目预处理
不符合
项目是否
符合条件
符合
项目扫描
候选函数提取
候选函数集合
初步报告生成
扫描报告生成
程序特征提取
数据预处理
向量化
预训练模型
分类识别
候选函数类型
识别结果

图 4.4: 系统进程视图

进程视图。进程视图处理系统的动态方面的设计，解释系统流程及其通信方式，并着重于描述系统的运行时行为，例如并发及同步行为，解决了并发，分布，集成器，性能和可伸缩性等问题。本文使用 UML 中的活动图来表示进程视图，如图4.4所示，下面分步骤对该图进行解释。

系统用户进入任务管理进程，创建并配置任务；

创建好的任务会被加入到任务列表；

用户可以在列表选择再次编辑任务或者启动任务；

用户启动任务后跳转到项目扫描进程，该进程负责扫描流程的调度；首先，该进程下载待扫描项目并进行预处理，预处理后判断该项目是否符合扫描条件，不符合则退出当前进程回到任务管理进程，符合则进行项目扫描，经扫描得到包含隐私数据传播路径图的初步报告以及候选加密函数的集合；

针对候选加密函数集合还需要进一步处理，此时系统进入候选加密函数识别进程；

首先，经过程序特征提取、数据预处理、泛化以及向量化得到特征向量，其次基于特征向量使用与训练的分类模型进行分类识别，得到候选函数类型识别结果并输出，此时退出当前进程，回到项目扫描进程；

上一步得到的候选函数类型识别结果与项目扫描进程得到的初步报告进行最终的扫描报告生成，输出扫描报告并退出当前进程，回到外层的进程；

此时系统完成了扫描检测的流程，任务状态更新为已完成检测，用户可以在系统查看详细的扫描检测结果报告；

此后，用户还可选择导出下载报告。

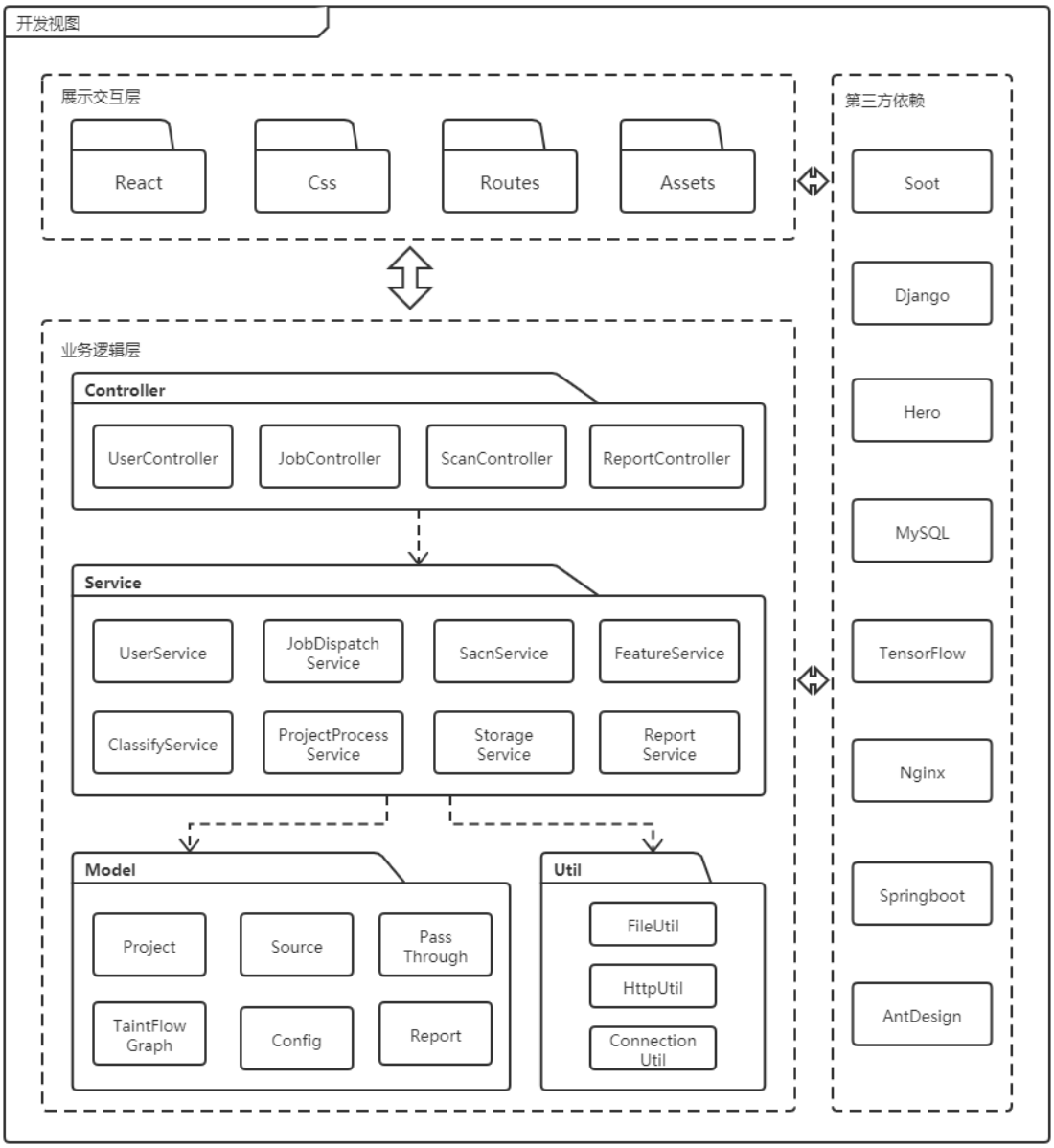

图 4.5: 系统开发视图

开发视图。开发视图描述了从开发者角度看到的程序应用系统，它是所有系统设计或者架构设计的基础视图，代表了系统开发所遵循的结构规范以及约束。图4.5展示了本系统的开发视图，包括三层：展示交互层、业务逻辑层以及第三方依赖服务层对系统结构进行开发者角度的详细描述。

展示交互层主要包括系统与用户交互的页面的相关模板引擎和静态资源，React 包为前端页面逻辑代码所在包，css 为页面样式配置文件包，routes 包为路

由数据目录，assets 包主要存放页面相关静态资源。

业务逻辑层使用了流行的 MVC 模式，M 是指业务模型，V 是指用户界面，C 则是控制器，通过该模式可以实现用户界面和模型的实现代码分离，从而使同一种后台逻辑实现能对应不同的前端页面表现。图中的 Controller 包是本文所有控制器所在包，每一个控制器都对应了一个系统功能单位，负责处理前段传来的对应功能的 HTTP 请求，并在对相应业务进行处理后返回带有数据的响应给前端页面进行展示。Controller 包下有四个具体的 Controller，分别为：UserController，负责处理用户信息相关的请求，如用户登录，权限识别等；JobController 负责扫描任务管理的相关请求，包括人物创建、任务编辑、任务配置等；ScanController 负责启动扫描的请求；ReportController 负责扫描报告相关请求，包括扫描报告的查看以及下载等。Service 包中是系统所有服务的集合，各个负责对控制器下发的业务需求提供具体的实现以及实际的执行。Model 包则包含了系统运行过程中会使用到的数据对象，包括 Project 项目对象、Source 数据变量源对象、TaintFlowGraph 污点传播图对象、Report 扫描报告对象等。Utils 包提供系统通用服务，如文件服务、Http 网络服务、数据库连接服务等。

第三方依赖服务层则是系统依赖的第三方的集合，包括程序分析使用的 Soot，Hero 框架、机器学习依赖的 Tensorflow 框架、负载均衡功能实现依赖的 Nginx 引擎等等。

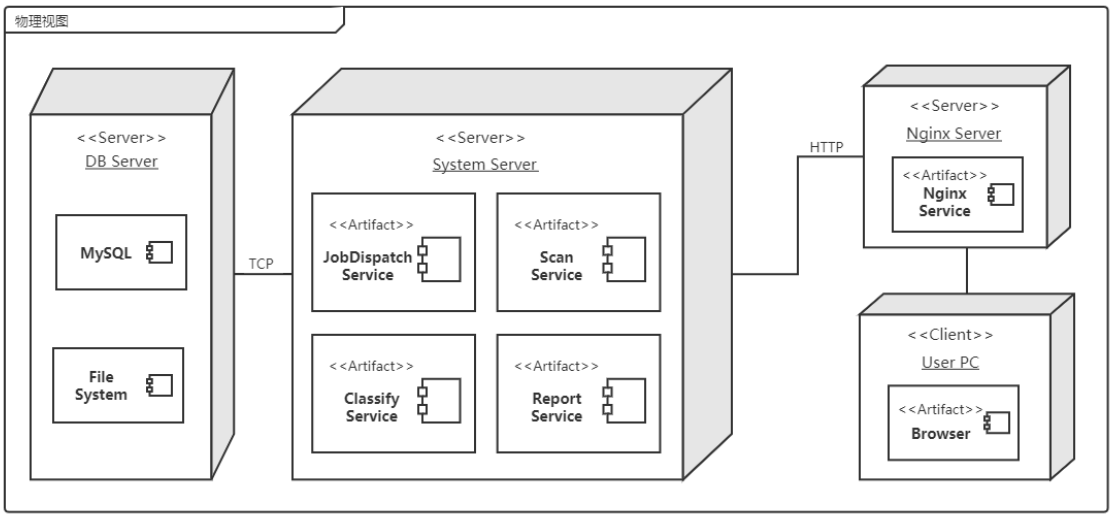

图 4.6: 系统物理视图

物理视图。物理视图从主要描述了系统的硬件组织结构，，反映了系统如何

处理资源利用，应对输入输出以及处理数据等过程，体现了系统在现实世界的存在形式。用户通过浏览器访问系统，在系统中的任一操作都会伴随着向服务端发起 HTTP 请求，该请求会经过 Nginx 服务器进行负载均衡，以保证请求分发到相对空闲的服务器，得到更短的响应时间。应用服务器即服务端上部署了系统的各个服务，包括 JobDispatchService 任务调度服务、ScanService 扫描服务、ClassifyService 分类识别服务以及 ReportService 扫描检测报告服务等。服务端通过使用 Mybait 框架与 Mysql 数据库相连接进行数据存储更新等操作，部分数据如模型参数数据以及训练样本数据等则以文件形式存储在服务器上。

4.3.3　持久化对象设计

本文系统使用 MySQL 数据库以及文件系统对系统使用的数据进行存储，主要存储内容有：用户信息（包括用户名以及登录令牌等）、任务信息、任务配置信息以及扫描检测报告等数据，按照需求分析以及系统设计，定义如下持久化对象。

表 4.11: User 表主要字段

字段名	字段类型	描述
ID	INT(11)	自增编号
Token	VARCHAR(200)	管理员发放的用户 Token，唯一标识以及登录凭证
CreateTime	DATETIME	用户创建时间

表4.11展示了用户信息表 User 表，该表用户存储客户信息。表中 ID 为一个用户的唯一标识，作为与其他表相连接时的外键；Token 为一字符串，用于用户登录，用户在注册系统时系统会随机生成一个令牌以供用户后续登录，该令牌相当于用户的登录密码，区别是不能更改，管理员可以对其进行管理；CreateTime 为该用户的创建时间。

表 4.12: Task 表主要字段

字段名	字段类型	描述
ID	INT(11)	自增编号
UserID	INT(11)	关联用户 ID
Name	VARCHAR(200)	任务名
ProjectPath	VARCHAR(200)	任务项目所在路径
CreateTime	DATETIME	任务创建时间

表4.12展示了任务信息 Task 表，该表存储任务元信息，记录任务元信息保证任务及其扫描结果的唯一性以及可查询性。表中 ID 为一个任务的唯一标识，作为与其他表相连接时的外键；UserID 为用户表 ID，由标识了创建该任务的唯一用户；Name 为任务名，由用户在人物创建时定义，或在编辑任务时更新；ProjectPath 表示任务目标项目文件在服务器上存储的路径；CreateTime 为该任务的创建时间。

