学校代码：10246
学　　号：19212010035

復旦大學

硕 士 学 位 论 文

（学术学位）

基于多源知识的开源软件漏洞的补丁识别方法

Finding Patches for Open Source Software Vulnerabiliies from multiple sources

院　　系：　软件学院
专　　业：　软件工程
姓　　名：　许聪颖
指 导 教 师：　陈碧欢　副教授
完 成 日 期：2021 年 12 月 15 日

目 录

摘 要

开源软件 (Open source software, OSS) 漏洞管理已然成为一个热点问题。开源漏洞数据库为解决漏洞问题提供十分有价值的数据信息，因此，漏洞数据库的数据质量也受到越来越多的关注和研究。具体的问题为：现有漏洞数据库中补丁的质量尚未研究清楚，此外，现有的补丁信息多由人工或基于启发式的识别方法进行收集。这种方法人工成本过高，且过于定制化无法应用于全部的 OSS 漏洞。

empirical study 该如何翻译比价好呢？实证研究？经验性研究？为了解决这些问题，首先，我们进行了实证研究，以了解当前商业旗舰漏洞数据库中开源软件漏洞补丁的质量和特征。我们的研究涵盖五个方面，包括：补丁的覆盖度、一致性、类型、基数和准确性。然后，基于研究的发现，我们提出了第一种名为 TRACER 的自动化方法，用于从多个来源查找开源漏洞的补丁。

实验评估表明：i) 与现有的基于启发式的方法相比，TRACER 能够为多达 273.8% 的 CVE 找到补丁；同时，准确性方面，将 F1 数值提高达 116.8%；ii) 与现有的漏洞数据库相比，TRACER 将召回率（recall）提高达 18.4% ；然而，12.0% 的 CVE 补丁未找到，精度（precision）下降约6.4%。

关键字：关键词 1，关键词 2，关键词 3

中图分类号：TP311

Abstract

Open source software (OSS) vulnerability management has become an open problem. Vulnerability databases provide valuable data that is needed to address OSS vulnerabilities. However, there arises a growing concern about the information quality of vulnerability databases. In particular, it is unclear how the quality of patches in existing vulnerability databases is. Further, existing manual or heuristic-based approaches for patch identification are either too expensive or too specific to be applied to all OSS vulnerabilities.

To address these problems, we first conduct an empirical study to understand the quality and characteristics of patches for OSS vulnerabilities in two state-of-the-art vulnerability databases. Our study is designed to cover five dimensions, i.e., the coverage, consistency, type, cardinality and accuracy of patches. Then, inspired by our study, we propose the first automated approach, named TRACER, to find patches for an OSS vulnerability from multiple sources. Our key idea is that patch commits will be frequently referenced during the reporting, discussion and resolution of an OSS vulnerability.

Our extensive evaluation has indicated that i) TRACER finds patches for up to 273.8% more CVEs than existing heuristic-based approaches while achieving a significantly higher F1-score by up to 116.8%; and ii) TRACER achieves a higher recall by up to 18.4% than state-of-the-art vulnerability databases, but sacrifices up to 12.0% fewer CVEs (whose patches are not found) and 6.4% lower precision. Our evaluation has also demonstrated the generality and usefulness of TRACER.

Keywords: Keyword1, Keyword2, Keyword3

CLC code: TP311

第 1 章 绪论

本章节概述了背景、研究目的与意义[1–2]。

1.1 研究背景

1.2 本文工作概述

1.3 本文篇章结构

如图1-1所示。如表1-1所示。

表 1-1 表格名称

AA \ BB	C1	C2
R1	1	2
R2	3	4

图 1-1 在上海混口饭吃真不容易啊

第 2 章 背景知识及相关工作【v1-doing】

本文的研究重点是开源软件漏洞的补丁定位问题，本章将首先介绍漏洞相关的背景知识，包括：通用漏洞披露（CVE）、漏洞通告（advisory）和漏洞补丁（vulnerability patch）；然后，从漏洞披露信息的质量、漏洞补丁的分析及应用三个方面介绍相关的研究工作。

2.1 背景知识

2.1.1 通用漏洞披露 (CVE)

通用漏洞披露 (Common Vulnerabilities and Exposures，CVE)[2]，是一个与网络安全有关的漏洞字典，收集各种信息安全漏洞并给予唯一编号以便于公众查阅及引用。

如图例2-1所示，每一个 CVE 条目都有唯一通用标识符（即：CVE-ID）、一段漏洞描述（即：Description）以及至少一个参考链接（即：References），该参考链接多为外部网站且包含与该漏洞相关的更详细的描述信息。

基于 CVE 收录的漏洞条目信息，美国国家漏洞数据库（NVD）、中国国家信息安全漏洞库（CNNVD）等与 CVE 数据完全同步的漏洞数据库被构建，用于为每个 CVE 条目提供更丰富的信息，如：修复信息、严重性评分、影响评级等。

2.1.2 漏洞通告【todo】

漏洞通告（advisory），也称漏洞警报。。。。是指。。。官方或第三方？再顺路引出相关的 sources，作为例子。

2.1.3 漏洞补丁【todo】

是指。。。。.Patch Git, SVN commit

CVE-ID

CVE-2017-11428 — Learn more at National Vulnerability Database (NVD)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

OneLogin Ruby-SAML 1.6.0 and earlier may incorrectly utilize the results of XML DOM traversal and canonicalization APIs in such a way that an attacker may be able to manipulate the SAML data without invalidating the cryptographic signature, allowing the attack to potentially bypass authentication to SAML service providers.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- MISC:https://duo.com/blog/duo-finds-saml-vulnerabilities-affecting-multiple-implementations
- MISC:https://www.kb.cert.org/vuls/id/475445

Assigning CNA

Duo Security, Inc.

