

面向项目版本差异性的漏洞识别技术研究

黄诚^{1,2}, 孙明旭¹, 段仁语¹, 吴苏晟¹, 陈斌¹

(1. 四川大学网络空间安全学院, 四川 成都 610065;

2. 广西密码学与信息安全重点实验室, 广西 桂林 541000)

摘要: 开源代码托管平台为软件开发行业带来了活力和机遇, 但存在诸多安全隐患。开源代码的不规范性、项目依赖库的复杂性、漏洞披露平台收集漏洞的被动性等问题都影响着开源项目及引入开源组件的闭源项目的安全, 大部分漏洞修复行为无法及时被察觉和识别, 进而将各类项目的安全风险直接暴露给攻击者。为了全面且及时地发现开源项目中的漏洞修复行为, 设计并实现了基于项目版本差异性的漏洞识别系统—VpatchFinder。系统自动获取开源项目中的更新代码及内容数据, 对更新前后代码和文本描述信息进行提取分析。提出了基于安全行为与代码特征的差异性特征, 提取了包括项目注释信息特征组、页面统计特征组、代码统计特征组以及漏洞类型特征组的共 40 个特征构建特征集, 采用随机森林算法来训练可识别漏洞的分类器。通过真实漏洞数据进行测试, VpatchFinder 的精确率为 84.35%, 准确率为 85.46%, 召回率为 85.09%, 优于其他常见的机器学习算法模型。进一步通过整理的历年部分开源软件 CVE 漏洞数据进行实验, 其结果表明 68.07% 的软件漏洞能够提前被 VpatchFinder 发现。该研究结果可以为软件安全架构设计、开发及成分分析等领域提供有效技术支撑。

关键词: 漏洞识别; 开源平台; 安全修复; 机器学习

中图分类号: TP393

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2021094

Vulnerability identification technology research based on project version difference

HUANG Cheng^{1,2}, SUN Mingxu¹, DUAN Renyu¹, WU Susheng¹, CHEN Bin¹

1. School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China

2. Guangxi Key Laboratory of Cryptography and Information Security, Guilin 541000, China

Abstract: The open source code hosting platform has brought power and opportunities to software development, but there are also many security risks. The open source code has poor quality, the dependency libraries of projects are

收稿日期: 2021-05-24; **修回日期:** 2021-10-12

通信作者: 孙明旭, 2018141424064@stu.scu.edu.cn

基金项目: 国家自然科学基金 (61902265); 四川省科技厅重点研发项目 (2020YFG0047); 广西密码学与信息安全重点实验室研究课题 (GCIS201921)

Foundation Items: The National Natural Science Foundation of China (61902265), Sichuan Science and Technology Program (2020YFG0047), Guangxi Key Laboratory of Cryptography and Information Security (GCIS201921)

引用格式: 黄诚, 孙明旭, 段仁语, 等. 面向项目版本差异性的漏洞识别技术研究[J]. 网络与信息安全学报, 2022, 8(1): 52-62.

Citation Format: HUANG C, SUN M X, DUAN R Y, et al. Vulnerability identification technology research based on project version difference[J]. Chinese Journal of Network and Information Security, 2022, 8(1): 52-62.

complex and vulnerability collection platforms are inadequate in collecting vulnerabilities. All these problems affect the security of open source projects and complex software with open source complements and most security patches can't be discovered and applied in time. Thus, the hackers could be easily found such vulnerable software. To discover the vulnerability in the open source community fully and timely, a vulnerability identification system based on project version difference was proposed. The update contents of projects in the open source community were collected automatically, then features were defined as security behaviors and code differences from the code and log in patches, 40 features including comment information feature group, page statistics feature group, code statistics feature group and vulnerability type feature group were proposed to build feature set. And random forest model was built to learn classifiers for vulnerability identification. The results show that VpatchFinder achieves a precision rate of 0.844, an accuracy rate of 0.855 and a recall rate of 0.851. Besides, 68.07% of community vulnerabilities can be early discovered by VpatchFinder in real open source CVE vulnerabilities. This research result can improve the current issue in software security architecture design and development.