表 4.13: TaskConfig 表主要字段

字段名	字段类型	描述
ID	INT(11)	自增编号
ProjectID	INT(11)	关联任务 ID
UserID	INT(11)	关联用户 ID
FilePath	VARCHAR(200)	配置文件所在路径
CreateTime	DATETIME	配置创建时间

表4.13展示了任务配置信息 TaskConfig 表，该表存储任务配置信息，由于一个任务可以有多个配置文件，故为了满足数据库的设计原则中的原子性，将其单独拆分为一个表。表中 ID 为一个任务配置的唯一标识，作为与其他表相连接时的外键；TaskID 为任务表 ID，标识了该配置关联的唯一任务；UserID 为用户表 ID，由标识了创建该任务的唯一用户；FilePath 表示任务配置文件在服务器上存储的路径；CreateTime 为该任务的创建时间。

表 4.14: Node 表主要字段

字段名	字段类型	描述
ID	INT(11)	自增编号
ParentID	INT(11)	父节点 ID
ReportID	INT(11)	关联报告 ID
NodeName	VARCHAR(200)	节点名
FileName	VARCHAR(200)	所在文件名
LineNum	INT(11)	所在行数
Type	VARCHAR(200)	节点类型

表4.14展示了隐私数据传播路径节点信息 Node 表，该表记录了隐私数据传播的各个程序节点，表中 ID 为一个节点的唯一标识，作为与其他表相连接时的外键；ParentID 为父节点 ID，标识了该改节点在传播图中的上一节点；TaskID 为任务表 ID，标识了该配置关联的唯一任务；NodeName 表示该节点的变量名或调用函数名；FileName 为该节点所在文件；LineNum 为该节点程序语句在文件中所在的行数；Type 为节点类型，标识了节点的程序语句类型，如赋值语句、变量定义、函数调用等。

表 4.15: Report 表主要字段

字段名	字段类型	描述
ID	INT(11)	自增编号
TaskID	INT(11)	关联任务 ID
TimeCost	INT(11)	任务耗时
CreateTime	DATETIME	报告生成时间
FilePath	VARCHAR(200)	报告文件路径

图4.15展示了扫描报告信息 Report 表，由于扫描报告为服务端在程序中生成文件，该表不存储所有的报告信息，而是报告的外部相关信息。表中 ID 为一个扫描报告的唯一标识，作为与其他表相连接时的外键；TaskID 为任务表 ID，标识了该扫描报告关联的唯一任务；TimeCost 为扫描耗时；CreateTime 为该扫描报告的生成时间；FilePath 表示扫描报告文件在服务器上存储的路径。

4.4　系统模块设计

上文已通过对系统的需求以及多角度的视图深入地阐述分析了本文隐私数据泄露风险检测系统的整体开发架构体系，并将系统分解成多个不同的服务模块。此节将就各模块从实际业务流程进行更深层细致的描述。。

4.4.1　可视化模块

可视化模块负责系统与用户交互的页面的展示逻辑，在界面接收用户的操作并想服务端发起 HTTP 请求，获得服务端的响应后利用返回的数据更新页面信息，给予用户反馈。该模块主要提供了对外功能的展示，功能包括扫描检测任务的相关管理操作（创建任务、查看任务、启动任务、查看报告等）。

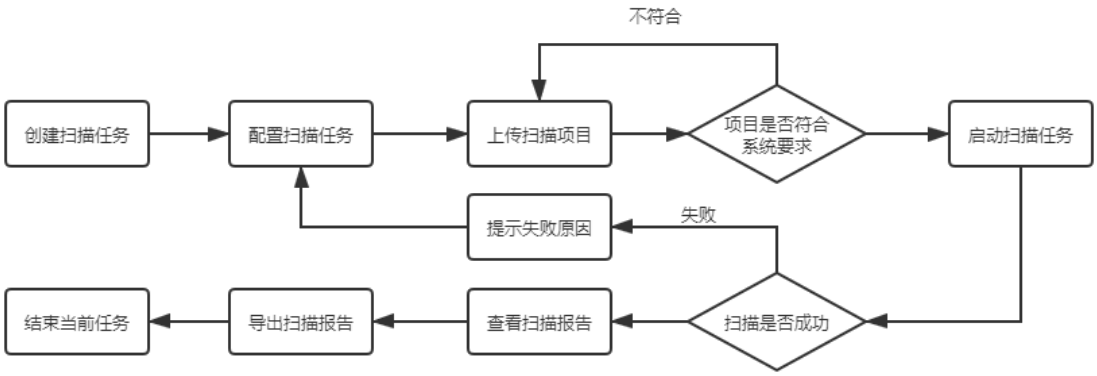

图 4.7: 可视化模块流程图

可视化模块总体流程图如图4.7所示，用户点击新建任务按钮进入新建任务流程后，最重要的是对待检测项目的上传，用户点击上传按钮后，可以选择上传本地文件也可以选择项目管理中的历史上传项目文件。上传项目文件包时，系统会对其进行检查，若不符合系统要求会提示用户重新上传。其后，用户填写任务基本信息以及扫描选项简单配置，包括项目名称、相关描述、项目配置文件等。任务创建成功后，该任务会加入到任务列表。在任务列表中用户可以对任一任务条目进行操作，可以对任务信息进行编辑，也可以直接开始执行该条任务。任务执行完成后用户同样可在任务条目中点击查看报告查看扫描检测的结果，还可以再报告详细界面进行报告的导出。上述内容即为一次完整的隐私数据泄露风险检测流程，流程中的每一步骤系统均会给予详细的指引，对于用户的错误操作会有相应的弹窗反馈，协助用户能更快速对系统功能上手。

4.4.2　候选加密函数生成模块

候选加密函数生成模块，主要是对隐私数据传播图的生成以及其中的候选加密函数的提取。该模块利用污点分析技术，以外部输入变量作为起点进行隐私数据传播的分析，可以得到隐私数据经过的程序语句以及可能泄露的程序位置，但污点分析难以识别清洁函数，本系统将其先定位加密函数，故此模块将路径中调用的函数提取出来作为候选加密函数，以供其他模块进一步分析。

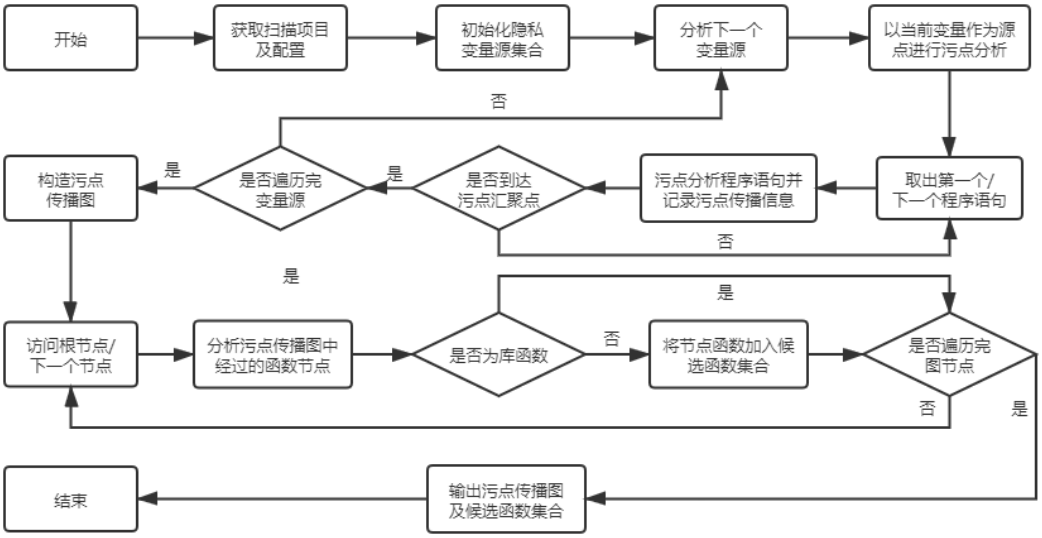

图 4.8: 候选加密函数生成模块流程图

候选加密函数生成模块总体流程图如图4.8所示，首先通过数据库及文件系统获取扫描系统机器配置，其后根据扫描配置初始化隐私变量源集合，对于变量源集合中的每一个变量进行循环处理：对于当前变量，将其作为源点进行污点分析，该过程包括对程序语句的迭代分析，记录分析的程序语句信息以及当前的污点传播信息，判断是否到达污点汇聚点；遍历完成变量源后由前述步骤得到的污点传播信息构建污点传播图；污点传播图构建完成后对该图进行遍历，分析该图中的所有节点，判断节点是否为候选加密函数，若符合则将其加入候选加密函数集合；遍历完成污点传播图节点后输出污点传播图（隐私数据变量传播路径图）以及候选加密函数集合，至此，候选加密函数生成流程完成。

4.4.3　业务处理模块

业务处理模块中，通过 HTTP 请求为可视化模块提供相应业务功能。业务处理模块负责对上传的项目进行预处理，调度控制扫描流程并通过 HTTP 以及 TCP 调用其他模块的服务，扫描流程调度完毕后实现扫描报告的生成。

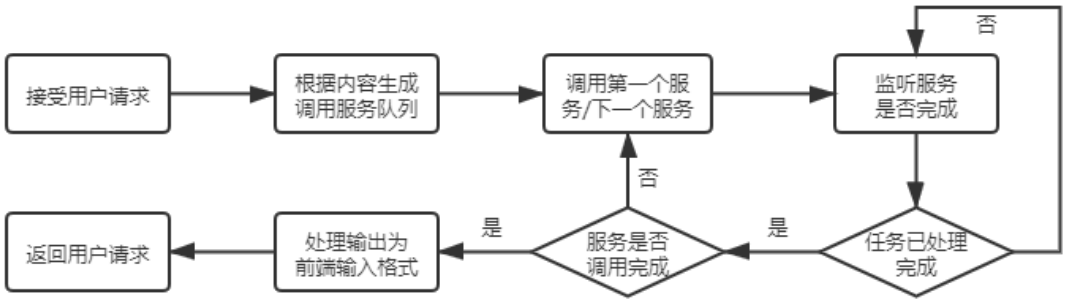

图 4.9: 业务处理模块流程图

业务处理模块总体流程图如图4.9所示，该模块接受用户的请求，根据请求内容生成需要调用的服务队列，根据队列顺序对服务依次进行调用，监听服务是否完成，服务完成后对于每一个服务的输出进行处理并作为下一个服务的输入，当队列中的服务调用完成后，将输出处理为前端所接受的格式并包装秤 HTTP 响应，返回用户请求，至此，一次业务处理流程完成。

4.4.4　程序特征提取模块

程序特征提取模块主要是提取候选加密函数的特征以便分类识别。该模块依靠 JavaParser、Soot 等程序分析框架生成程序的控制流图、抽象语法树以及 token 序列，以提取程序的控制流、结构以及词法特征，并进行聚合以及泛化，得到程序的图表示，其后利用图核函数对程序的图表示进行向量化，得到函数的特征向量