Date Record Created

20170718 — Disclaimer: The record creation date may reflect when the CVE ID was allocated or reserved, and does not necessarily indicate when this vulnerability was discovered, shared with the affected vendor, publicly disclosed, or updated in CVE.

Phase (Legacy)

Assigned (20170718)

Votes (Legacy)

Comments (Legacy)

Proposed (Legacy)

N/A

This is a record on the CVE List, which provides common identifiers for publicly known cybersecurity vulnerabilities.

SEARCH CVE USING KEYWORDS: Submit
You can also search by reference using the CVE Reference Maps.

For More Information: CVE Request Web Form (select "Other" from dropdown)

图 2-1　CVE-2017-11428

2.2　相关工作

2.2.1　漏洞信息质量

漏洞数据库（例如：CVE、NVD）被广泛关注，其中的漏洞信息也被广泛参考使用。随着漏洞数据库积累的漏洞数据越来越多，研究人员也越来越关注其中的漏洞信息的质量。

Nguyen 和 Massacci[3]最早揭示了 NVD 数据库中漏洞所影响的软件版本信息的不可靠性。为了提高该信息的可靠性，Nguyen[4]和 Dashevskyi 等人[5]开发了工具以确定某一旧软件版本是否会受到新披露的漏洞的影响。他们认为：如果旧版本包含修复漏洞所更改的源代码行，则该版本被视为受漏洞影响。Dong 等人[6]从漏洞的描述信息中识别受漏洞影响的软件名称和版本，并与漏洞报告所提供的软件名称和版本信息进行对比。他们发现漏洞数据库中会遗漏真正受漏洞影响的版本，也会错误地包含了不受漏洞影响的版本。Chen 等人[7]识别受漏洞影响的开源库信息。Chaparro 等人的工作[8]检测漏洞描述中是否缺少用于重现漏洞的关键步骤或预期效果信息。Mu 等人的工作[9]揭示了漏洞报告丢失重现漏洞信息的普遍性。以上工作已侧重于漏洞信息的多个方面，而本文的工作重点则是研究漏洞的补丁信息，并尝试自动化地从不同来源漏洞报告的综合信息中

定位漏洞补丁。

近期，Tan 等人完成了一项与本文的研究问题相似的工作[10]。他们使用深度学习排名算法对代码仓库中的提交（commit）历史进行排名，把排在首位的提交当作为漏洞的补丁提交。他们的工作包含两个假设：（1）CVE 中，受漏洞影响的软件的代码仓库已知；（2）漏洞与其补丁提交在数量上是一对一的映射关系。然而，事实上，受漏洞影响的软件的代码仓库并不已知，而需人工识别；此外，漏洞与其补丁提交在数量上存在一对多的关系（Sec.3.6）。

2.2.2 漏洞补丁分析

当前，有多中补丁分析相关的任务可用于提高软件安全性，如：补丁的生成和部署[11–13]、补丁的存在性测试[14–16]以及秘密补丁识别[17–20]。

此外，研究人员也已为 Java[21]、C/C++[22]以及特定开源项目[23]构建安全补丁数据集。基于这些数据集，研究人员已开展实证研究以表征漏洞及其补丁[24–27]。在这些工作中，补丁信息多由人工识别[13,15–21,24]，或通过启发式规则识别，例如：在 CVE 的引用链接中查找补丁提交信息[12,22–23,25–26]，以及在提交信息（commit message）中搜索 CVE 标识符[22–23,27]。这些工作存在的问题为：通过人工收集成本过高，且耗时较长；然而，启发式规则的方法又不足以找到或是找全补丁。

2.2.3 漏洞补丁应用

漏洞补丁信息可被用于多种软件安全性任务。例如，基于漏洞补丁生成漏洞攻击程序[28–29]，通过软件成分分析以确定项目是否使用包含漏洞的第三方库，并判定该漏洞所影响的函数是否被调用[30–33]，以及通过学习漏洞特征[34–37]、通过匹配漏洞签名[38–39]、通过匹配漏洞和补丁检测签名[40–42]来检测程序中的漏洞。

与上一小结的补丁分析工作类似，这些工作中的 CVE 补丁主要通过人工识别[30–32,41]、基于启发式规则的方法[29,33–35,37]或直接取自为特定项目建立 CVE 和补丁之间映射关系的安全公告[38–40]。但是，人工识别的成本过高，而且基于启发式规则的方法找到或是找全补丁。

第 3 章 经验研究

为了了解已有漏洞数据库中开源软件漏洞补丁的质量和特征，本文开展了一项针对当前高质量漏洞数据库的经验研究。本章节将介绍经验研究的设计、数据收集与解析以及经验研究的结果分析。

