面向代码审查识别个人数据处理

Feiyang Tang[1], BjarteM.Østvold[1]和 Magiel Bruntink[2]挪威计算中心，奥斯陆，挪威[2]*Software Improvement Group*，阿姆斯特丹，荷兰{*feiyang, bjarte*}*@nr.no, m.bruntink@sig.eu*

arXiv:2301.01568v1 [cs.SE] 4 Jan 2023

关键词: 数据隐私保护、代码审查、静态分析

摘要: 代码审查是软件开发生命周期中的关键环节，用于评估并提升代码的有效性和正确性、发现安全问题，并通过遵循最佳实践来提高代码质量。由于立法推动了对个人数据保护日益增长的需求，代码审查人员需要了解个人数据在软件系统中的存储位置及其处理方式。尽管近期大多数关于代码审查的研究都聚焦于安全漏洞，但代码审查人员不易实施隐私相关技术，因此将其纳入代码审查流程颇具挑战性。本文介绍一项正在进行的新方法研究，该方法旨在识别个人数据处理，帮助开发人员和代码审查人员开展隐私分析，并遵守《通用数据保护条例》（GDPR）等法规。

1 引言

《通用数据保护条例》（GDPR）为欧盟的数据保护奠定了法律基础，并通过提升个人的数据保护权利，加强了整个欧洲范围内的个人数据保护。对于不遵守该法规的企业，该条例还规定了高额罚款，最高可达其全球年度收入的4%。许多IT系统提供商，尤其是软件生产企业，可能需要改造其系统，以遵守《通用数据保护条例》。预计这将需要付出大量努力（Blume，2016）。因此，为行业中的软件工程师提供有效且系统的方法，将数据保护融入软件开发，是一个必不可少且具有重要价值的研究课题（Lenhard 等，2017）。组织正将安全措施前移至软件开发生命周期中的各个环节，例如代码审查，以防止软件安全漏洞（Braz 和 Bacchelli，2022）。同样，为了遵循隐私保护设计原则并执行隐私分析任务，代码审查人员也将受益于类似于安全领域所使用的工具，以识别软件中的隐私相关模式。

开发人员使用数据安全术语来应对隐私问题，而这套词汇将他们对隐私的理解局限于组织之外的威胁（Hadar 等人，2018）。然而，尽管数据安全是数据隐私的主要前提，软件中的隐私保护仍与传统的安全相关漏洞大不相同——同。Bambauer 指出：“安全与隐私可以而且应该被视为彼此独立的问题”（Bambauer，2013年）。开发人员难以将法律、伦理和社会层面的隐私关切转化为具体的技术和解决方案（Notario 等人，2015）。

评估隐私不仅涉及在软件中查找个人数据，还包括评估相关处理是否合规。《通用数据保护条例》将处理定义为：“对个人数据或个人数据集执行的任何操作或一系列操作，无论是否通过自动化手段进行。”该定义涵盖对个人数据执行的广泛操作，例如收集、记录、组织、整理、存储、适配或修改、检索、传输等。隐私评估任务引出一个问题：我们如何协助代码审查人员和软件开发人员评估个人数据处理？通过识别系统中的个人数据及相关处理，代码审查人员可以发现有价值的模式，并利用这些模式重新设计系统，使其更加隐私友好，或开展隐私分析。

本文介绍一项正在进行的研究工作，提出一种新方法，旨在帮助开发人员和代码审查人员识别个人数据处理，以便后续用于隐私分析。这使开发人员和代码审查人员能够协助组织完成各种重要的隐私

相关任务，例如完成数据保护影响评估（DPIA）并制定隐私政策。

2 相关工作

软件开发过程中的一个关键步骤是代码审查，它包括人工审查和/或自动化审查。代码审查的主要目标是评估并提升代码的效能和正确性，找出安全问题，并通过遵循最佳实践来提高代码质量（McIntosh 等人，2014）。为自动评估代码，人们开发了各种漏洞检测工具。这类工具也称为源代码分析器或静态分析工具，因为它们无需执行程序即可分析其代码（McGraw，2008）。