Keywords: vulnerability detection, open source platform, security patch, machine learning

0 引言

《2021 年开源软件供应链安全风险研究报告》显示,自 2015 年起开源组件中漏洞数逐年递增,其中 2020 年报告的开源代码漏洞数量环比 2019 年增长近 40%。漏洞数量激增的同时,缺乏有效的漏洞识别措施和全面的漏洞信息收集渠道是目前较为严重的问题。对开发者而言,项目引入的开源组件是否存在漏洞对项目整体的安全性至关重要,当开源组件发布一个漏洞修复后,引用者可能仍使用着包含漏洞的旧版本。虽然有如 Dependabot^[1]这样对依赖库版本进行更新的自动化工具,但并非适用于所有项目。一方面,所引组件的更新内容未必符合项目需求;另一方面,只有完全信任组件开发者,并保证其发布的版本向下兼容,才能确保安全性。大型项目通常有多个依赖库,繁重的工作量给维护人员带来了巨大挑战^[2]。为了优先考虑漏洞修复,他们倾向参考公共漏洞和暴露 (CVE, common vulnerabilities & exposures)、美国国家信息安全漏洞库 (NVD, national vulnerability data base) 等漏洞披露平台。然而实际上被公开的漏洞仅占真实漏洞数量的少数^[3],即使能够公开发布,也存在几周甚至数月的延迟,在此周期内,项目仍可能遭受攻击。因此,面向开源代码托管平台开发一个有效的漏洞识别系统,一方面可以感知安全态势,预警漏洞信息;另一方面,能够帮助开发者和软件维护者检测引用的开源组件是否

存在漏洞,保证项目的安全性。

针对漏洞识别的相关研究主要围绕源码审计^[4-5]、漏洞模式匹配^[6-7]、语法结构分析^[8-9]以及深度学习^[10]等方面展开。然而,如果将上述研究应用于开源社区,会遗漏项目更新时一些重要的描述信息。少部分研究工作考虑了代码更新过程,文献[11-15]提出安全更新和普通更新在部分基本特性上的差异,但由于数据不足和特征有效性较差等缺点,没有实现足够高的检测精度。

为解决上述问题,本文提出基于项目版本差异性的漏洞识别系统—VpatchFinder。针对 C/C++ 项目,通过 CVE 和 GitHub 构建数据集,结合多种过滤方法改善数据质量,接着提出 40 个特征以区分漏洞和非漏洞数据,尝试和比较多种机器学习技术,最终采用随机森林作为分类算法。本文贡献总结如下。

1) 通过爬取 CVE 网站和 GitHub 社区构建了一份真实的 C/C++ 项目更新内容数据集,提出多种过滤方法改善其数据质量,并将数据集进行开源。

2) 从安全行为和代码特征等方面对漏洞的表现形式展开研究,提出了 4 个特征组 (注释信息特征组、页面统计特征组、代码统计特征组和漏洞类型特征组),共 40 个特征,并通过实验证明了所选特征的有效性。

3) 设计并实现了漏洞识别系统—VpatchFinder,该系统在测试集上具有 84.35% 的精确率和 85.09% 的召回率。实验表明,在 VpatchFinder

的应用下, 68.07% 的社区漏洞可以比 CVE 提前发现, 同时检测出 192 条未公开披露的漏洞实例。

1 相关工作

有关开源代码托管平台漏洞检测的研究方法可分为 3 类: 基于源码检测的方法、基于公开线索挖掘的方法和基于更新内容差异性分析的方法。

1.1 基于源码检测的方法

源代码中含有丰富的语义信息, 目前针对源码漏洞检测的研究工作较为成熟, 少部分研究技术可以应用于开源平台。

Neuhaus 等^[16]考虑代码的相似性, 基于导入和调用函数的行为识别漏洞。该方法易扩展到其他语言, 但检测的覆盖面不足, 只适合检测特定类型的漏洞。郑荣锋等^[17]提取恶意代码行为指纹, 通过指纹匹配度量不同恶意代码的相似性。Kong 等^[18]结合多个静态分析工具, 提出基于数据融合的漏洞检测方法。Sonnekalb^[19]通过匹配不同的机器学习架构和代码表示方式得到最合适的漏洞检测模型。李元诚等^[20]从一些常见的漏洞关键点入手, 从代码中提取关键漏洞片段, 并提出一种新型神经网络—DCnnGRU, 通过保留代码上下文调用关系解决梯度消失问题。上述研究均有不错的检测效果, 但对代码的完整性要求较高, 无法检测单一的代码片段。树和图的技术常用于分析源代码, 文献[21-22]通过抽象语法树提取源代码结构, 文献[9,23-25]采用图的技术表征漏洞代码片段。Yamaguchi 等^[8]将抽象语法树和机器学习相结合, 模型的 $F1$ 值达到了 82.43%。然而基于树和图的技术存在相同的问题: 计算量大, 且需要完整的代码结构, 因此难以实现大规模应用。