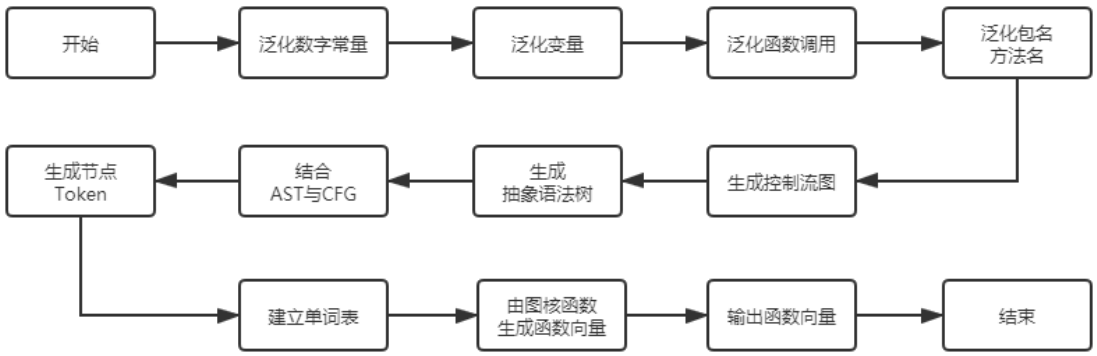

图 4.10: 程序特征提取模块流程图

程序特征提取模块总体流程图如图4.10所示，该模块接受函数体并将其处理为特征向量。首先，对于函数中的数据子长量进行泛化，去除冗余信息以免影响后续分类；其次对变量、函数调用、方法名等一次进行泛化处理；接下来对于函数生成其控制流图、抽象语法树，并将其结合，生成函数结构图，对于图中的每个节点的程序语句信息生成 Token，对于图中所有的 Token 内容生成单词表，最后由图核函数生成函数图表示的向量，输出函数的特征向量，至此，程序特征提取模块结束。

4.5　本章小结

本章首先对本文的隐私数据泄露风险检测系统给出了一个系统的整体概述，介绍了本文系统的总体框架结构；其次，分析并给出了本文系统的两种功能/非功能需求，并根据需求设计出了系统用例，以表格的形式做出了详细描述；随后，以 4+1 视图的形式对系统的总体设计和架构进行了说明；其后对本系统的持久化数据对象的设计做出了说明；最后对系统的各个模块进行了模块设计的描述，阐述了各个模块的功能职责以及简要的流程。

第五章　详细设计与实现

本章主要介绍本文隐私数据泄露风险检测系统的详细设计与实现。根据第四章进行的需求分析与概要设计，本系统被划分为共五个模块，其中可视化模块属于前端层，主要负责监听用户请求并将请求传递到后方以及接受后端传输的请求响应结果，涉及代码逻辑较少，故本章不涉及该模块的详细设计实现，将对系统其余四个模块进行详细介绍。本章以各模块功能的核心类图以及关键代码对系统各部分的实现做出详细介绍，部分功能辅以时序图描述各功能组件相互调用的顺序过程，并以截图的形式展示了本文系统的核心功能页面。

5.1　业务处理模块

业务处理模块主要负责接收扫描检测业务相关的请求并调度系统的其它服务完成扫描的全部流程。该模块的功能首先是任务管理与分发，包含了对于用户创建任务的管理以及任务启动后的调度；其次是扫描主流程开始前的扫描项目预处理，包含了扫描入口配置以及扫描环境配置等；最后是扫描流程完成后扫描报告生成的功能，下面将对这三部分分别进行详细描述。

5.1.1　任务管理与分发的设计与实现

任务管理与分发部分主要负责与任务相关的操作以及对扫描流程的调度功能。该部分实现逻辑主要集中在对人物的管理，包括接收用户创建任务的请求并存储任务信息以及用户上传的项目至系统服务器，在任务创建完成后接收用户编辑任务信息的请求并更新数据库中的数据，以及在接收用户启动任务的请求后调度项目预处理服务、扫描服务、分类服务、报告服务等服务完成完整的扫描流程。

任务管理与分发部分核心类图如图5.1所示，JobDispatchController 控制器主要提供任务相关的功能，包括创建任务、执行任务、停止任务、更新任务信息、获取任务状态以及获取扫描任务结果等功能。JobDispatchService 承担任务创建、更新以及获取任务状态的主要逻辑，并调度 ProjectProcessService、ScanServicce 以及 ClassifyService 实现执行扫描任务的功能，且调度 ReportService 实现扫描报告生成的功能。JobDispatchService 通过调用 StorageService 与系统数据库交互以实现任务的创建以及更新；任务状态的获取则通过查询扫描结果是否生成来实现；对于各个服务的调度则通过将流程中各服务加入到等待的任务队列中，并监听该异步队列中个任务的状态，当监听到新消息时，获取当前的中间结果并

作为回调函数的输入继续下一步骤。ProjectProcessService 主要提供对待检测项目的预处理服务，包括对项目文件的读取、项目数据配置、项目扫描入口配置、项目扫描环境配置等；ScanService 主要提供基于污点分析的隐私数据变量传播路径的生成以及候选加密函数的提取；ClassifyService 服务则主要提供对候选加密函数的分类识别；ReportService 主要提供报告生成、报告文件导出的服务，以上被 JobDispatchService 调度的服务都将在本章余下小节进行详细介绍。

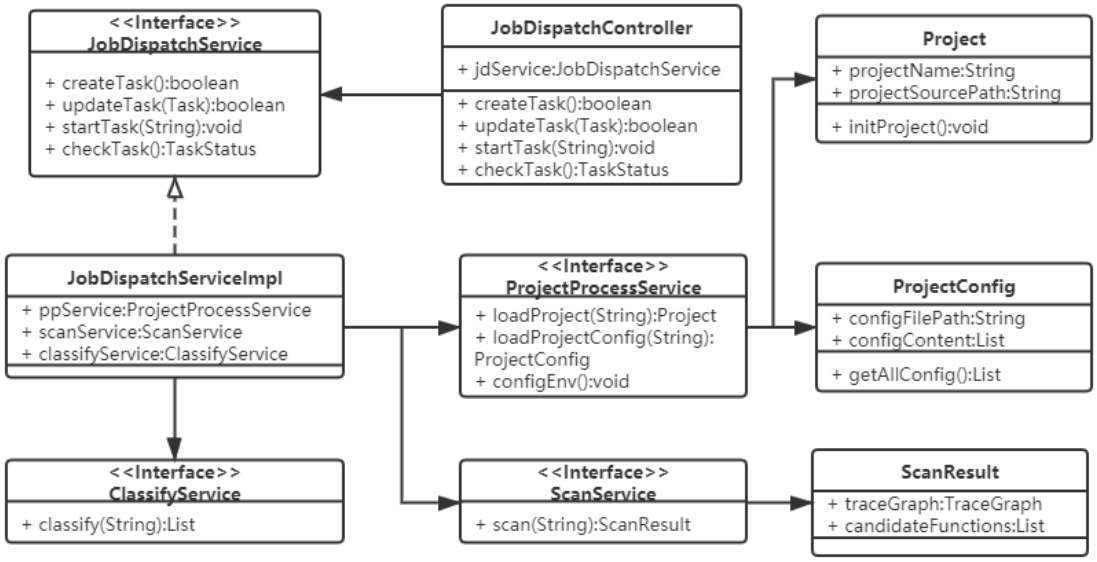

图 5.1: 任务管理与分发相关核心类图

任务管理与分发时序图如图5.2所示，JobDispatchController 负责接收用户的请求，接收到创建任务、更新任务、查看任务状态的请求时，JobDispatchController 调用 JobDispatchService 的 createTask、updateTask、checkTask 方法进行任务创建并返回创建结果、更新任务信息并返回更新结果、查看任务状态并返回任务状态信息。当接收到执行任务请求时，JobDispatchController 调用 JobDispatchService 的 startTask 方法启动整个漏洞扫流程。JobDispatchService 首先会调用 ProjectProcessService 的 loadProject 方法进行待扫描项目的下载以及预处理配置，返回项目 Project 对象；随后调用 ScanService 的 scan 方法进行项目的隐私传播路径扫描，返回隐私数据传播图以及候选加密函数集合；其次调用 ClassifyService 进行候选加密函数的分类，返回候选加密函数的识别结果；最后调用 ReportService 进行扫描报告的生成并存储至数据库。

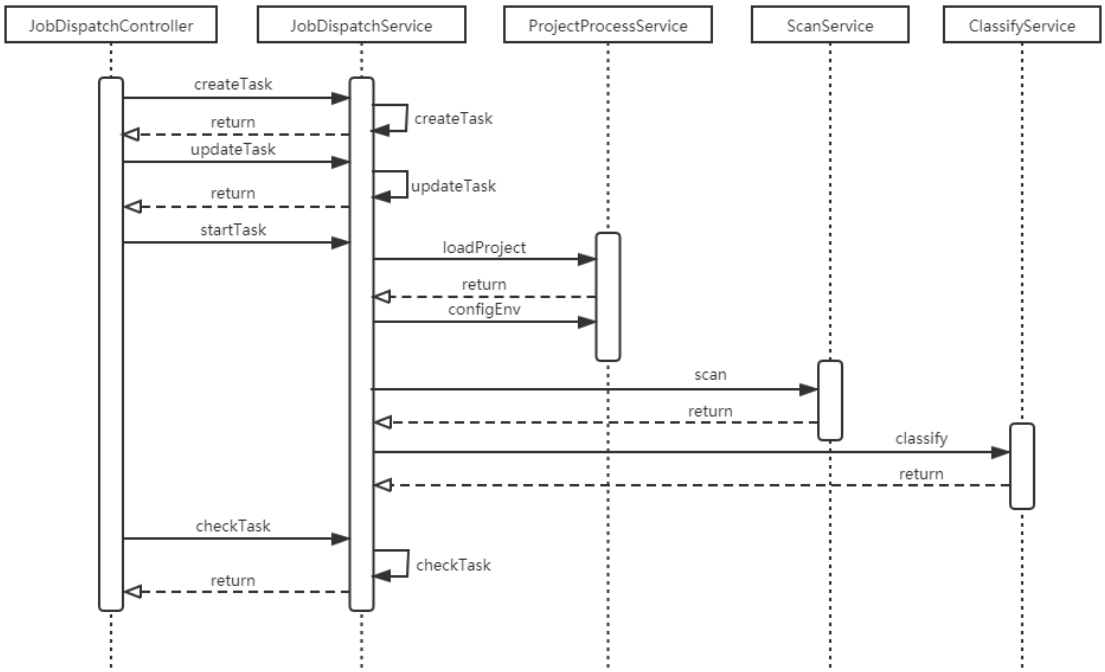

图 5.2: 任务管理与分发时序图

5.1.2　扫描项目预处理的设计与实现

在对项目进行扫描之前，业务处理模块会先对项目进行过预处理，为后续的扫描服务提供包含完整项目信息的项目对象 Project。该部分功能包括将项目文件导入系统并转化为类对象、导入用户配置文件并转化为具体的配置对象、根据配置定位扫描入口得到扫描入口 EntryPoint 的集合、最后是对扫描环境的配置，包括污点分析依赖框架的所有相关配置等等。

项目预处理功能部分的核心类图如图5.3所示，ProjectProcessService 为预处理服务对外的接口，对外提供 loadProject 方法，对外返回处理好的 Project 对象。loadProject 方法的实现首先是将项目文件加载进系统内，随后调用 ConfigHandler 进行扫描配置的加载并调用 EntryPointHandler 进行扫描入口的获取。配置处理器 ConfigHandler 以及扫描入口处理器 EntryPointHandler 均继承自抽象类处理器 Handler，该类有唯一方法 handle 表示该处理器的目标任务。ConfigHandler 处理器由于需要处理用户配置以及系统配置，所以拥有两个处理方法，handleProjectConfig 以及 handleEnvConfig。