3.1 研究设计

3.1.1 研究问题

- **RQ1 覆盖率分析：**漏洞库中，漏洞补丁信息的覆盖度如何？即：有多少漏洞包含补丁信息？(Sec. 3.3)
- **RQ2 一致性分析：**不用漏洞库间，漏洞补丁信息的一直性如何？即：有多少漏洞在漏洞数据库中具有相同的补丁？(Sec. 3.4)
- **RQ3 补丁类型分析：**漏洞补丁的类型有哪些？(Sec. 3.5)
- **RQ4 补丁映射分析：**漏洞与其补丁在数量上的映射关系是怎样的？(Sec. 3.6)
- **RQ5 补丁准确性分析：**漏洞库中，漏洞的补丁信息准确度如何？(Sec. 3.7)

其中，RQ1 可用来衡量不同漏洞数据库中开源软件漏洞的补丁缺失程度，RQ2 用来量化不同漏洞数据库中漏洞补丁的不一致程度，RQ3 和 RQ4 用来统计常见的补丁类型以及开源漏洞及其补丁之间的映射关系，RQ5 用以评估不同漏洞数据库中漏洞补丁信息的准确性。总的来说，RQ1、RQ2 和 RQ5 的结果旨在从不同的角度评估补丁质量，并挖掘出对自动化补丁识别方法的需求；RQ3 和 RQ4 旨在从不同角度捕捉开源软件漏洞补丁的特征，并为自动化补丁识别方法的设计提供启发。

3.1.2 评估标准【todo】

precision、recall。。。

3.2 数据准备

3.2.1 漏洞数据库选择

本文前期调研了来自安全社区、工业界和学术界的漏洞数据库。在该章节的经验研究中，本文首先排除了来自安全社区的数据库（例如，CVE 和 NVD）。因为它们没有结构化的补丁信息，而补丁信息多是隐藏在参考链接中；此外，CVE

和 NVD 数据库不仅仅包含开源软件漏洞，还包括闭源软件、系统及硬件相关的漏洞。本文还排除了来自学术界的数据集[21–23,25–27,43–44]，这是因为这些数据集中的漏洞通常限定于特定程序语言（例如：Python、Java）而非面向所有开源软件，此外，由于长期缺乏维护，这些漏洞数据集缺失较新的漏洞数据。

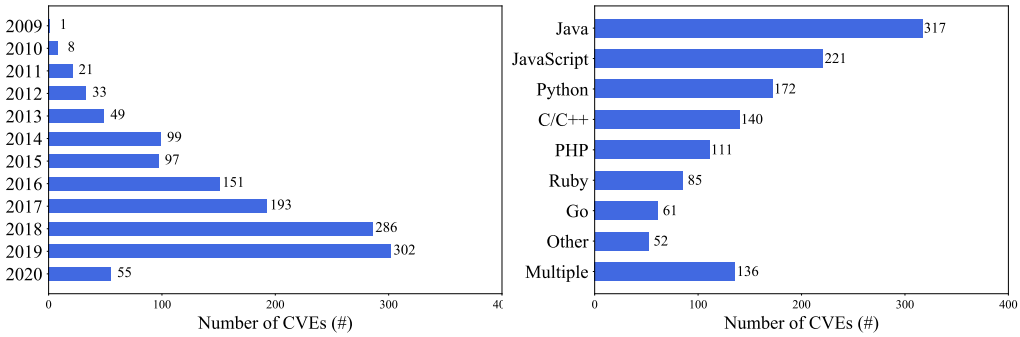
本文关注到 BlackDuck[45]、WhiteSource[46]、Veracode[47] 和 Snyk[48] 四家公司提供软件成分分析（Software Composition Analysis）服务，该服务识别当前软件系统中使用的开源成本（即：第三方库），并报告开源成分相关的漏洞。因此，这四家公司需要构建尽可能完整且包含详细漏洞信息的漏洞库作为基础，本文便首先将这四家公司的漏洞数据库作为研究对象。进一步调研发现，某些公司并未公开漏洞数据库，或是公开的信息中不包含漏洞的补丁信息。最终，本文选定 Veracode 和 Snyk 的漏洞数据库作为研究对象，下文中简称为：DB_A 和 DB_B。

- **Black Duck：**该公司的安全公告中共包含 157,000 多个漏洞，涵盖 90 多种编程语言。其中，数千个漏洞尚未被 NVD 收录。这些漏洞信息有特定的专家团队进行维护，以确保漏洞数据的完整性和准确性。然而，157,000 多个漏洞中开源漏洞的具体数量并未公开，且具体的漏洞信息也并未公开。
- **Sonatype：**该公司声称：*"OSS Index is a free catalogue of open source components and scanning tools to help developers identify vulnerabilities, understand risk, and keep their software safe."* Sonatype 的 OSS 索引支持 20 多个生态系统（即：Maven、npm、Go、PyPI 等）。该公司的漏洞公告包含：漏洞描述、受漏洞影响的组件和版本、CVSS 向量和参考链接等信息①。
- **WhiteSource：**该公司从 NVD 及其他安全公告平台和问题追踪系统（issue tracking system）中收集的漏洞超过 175,000 个，涵盖 200 多种编程语言。
- **Veracode：**该公司的漏洞数据库涵盖 10 多种编程语言相关的 18,000 多个漏洞链接，公开的漏洞信息包括：受漏洞影响的组件和版本范围、库修复说明、参考资料等。
- **Snyk：**该公司的漏洞数据库②是由经验丰富的安全研究团队持续维护，通过关注安全公告、Jira issue 报告，Github commits 等方式自动识别安全漏洞相关的报告。该公司的数据库涵盖超过 10 个编程语言生态系统，如：Maven、npm、Go、Composer 等。该数据库提供漏洞的详细信息，包括：受漏洞影响的组件、版本范围、修复方法、参考链接等③。