1.2 基于公开线索挖掘的方法

开源代码托管平台公开的报告区或评论区蕴含着漏洞关键信息, 可以通过分析公开线索来识别漏洞。

Shin 等^[26]对开源项目 Linux 和 Firefox 展开研究, 采用机器学习对代码更新日志进行漏洞识别, 不足之处在于扩展性较差, 不能普遍适应大部分开源项目。Tian 等^[11]考虑漏洞被访问的频率以及

提交人员的经验特征等特性。Neil 等^[27]跟踪多个开源库, 监控用户在公共报告区发布的问题和回复, 寻找与漏洞相关的术语。这两项研究思路较新颖, 但缺点是它们忽略了代码中蕴含的大量信息。

1.3 基于更新内容差异性分析的方法

漏洞修复和普通更新在代码、文本描述、修改程度和变化数值的统计等方面存在明显差异, 但由于数据稀少和更新内容格式不统一等问题, 仅少数工作考虑到代码更新过程。

曹琰等^[28]结合程序依赖图, 通过漏洞源文件和补丁文件代码结构的差异掌握修补漏洞的方式。Sabetta 等^[3]直接将代码视为文本, 通过分析移除和增加的代码来识别和漏洞相关的代码更改, 缺点在于分类精度并不高。Ponta 等^[29]针对 Java 收集了一份漏洞修复数据集, 并将其开源以供学术界和工业界使用。Zaman 等^[13]研究 Firefox 中安全更新和功能性更新的差异。Perl 等^[12]提出漏洞修复和普通更新的一些差异性特征。Li 等^[14]对不同类别的更新进行第一次大规模实证研究。Wang 等^[15]在以上研究的基础上, 提出一些句法特征。但上述工作均没有实现足够高的检测精度, 仅考虑了漏洞修复和普通更新在部分基本特性上的差异, 而忽略了对更新内容的描述以及特定编程语言对应的漏洞类型等特征。此外, 数据收集并不充分, 也没有进行有效的数据过滤, 存在数据混杂的问题。

2 VpatchFinder 系统的实现

VpatchFinder 系统主要包含数据收集、特征提取、模型训练以及分类检测 4 个部分, 其系统框架如图 1 所示。

2.1 数据类型及来源

C/C++ 是近年来被曝出漏洞数量最多的编程语言, VpatchFinder 支持检测 C/C++ 项目。GitHub 中代码提交的 patch 页面主要包含以下内容: 提交者个人信息; 对本次更新内容的简要描述; 本次提交变化的文件数、移除的行数以及添加的行数; 移除和添加的具体代码以及它们上下几行的代码。patch 页面示例如图 2 所示。对漏洞修复而言, patch 页面包含许多漏洞的关键信息。