CodeQL[1], 和 Semgrep（r2c，2022）[2] 是两款广受欢迎的静态分析代码审查工具。CodeQL 将代码视为数据，并将问题建模为查询。从代码中提取这些查询后，会针对数据库执行查询。数据库是一个目录，其中包含数据、用于显示查询结果的源引用、查询结果和日志文件。Semgrep 会在经过解析的程序（以抽象语法树（AST）表示）中匹配语法模式，而不是将程序视为字符串，在其中匹配字符串或正则表达式模式。与 CodeQL 相比，Semgrep 大大简化了自定义规则的编写；CodeQL 则要求使用 QL（一种声明式面向对象查询语言）来定义规则。

目前，专注于通过代码审查识别隐私相关漏洞的已发表研究相对较少，而且将现有安全知识应用于隐私领域存在问题，我们将在第 3 节对此进行解释。也有一些关于识别个人数据的研究，对我们的研究很有价值。Fugkeaw 等人（Fugkeaw 等人，2021）提出了 AP2I，使组织能够自动识别和管理本地文件系统中的个人数据。ReCon（Ren 等人，2016）通过监控网络流量，利用机器学习识别可能发生的个人数据泄露。van der Plas 等人（van der Plas，2022）使用 CodeBERT（一种类 RoBERT 的 Transformer 模型）识别 Git 提交中的个人数据。

[1]https://codeql.github.com/ [2] https://semgrep.dev/

3 背景与挑战

数据隐私分析正变得与安全漏洞发现同等重要，并为数据安全困境带来了新的维度（Bertino，2016）。如果代码审查人员能够像进行安全分析那样开展类似的隐私分析，将大有裨益。

当前最先进的方法主要侧重于安全分析。尽管数据安全是数据隐私的一项基本要求，但分析领域和识别流程却大不相同（Jain 等人，2016）。简单地采用安全机制和思维方式来分析隐私可能会产生误导，甚至造成危害（Bambauer，2013）。

将近期关于代码审查期间评估软件隐私的研究整合起来颇具挑战。在程序分析方面，有三种广为人知的隐私分析方法。首先，基于字节码的静态分析需要项目编译，而动态污点分析则需要执行项目。这对于代码审查人员而言，既不切实际，也难以高效实施。基于机器学习的技术同样难以实施，因为它需要大量且多样化的训练数据集。获取和生成此类数据集需要额外投入，也可能超出代码审查人员的能力范围。最后，基于用户界面控件的文本分析受限于特定领域的用户界面属性。例如，使用基于安卓健康类移动应用训练的模型的金融网络应用程序，不太可能从中受益。代码审查人员需要一种易于部署、高效且适应性强的方法（Buse 和 Zimmermann，2012）。

由于隐私本身的复杂性以及个人数据定义的流动性，在代码库中识别个人数据的处理活动颇具挑战。

在以下段落中，我们将重点介绍代码审查过程中与此任务相关的两个最主要挑战。

3.1 个人数据定义的模糊性

《通用数据保护条例》第4条第1款将个人数据定义为：

> 与已识别或可识别的自然人（“数据主体”）有关的任何信息；可识别的自然人是指能够直接或间接识别的个人，尤其是通过姓名、身份号码、位置数据等标识符识别的个人，

> 或通过在线身份标识，或通过一个或多个特定因素识别的个人，这些因素涉及该自然人的生理、心理、遗传、精神、经济、文化或社会身份。

《通用数据保护条例》中个人数据的定义非常宽泛，几乎任何信息只要涉及某个特定个人，都可能被认定为个人数据，例如某人穿着红色衬衫这一事实（Berˇciˇc 和 George，2009）。这一概念在语义上也存在歧义。