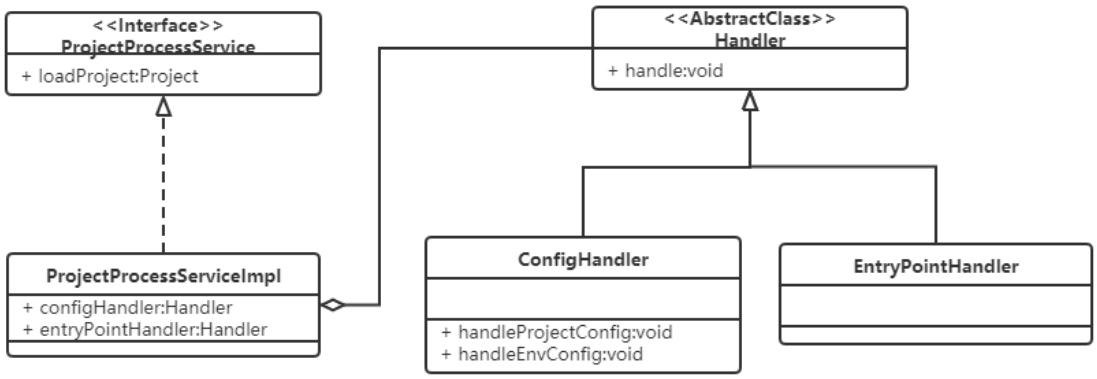

图 5.3: 扫描项目预处理相关核心类图

下文代码展示了项目预处理部分的核心实现，loadProject 方法中首先是通过项目的存储路径，通过 Soot 将项目类文件加载进系统内存中，并创建一个空的 Project 对象，将类文件对象集合加入其中；其后调用 ConfigHandler 的 handleProjectConfig 方法将项目扫描配置加载进系统内获得配置对象 Config；其后以 Project 对象以及 Config 对象作为参数，调用 EntryPointHandler 的 handle 方法生成扫描的入口集合，加入到 Project 对象中，至此完成对项目的预处理。

```
1  private void loadProject() {
2      //清除之前有可能的Soot instance
3      G.reset();
4      //设置Soot相关配置
5      initSoot();
6      //加载Sink配置
7      loadSinkConfig();
8      projectClasses.addAll(Scene.v().
9          getApplicationClasses());
10     projectClassNames = new ArrayList<>();
11     for (SootClass sc : projectClasses) {
12         projectClassNames.add(sc.getName());
13     }
14     logger.info("开始扫描外部输入");
15     EntryPointHandler.initialEntryPoints(this);
16     logger.info("结束扫描外部输入");
17     if (this.getEntryPointList() == null) {
18         logger.warn("本项目无可发现的外部输入！");
19         return;
20     }
21     List<SootMethod> entryPointMethods =
```

```
22          new ArrayList<>();
23      for (EntryPoint entryPoint : entryPointList) {
24          entryPointMethods.add(entryPoint.getMethod());
25      }
26      Scene.v().setEntryPoints(entryPointMethods);
27      //启用CHA,生成CallGraph
28      logger.info("开始生成Call Graph");
29      CHATransformer.v().transform();
30      logger.info("生成Call Graph完成");
31  }
```

5.1.3　相关系统页面展示

系统主界面如图5.4所示，左侧为功能栏，其中包括首页、任务管理、项目管理、配置管理和关于；界面上方分别设有搜索栏、用户信息栏和注销登录按钮；界面中心则为创建项目按钮，用户可以直接进行扫描检测任务的创建。

任务创建界面如5.5所示，界面上用户需要输入的信息为任务名称、任务描述、任务配置文件以及扫描项目；用户将信息输入完成后点击最下方的确认按钮即可完成任务的创建。

任务管理界面如图5.6所示，系统中显示了该用户所关联的全部任务，每行任务信息主要包括：任务姓名、任务描述、创建的时间以及任务状态;用户通过直接点击对每项任务的编辑、删除以及开始操作按钮来完成对每项任务的操作。

图 5.4: 系统主页面

图 5.5: 任务创建页面

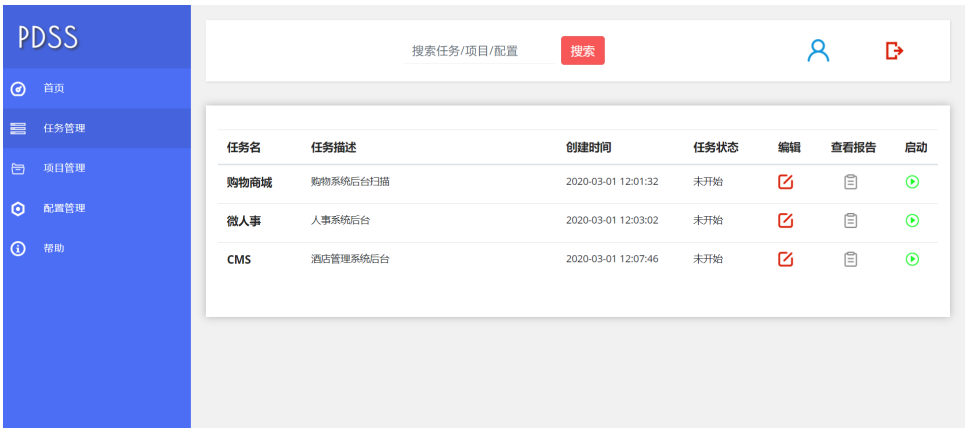

图 5.6: 任务管理页面

5.1.4　扫描报告生成的设计与实现

在调度完成扫描服务的进程后，业务处理模块负责综合各个服务的处理结果生成最终的扫描报告，该部分功能实现的核心类图如图5.7所示，其中 ReportLogic 为扫描报告服务入口逻辑，主要有获取在线报告内容和下载报告文件两个方法，通过调用并处理 ReportService 提供的系统服务结果来实现。ReportService 提供了扫描检测报告相关的系统服务，主要包含生成报告、获取报告、保存报告等功能。ReportDao 负责报告服务的数据连接部分，提供数据库报告数据的增改查服务。MySQLUtil 为封装好的 MySQL 接口，为系统提供统一访问 MySQL 数据库功能。

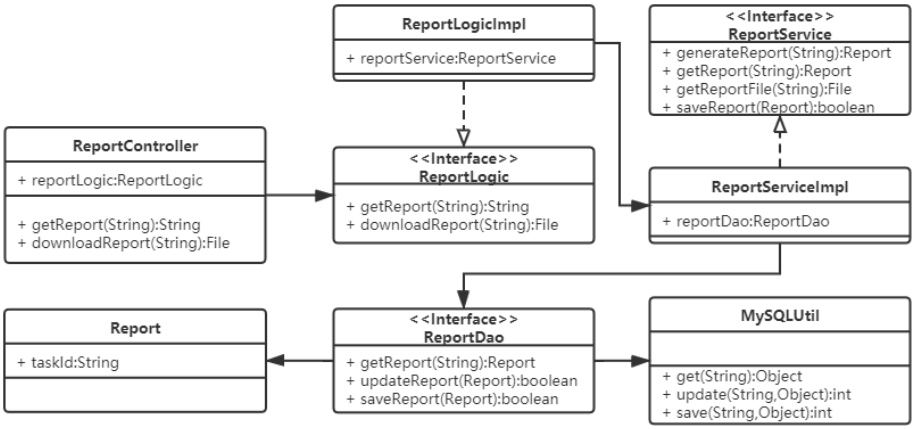

图 5.7: 扫描报告功能相关核心类图

获取报告时序图如图5.8所示。ReportController 接收前端模块发来的扫描检测报告相关的请求，对外部提供了获取报告以及下载报告的功能接口，其中具体的逻辑实现在 ReportLogic 类中。调用获取报告接口时，ReportLogic 调用 ReportService 中的 getReport 方法，该方法返回报告数据。ReportService 首先查询对应人物的信息，若数据库中存在任务报告的信息，则直接读取数据库中的报告路径，将路径中的文件内容读取后返回；否则，调用 generateReport 方法生成报告，生成完成后存储报告文件并更新数据库条目，同时返回报告内容。

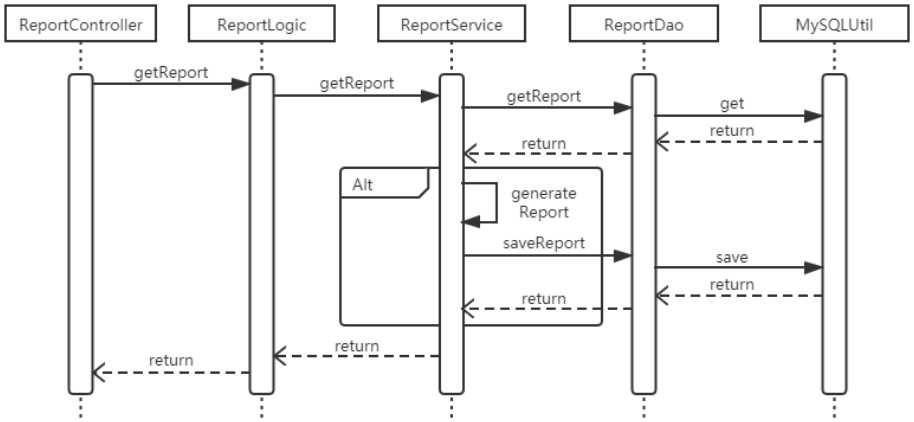

图 5.8: 获取扫描报告时序图

下面的代码展示了 ReportService 的 generateReport 方法的核心代码，首先调用任务服务获取原始的任务结果，其次生成当前报告对应的存储路径，接着根

据原始结果进行再组织，并将报告导出成 JSON 文件格式，存储至服务器，最后保存报告文件的路径等相关信息到数据库中。

```
public void generateReport(String taskId) {
    String taskResult = taskService.getTaskResult(taskId);
    //基于任务信息生成报告存储路径
    String reportPath = generateReportPath(taskId, Consts.REPORT_NAME);
    //生成报告JSON文件并上传服务器
    FileUtil.writeDataToJsonFile(taskResult, reportPath
    + REPORT_JSON_UPLOAD_PATH, REPORT_JSON_FILENAME);
    //调用保存报告方法将报告文件信息存储到数据库
    saveReport(taskId, reportPath);
    ……
}
```

5.1.5　相关系统界面展示

如图5.9为系统的扫描检测报告展示界面，系统以表格的形式展示了扫描的具体结果，若要查看没想数据的具体传播路径还可点击表格中的查看按钮进行查看，具体路径界面如图5.10所示，关键语句及变量标红以便用户查看，标绿的函数方法则为检测识别出的匿名清洁函数，将隐私变量进行了加密处理。

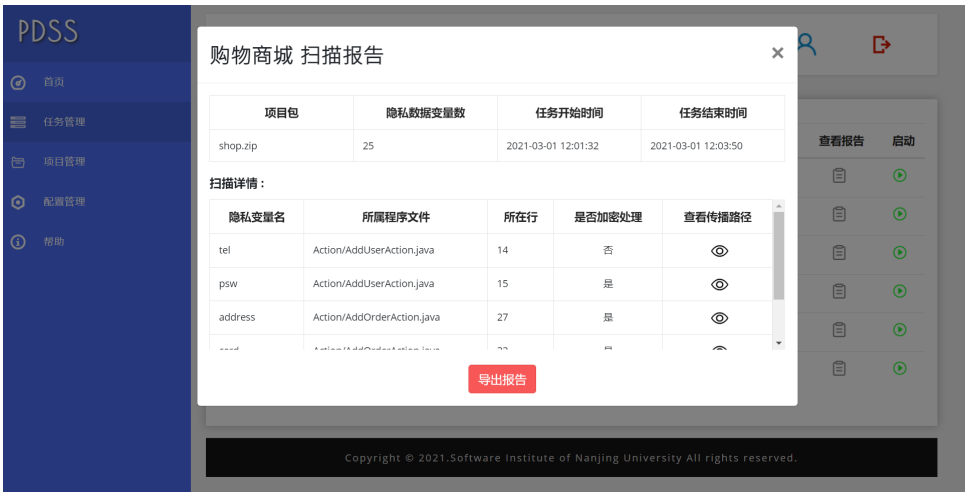

图 5.9: 扫描结果页面

图 5.10: 具体路径页面

5.2　候选加密函数生成模块

候选加密函数生成模块主要负责以项目的扫描入口 EntryPoint 集合即外部输入隐私数据集合作为起点，逐个进行污点分析获得隐私数据在项目程序中的传播路径，生成污点传播路径图，并在图中提取调用的项目函数，作为候选的加密函数集合，后续对其进行分类识别以判断隐私数据是否在有泄露风险的操作前进行加密处理。故本节如上所述将候选加密函数生成模块分为两个部分：污点传播路径分析以及候选加密函数提取进行详细介绍。