① https://ossindex.sonatype.org

② https://snyk.io/product/vulnerability-database/

③ https://snyk.io/vuln



(a) 漏洞年份分布统计 (b) 程序语言分布统计

图 3-1 数据集中漏洞年份及程序语言分布统计

3.2.2 广度数据集构建

为了量化分析漏洞数据库中补丁的缺失程度以及不同数据库间补丁的不一致性（即：RQ1 和 RQ2），本文基于 DB_A 和 DB_B 构建了一个开源软件漏洞的广度数据集用以实验分析。截至 2020 年 4 月 7 日，分别从 DB_A 和 DB_B 中获取了8,630和5,858个 CVE 漏洞。

3.2.3 深度数据集构建

为了准确地量化分析漏洞补丁的类型、映射关系和准确性（即：RQ3、RQ4 和 RQ5），本文还构建了开源软件漏洞的深度数据集。该数据集的漏洞数量少于广度数据集，但每个漏洞都包含补丁信息。为了确保漏洞补丁信息完整性和准确性，其中所有漏洞的补丁信息都是由人工识别。

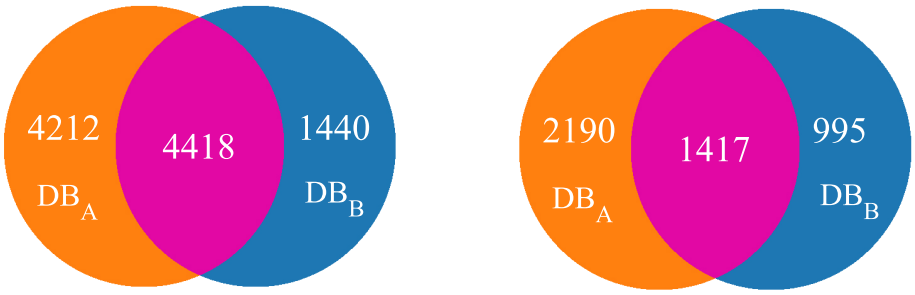
在构建过程中，为了确保数据集能够涵盖足够多的漏洞且不至于在人工手动识别补丁的环节中产生难以完成的工作量，本文将在 DB_A 和 DB_B 都含有补丁信息的漏洞作为研究对象，最终挑选了1,417个 CVE 漏洞。对于每个 CVE 漏洞，首先，两位研究人员通过分析 DB_A 和 DB_B 数据库报告的补丁、查看 NVD 中的漏洞描述和参考链接信息以及搜索 GitHub 代码仓库的提交历史和其他网络资源，分别找到了其补丁；然后，两位研究人员再一起分析补丁结果不一致的漏洞数据直到达成共识。由于公开的信息有限，1,417个 CVE 漏洞中的122个 CVE 漏洞无法找到补丁信息，最终，深度数据集共包含了1,295个 CVE 漏洞。

本文进一步分析了深度数据集中1,295个 CVE 开源软件漏洞的年份和程序语言分布情况，以评估该数据集是否具有代表性。如图3-1a所示，CVE 的数量逐年增加，这与 Snyk 的报告[49]一致。此外，本文通过分析补丁中更改的源文件类型来确定 CVE 的编程语言。如图3-1b所示，深度数据集中的 CVE 漏洞涵盖了七种较为常用的程序语言，具有较好的语言多样性。因此，可以认为该深度数据集对于开源软件漏洞具有较好的代表性。

表 3-1　补丁一致性结果

补丁一致	存在性不一致			内容不一致		
	总数	无漏洞信息	无补丁信息	总数	包含关系	非包含关系
907 （19.7%）	3,185 （69.2%）	1,392 （30.2%）	1,793 （39.0%）	510 （11.1%）	176 （3.8%）	334 （7.3%）

3.3　RQ1：覆盖率分析



(a) 开源软件漏洞　　(b) 含补丁信息的开源软件漏洞

图 3-2　DB_A 与 DB_B 间数据交集

如图3-2a所示，DB_A 和 DB_B 数据库中共有的 CVE 漏洞为4,418个，同时 DB_A 和 DB_B 分别包含4,212和1,440个特有的 CVE 漏洞。如图3-2b所示，DB_A 中3,607(41.8%)个 CVE 漏洞含有补丁信息，DB_B 中2,412(41.2%)个 CVE 漏洞含有补丁信息。总体而言，DB_A 和 DB_B 数据库共有10,070个开源软件 CVE 漏洞，其中4,602(45.7%)个漏洞至少有一个数据库提供了补丁信息。

由此可见，数据库 DB_A 和 DB_B 中漏洞的补丁覆盖率都较低（41.8% 和 41.2%），漏洞补丁缺失的情况较为普遍。同时，这也体现出自动化补丁查找方法的必要性，用以填补缺失的补丁信息。