与某些数据从一开始就是匿名数据（例如与任何真实人物都无关联的天气传感器数据）不同，其他数据起初可能属于个人数据，但之后可以通过有效处理，使其不再与已识别或可识别的自然人相关联。这凸显了个人数据分类的灵活性（Finck 和 Pallas，2020）。

同一数据点是否属于个人数据，取决于具体情境，因此可能受法规约束，也可能不受约束。这意味着，软件中的个人数据类别会因软件及其背后的处理活动而异。例如，血压等健康数据以及医疗记录对健康应用程序而言属于敏感数据，而位置数据对导航软件而言则属于敏感数据。

> 即使我们承认，从内容上看，只要某项信息能够与个人相关联，就可以将其视为个人数据，但《通用数据保护条例》对个人数据的定义在结构上仍相当模糊，因为并非总能明确个人的每条“记录”必须具有什么样的结构，才能被视为个人数据（Voss 和 Houser，2019）。

由于相关法律中个人数据定义含糊不清，我们在实践中很难确定一个清晰且固定的标识符，以便在代码中准确定位个人数据。

3.2 什么算作敏感数据处理?

数据主体可以同意出于特定目的进行数据处理。这是通常采用的法律依据，但只是考量因素之一。数据处理也可能“为履行数据主体作为一方当事人的合同所必需，或为应数据主体的请求，在订立合同前采取措施所必需。”（Voss 和 Houser，2019）

遗憾的是，原则上对合同与同意之间关系的担忧，在现实中往往通过忽视同意要求而被规避（Pormeister，2017）。

我们无法依靠现有的隐私政策和书面同意来发现代码库中的个人数据处理。这要求我们考虑代码库中所有潜在的个人数据处理。稍后，我们将在第 4.1.2 节说明如何定义并识别软件中的相关处理。

4 方法

我们提出了一种方法，用于识别代码库中个人数据处理的实例，并以便于代码审查的方式呈现这些实例。

该方法主要分为三个阶段：模式匹配、标注和结果分组。输入是由源代码文件组成的代码库。随后，静态分析器将根据我们的规则和模式分析这些源代码文件。静态分析器发现的代码片段会根据其包含的各种特征进行标注。最后，我们允许用户按单个或多个标签对结果进行分组，从而根据个人需求探索分析结果。

我们的方法示意图如图 1 所示。

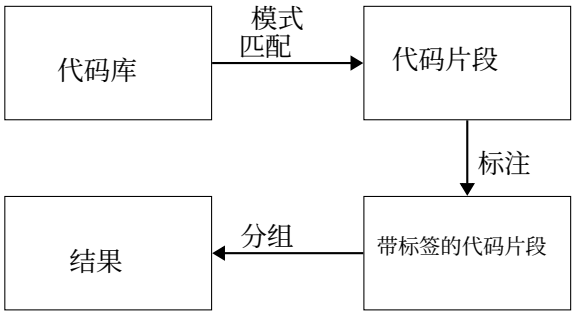

图 1： 方法

4.1 设计选择

在以下段落中，我们将讨论实现该方法时所做的设计选择。

4.1.1 发现类型

我们希望预设一份需要定位的基本个人数据清单，其中主要包括个人身份标识和特征数据，例如全名、电子邮件地址、性别、性取向和年龄。我们将其称为固定个人数据。

根据不同类型的软件，我们会为其定制默认列表。例如，对于银行/金融应用，列表中可能包含银行账

户号码、信用评分和薪资信息。这类个人数据取决于具体软件中的处理类型，我们将其称为情境化个人数据。

根据我们在软件中定位所提及个人数据的方式，可以将其在代码中的出现简单分为两种类型。

第一种是明文。这包括个人数据以明文出现的所有位置。它是逐字或直接呈现的个人数据。例如，信用卡号出现在 SQL 查询中，或者电子邮件地址被传递给日志函数。

另一种类型更为常见且更为隐蔽，即个人数据存储在变量或对象中。根据编程语言类型的不同，对象类型可能是局部变量、类实例或原型。这意味着我们的目标是找出处理此类数据的代码。