5.2.1　污点传播路径分析的设计与实现

污点传播路径分析部分主要依赖 Soot 框架记性程序流分析，其核心类图如图5.11所示，其中 Transformer 以及 SceneTransformer 为 Soot 的原生类，TanitTransformer 继承自 SceneTransformer，是作用于全局的过程间的分析变换器；TaintAnalysis 中包含了污点分析过程中的流变换函数的实现，继承了 Soot 中原生的前向分析器 ForwardAnalysis；TaintAnalysisRunner 是污点传播分析的启动器，设计为单例模式，同一时间只能运行一个污点分析器。

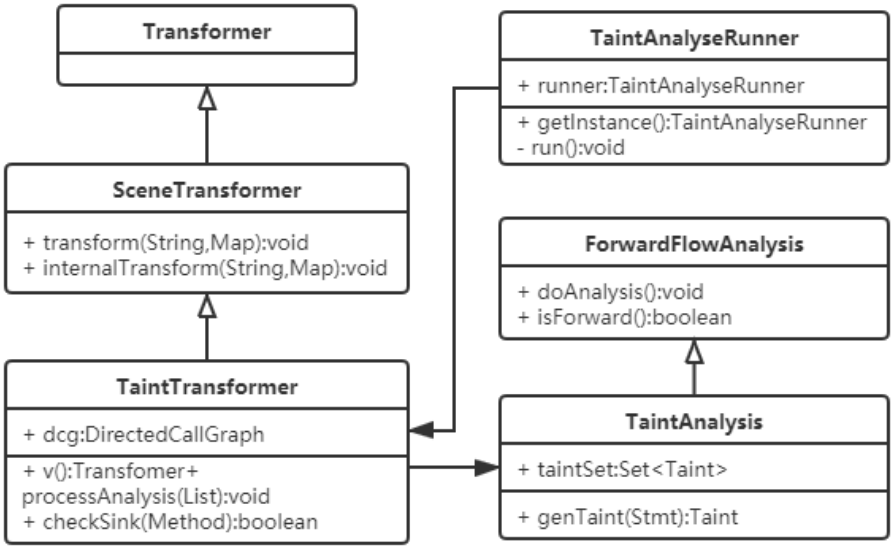

图 5.11: 污点传播路径分析类图

污点传播路径分析得到的污点传播图的相关类图如下图5.12所示。TaintGraph 为污点传播图，由点集合 Vertexs 以及编辑和 Edges 组成，提供添加点的方法 addVertex、添加边的方法 addEdge，以及判断点是否存在图中的方法 contains；点对象 Vertex 为污点类 Taint 的父类，而源点类 Source 以及污点汇聚点类 Sink 在位 Taint 类的子类；而 Edge 自包含两个成员变量 from 和 to，均为 Vertex 对象。

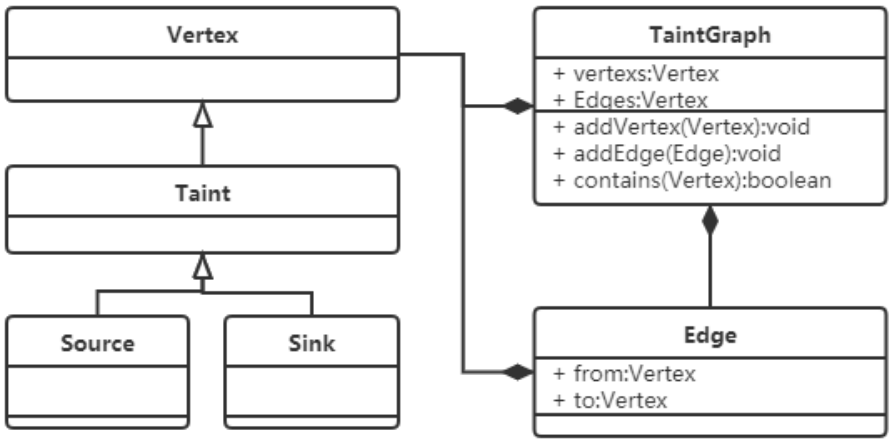

图 5.12: 污点传播图类图

下文代码展示了污点传播路径分析的核心实现，主要是利用流函数，在程序流分析时判断当前变量是否能传播到后文中，并记录当前的污点传播信息。

```
protected void internalTransform(String phaseName,
    Map<String, String> options) {
    //整体的CallGraph
    CallGraph cg = Scene.v().getCallGraph();
    //过滤javalib方法以及init方法
    SootMethodFilter filter = new SootMethodFilter();
    Iterator<MethodOrMethodContext> heads =
        cg.sourceMethods();
    List<SootMethod> headMethods = new ArrayList<>();
    while (heads.hasNext()) {
        headMethods.add(heads.next().method());
    }
    Iterator<SootMethod> headMethodsIterator =
        headMethods.iterator();
    //得到过滤后的调用图
    DirectedCallGraph dcg = new
        DirectedCallGraph(cg, filter,
        headMethodsIterator, true);
    this.dcg = dcg;
    //处理当前Project的entryPointList中
    //跟项目方法无调用关系的成员
    //processNonCallEntry(headMethods);
    for (EntryPoint entryPoint :
        Project.getCurrentProject().getEntryPointList()){
        for (Source src : entryPoint.getSources()) {
            generateFlow(entryPoint.getMethod(), src);
        }
    }
}
```

5.2.2　候选加密函数提取的设计与实现

候选加密函数提取部分基于污点传播图进行处理，其核心实现如下代码所示，对图中的顶点进行遍历，若当前顶点为函数调用，则判断当前被调用的函数是否为 Java 库函数或外部依赖库函数，若是项目函数则将其加入到候选函数的集合中。

```
public List<Function> retrieveFunction(){
    ...
    if (stmt instanceof InvokeStmt) {
            List<Integer>list=getTaintParamIndex(stmt.getInvokeExpr());
            if (!list.isEmpty()) {
                boolean isLib = isLibMethod(stmt.getInvokeExpr()
```

```
                .getMethodRef().getDeclaringClass().getName(),
                stmt.getInvokeExpr().getMethodRef().getSubSignature()
                .toString());
                PassThrough passThrough = new PassThrough
                (className, methodSig,lineNumber,
                stmt.getInvokeExpr().getMethodRef().getSubSignature()
                .toString(),stmt,isLib,list,
                retrieveBaseClassName(stmt.getInvokeExpr()));
                passThroughs.add(passThrough);
                if(stmt.getInvokeExpr().getMethodRef().getName()
                .equals("<init>")){
                    JSpecialInvokeExpr specialInvokeExpr=
                        (JSpecialInvokeExpr)stmt.getInvokeExpr();
                    taintLocalSet.add((Local) specialInvokeExpr.getBase());
                    Taint taint = genTaint(stmt, lineNumber,
                    (Local) specialInvokeExpr.getBase());
                    returnVar.add(taint);
                }
            }
        }
        ...
}
```

5.3　程序特征提取模块

程序特征提取模块提供了对于候选加密函数的程序特征正的提取的功能，最大限度保留函数的程序特征以便使后续的分类识别效率最大化。

5.3.1　程序特征图提取的设计与实现

程序特征图提取部分生成函数体的图表示,其核心类图如图5.13所示。MethodGraphGenerator 为程序图生成其，为特征服务 FeatureService 所调用，提供了 createACGraph 方法，即生成生成 ACGraph。此处的 ACGraph 为本文所采取的程序图表示，融合了程序控制流图 CFG 以及程序的抽象语法树 AST，并对各个节点进行 Token 序列的转换，得到了最终的程序图表示，最大限度保留程序的控制、结构以及文本特征。MethodGraphGenerator 调用 ControlFlowGraphBuilder 进行控制流图的生成并表用 AbstractSyntaxTreeBuilder 进行抽象语法树的生成，融合后调用 Tokenizer 类的 tokenize 方法对图节点进行处理，最终生成 ACGraph，并以 json 文件作为暂时的存储形式输出到当前工作目录。

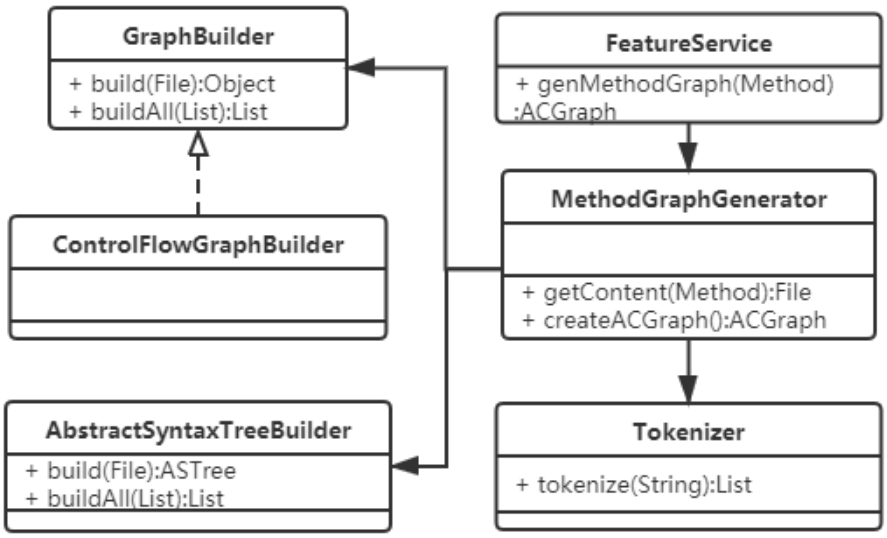

图 5.13: 程序特征图提取相关核心类图

程序特征图提取的时序图如图5.14所示，FeatureService 调用 MethodGenerator 类的 createACGraph 方法进行程序图的生成，MethodGenerator 依次调用 AbstractSyntaxTreeBuilder、ControlFlowGraphBuilder 以及 Tokenizer 的生成方法并获取返回对象，进行处理后返回最终的程序图对象。

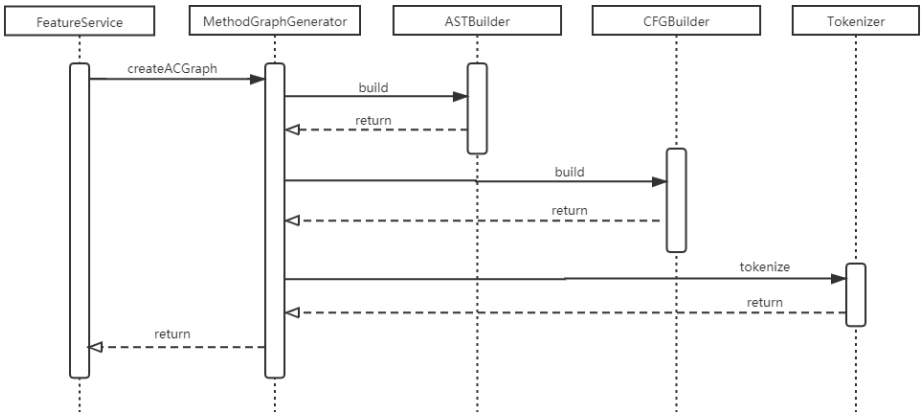

图 5.14: 程序特征图提取时序图

程序特征图提取的核心实现如下代码所示，调用相关服务生成各个图以后依次遍历 CFG 的各个节点同时将其关联对应的 AST 节点，遍历完成后循环调用 Tokenize 服务对图中各节点进行处理，得到完整的 ACGraph 对象，并导出到