3.4　RQ2：一致性分析

为了分析两个数据库之间的补丁一致性情况,本节主要关注带有补丁的CVE漏洞，即：图3-2b中的 CVE 漏洞。考虑到漏洞补丁的个数可能不唯一，可能为一组补丁信息，当两个数据库针对同一漏洞提供的补丁集完全相同时，判定为补丁信息一致。本文将补丁信息不一致分为存在性不一致和内容不一致。前者是指某一个数据库为该 CVE 漏洞提供了补丁信息，而另一个数据库不存在该 CVE，或是存在该 CVE 但不存在补丁信息，这反映了出数据库 DB_A 和 DB_B 中开源软件漏洞及其补丁信息的不完整性。后者是指两个数据库都存在该 CVE 的补丁信息，但它们的补丁集并不一致，是包含关系或非包含关系的不一致情况，这反映了出数据库 DB_A 和 DB_B 中漏洞补丁信息可能是不准确的。

表3-1中展示了补丁一致性的分析结果。其中，第一列为在 DB_A 和 DB_B 中具有一致补丁集的 CVE 的数量（907，19.7%），第二到第四列为补丁存在性不一致的 CVE 数量，最后三列为补丁集内容不一致的 CVE 数量。可以发现：4,602个 CVE 中，（1）只有907(19.7%)的漏洞在 DB_A 和 DB_B 中有一致的补丁信息；（2）超过三分之二（即：3,185(69.2%)）的 CVE 漏洞补丁信息存在不一致情况，其中1,392(30.2%)个 CVE 漏洞不在 DB_A 或 DB_B 中，1,793(39.0%)个 CVE 漏洞都存在于 DB_A 或 DB_B 但在没有补丁信息；（3）510(11.1%)个 CVE 漏洞补丁内容不一致，其中，176(3.8%)个 CVE 的来自于某一个数据库的补丁集包含来自另一个数据库的补丁集，334(7.3%)个 CVE 的来自 DB_A 和 DB_B 的补丁集即不同也不包含。

这些结果表明，DB_A 和 DB_B 间存在较多不一致的补丁信息，进而证明数据库中补丁信息的准确性也值得进一步研究。

3.5 RQ3：补丁类型分析

通过人工手动查找，深度数据集中1,295个 CVE 漏洞共收集到3,043个补丁。具体来说，可能是因为 GitHub 在开源软件中被广泛使用，2,852(93.7%)的补丁都是为 GitHub commit 形式，136(4.5%)的补丁为 SVN commit 形式，且仅有55(1.8%)的补丁为来自其他 Git 平台的 commit 形式。

此外，从 CVE 的角度统计来看，1,295个 CVE 中1,202(92.8%)的 CVE 有 GitHub commit 类型的补丁，4(0.3%)的 CVE 有 SVN commit 类型的补丁。由于由于很多项目是从 SVN 切换为 Git 管理，48(3.7%)的 CVE 即有 GitHub commit 又有 SVN commit 类型的补丁。只有30(2.3%)的 CVE 的补丁全为来自其他 Git 平台的 commit 形式。这些结果表明，开源软件漏洞的补丁类型主要为 GitHub 和 SVN commit。

3.6 RQ4：补丁映射分析

基于深度数据集中1,295个 CVE 漏洞及其补丁数据，本节将分析开源漏洞及其补丁间在数量上的映射关系。本文将 CVE 与其补丁之间的映射关系分为三种类型，即：一对一、一对一组及一对多。

一对一是指一个补丁来修复某个漏洞，后文中简记为：*SP*（Single Patch）。如图3-3所示。567(43.8%)的 CVE 与其补丁具有一对一的映射关系（*SP*）

一对一组是指 CVE 与其补丁在数量上非一对一关系，一个漏洞有多个 commit 信息，然而，这些 commit 都是等效的，任何一个补丁都足以修补漏洞，后文中简记为：*MEP*（Multiple Equivalent Patch）。等效的补丁是指代码变更完全一样的两个 commit，主要有两个原因：（1）通过 GitHub 中的 Pull Request 功

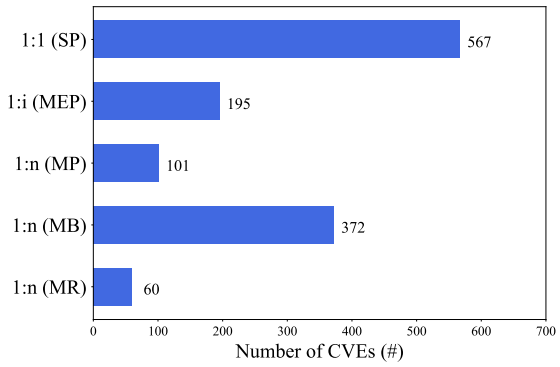


图 3-3 Mapping Cardinalities between CVEs and Patches

能修补 CVE，此时拉取请求提交（requested commit）和合并提交（merged commits）是该漏洞的等效补丁集。例如，python-jose@89b463[50]是拉取请求提交（requested commit），python-jose@73007d[51]是用于修复 CVE-2016-7036 的合并提交（merged commits），这两个 commit 是等效的。（2）一些开源软件的仓库是由 SVN 迁移到 GitHub，因此，在同一软件的 SVN 和 GitHub 的代码仓库中分别有用于修补该 CVE 的 commit，且这两处的 commit 中代码变更完全一样且完全等效的。例如，james-hupa 代码仓库从 SVN 迁移到了 GitHub，SVN commit james-hupa@1373762[52]与 GitHub commit james-hupa@aff28a[53]是等效的。如图3-3所示，567(43.8%)的 CVE 与其补丁具有一对一的映射关系（*SP*），195(15.1%)的 CVE 与其补丁为一对一组映射关系（*MEP*）