4.1.2 处理类型

简单地定位每个个人数据实例会产生大量结果。其中许多结果无法直接帮助代码审查员开展工作，而代码审查员的任务是找出有意义的处理。我们希望采用混合方法，尽可能涵盖更多处理。

处理个人数据是一种特定行为。这促使我们采用第一种方法：使用动作名称标签查找相关处理。我们采用了 DPV 第 3 节中的大多数动词（Pandit 等人，2019）[3]。这些词汇有助于我们在软件中查找相关流程。

第二种方法是识别外部库。我们知道，现代应用程序依赖各种 API 来实现不同目标。因此，获取相关 API 列表，并检测是否有个人数据流入这些 API，有助于我们找到有意义的模式。

4.2 模式匹配

第一步是将代码库（由源代码文件组成）交由静态分析器进行模式匹配。我们选择 Semgrep 作为分析器，因为它的规则易于使用，处理速度也很快。根据个人数据不同的语法特征（如第 4.1.1 节所述），我们采用结合两种不同分析类型的混合方法。

- 使用正则表达式匹配明文中的个人数据。
- 通过污点分析，在每个文件中查找符合我们标准的流程，这些流程连接源（个人数据进入分析范围的位置）和汇（个人数据被处理的位置）。

目前，我们的个人数据处理规则支持 Java、JavaScript 和 TypeScript 作为主要分析领域。不过，我们用于识别明文个人数据的规则适用于 Semgrep 支持的绝大多数语言。

4.2.1 源与汇

我们的原型将源分为九个不同类别。如第 4.1.1 节所述，我们将固定个人数据分为四个不同类别：账户、联系人、国家身份证件和个人身份标识。此外还包括五个情境化个人数据类别，例如位置、健康和财务数据。此外，我们还提供用于识别个人数据处理的模板，并允许代码审查人员和开发人员只需输入相关关键词，即可提交其他个人数据。随后，系统会自动生成相应规则，供将来使用。

汇分为五种主要类型。其中三种是操作类型：数据操作（**M**）、数据传输（**T**）和数据创建/删除（**C/D**）。另外两种是特殊类型：数据库（**DB**）和加密（**E**）。

汇的名称可能包含特定类型的源。例如，setLatitude(100,100) 不会将任何源传入该方法，但包含一个源标识符 Latitude 和一个汇标识符 set，表明它会直接将值作为源处理并传入汇。我们将这种特殊类型的汇称为特定于源的汇。当特定于源的汇调用任何内容时，我们会将该特定于源的汇标记为新的源，并将特定于源的汇的调用方标记为新的汇。例如，在 gpsTracker.setLatitude(100,100) 中，setLatitude 成为新的源，而 gpsTracker 成为新的汇。

受 Privado [4] 使用正则表达式识别 Java 应用程序中与《通用数据保护条例》相关数据的启发，下面的图2展示了一条示例 Semgrep 规则，该规则匹配账户 数据源进入运输**(T)**汇这一模式；图5则展示了检测到的示例代码片段。