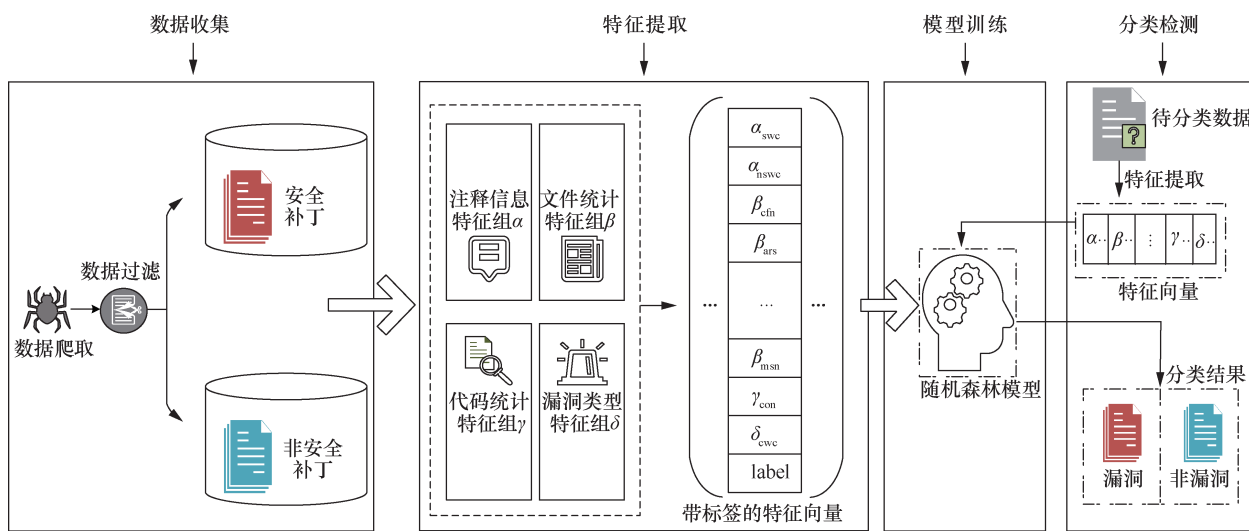


图 1 VpatchFinder 系统框架
Figure 1 Framework of VpatchFinder

```

From d9cd01ede97f4145af9781d448c62a3318952719 Mon Sep 17 00:00:00 2001
From: Tomash Brechko <tomash.brechko@gmail.com>
Date: Mon, 2 Nov 2009 18:15:54 +0100
Subject: [PATCH] Use strcmp when checking for large ascii multigets.

---
 memcached.c | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)

diff --git a/memcached.c b/memcached.c
index ed67eb06e..3e2e9c59e 100644
--- a/memcached.c
+++ b/memcached.c
@@ -3148,7 +3148,9 @@ static int try_read_command(conn *c) {
     ++ptr;
 }

- if (strcmp(ptr, "get ") && strcmp(ptr, "gets ")) {
+ if (ptr - c->rcurr > 100 ||
+     (strcmp(ptr, "get ", 4) && strcmp(ptr, "gets ", 5))) {
+
     conn_set_state(c, conn_closing);
     return 1;
 }

```

图 2 patch 页面示例
Figure 2 Example of the patch page

研究安全更新的挑战之一是缺乏足够的数据。CVE 为官方漏洞披露平台，收录着准确可靠的漏洞数据。通过 CVE 条目的参考链接追溯至 GitHub 中的代码提交，并根据固定的网址格式获取 patch 页面信息。

对于非漏洞数据，即普通的代码更新，从 GitHub 随机收集多个 C/C++ 项目的历史提交 patch 信息，并根据 hash 值的唯一性筛除非漏洞数据集中的漏洞数据。

经上述方法获取的两类数据存在明显的质量问题。一些 C/C++ 项目的漏洞修复也包含少量对除了“.c”和“.cpp”之外文件的修改，不利于统

一的特征提取。另外，非漏洞数据集中可能会掺杂部分未公开的漏洞，尤其是在一些大的复合型代码更新中。本文提出基于正则表达式和大型更新人工查验的方法过滤数据。

2.2 特征提取

提取有效特征来区分漏洞修复和普通更新是关键。分析 patch 页面，提出 4 个特征组（注释信息特征组、页面统计特征组、代码统计特征组和漏洞类型特征组），共 40 个特征，其中 11 个为本文首次提出，如表 1 所示。

2.2.1 注释信息特征组

每个 patch 页面都有一段形如“Subject: ”的字符串，这是代码提交者对此次更新内容的注释。基于漏洞修复和普通更新在注释信息上的差异，提出两个特征。

(1) Subject 安全关键词统计

虽然 Subject 描述较为精简，但研究发现，一些关键词经常出现在漏洞修复中而很少出现在普通更新中，称这类词语为安全关键词。将从 CVE 和 NVD 抓取的描述性文档作为语料库，结合 TF-IDF 算法^[30]，得到安全关键词的排序。从 0 至 100 进行递归，通过多次测试发现当选取 30 个安全关键词（overflow、buffer、leak 等）作为统计标准时，实验效果最优。采用词袋（BoW, bag-of-words）模型对 Subject 段落进行特征统计。

表 1 选取的所有特征
Table 1 Summary of all the extracted features

特征组类别	特征名称	符号	特征序号	来源
注释信息特征组 α	Subject 安全关键词统计	α_{swc}	1	首次提出
	Subject 非安全关键词统计	α_{nswc}	2	首次提出
页面统计特征组 β	变化的文件数量	β_{cfn}	3	文献[13,14]
	变化的修改块数量	β_{ccn}	4	文献[11,12]
	变化的行的数量	β_{cln}	5~10	文献[12,13]
	变化的字符数量	β_{ccn}	11~16	文献[11,15]
	添加代码与移除代码的相似程度	β_{ars}	17~19	文献[15]
	出现相同的代码更改的最大次数	β_{msn}	20	首次提出
	patch 文件大小	β_{fis}	21	首次提出
代码统计特征组 γ	变化的条件语句的数量	γ_{ccn}	22~27	文献[11,15]
	变化的循环语句的数量	γ_{cln}	28~33	文献[11,15]
	变化的算术、逻辑和关系运算符的总数量	γ_{con}	34~39	首次提出
漏洞类型特征组 δ	变化的代码中漏洞关键函数统计	δ_{cwc}	40	首次提出

