学校代码：10246
学　　号：19212010035

復旦大學

硕 士 学 位 论 文

（学术学位）

基于多源信息的开源软件漏洞补丁定位方法

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 13 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

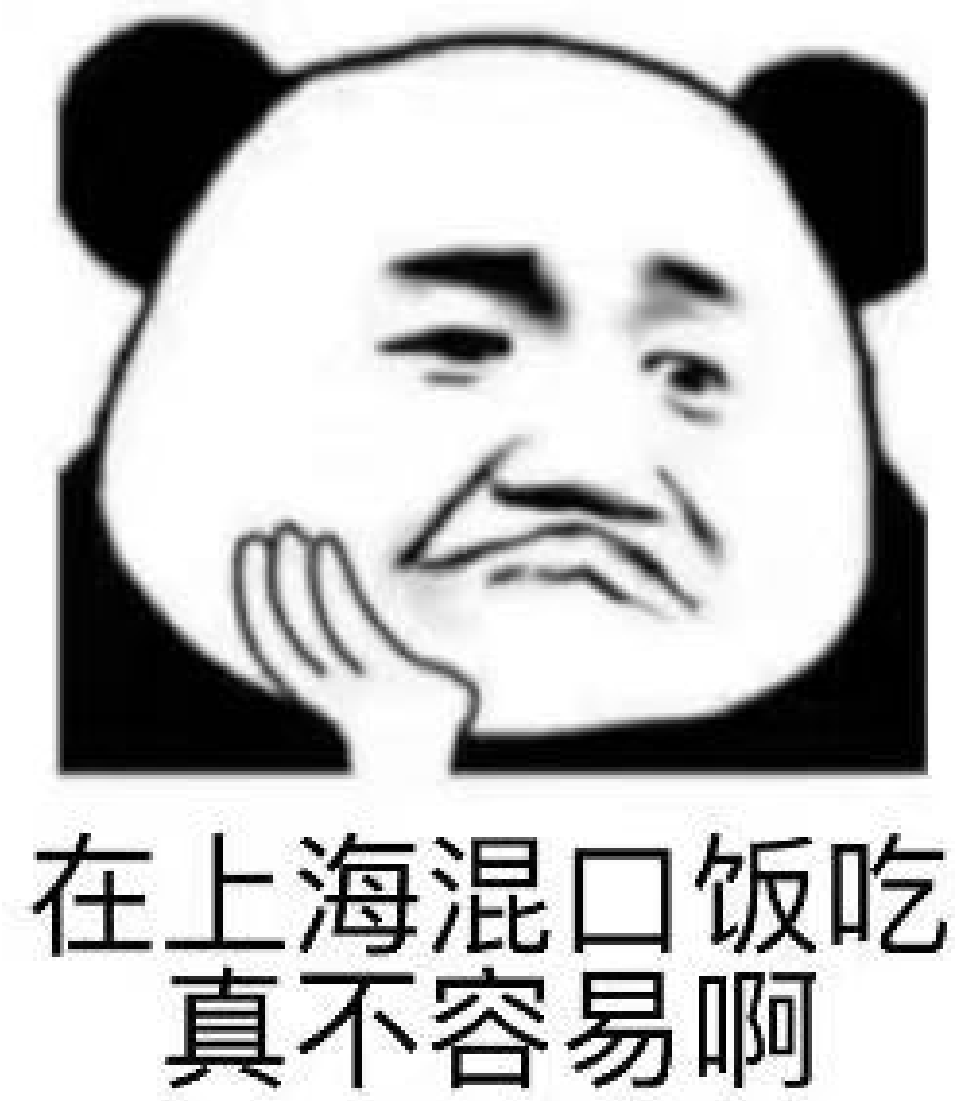

图 1-1 在上海混口饭吃真不容易啊

第 2 章 背景知识及相关工作

本文主要问题是 finding，。。。。

本章将介绍本文工作中的背景知识，包括：通用漏洞披露（CVE）、漏洞通告（advisory）和漏洞补丁（vulnerability patch）；此外，将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本文的相关工作。

2.1 背景知识

2.1.1 通用漏洞披露 (CVE)

通用漏洞披露 (Common Vulnerability Exposure，CVE)，是指。。。被广泛关注和参考。官方或第三方平台？再顺路引出相关的 sources，作为例子。

2.1.2 漏洞通告

漏洞通告（advisory），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

是指。。。。.Patch Git, SVN commit

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。董等人[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。陈等人[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。穆等人的工作[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.**??**）。

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（commitmessage）中搜索 CVE 标识符[22**?** –23]。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

本章节介绍了本文方法。

3.1 研究设计

3.2 数据收集及解析

3.2.1 研究问题

RQ1234

3.2.2 研究问题

precision、recall。。。

3.3 RQ1：覆盖率分析

3.4 RQ2：一致性分析

3.5 RQ3：补丁类型分析

3.6 RQ4：补丁映射分析

3.7 RQ5：补丁准确性分析

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.

致谢

致谢致谢致谢