4.3 标注

Semgrep 识别出的发现以不同长度的代码片段（由语句和表达式组成）的形式呈现。每项发现都包含

[3] https://w3c.github.io/dpv/dpv/

[4] https://www.privado.ai

```yaml
rules:
  - id: account-data-transportation
    languages:
      - javascript
      - java
      - typescript
    mode: taint
    message: Match found
    pattern-sinks:
      - patterns:
          - pattern: $SINK(..., $Z, ...)
          - metavariable-regex:
              metavariable: $SINK
              regex: (?i)(.*(send|move|connect|escap|stream|redirect|
              erase|query|share|stor|transfer|transmit|move).*)
    pattern-sources:
      - pattern-regex: (?i).*(?:account|user|customer|doctor|patient|
      policyholder|insurer|claimant)[^\\s/(;)|,=!>]{0,3}(id|number|no|
      num)
      - pattern-regex: (?i)(?:facebook|twitter|instagram|linkedin|
      pinterest|behance|dribble)[^\\s/(;)|,=!>]{0,2}(?:id|account|
      username|handle)
      - pattern-regex: (?i).*(?:db|database|jira|sql|postgres|mongo|aws)
      [^\\s/(;)|,=!>]{0,3}(psw|pswd|password|passwrd)
      - pattern-regex: (?i)(.*(?:db|database|jira|sql|postgres|mongo|
      aws)[^\\s/(;)|,=!>]{0,3}user[^\\s/(;)|,=!>]{0,3}name)|(.*(account|
      customer|doctor|patient|teacher|student|person|organi[zs]ation|
      company)[^\\s/(;)|,=!>]{0,3}name)
    severity: WARNING
```

图 2： Semgrep 规则：查找从账户数据源到运输汇的个人数据流

```
this.usersService.updateUser(newUser.id,
                             {defaultOrganizationId:
                              newUser.organizationId})
                               .catch((error) => {
      console.error('Error while updating default
                     organization id', error);
    });
```

图 3： Semgrep 检测到的代码片段（来自 ToolJet），展示了从账户个人数据到运输汇的流程。

至少一个检测到的汇和一个源（或一个从源接收值的对象）。我们使用以下符号来抽象每个代码片段中可能的源和汇的结构。

- O 遍历源
- I 遍历汇
- Io 遍历特定于源的汇

我们用 O 简写可能为空的序列 $O1, \cdots, O_n$。这里，下划线表示表达式的占位符，该表达式在隐私方面无关紧要——它既不是源也不是汇，也不包含来自源的值。

下面列出了我们在每个代码片段中观察到的源与汇之间常见的流抽象。每种抽象都代表一种典型的流，例如，①表示3 有值经过汇流向源，这些值可以来自非隐私敏感值（1），也可以来自另一个源（2），还可以是从另一个源对象内部引入一个汇（3 ）。

① $O = _.I(_)$
② $O_2 = _.I(O_1, _)$
③ $O_2 = _.O_1.I(_)$
④ $_ = _.O.I(_)$
⑤ $_ = _.I(\bar{O}, _)$
⑥ $_.O.I(_)$
⑦ $_.O.I(_, \bar{O})$
⑧ $_.I^O(_)$
⑨ $_.I^O(_, \bar{O})$
⑩ $_.I(\bar{O}, _)$

对于每个识别出的代码片段，我们使用 22 个标签对其进行标注（9 种源类型、5 种汇类型、5 种特定于源的汇类型，以及 3 种敏感级别变化类型），这些标签列于表 1 中。除了源与汇的定义外，我们还引入了一个重要标签：敏感级别。在一个识别出的代码片段中，敏感级别可以升高、降低或保持不变。

O九种源类型：$\{O^1, O^2, \ldots\ldots, O^9\}$ I五种汇类型：$\{I^1, I^2, \ldots\ldots, I^5\}$ Io五种特定于源的汇类型：$\{I^{O^1}, I^{O^2}, \ldots\ldots, I^{O^5}\}$ $\mathcal{S}$ 敏感级别变化：{上升、下降、相同}

表 1：分配给每个代码片段的标签

敏感级别并非所有结果在个人数据处理方面都具有相同的敏感级别。处理后，源中的数据可能保持相似的敏感级别、变得更敏感，或变得不那么敏感。

- $\mathcal{S} =$ `up`: ①, ④, ⑤
- $\mathcal{S} =$ `equal`: ②, ③, ⑥, ⑦, ⑧, ⑨
- $\mathcal{S} =$ `down`: ⑩