临时的 json 文件，以供后续 python 语言实现的向量化功能实现部分进行使用。

```
public static void retrieveMethodFromClassDDG(
    ACDGraph ACDGraph,DataDependenceGraph classDDG,
    String methodName) throws FileNotFoundException{
    Iterator<PDNode> iterator = classDDG.
        allVerticesIterator();
    CFNode methodCFGRoot = null;
    PDNode methodDDGRoot = null;
    for (CFNode node : classDDG.getCFG().
        getAllMethodEntries()) {
        if (node.getProperty("name").equals(methodName)){
        ACDGraph.name +=
            (String) node.getProperty("name");
        methodCFGRoot = node;
        methodDDGRoot =
            (PDNode) node.getProperty("pdnode");
        break;
        }
    }
    ACDGraph.CFGRoot = methodCFGRoot;
    ACDGraph.DDGRoot = methodDDGRoot;
    bfsCFG(ACDGraph, classDDG.getCFG(), methodCFGRoot);
    bfsDDG(ACDGraph, classDDG, methodDDGRoot);
}
```

5.3.2　建立单词表与向量化设计与实现

建立单词表与向量化部分基于上一节得到的程序图的 json 文件进行分析处理，最终得到函数的特征向量，以供分类模型的输入。首先需要读取 json 文件，其实现如下代码所示，该代码中为读取带有标签的训练数据。

```
def readFromJson(folderDir):
    fileList=os.listdir(folderDir)
    labels=[]
    graphs=[]
    for fileName in fileList:
        with open(folderDir+"/"+fileName, 'r') as load_f:
            #print(folderDir+"/"+fileName)
            graphJson = json.load(load_f)
            labels.append(graphJson['label'])
            graphs.append(graphJson)
    return graphs,labels
```

其次是根据 json 文件中记录的图的点和边信息重构图表示，其实现如下代码所示。

```
def create_graph_of_json(graphJsons):
    graphs=[]
    vocab = dict()
    for gJson in graphJsons:
        G = nx.Graph()
        for vertexJson in gJson["vertexList"]:
            G.add_node(vertexJson["name"])
            if vertexJson["name"] not in vocab:
                vocab[vertexJson["name"]] = len(vocab)
            G.nodes[vertexJson["name"]]['label'] =
            vocab[vertexJson["name"]]
            #判断途中是否有当前顶点，若无则加入
        for edgeJson in gJson["edgeList"]:
            G.add_edge(edgeJson["source"],
            edgeJson["target"])
        graphs.append(G)
    return vocab,graphs
```

建立单词表核心实现如下代码所示，在遍历点信息的过程中将各单词加入掉单词表 dict 中，并将单词表导出到 pkl 文件中。Python 的 pickle 模块针对 python 中的数据对象提供了易于使用的持久化功能，可以将 Python 中的数据对象当前的所有状态属性以.pkl 文件的形式导出存储在本地磁盘上，当需要还原该数据对象继续使用时，能迅速读取转换为原对象，故此处使用 pickle 进行单词表的存储。

```
def prepareVocab():
    graphJsons_train, labels_train = readFromJson("path")
    vocab,graphs = create_graph_of_json(graphJsons_train)
    with open("vocab" + '.pkl', 'wb') as f:
        pickle.dump(vocab, f, pickle.HIGHEST_PROTOCOL)
```

向量化的核心实现如下代码所示，此处使用 Weisfeiler-Lehman 函数作为图核函数进行训练，得到图的向量标识。

```
# Initialize a Weisfeiler-Lehman subtree kernel
gk = WeisfeilerLehman(n_iter=1, normalize=False,
base_graph_kernel=VertexHistogram)
# Construct kernel matrices
K_train = gk.fit_transform(G_train)
K_test = gk.transform(G_test)
```

5.4　分类识别模块

分类识别模块主要负责对候选加密函数进行分类识别，以判断其是具有加密功能的函数。

5.4.1　分类模型训练预测的设计与实现

分类模型训练预测流程架构图如图5.15所示，分成数据预处理、基于图核函数技术的函数向量生成以及基于支持向量机的函数分类预测三个子模块，除了三个主模块该架构还包括模块所涉及的所有相关数据和对模型的评估方法。其中，数据预处理子模块对带标签的加密函数训练数据集进行相关解析。首先，对变量名进行泛化去除影响，其次生成程序结构图并将不同的结构图进行融合。基于图核函数技术的函数向量生成子模块首先进行单词字典的构建，其次对图核函数进行训练，最后生成特征向量。基于支持向量机的函数分类预测子模块的工作流程主要包含以下几步：划分数据集为训练集和测试集，初始化分类模型，模型迭代训练，最后获得训练好的函数分类器。。

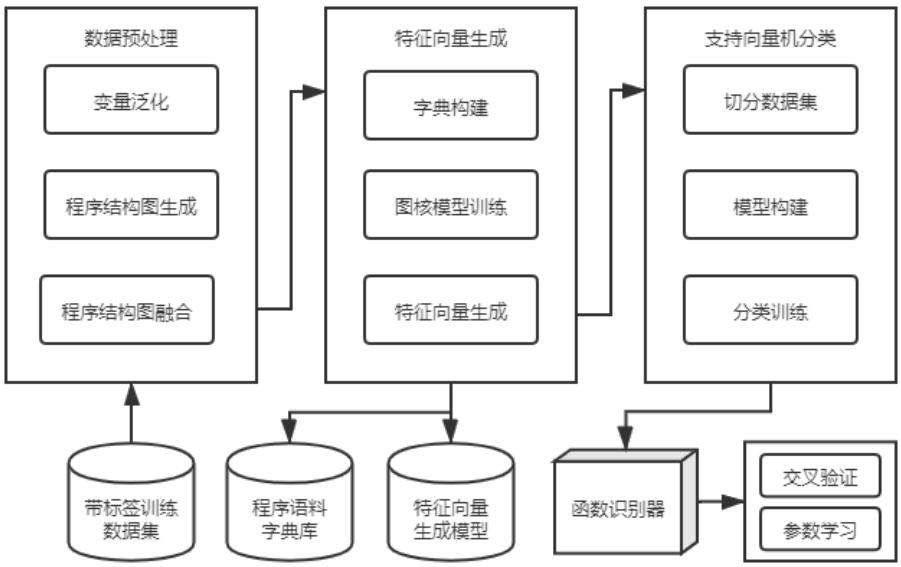

图 5.15: 模型训练构架图

模型训练预测的核心实现如下代码所示

```
def train(slice_dir: str, label_dir: str, train_percent: float = 1,
    saveto: str = None, data_length: int = 0)
    ......
    dataset = TextDataset(slice_dir, label_dir,
    preprocessing.preprocessing, data_length)
    train_data, test_data = dataset.divide(train_percent)
    ......
    # Train an SVM classifier and make predictions
    clf = SVC(kernel='precomputed')
    clf.fit(K_train, labels_train)
    y_pred = clf.predict(K_test)
```

5.5　本章小结

本章主要对程序隐私数据扫描系统的关键模块进行详细设计并详述了相关实现细节。本章从系统的业务处理模块、候选加密函数生成模块、程序特征提取模块和分类识别模块四个模块进行介绍。针对每个模块，首先详细介绍该核心类模块的详细设计，然后使用该模块核心类型图和顺序表等方式来说明该模块的具体实现，最后给出该核心类模块的一些关键源代码进行辅助说明并通过一些实际截图的形式来说明本文系统的各个页面。

第六章　系统测试与实验分析

本章将基于前两章对本文系统实现的描述，对系统提供的功能、性能以及技术方法的应用效果进行测试与评估。本章首先介绍测试与试验所基于的测试环境，包括软件和硬件环境，其后基于该测试环境进行系统的功能、性能测试并以图表的形式展示测试的结果，最后对隐私数据扫描方法 PDTP 的应用效果进行评估。

6.1　测试环境

表6.1展示了本文系统测试的软硬件环境，在此环境基础上开展系统功能测试、性能测试以及实验分析。测试环境分为客户端和服务端，两部分的硬件配置相同，由于扫描是计算密集型任务，硬件上选择了较高配置的 Intel i7 CPU 以及 16G 的内存容量，操作系统也均为 Windows10 专业版。其中客户端使用目前主流的 Chrome 浏览器作为与用户交互的系统平台，客户端实现代码项目使用 React 框架进行构建，通过 Webpack 进行打包。客户端服务器与服务端服务器之间使用 RESTful 风格的 API 通过 HTTP 库 Axios 进行通信。服务端分为两部分，一部分使用 Spring Boot 框架进行构建，负责客户端进行交互并调用另一部分的服务进行业务功能的实现，主要使用的相关模块工具包括支持 Java 程序运行的开发工具包 JDK、Java 程序分析优化框架 Soot、系统运行部署容器平台 Docker 以及数据存储服务 MySQL；另一部分使用 Django 框架进行构建，运行的程序包括 Python、Scikit-learn 等。

表 6.1: 系统测试环境

	软件/设备	配置/版本
机器配置	CPU	型号为 Intel(R) Core(TM) i7 8550U CPU @ 1.80GHz
	内存	容量为 16.0GB
	操作系统	Windows10 专业版
客户端	浏览器	Chrome 88.0（64 位）
	React	16.4.1
	Axios	0.21.1
	Webpack	3.8.1
服务端	JDK	$1.8.0_201$
	Spring Boot	2.1.5
	Soot	4.2.1
	Python	3.6
	Django	2.2.5
	Scikit-learn	0.23.2
	grakel	0.1.8
	Docker	18.09.2
	MySQL	8.0.12

6.2　功能测试与性能测试

6.2.1　功能测试

功能测试又名黑盒测试，指的是对系统应用在未知源代码的情况下，仅根据系统的外部表现进行测试的方法，该方法一般根据系统的功能需求进行测试用例的设计，并根据各项测试用例逐一进行实践，通过上述过程来验证系统是否符合用户的需求预期，同时不需要测试软件系统的内部结构和处理过程。本节将依据表4.1中所分析的用户需求设计测试哟用例进行功能测试。

表6.2展示了用户注册登录测试用例,包括了该用例的测试功能、测试步骤和预期结果以及测试结果。本用例主要关注在不同条件下（注册新用户、注册已注册用户、登录正确用户信息、登录为注册用户、登录错误用户信息）系统能否处理用户输入返回正确结果并显示提示信息。

表 6.2: 用户登录注册测试用例

测试 ID	UC1
测试名称	用户登录注册
测试功能	用户可以通过系统进行登录和注册操作
测试步骤	1. 进入系统登录界面，点击注册按钮，输入新用户名进行注册； 2. 进入系统登录界面，点击注册按钮，输入已注册用户名进行注册； 3. 进入系统登录界面，输入已注册用户名、正确用户令牌进行登录； 4. 进入系统登录界面，输入已注册用户名、错误用户令牌进行登录； 5. 进入系统登录界面，输入未注册用户名、任意用户令牌进行登录。
预期结果	1. 用户注册操作成功，弹窗显示注册成功信息以及生成的用户令牌； 2. 用户注册操作失败，弹窗显示注册失败信息并提示该用户名已被注册； 3. 用户登录操作成功，系统跳转至功能主界面； 4. 用户登录操作失败，弹窗显示登录失败并提示用户密码或用户名错误； 5. 用户登录操作失败，弹窗显示登录失败并提示用户未注册。
测试结果	通过

表6.3展示了扫描任务创建测试用例相关的测试过程与预期结果，主要关注项目文件是否能正常上传、配置文件能否正确关联、任务创建是否能正常完成以及系统能否正确显示已创建的任务信息。

表6.4展示了扫描任务管理测试用例相关的测试过程与预期结果，主要关注系统能否正确展示用户的任务信息，并正确完成任务的编辑以及删除操作，正常启动任务并及时更新任务状态。