(2) Subject 非安全关键词统计

除了漏洞修复，其他代码更新主要分为以下几类：增添或删除功能；改进代码质量；加强性能或效率；修复与安全性无关的错误或消除警告。当涉及以上类别的更新时，一些词语更多地出现在 Subject 描述中，称这类词语为非安全关键词。对于非安全关键词，统计大量应用软件的功能更新以及其他非安全更新的描述文档，结合 TF-IDF 以及递归测试，选取 18 个非安全关键词（function、warning、add 等）作为判断标准。同样采用 BoW 模型统计 Subject 段落。

2.2.2 页面统计特征组

关注 patch 页面中代码外部的一些统计性描述，基于漏洞修复和普通更新在统计数值上的差异构建特征集。

(1) 变化的文件数量

统计更新过程发生变化的文件数量。相比漏洞修复，普通更新影响的文件范围通常更广，如添加某功能模块时往往需要衔接多个文件，漏洞修复则大多在单个文件内完成更新。

(2) 变化的修改块数量

统计更新过程变化的修改块数量。定义修改块为 patch 页面内由字符串“@@”起始的一次单位修改，它的数量反映了代码结构修改的复杂程度。非安全更新的逻辑通常较复杂，包含多个修

改块，而大部分漏洞修复仅有一个修改块。

(3) 变化的行的数量

非安全更新多为功能更新，需要引入大量代码行来实现新功能，或修改较多代码行以提高软件效率。而对于漏洞，往往一个不恰当的边界值或一种错误的数据类型就导致极为严重的后果，因此很多漏洞修复仅涉及一行或几行的修改。为了提升特征集的鲁棒性，统计更新过程移除与添加的组合特征，即移除行数、添加行数、添加行数减移除行数的绝对值、添加行数加移除行数、添加行数是否大于移除行数以及添加行数与移除行数的比例共 6 个特征。

(4) 变化的字符数量

漏洞修复往往对代码库影响较小，仅有可能修改较少的代码。对于变化的字符数量，同样统计移除和添加的组合特征。

(5) 添加代码与移除代码的相似程度

更新前后代码的相似性表示修改幅度大小。首先标准化移除和添加的代码，以防止空格、注释和用户自定义变量等因素的影响。在此基础上，采用 Edit Distance 算法^[31]进行相似性统计，它计算由一个字符串转换成另一个字符串所需要的最少编辑操作次数。一次更新可能包含多个修改块，分别统计所有修改块中最小、平均和最大的相似性数值。

(6) 出现相同的代码更改的最大次数

统计更新过程中出现相同代码更改的最大次数。研究发现,部分漏洞修复会对同一修改模式重复多次,如更正多个变量的数据类型,或多次调整条件语句临界值等,而普通更新中则较少存在上述情况。

(7) Patch 文件大小

文件大小代表修改信息的多少。相比漏洞修复,其他更新的 patch 文件通常会更大。将 patch 页面下载为 txt 文件,并获取文件大小,为了防止数值过于离散,将其统一取整并转化为“kB”级单位。

2.2.3 代码统计特征组

关注 C/C++ 代码内部的统计内容,依据编程语言特性从代码中寻找漏洞修复的相关信息。

(1) 变化的条件语句的数量

C/C++ 的漏洞修复容易涉及条件控制语句,即通过添加新条件或修正现有条件来修复由条件不当引发的漏洞。定位 patch 页面中的移除行和增加行,分别统计它们包含条件语句的数量,并计算移除和添加的组合特征。

(2) 变化的循环语句的数量

循环语句中的控制条件临界值也极易导致漏洞。统计移除行和添加行中循环语句的数量,并计算移除和添加的组合特征。

(3) 变化的算术、逻辑和关系运算符的总数量

C/C++ 运算符是说明特定操作的符号,用于连接多个操作数或构成表达式,易被恶意代码利用。相比普通更新,漏洞修复有更大的可能涉及运算符的修改。统计算术、逻辑和关系运算符的总数量,同样计算移除和添加的组合特征。

2.2.4 漏洞类型特征组

C/C++ 频繁出现的漏洞类型通常比较固定,如常见的溢出问题、指针越界、内存泄露等^[32],固定的某类漏洞在 C/C++ 代码中也有特殊的表现。关注更新前代码中能够体现漏洞的关键函数,提出漏洞类型特征组。

C/C++ 中的一些函数在设计时忽略了安全性,存在较大风险。针对常出现的漏洞类型,列举一些容易引发漏洞的函数: mallo-c()、strcpy()、free()、gets()、strcat()、fgets()、sprintf()、memset()、

memcpy()、str-dup()。经迭代测试,选取上述 10 个关键函数的函数名作为语料库,通过抽象语法树解析更新前的代码以提取函数名,采用 BoW 模型进行特征统计。