4.4 结果展示

Johnson 等人（Johnson 等人，2013年）指出：“由于结果被直接倾倒在代码审查员的屏幕上，且没有明确的结构，导致他们不得不花费大量时间弄清楚需要做什么。”这表明，如果静态分析工具以不合理的方式呈现未分组的代码片段，开发人员和代码审查人员可能无法从中受益。

为了解决这一问题，我们提出了一种两阶段技术，用于处理 Semgrep 的发现结果，并以智能的方式呈现给代码审查人员。

为每个代码片段添加标签后，我们开始根据其标签及其他标准对其进行分组并呈现。分组标准不仅包括标签，还包括其他属性：

- 相邻结果将合并（位于同一文件且行号差在阈值范围内）；

- 相同或相似的源/汇名称；
- 相同的应用程序接口用法（例如，所有与同一个应用程序接口 MongoDB 相关的代码片段）。

图4展示了结果呈现方式的直观示例。结果分为两个独立部分：纯文本结果和流结果。用户可以灵活选择任意标签或标签组合来筛选结果。

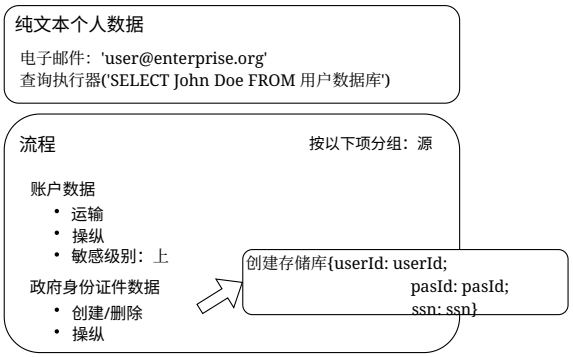

图4： 结果示例。个人数据出现位置位于顶部，个人数据处理代码位于底部。

5 演示

我们在 Semgrep 中创建了规则，尝试捕获尽可能多的有用发现，以供分析。我们在此分析的软件是 ToolJet[5]，，这是一个用于构建基于 React 的 Web 应用程序的开源低代码框架。ToolJet 的实现主要使用 JavaScript 和 TypeScript。用户可以使用 ToolJet 预构建的用户界面控件，连接数据库、API端点和外部服务等数据源，从而构建内部工具。这意味着 ToolJet 有许多处理个人数据的部分，因此它是一个很好的起始示例。

我们的 Semgrep 规则从 ToolJet 的源代码中共生成了 1,589 条结果。我们手动审查了每条结果，并计算了各类别的精确率。如果某条结果能够清楚地表明正在处理个人数据，我们就认为它与隐私代码审查相关，可能对此有所帮助。令人惊讶的是，大多数假阳性来自个人数据出现检测器（精确率仅为 46.6%），而大多数个人数据处理结果都与分析相关（对于识别出 50 个以上代码片段的类别，平均精确率为 90.9%）。

[5]https://github.com/ToolJet/ToolJet

详细统计数据见表 2 和表 3。

	M	T	C/D	DB	E	L
账户	66	171	84	24	-	21
联系人	89	175	36	3	-	3
个人身份标识	56	133	41	7	1	4
在线身份标识	6	26	1	-	-	1
位置	1	2	-	-	-	-

表2：各个已识别源与汇对应的代码片段数量；“-”表示我们的方法未检测到代码片段的标签。汇类型包括：数据操作**(M)**、数据传输 **(T)**、数据创建/删除 **(C/D)**、数据库 **(DB)**、加密**(E)**以及日志 **(L)**。

	M	T	C/D	DB	E	L
账户	90.9	90.6	95.2	91.67	-	95.2
联系人	89.9	94.9	80.6	*	-	*
个人身份标识	92.9	81.9	85.4	*	*	*
在线身份标识	*	84.6	*	-	-	*
位置	*	*	-	-	-	-