一对多是指 CVE 与其补丁在数量上为一对多的关系，即：多个非等效的补丁用来修复某个漏洞。如图3-3所示，533(41.2%)的 CVE 与其补丁为一对多的映射关系，可以进一步分为三种类型:

- 一个 CVE 是通过一个分支中的多个独立 commit 来修复的。这是因为该 CVE 较难修复需多次提交，或是后期发现初始的补丁不足以修复漏洞便追加了补丁。后文中简记为：*MP*（Multiple Patch），占比7.8%（101）。例如，CVE-2017-17837 由三个 commit deltaspike@4e2502[54]、deltaspike@72e607[55]和 deltaspike@d95abe[56]修复。
- 一个 CVE 由多个分支中的多个补丁集修复。这是因为该漏洞影响开源软件的多个版本，每个版本又都在单独的分支上维护。后文中简记为：*MB*（Multiple Branches），占比28.7%（372)。例如，CVE-2019-19118 影响了 django 框架的 2.1.x、2.2.x、3.0.x 和 3.2.x 版本，commit django@103ebe[57]、django@36f580[58]、django@092cd6[59]和 django@11c5e0[60]分别修复了受影响的四个版本分支中，其中，commit django@103ebe 与其他 commit 的代码变更并不相同。
- 一个 CVE 由多个存储库中的多个补丁集修复。这是因为 CVE 影响了多个开源软件或一个开源库的多个版本，而这些版本是分布在独立的代码仓库中维护。

表 3-2 DB_A 和 DB_B 补丁准确性评估结果

映射类型	数量	DB_A			DB_B		
		Pre.	Rec.	F1	Pre.	Rec.	F1
1:1 (SP)	567	0.908	0.915	0.910	0.900	0.921	0.906
1:*i* (MEP)	195	0.935	0.898	0.902	0.924	0.909	0.906
1:*n* (MP)	101	0.923	0.483	0.616	0.911	0.520	0.638
1:*n* (MB)	372	0.941	0.510	0.620	0.932	0.436	0.555
1:*n* (MR)	60	0.913	0.610	0.695	0.964	0.526	0.636
Total	1,295	0.923	0.748	0.793	0.917	0.730	0.771

文中简记为：*MR*（Multiple Repositories ），占比4.6%（60)。例如，CVE-2016-5104 影响了 libimobiledevice 和 libusbmuxd 两个开源软件，commit libimobiledevice@df1f5c[61] 和 libusbmuxd@4397b3[62] 分修复了受影响的两个开源库。

这些结果展示了 CVE 及其补丁之间映射关系的多样性。在后文设计自动化补丁查找方法时，应充分考虑该特征。

3.7 RQ5：补丁准确性分析

本文使用精度（precision）、召回率（recall）和 F1 值（F1-score）作为评估补丁准确性的指标。对于具有两个等效补丁的 CVE，报告两个等效补丁之一的数据库的精度和召回率都为 1，而报告两个等效补丁之一和另一个不相关补丁的数据库的精度为 1 和召回率为 0.5。

如表3-2所示，分解了两个数据库在映射基数方面的准确性结果。第一列为 CVE 与补丁的映射类型，第二列为每种映射类型的 CVE 数量，最后六列分别为数据库 DB_A 和 DB_B 中 CVE 补丁的准确率、召回率和 F1 值。其中，DB_A 和 DB_B 对于 *SP* 和 *MEP* 类型的 CVE 可实现约 90% 的精度和召回率。同时，对于 *MP*、*MB* 和 *MR* 类型的 CVE，可达到 90% 以上的高精度，但约 50% 的召回率。例如，对于漏洞 CVE-2017-17837，DB_A 和 DB_B 仅报告三个补丁中的一个；对于漏洞 CVE-2019-19118，DB_A 报告四个补丁中的两个，而 DB_B 仅报告四个补丁中的一个。

这些结果表明，漏洞数据库 DB_A 和 DB_B 经常会遗漏一些漏洞的补丁信息，尤其是对于具有多个补丁的 CVE 漏洞。对于安全服务用户来说，这会给漏洞的及时检测和修复带来较大挑战，这也反映出自动化补丁查找方法的必要性。

第 4 章　TRACER：基于多源信息的漏洞补丁定位方法

本章节系统实现。

4.1　方法概述

4.2　步骤一：构建多源信息网络

4.3　步骤二：精选补丁节点

4.4　步骤三：补丁扩增

第 5 章 实验验证及结果分析

本章节实验评估。

5.1 实验设计

5.2 RQ6：准确性验证

5.3 RQ7：削弱性分析

5.4 RQ8：敏感度分析

5.5 RQ9：通用性分析

5.6 RQ10: 实用性能分析

5.7 讨论

第 6 章 相关工作

第 7 章 总结与展望

本章节总结与展望。

7.1 本文总结

本文总结本文总结本文总结本文总结本文总结本文总结本文总结本文总结

7.2 未来展望

未来展望未来展望未来展望未来展望未来展望未来展望未来展望未来展望

参考文献

[1] 贾培养, 孙鸿宇, 曹婉莹, 等. 开源软件漏洞库综述[J]. 信息安全研究, 2021: 566-574.

[2] MITRE. About cve[EB/OL]. 2021. https://cve.mitre.org/.