2.3 分类算法

在 VpatchFinder 中采用随机森林作分类器。随机森林为一种集成学习算法,其输出结果由多棵决策树投票产生,在一定程度上可以避免模型的过拟合问题^[33],以应对真实环境中数据的复杂性。

3 实验及结果分析

本节评估所选特征的有效性,及 VpatchFinder 的分类性能,比较 VpatchFinder 与 CVE 发现漏洞的及时性,并展示检测出的部分未公开漏洞实例。

3.1 实验数据

采用 C/C++ 收集漏洞数据集和非漏洞数据集两类数据,它们提供真实环境下漏洞修复和普通更新的实例。

(1) 漏洞数据集

CVE 公开披露了大量漏洞,跟踪 1999 年至 2020 年 6 月所有 CVE 条目,通过引用链接回溯至 GitHub,筛除 C/C++ 以外的项目,并根据统一的网址格式获取 patch 页面信息,正则过滤非“.c”和“.cpp”文件,构建了 2 688 条漏洞数据。

(2) 非漏洞数据集

非漏洞数据集的数量众多,为防止数据集偏向某种特定类型,必须保证随机性和全面性。在 GitHub 中选取 62 个 C/C++ 项目(micropython、tcpdump 等),通过 git 命令获取项目历史提交 hash 值,利用 hash 值的唯一性过滤漏洞数据,并根据固定网址格式获取 patch 页面。考虑两类数据的平衡性,从获取的非漏洞数据中随机选出 3 100 条。在上述数据中,手动检查大型更新中是否包含小型的漏洞修复,如果有,则丢弃该数据,构建了 3 082 条非漏洞数据。

3.2 特征分析

调用随机森林中的 feature_importance 函数对 40 个特征展开重要性分析,前 12 位的排名和贡献能力如图 3 所示。从图 3 可以看出,特征贡献

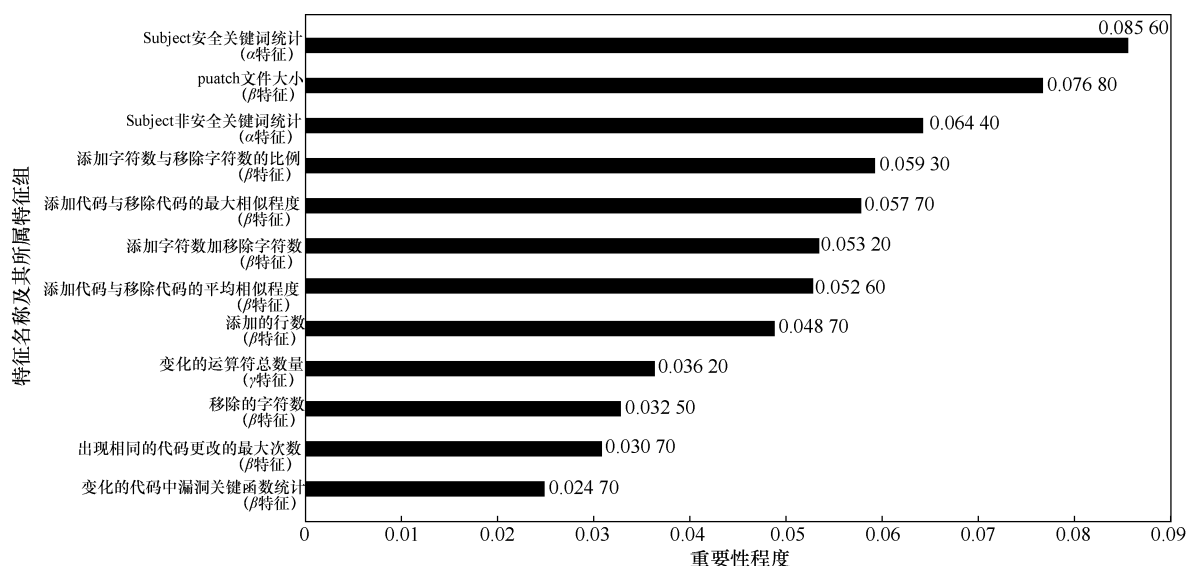


图 3 特征重要性排名

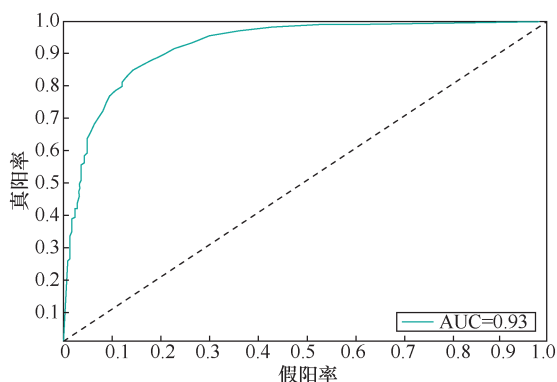
Figure 3 Ranking of the feature importance