表6.5展示了扫描检测配置管理测试用例相关的测试过程与预期结果，主要关注系统能否正常完成配置信息的显示，正确处理配置的创建、编辑以及删除操作。

表 6.3: 扫描任务创建测试用例

测试 ID	UC2
测试名称	任务创建
测试功能	用户可以通过系统进行扫描任务创建操作
测试步骤	1. 用户在系统主页面点击创建任务图标，进入创建扫描任务页面； 2. 用户填写项目名称和项目描述； 3. 用户点击“选择配置”，并选择需要关联到扫描任务的已有配置； 4. 用户点击“添加”，并上传新的配置文件已关联当前项目； 5. 用户点击上传扫描项目图标，并上传待扫描项目； 6. 用户点击确认创建的图标完成扫描任务的创建。
预期结果	1. 系统弹出扫描任务创建弹窗并显示填写指引； 3. 系统弹窗显示当前已有的配置供用户选择，用户选择后将以小标签形式显示在任务创建界面； 4. 系统接收新配置文件信息，并将其上传至用户配置库，用户可在配置管理处查看，上传完成后同样以小标签形式显示在任务创建界面； 5. 系统接收上传的项目文件，上传成功后，系统以弹出框提示“上传成功”，并且正确显示文件名称 6. 系统提示任务已经创建并指引用户到任务管理区域对任务进行后续处理，任务管理界面的任务列表中显示出方才新建的任务且任务信息正确。
测试结果	通过

表 6.5: 扫描检测配置管理测试用例

测试 ID	UC4
测试名称	扫描检测配置管理
测试功能	用户可以通过系统进行扫描配置管理操作
测试步骤	1. 进入系统配置管理界面，用户可对所有创建的扫描配置进行管理； 2. 用户选择一项配置，点击编辑按钮，系统弹出配置编辑弹窗； 3. 用户点击“新建配置”按钮，系统显示新建配置界面； 4. 用户选择一项配置，点击删除按钮以删除扫描配置。
预期结果	1. 系统显示用户创建的所有配置的列表； 2. 系统弹出配置编辑弹窗，任务信息正确，用户编辑过后系统能正确显示编辑后的配置信息； 3. 系统提示配置创建成功，并自动刷新页面，配置列表新增方才创建的配置条目； 4. 系统提示配置删除成功，并自动刷新页面，该配置将不存在于列表中。
测试结果	通过

表 6.4: 扫描检测任务管理测试用例

测试 ID	UC3
测试名称	扫描检测任务管理
测试功能	用户可以通过系统对已经创建的扫描检测任务进行管理活动
测试步骤	1. 点击左侧功能栏的任务管理按钮进入系统任务管理界面，用户可以查看所有已创建任务的信息列表； 2. 用户选择一项任务，点击编辑按钮，系统弹出任务编辑弹窗； 3. 用户选择一项任务，点击启动按钮以启动扫描任务； 4. 用户选择一项任务，点击删除按钮以删除扫描任务。
预期结果	1. 系统显示用户创建的所有任务的列表； 2. 系统弹出任务编辑弹窗，任务信息正确，用户编辑过后系统能正确显示编辑后的任务信息； 3. 系统提示任务启动成功，并自动刷新页面，任务状态改变为“运行中”； 4. 系统提示任务删除成功，并自动刷新页面，该任务将不存在于列表中。
测试结果	通过

表6.6展示了查阅扫描检测报告测试用例的相关信息，主要关注系统能否展示正确的扫描结果报告信息并提供正常的报告下载功能。

表 6.6: 查阅扫描检测报告测试用例

测试 ID	UC5
测试名称	查阅扫描报告
测试功能	用户在当前任务系统处理完成后，可通过任务管理界面对当前任务的扫描检测报告进行查看，并可在报告详情页进行导出下载
测试步骤	1. 用户进入任务管理页面并选择一项“已完成”状态的任务； 2. 用户点击当前任务的“查看报告”图形按钮，系统弹出扫描报告弹窗； 3. 用户点击扫描报告界面中某一项扫描项的“查看详情”按钮，系统显示该任务的扫描检测结果详情界面； 4. 用户点击扫描报告界面的“导出报告”以进行报告的导出。
预期结果	1. 系统显示多有已完成的任务列表； 2. 系统显示扫描报告弹窗，内容包括扫描报告的简要版本； 3. 系统页面显示该扫描数据项的详细传播路径信息； 4. 系统提示报告导出成功并自动开始报告的下载。
测试结果	通过

上述测试用例均已按照步骤依次执行，所有测试用例的执行结果符合测试预期，说明本系统达到了基本的用户需求。

6.2.2　性能测试

本小节选用当前应用最广的主流 Web 应用压力测试工具 JMeter 对本系统进行性能测试。测试的形式是对系统的常用也即流量最大的接口进行并发测试，通过测试接口可以反映系统在应对多用户同时访问时的性能表现。本小节选用系统的用户登录接口以及查看扫描检测报告的接口进行性能测试，测试结果如表6.7。从表中可以看出该压力测试的设置是用户线程共 200 个，在 2 秒内向系统进行请求，该数据基本模拟一个中小流量的网站的用户访问频度，符合本文系统的实际应用情况。图表中的结果显示，对于系统的登录接口，响应时间平均值、最大值、99 线以及吞吐量分别为 21ms，26ms、23ms 以及 200 条每秒；对于查看扫描检测报告接口，响应时间平均值、最大值、99 线以及吞吐量分别为 53ms、76ms、71ms 以及 97 条每秒，两个接口均未出现错误请求的异常情况，从上述结果可以得出本文系统能在可接受的时间范围内给出响应并且承受一定量用户的并发访问。如图6.1所示为查看扫描检测报告的接口的响应时间推移图，图中以 10ms 为间隔显示出了该接口响应时间随时间的变化，从图像上可以看出改接口的响应时间并未出现很大的波动，一直在 40ms-80ms 的范围之内，故本系统在能应对一定量并发的压力情况下也能保持稳定的运行，具有较为良好的性能。

表 6.7: 接口压力测试结果

接口名	请求量	平均响应 (ms)	中位数 (ms)	99 百分位 (ms)	最小值 (ms)	最大值 (ms)	异常率 (%)	吞吐量 (ms)
登录系统	200	21	21	23	19	26	0.00	200.6/sec
查看扫描检测报告	200	53	53	71	44	76	0.00	97.5/sec

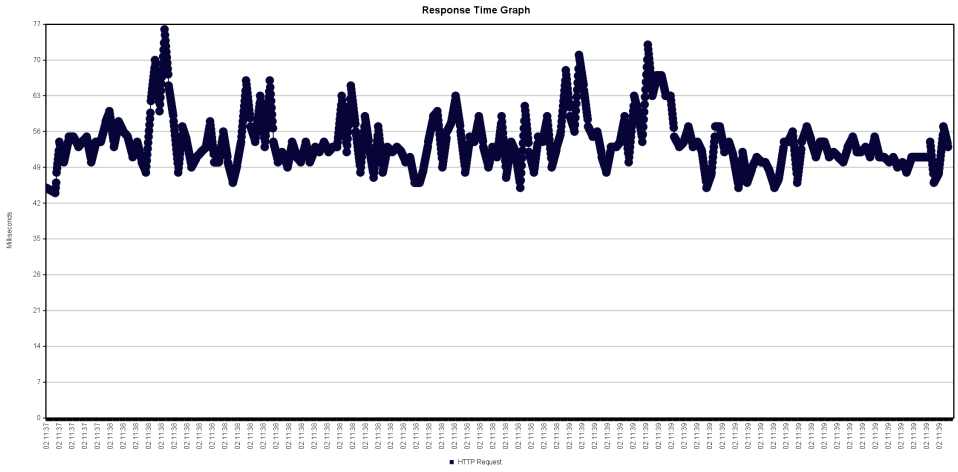

图 6.1: 查看扫描检测报告接口响应时间图

6.3　实验设计

本文针对现有的隐私漏洞检测工具技术误报率高，无法定位完整的隐私数据等问题，提出基于污点分析以及机器学习的隐私数据路径定位分析方法，以得到更为准确的隐私数据泄露风险检测结果并提供完整的隐私数据视图。本节实验将本系统的扫描检测结果与传统的漏洞扫描工具进行对比，已验证本系统可以有效降低漏报及误报率。本节将从实验对象、实验流程以及评估指标来描述实验的设计，二实验的结果则将在下一小节进行分析。

6.3.1　实验对象

本文采用两个真实项目进行实验，分别是购物网站后台项目以及移动社交软件后台项目。下表为实验对象的具体数据，包括实验对象名、实验对象描述、包含的隐私数据数量、具有隐私泄露风险的数据变量数量、不具有隐私泄露风险的数据变量数量等。

表 6.8: 实验对象信息

名称	描述	变量/用例数量	具有隐私泄露风险的数据变量数量/真实用例	不具有隐私泄露风险的数据变量数量/误报用例
onlineHall	购物网站后台项目	53	29	24
justHi	移动社交软件后台项目	49	32	17

6.3.2　评估指标

本文对隐私数据泄露风险的检测问题可以看作是二分类问题，即存在风险和不存在风险，则针对该问题的解决方法的实践效果可以通过相应的定量指标进行评估。本文对隐私数据泄露风险检测效果的评价指标包括：准确率、精确率、召回率以及 F1-Score，下面对以上指标进行简要介绍。

TP: Ture Positive，把正例判断为正的数目。

FN: False Negative，把正例错判为负的数目。

FP: False Positive，把负例错判为正的数目。

TN: True Negative，把负例判为负的数目。

准确率：准确率是指在实验的所有结果中，判断正确的样本数量，这里包括 TP 以及 TN，占所有样本数量的比例，反映了测定值与真值相符合的程度，其公式为：$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$。

精确率：精确率是相对于预测结果而言的，它表示的是预测为正的样本中有多少是对的，反映了方法的预测效果，公式为：$Precision = \frac{TP}{TP+FP}$。

召回率：召回率是相对于正样本而言的，即样本中有多少正样本被预测正确，公式为：$Recall = \frac{TP}{TP+FN}$。

F1-Score：F1-Score 综合了精准率与召回率，反映了实验方法是否能有效平衡漏报和误报，具体的计算方式则是上述两个值的调和平均，F1-Score 具有一定的可参考性，其公式为：$F1-Score = \frac{2PrecisionRecall}{Precision+Recall}$。

6.3.3　实验流程

本节实验通过与当前传统的漏洞扫描工具（Find Security Bug）进行对比已说明本文提出的方法能够有效降低风险检测的误报率，本文实验的流程如下：

（1）使用本文隐私数据泄露风险扫描检测系统以及 Find Security Bug 工具分别对实验对象进行扫描检测，得到两个系统在实验对象上检测的正误结果。

（2）基于上一步得到的判断结果，针对两系统的实验数据分别计算上一小节中提到的实验评估指标并进行比较。

（3）对两系统的实验进行三次，最终结果取三次实验数据的平均值。

6.4　实验结果与分析

表 6.9: 实验结果

	准确率	精确率	召回率	F1-Score
Find Security Bug	67.64%	64.89%	100.00%	66.23%
本系统	88.23%	91.52%	88.52%	90.00%

经过上一节所述的实验流程得到的最终实验结果数据如图6.9所示。从数据对比可以看出，Find Security Bug 保证了极高的召回率，达到了 100.00%，即所有真实存在泄露风险的隐私数据都被成功检测出来，这也与当前方法着重于不漏报相呼应；但相应的，为了对风险漏洞不漏报，部分并非有风险的样本也被判定有泄露风险，从数据上就可以看出，该工具的准确率仅为 67.64%，这意味着大量误报的存在。观察本文系统的实验数据可以看出，本文系统在损失了 11.48% 的小部分召回率（即漏报了少部分风险），准确率达到了近 88%，较 Find Security Bugs 将准确率提高 20%，在精确率和 F1-score 指标上也较高。综上，相较于传统隐私漏洞扫描器，本系统能够以较低的成本（召回率）更为准确地检测隐私数据泄露风险。