[3] NGUYEN V H, MASSACCI F. The (un) reliability of nvd vulnerable versions data: An empirical experiment on google chrome vulnerabilities[C]//Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. 2013: 493-498.

[4] NGUYEN V H, DASHEVSKYI S, MASSACCI F. An automatic method for assessing the versions affected by a vulnerability[J]. Empirical Software Engineering, 2016, 21(6): 2268-2297.

[5] DASHEVSKYI S, BRUCKER A D, MASSACCI F. A screening test for disclosed vulnerabilities in foss components[J]. IEEE Transactions on Software Engineering, 2019, 45(10): 945-966.

[6] DONG Y, GUO W, CHEN Y, et al. Towards the detection of inconsistencies in public security vulnerability reports[C]//Proceedings of the 28th USENIX Security Symposium. 2019: 869-885.

[7] CHEN Y, SANTOSA A E, SHARMA A, et al. Automated identification of libraries from vulnerability data[C]//Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice. 2020: 90-99.

[8] CHAPARRO O, LU J, ZAMPETTI F, et al. Detecting missing information in bug descriptions[C]//Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. 2017: 396-407.

[9] MU D, CUEVAS A, YANG L, et al. Understanding the reproducibility of crowd-reported security vulnerabilities[C]//Proceedings of the 27th USENIX Security Symposium. 2018: 919-936.

[10] TAN X, ZHANG Y, MI C, et al. Locating the security patches for disclosed oss vulnerabilities with vulnerability-commit correlation ranking[C]//Proceedings of the 28th ACM Conference on Computer and Communications Security. 2021.

[11] MULLINER C, OBERHEIDE J, ROBERTSON W, et al. Patchdroid: Scalable third-party security patches for android devices[C]//Proceedings of the 29th Annual Computer Security Applications Conference. 2013: 259-268.

[12] DUAN R, BIJLANI A, JI Y, et al. Automating patching of vulnerable open-source software versions in application binaries.[C]//Proceedings of the 26th Annual Network and Distributed System Security Symposium. 2019.

[13] XU Z, ZHANG Y, ZHENG L, et al. Automatic hot patch generation for android kernels[C]//Proceedings of the 29th USENIX Security Symposium. 2020: 2397-2414.

[14] ZHANG H, QIAN Z. Precise and accurate patch presence test for binaries[C]// Proceedings of the 27th USENIX Security Symposium. 2018: 887-902.

[15] JIANG Z, ZHANG Y, XU J, et al. Pdiff: Semantic-based patch presence testing for downstream kernels[C]//Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020: 1149-1163.

[16] DAI J, ZHANG Y, JIANG Z, et al. Bscout: Direct whole patch presence test for java executables[C]//Proceedings of the 29th USENIX Security Symposium. 2020: 1147-1164.

[17] XU Z, CHEN B, CHANDRAMOHAN M, et al. Spain: security patch analysis for binaries towards understanding the pain and pills[C]//Proceedings of the IEEE/ACM 39th International Conference on Software Engineering. 2017: 462-472.

[18] ZHOU Y, SHARMA A. Automated identification of security issues from commit messages and bug reports[C]//Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. 2017: 914-919.

[19] SABETTA A, BEZZI M. A practical approach to the automatic classification of security-relevant commits[C]//Proceedings of the IEEE International Conference on Software Maintenance and Evolution. 2018: 579-582.

[20] CHEN Y, SANTOSA A E, YI A M, et al. A machine learning approach for vulnerability curation[C]//Proceedings of the 17th International Conference on Mining Software Repositories. 2020: 32-42.

[21] PONTA S E, PLATE H, SABETTA A, et al. A manually-curated dataset of fixes to vulnerabilities of open-source software[C]//Proceedings of the IEEE/ACM 16th International Conference on Mining Software Repositories. 2019: 383-387.

[22] FAN J, LI Y, WANG S, et al. Ac/c++ code vulnerability dataset with code changes and cve summaries[C]//Proceedings of the 17th International Conference on Mining Software Repositories. 2020: 508-512.

[23] JIMENEZ M, LE TRAON Y, PAPADAKIS M. Enabling the continous analysis of security vulnerabilities with vuldata7[C]//Proceedings of the IEEE International Working Conference on Source Code Analysis and Manipulation. 2018: 56-61.

[24] ZAMAN S, ADAMS B, HASSAN A E. Security versus performance bugs: a case study on firefox[C]//Proceedings of the 8th working conference on mining software repositories. 2011: 93-102.

[25] LI F, PAXSON V. A large-scale empirical study of security patches[C]// Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 2201-2215.

[26] LIU B, MENG G, ZOU W, et al. A large-scale empirical study on vulnerability distribution within projects and the lessons learned[C]//Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering. 2020: 1547-1559.

[27] ANTAL G, KELETI M, HEGEDŰS P. Exploring the security awareness of the python and javascript open source communities[C]//Proceedings of the 17th International Conference on Mining Software Repositories. 2020: 16-20.

[28] BRUMLEY D, POOSANKAM P, SONG D, et al. Automatic patch-based exploit generation is possible: Techniques and implications[C]//Proceedings of the IEEE Symposium on Security and Privacy. 2008: 143-157.

[29] YOU W, ZONG P, CHEN K, et al. Semfuzz: Semantics-based automatic generation of proof-of-concept exploits[C]//Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. 2017: 2139-2154.