度分布较为均匀,充分保证了模型的稳定性。此外,本文提出的特征,如“Subject 安全关键词统计”“patch 文件大小”等的贡献能力数值均排在前列,验证了所选特征的有效性。

3.3 VpatchFinder 分类测试结果

采用十折交叉验证,通过绘制接收者操作特征曲线 (ROC, receiver operating characteristic curve) 评估系统的分类能力。基于相同数据集将随机森林与其他流行的机器学习算法做比较。

VpatchFinder 在随机森林下的 ROC 曲线如图 4 所示,曲线下面积 (AUC, area under curve) 值为 0.93,这表明 VpatchFinder 能够准确检测开源社区中的漏洞。

图 4 随机森林下的 ROC 曲线
Figure 4 ROC of random forest

不同算法的准确率、召回率、精确率以及 $F1$ 值的评估结果如图 5 和表 2 所示。其中, $F1$ 值是精确率和召回率的调和均值,可作为综合评价指标。结果表明,随机森林的分类效果最优。

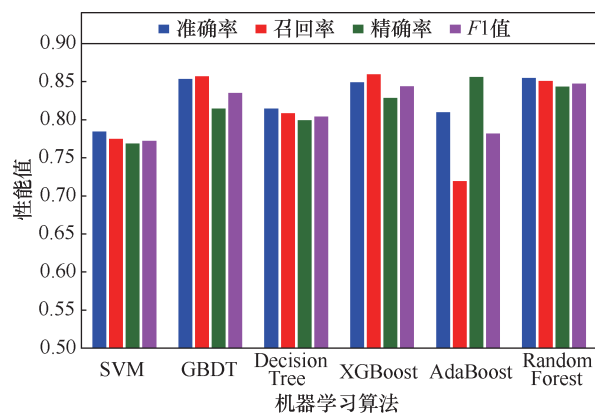


图 5 不同机器学习算法的性能对比

Figure 5 Performance comparison of different machine learning algorithms

3.4 VpatchFinder 及时性分析

VpatchFinder 的主要优势在于能够及时发现漏洞,相比官方网站漏洞披露的被动性,VpatchFinder 可以提供更全面、更及时的漏洞情报。为验证系统发现漏洞的及时性,对 VpatchFinder 的检测时间与 CVE 的漏洞发布时间展开比较。2020 年 6 月至 2021 年 3 月初,CVE 共公开发布了 275 条包含 commit 链接且属于 C/C++项目的漏

洞，它们来自 GitHub 的 Linux、Tensorflow、LibreDWG 和 FreeRDP 等项目，将上述 275 条数据开源处理，同时将它们作为待检测任务输入 VpatchFinder，在分类阈值设置为 0.5 的条件下，系统共报告了 238 条漏洞信息，检测的召回率为 86.55%。统计 238 条漏洞数据的时间信息，将 CVE 公开发布漏洞的日期作为 CVE 时间，将 commit 的提交时间作为 VpatchFinder 的检测时间，绘制日期天数差的累计分布曲线，结果如图 6 所示。X 轴上的正值表示 VpatchFinder 早于 CVE 发现漏洞的天数，有将近 60% 的数据处于 0 至 100 天这一区间，说明 VpatchFinder 通常可以比 CVE 提前 1 至 100 天发现漏洞。另外，分别统计 238 条数据中检测时间早于 CVE、等于 CVE 和晚于 CVE 的漏洞条数及所占比例，68.07% 的漏洞可以提前被 VpatchFinder 发现，结果如表 3 所示。上述实验表明，VpatchFinder 系统具备提前发现漏洞的能力，能够更早地进行预测，增加安全人员的应对时间，从而减少漏洞的威胁性。

表 2 随机森林和其他算法的评估数值结果
Table 2 Numerical results of random forest and other algorithms

算法	准确率	召回率	精确率	F1 值
SVM	0.784 2	0.774 9	0.768 8	0.772 0
GBDT	0.853 2	0.856 9	0.814 3	0.835 0
DT	0.814 7	0.808 5	0.799 3	0.803 8
XGBoost	0.849 2	0.859 3	0.828 6	0.843 7
AdaBoost	0.809 8	0.719 2	0.856 2	0.781 7
RF	0.854 6	0.850 9	0.843 5	0.847 2