6.5　本章小结

本章主要对本文实现的隐私数据泄露风险扫描检测系统进行测试，并对系统提出的 PDTP 方法进行了实验评估。本章首先描述了本文测试以及实验的软硬件环境；其次是对功能测试用例的设计，该部分与第四章的内容相对应，并对测试用例逐一执行得到功能测试结果；接着是对 U 系统的性能测试，该部分是对系统的常用接口利用流行的压力测试工具进行并发测试，以验证系统的性能，并使用表格以及响应时间图进行了具体说明；随后是对隐私数据定位分析方法的实验分析，通过与传统工具的对比实验说明了本文系统能有效扫描项目中的隐私数据并给出准确的漏洞风险分析。

第七章　总结与展望

7.1　总结

互联网大数据时代已经到来，个人信息开始越来越多的暴露到软件介质中工企业组织保管分析，随之而来的是层出不穷的隐私安全问题。为了保护个人隐私信息的安全，各国组织相继出台个人信息保护法规，要求企业组织对个人隐私信息进行数据合规工作。对个人信息数据的合规工作首要的就是对软件内的隐私相关数据以及可能存在的漏洞进行定位，目前的多数静态扫描工具采用过度近似策略以保证结果的完备性，这导致其误报率过高，反而加大了工程师的工作量。针对这一问题，本文设计并实现结合污点分析以及机器学习的程序隐私数据扫描系统，通过污点分析定位隐私数据在程序系统中传播的路径并且使用机器学习技术对关键函数进行识别判断其是否为匿名处理函数，有效降低扫描工具误报率并提供完整的数据传播路径，为开发人员节约了数据合规成本。本文主要工作如下：

（1）本文提出了基于污点分析以及机器学习的隐私数据路径定位分析方法PDTP，首先，以程序外部输入个人信息相关变量作为起点进行污点分析，得到数据传播的基本路径并生成污点传播图；其次，从污点传播图提取传播经过的函数作为候选加密函数；随后对候选加密函数进行程序图特征的提取，并基于图核函数进行函数特征向量的生成；最后基于函数特征向量，应用机器学习分类模型，此处应用的是支持向量机 SVM，进行函数的分类识别，判断变量是否进行了加密操作以筛除无泄露风险的隐私变量，得到最终的隐私数据路径扫描分析结果。

（2）本文基于 PDTP 实现了一个程序隐私数据泄露风险检测系统，用户可以上传软件源代码项目至系统并创建扫描任务，系统在对项目扫描完成后会生成详尽的扫描检测报告供用户查看。系统被设计为有良好的可拓展性，对于污点分析、程序图特征提取、程序特征向量生成以及程序分类识别等，都提供了对外的接口，以便后续可以整合到其他系统或进行功能扩展。

（3）本文在真实项目上进行了实证研究，并与现有的开源工具 Find Security Bug 相对比，以评估本文方法的有效性。实验结果表明，相对于比较的工具，本文系统在损失小部分召回率的基础上较大幅度地提高了风险检测的准确性（20%），体现了本文方法及工具在风险检测方向具有一定的价值。

7.2 展望

本文隐私数据泄露风险程序扫描检测系统在可适用范围、用户体验以及实验评估等方面还有较大的提升进步空间，未来计划就以下几个方面进行改进：

（1）本文系统实现目前只针对于 Java 语言开发的 Java Web 工程，计划在未来扩展到其他语言，最终目标是打破语言的限制，开发一个通用的工具。

（2）本文系统目前只适应于传统的 Java Web 项目，不能适配现有的框架如 Spring，未来可以通过结合动态分析突破框架将类对象运行时自动包装导致静态分析难以应用的问题。

（3）当前的实验评估由于时间原因只选取了两个真实项目进行实践，其对于普遍的程序应用的代表性有限，在后续的实验中需要考虑扩充更多类型的实验对象。

（4）当前系统只考虑加密作为数据脱敏的方法，在未来可以扩展更多的数据脱敏处理方式。

（5）当前系统只考虑针对系统默认配置以及用户配置规定的规范的变量名进行扫描，实际项目中存在不规范的变量命名，在未来可以对不规范变量也进行识别分析是否为隐私变量。

参考文献

[1] F. Nielson, H. R. Nielson, C. Hankin, Principles of program analysis, Springer Science & Business Media, 2004.

[2] 李珍, 邹德清, 王泽丽, 金海, 面向源代码的软件漏洞静态检测综述, 网络与信息安全学报 005 (001) (2019) 1–14.

[3] I. Chowdhury, M. Zulkernine, Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities, Journal of Systems Architecture 57 (3) (2011) 294–313.

[4] N. H. Pham, T. T. Nguyen, H. A. Nguyen, T. N. Nguyen, Detection of recurring software vulnerabilities, in: IEEE/ACM International Conference on Automated Software Engineering, 2010.

[5] S. Neuhaus, T. Zimmermann, C. Holler, A. Zeller, Predicting vulnerable software components, in: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007, 2007.

[6] D. Rattan, R. Bhatia, M. Singh, Software clone detection: A systematic review, Information and Software Technology 55 (7) (2013) 1165–1199.

[7] H. Liang, L. Wang, D. Wu, J. Xu, Mlsa: A static bugs analysis tool based on llvm ir (2016) 407–412.

[8] F. Cassez, A. M. Sloane, M. Roberts, M. Pigram, P. Suvanpong, P. G. D. Aledo, Skink: Static analysis of programs in llvm intermediate representation.

[9] J. Thomé, L. K. Shar, D. Bianculli, L. Briand, Search-driven string constraint solving for vulnerability detection, in: 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), 2017, pp. 198–208.

[10] R. Mahmood, Q. Mahmoud, Evaluation of static analysis tools for finding vulunerbailities in java and c/c++ source code, CoRR.

[11] H. Shahriar, M. Zulkernine, Mitigating program security vulnerabilities: Approaches and challenges, ACM Computing Surveys 44 (3) (2012) 1–46.

[12] J. Viega, J. T. Bloch, Y. Kohno, G. Mcgraw, Its4: a static vulnerability scanner for c and c++ code, in: Proceedings 16th Annual Computer Security Applications Conference (ACSAC'00), 2002.

[13] S. Russell, P. Norvig, Artificial intelligence: a modern approach.

[14] T. S. Guzella, W. M. Caminhas, A review of machine learning approaches to spam filtering, Expert Systems with Applications 36 (7) (2009) 10206–10222.

[15] G. Caruana, M. Li, A survey of emerging approaches to spam filtering, ACM Computing Surveys (CSUR) 44 (2) (2008) 1–27.

[16] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, E. Vázquez, Anomaly-based network intrusion detection: Techniques, systems and challenges, computers & security 28 (1-2) (2009) 18–28.

[17] C. V. Zhou, C. Leckie, S. Karunasekera, A survey of coordinated attacks and collaborative intrusion detection, Computers & Security 29 (1) (2010) 124–140.

[18] H. Perl, S. Dechand, M. Smith, D. Arp, F. Yamaguchi, K. Rieck, S. Fahl, Y. Acar, Vccfinder: Finding potential vulnerabilities in open-source projects to assist code audits, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, pp. 426–437.

[19] O. Ferschke, I. Gurevych, M. Rittberger, Flawfinder: A modular system for predicting quality flaws in wikipedia., in: CLEF (Online Working Notes/Labs/Workshop), 2012, pp. 1–10.

[20] Z. Li, D. Zou, S. Xu, H. Jin, H. Qi, J. Hu, Vulpecker: an automated vulnerability detection system based on code similarity analysis, in: Proceedings of the 32nd Annual Conference on Computer Security Applications, 2016, pp. 201–213.

[21] S. Kim, S. Woo, H. Lee, H. Oh, Vuddy: A scalable approach for vulnerable code clone discovery, in: 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 595–614.

[22] F. Yamaguchi, F. Lindner, K. Rieck, Vulnerability extrapolation: Assisted discovery of vulnerabilities using machine learning, in: Proceedings of the 5th USENIX conference on Offensive technologies, 2011, pp. 13–13.

[23] F. Yamaguchi, M. Lottmann, K. Rieck, Generalized vulnerability extrapolation using abstract syntax trees, in: Proceedings of the 28th Annual Computer Security Applications Conference, 2012, pp. 359–368.

[24] S. Heelan, Vulnerability detection systems: Think cyborg, not robot, IEEE Security & Privacy 9 (3) (2011) 74–77.

[25] 王蕾, 李丰, 李炼, 冯晓兵, 污点分析技术的原理和实践应用, 软件学报 (4).

[26] O. Tripp, S. Guarnieri, M. Pistoia, A. Aravkin, Aletheia: Improving the usability of static security analysis, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, 2014, pp. 762–774.

[27] F. Yamaguchi, A. Maier, H. Gascon, K. Rieck, Automatic inference of search patterns for taint-style vulnerabilities, in: 2015 IEEE Symposium on Security and Privacy, IEEE, 2015, pp. 797–812.

[28] O. Tripp, M. Pistoia, S. J. Fink, M. Sridharan, O. Weisman, Taj: effective taint analysis of web applications, ACM Sigplan Notices 44 (6) (2009) 87–97.

[29] N. Jovanovic, C. Kruegel, E. Kirda, Pixy: A static analysis tool for detecting web application vulnerabilities, in: 2006 IEEE Symposium on Security and Privacy (S&P'06), IEEE, 2006, pp. 6–pp.

[30] 刘芳, 李戈, 胡星, 金芝, 基于深度学习的程序理解研究进展, 计算机研究与发展 56 (8) (2019) 1605–1620.

[31] C. Cummins, P. Petoumenos, Z. Wang, H. Leather, Synthesizing benchmarks for predictive modeling, in: 2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), IEEE, 2017, pp. 86–99.

[32] X. Gu, H. Zhang, D. Zhang, S. Kim, Deep api learning, in: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2016, pp. 631–642.

[33] M. White, C. Vendome, M. Linares-Vásquez, D. Poshyvanyk, Toward deep learning software repositories, in: 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories, IEEE, 2015, pp. 334–345.

[34] A. Bhoopchand, T. Rocktäschel, E. Barr, S. Riedel, Learning python code suggestion with a sparse pointer network, arXiv preprint arXiv:1611.08307.

[35] S. Iyer, I. Konstas, A. Cheung, L. Zettlemoyer, Summarizing source code using a neural attention model, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2016, pp. 2073–2083.

[36] X. Hu, G. Li, X. Xia, D. Lo, S. Lu, Z. Jin, Summarizing source code with transferred api knowledge.

[37] M. Allamanis, H. Peng, C. Sutton, A convolutional attention network for extreme summarization of source code, in: International conference on machine learning, PMLR, 2016, pp. 2091–2100.

[38] J. Li, Y. Wang, M. R. Lyu, I. King, Code completion with neural attention and pointer networks, arXiv preprint arXiv:1711.09573.

[39] C. Liu, X. Wang, R. Shin, J. E. Gonzalez, D. Song, Neural code completion.

[40] P. Yin, G. Neubig, A syntactic neural model for general-purpose code generation, arXiv preprint arXiv:1704.01696.

[41] M. Rabinovich, M. Stern, D. Klein, Abstract syntax networks for code generation and semantic parsing, arXiv preprint arXiv:1704.07535.

[42] M. Allamanis, M. Brockschmidt, M. Khademi, Learning to represent programs with graphs, arXiv preprint arXiv:1711.00740.

[43] M. Allamanis, M. Brockschmidt, Smartpaste: Learning to adapt source code, arXiv preprint arXiv:1705.07867.

[44] M. Brockschmidt, M. Allamanis, A. L. Gaunt, O. Polozov, Generative code modeling with graphs, arXiv preprint arXiv:1805.08490.

[45] T. Ben-Nun, A. S. Jakobovits, T. Hoefler, Neural code comprehension: A learnable representation of code semantics, arXiv preprint arXiv:1806.07336.

[46] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, K. M. Borgwardt, Weisfeiler-lehman graph kernels., Journal of Machine Learning Research 12 (9).

[47] P. B. Kruchten, The 4+1 view model of architecture, IEEE Software 12 (6) (1995) 45–50.

简历与科研成果

致　谢