表3：每种已识别源与汇的代码片段相关性精确率（%），“-”表示我们的方法未检测到任何代码片段的标签，“*”表示我们的方法检测到少于 10 个结果的标签。汇类型包括：数据操作 **(M)**、数据传输 **(T)**、数据创建/删除 **(C/D)**、数据库 **(DB)**、加密**(E)** 和 日志 **(L)**。

图5展示了一个简单而有趣的分组结果示例，说明个人数据 userId 如何从*appusers.service.ts* 中的本地存储库中检索出来，然后用于生成许多数据结构，例如*app service.ts*中的 app 对象。

```
async create(user: User): Promise<App> {
  const app = await this.appsRepository.save(
    this.appsRepository.create({
      name: 'Untitled app',
      createdAt: new Date(),
      updatedAt: new Date(),
      organizationId: user.organizationId,
      userId: user.id,
    })
  );
```
app.service.ts

```
async create(user: User, appId: string, organizationUserId: string,
             role: string): Promise<AppUser> {
  const organizationUser = await this.organizationUsersRepository.
                findOne({ where: { id: organizationUserId } });

  return await this.appUsersRepository.save(
    this.appUsersRepository.create({
      appId,
      userId: organizationUser.userId,
      role,
      createdAt: new Date(),
      updatedAt: new Date(),
    })
  );
}
```
app_users.service.ts

图5： 分组示例结果展示了 organizationUserId 如何在函数之间流动。

5.1 未来工作

由于我们的目标是识别源代码中所有相关的个人数据处理，因此减少假阴性是我们的首要任务。然而，在我们的案例中，假阳性并非主要问题。个人数据处理具有微妙性，若没有人工协助，判断其相关性尤其困难。将分析范围限定于某些特定模式，有助于简化人工分析。因此，我们需要构建隐私分类体系。以 Ethyca 的分类体系（Ethyca，2022）为例，我们可以调整标签，使其与该分类体系采用的技术相匹配。

作为本文的延伸，我们提出在一篇尚未发表（正在评审中）的稿件（Tang 等，2023）中，自动映射个人数据，以协助开发人员和代码审查人员识别与隐私相关的代码。该映射基于静态分析，可自动检测个人数据及处理这些数据的代码，并为个人数据流提供语义信息。

6 结论

本文介绍了一项正在进行的工作，提出了一种新颖且可定制的方法，用于在代码审查中识别个人数据处理。这项分为三个阶段的技术，首先使用 Semgrep 根据源和汇规则匹配代码中的模式，然后将模式匹配生成的代码片段关联到一组行为标签，最后对结果进行分组，以减轻代码审查人员的工作负担。我们的演示表明，该方法能够有效且可行地从代码库中收集并呈现与个人数据处理相关的代码片段。

随着该方法架构的持续完善（细化源和汇规则、采用更有意义的标签，以及增加分组标准），未来工作将重点扩展案例研究，纳入来自不同领域的更多开源软件，并对由此产生的平台开展全面的用户评估。

致谢

本研究是 Privacy Matters (PriMa) 项目的一部分。PriMa 项目获得了欧盟地平线2020研究与创新计划的资助，资助依据为玛丽·斯克沃多夫斯卡-居里资助协议（编号：860315）。