[30] PASHCHENKO I, PLATE H, PONTA S E, et al. Vulnerable open source dependencies: Counting those that matter[C]//Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. 2018: 1-10.

[31] PONTA S E, PLATE H, SABETTA A. Detection, assessment and mitigation of vulnerabilities in open source dependencies[J]. Empirical Software Engineering, 2020, 25(5): 3175-3215.

[32] PASHCHENKO I, PLATE H, PONTA S E, et al. Vuln4real: A methodology for counting actually vulnerable dependencies[J]. IEEE Transactions on Software Engineering, 2020.

[33] Wang Y, Chen B, Huang K, et al. An empirical study of usages, updates and risks of third-party libraries in java projects[C]//ICSME. 2020: 35-45.

[34] LI Z, ZOU D, XU S, et al. Vulpecker: an automated vulnerability detection system based on code similarity analysis[C]//Proceedings of the 32nd Annual Conference on Computer Security Applications. 2016: 201-213.

[35] LI Z, ZOU D, XU S, et al. Vuldeepecker: A deep learning-based system for vulnerability detection[C]//Proceedings of the 25th Annual Network and Distributed System Security Symposium. 2018.

[36] ZHOU Y, LIU S, SIOW J, et al. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks[C]// Proceedings of the 32nd Annual Conference on Neural Information Processing Systems. 2019: 10197-10207.

[37] JIMENEZ M, RWEMALIKA R, PAPADAKIS M, et al. The importance of accounting for real-world labelling when predicting software vulnerabilities[C]// Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2019: 695-705.

[38] JANG J, AGRAWAL A, BRUMLEY D. Redebug: finding unpatched code clones in entire os distributions[C]//Proceedings of the IEEE Symposium on Security and Privacy. 2012: 48-62.

[39] KIM S, WOO S, LEE H, et al. Vuddy: A scalable approach for vulnerable code clone discovery[C]//Proceedings of the IEEE Symposium on Security and Privacy. 2017: 595-614.

[40] XU Y, XU Z, CHEN B, et al. Patch based vulnerability matching for binary programs[C]//Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2020: 376-387.

[41] XIAO Y, CHEN B, YU C, et al. {MVP}: Detecting vulnerabilities using patch-enhanced vulnerability signatures[C]//Proceedings of the 29th USENIX Security Symposium. 2020: 1165-1182.

[42] CUI L, HAO Z, JIAO Y, et al. Vuldetector: Detecting vulnerabilities using weighted feature graph comparison[J]. IEEE Transactions on Information Forensics and Security, 2021, 16: 2004-2017.

[43] GKORTZIS A, MITROPOULOS D, SPINELLIS D. Vulinoss: a dataset of security vulnerabilities in open-source systems[C]//Proceedings of the 15th International Conference on Mining Software Repositories. 2018: 18-21.

[44] NAMRUD Z, KPODJEDO S, TALHI C. Androvul: a repository for android security vulnerabilities[C]//Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering. 2019: 64-71.

[45] Black duck[EB/OL]. https://www.synopsys.com/content/dam/synopsys/sig-assets/datasheets/bdknowledgebase-ds-ul.pdf.

[46] Whitesource[EB/OL]. https://www.whitesourcesoftware.com/vulnerability-database/.

[47] Veracode[EB/OL]. https://sca.veracode.com/vulnerability-database/search.

[48] Snyk[EB/OL]. https://snyk.io/vuln.

[49] SNYK. State of the open source security report[EB/OL]. 2019. https://snyk.io/wp-content/uploads/sooss_report_v2.pdf.

[50] [EB/OL]. https://github.com/mpdavis/python-jose/commit/89b46353b9f611e9da38de3d2fedf52331167b93.

[51] [EB/OL]. https://github.com/mpdavis/python-jose/commit/73007d6887a7517ac07c6e755e494baee49ef513.

[52] [EB/OL]. https://svn.apache.org/viewvc?view=revision&revision=1373762.

[53] [EB/OL]. https://github.com/apache/james-hupa/commit/aff28a8117a49969b0fc8cc9926c19fa90146d8d.

[54] [EB/OL]. https://github.com/apache/deltaspike/commit/4e2502358526b944fc5514c206d306e97ff271bb.

[55] [EB/OL]. https://github.com/apache/deltaspike/commit/72e607f3be66c30c72b32c24b44e9deaa8e54608.

[56] [EB/OL]. https://github.com/apache/deltaspike/commit/d95abe8c01d256da2ce0a5a88f4593138156a4e5.

[57] [EB/OL]. https://github.com/django/django/commit/103ebe2b5ff1b2614b85a52c239f471904d26244.

[58] [EB/OL]. https://github.com/django/django/commit/36f580a17f0b3cb087deadf3b65eea024f479c21.

[59] [EB/OL]. https://github.com/django/django/commit/092cd66cf3c3e175acce698d6ca2012068d878fa.

[60] [EB/OL]. https://github.com/django/django/commit/11c5e0609bcc0db93809de2a08e0dc3d70b393e4.

[61] [EB/OL]. https://github.com/libimobiledevice/libimobiledevice/commit/df1f5c4d70d0c19ad40072f5246ca457e7f9849e.

[62] [EB/OL]. https://github.com/libimobiledevice/libusbmuxd/commit/4397b3376dc4e4cb1c991d0aed61ce6482614196.

致谢

致谢致谢致谢