参考文献

Bambauer, D. E.（2013）。隐私与安全。《刑法与犯罪学杂志》，103:667。Berˇciˇc, B. 和 George, C.（2009）。使用关系数据库设计原则识别个人数据。《国际法律与信息技术杂志》，17(3):233–251。Bertino, E.（2016）。数据安全与隐私：概念、方法与研究方向。载于《2016年IEEE第40届年度计算机软件与应用会议（COMPSAC）》，第1卷，第400–407页。IEEE。Blume, P.（2016）。欧盟《通用数据保护条例》对公共部门的影响。《数据保护与隐私杂志》，1(1):53–63。Braz, L. 和 Bacchelli, A.（2022）。现代代码审查期间的软件安全：开发人员视角。《arXiv预印本 arXiv:2208.04261》。Buse, R. P. 和 Zimmermann, T.（2012）。软件开发分析的信息需求。载于《2012年第34届国际软件工程会议（ICSE）》，第987–996页。IEEE。Ethyca（2022）。Fides 语言。https://ethyca.github.io/ fideslang/。（访问日期：2022年11月15日）。Finck, M. 和 Pallas, F.（2020）。不可识别之人——根据《通用数据保护条例》区分个人数据与非个人数据。《国际数据隐私法》，10(1):11–36。Fugkeaw, S., Chaturasrivilai, A., Tasungnoen, P. 和 Techaudomthaworn, W.（2021）。AP2I：自适应个人身份信息扫描与同意发现系统。载于《2021年第13届知识与智能技术国际会议（KST）》，第231–236页。IEEE。Hadar, I., Hasson, T., Ayalon, O., Toch, E., Birnhack, M., Sherman, S. 和 Balissa, A.（2018）。隐私设计者：软件开发人员的隐私意识。《实证软件工程》，23(1):259–289。Jain, P., Gyanchandani, M. 和 Khare, N.（2016）。大数据隐私：技术视角与综述。《大数据杂志》，3(1):1–25。Johnson, B., Song, Y., Murphy-Hill, E. 和 Bowdidge, R.（2013）。软件开发人员为何不使用静态分析工具查找软件缺陷？载于《2013年第35届国际软件工程会议（ICSE）》，第672–681页。IEEE。Lenhard, J., Fritsch, L. 和 Herold, S.（2017）。隐私模式研究的文献综述。载于《2017年第43届Euromicro软件工程与高级应用会议（SEAA）》，第194–201页。IEEE。McGraw, G.（2008）。用于安全性的软件自动化代码审查工具。《计算机》，41(12):108–111。McIntosh, S., Kamei, Y., Adams, B. 和 Hassan, A. E.（2014）。代码审查覆盖率和代码审查参与度对软件质量的影响：qt、vtk 和 itk 项目的案例研究。载于《第11届软件仓库挖掘工作会议论文集》，第192–201页。Notario, N., Crespo, A., Martˊın, Y.-S., Del Alamo, J. M., Le Mˊetayer, D., Antignac, T., Kung, A., Kroener, I. 和 Wright, D.（2015）。PRIPARE：整合隐私

最佳实践与隐私工程方法论。载于《2015年IEEE安全与隐私研讨会》，第151–158页。IEEE。Pandit, H. J., Polleres, A., Bos, B., Brennan, R., Bruegger, B., Ekaputra, F. J., Fern´andez, J. D., Hamed, R. G., Kiesling, E., Lizar, M. 等（2019）。创建数据隐私词汇表。载于《OTM联合国际会议：迈向有意义的互联网系统》，第714–730页。施普林格。Pormeister, K.（2017）。以“23andMe”为例，为履行数字消费者合同而处理敏感个人数据的知情同意。《欧洲消费者与市场法杂志》，6(1)。r2c（2022）。Semgrep。https://semgrep.dev/。（访问日期：2022年11月15日）。Ren, J., Rao, A., Lindorfer, M., Legout, A. 和 Choffnes, D.（2016）。ReCon：揭示并控制移动网络流量中的个人身份信息泄露。载于《第14届移动系统、应用与服务年度国际会议论文集》，第361–374页。Tang, F., Østvold, B. M. 和 Bruntink, M.（2023）。在源代码中映射个人数据以实现GDPR合规。van der Plas, N.（2022）。检测 Git 提交中的个人身份信息。《代尔夫特理工大学硕士论文》。Voss, W. G. 和 Houser, K. A.（2019）。个人数据与《通用数据保护条例》：为美国公司提供竞争优势。《美国商法杂志》，56(2):287–344。