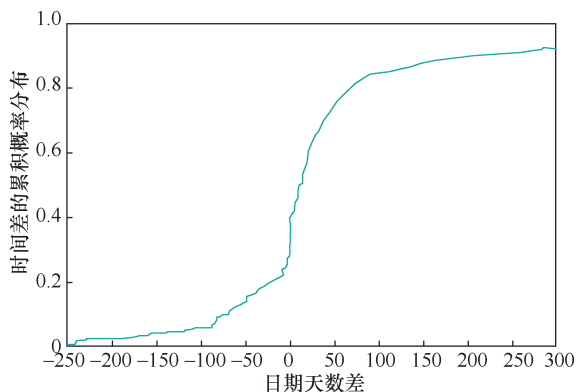


图 6 VpatchFinder 与 CVE 时间差的累计分布曲线
Figure 6 Cumulative distribution curve of time difference between VpatchFinder and CVE

表 3 VpatchFinder 与 CVE 时间对比
Table 3 Time comparison of VpatchFinder and CVE

检测时间状况	数量	占比
检测时间早于 CVE	162	68.07%
检测时间与 CVE 相同	11	4.62%
检测时间晚于 CVE	65	27.31%

3.5 识别的秘密漏洞实例

本节验证 VpatchFinder 发现秘密漏洞修补的能力，并展示挖掘出的一些未公开漏洞实例。首先，收集 GitHub 中 7 个大型 C/C++ 项目 (tcpdump、pytorch、radare2、electron、micropython、ImageMagickphp-src、Php-src) 的历史提交，并根据 hash 值筛选 CVE 已发布的漏洞和训练集中已有的数据。将 VpatchFinder 与广泛应用于 C/C++ 代码缺陷检测的静态分析系统 Cppcheck 进行检测结果的对比，将 commit 的 url 输入 VpatchFinder，将修改前的源代码文件输入 Cppcheck，接着人工检验所有分类结果为漏洞的数据，即结合专家分析和官方查询等方式判断其是否属于真正的漏洞。考虑人工检验的巨大工作量，仅从收集的 103 272 条数据中随机选取 10 000 条输入两种检测系统，检测结果如表 4 所示。

表 4 VpatchFinder 与 Cppcheck 检测结果对比
Table 4 Comparison of detection results between VpatchFinder and Cppcheck

检测系统	报告漏洞数量	人工检验漏洞数量	正确率	误报率
VpatchFinder	236	192	81.36%	18.64%
Cppcheck	318	182	57.23%	42.77%

值得说明的是，Cppcheck 报告的错误信息数量不包括警告及代码风格等问题。结果显示 VpatchFinder 的误报率远低于 Cppcheck，且两者检测到了相同的 154 条漏洞信息。以上实验表明 VpatchFinder 具有较高的实用价值。截至 2021 年 4 月，以上漏洞均仍未被 CVE 公开披露。详细分析其中的 7 个实例，如表 5 所示。表中第一列为漏洞项目来源及 commit 的 hash 值，第二列为漏洞修复在 GitHub 中的提交日期。实验表明，开源社区存在很多尚未公开披露的漏洞，类似 VpatchFinder 的漏洞识别系统对开源平台用户极为重要。

表 5 已识别的秘密漏洞实例
Table 5 Instances of Identified secret vulnerability

漏洞来源	提交时间	漏洞类型	漏洞描述
tcpdump(5e48...fb8)	Mon,13 Jul 2015	栈溢出	print-juniper.c 文件中的函数 juniper_parseHeader()存在缓冲区溢出漏洞,攻击者可利用此处实施拒绝服务攻击,导致应用程序崩溃
radare2(9bd0...7fb)	Fri,15 Nov 2013	栈溢出	cmd_write.c 文件中未判断变量的数据长度,可能引发栈溢出漏洞
micropython(2daa...93b)	Fri,1 Sep 2017	栈溢出	未进行输入数据格式的规范性检验,攻击者可以人为构造可利用代码溢出 modstruct.c 文件中的结构体缓冲区,实施恶意攻击
ImageMagick(a464...0d0)	Mon,11 Feb 2019	堆溢出	处理 SVG 图像时,未检查变量 message 的数据长度,导致堆溢出漏洞
php-src(6ebe...d3c)	Thu,27 May 2010	空指针	在 mysqlnd.c 文件中,调用指针变量 result 前缺乏判空步骤,远程攻击者可以绕过访问限制或进行拒绝服务攻击,导致应用程序崩溃
php-src(cdd9...004)	Wed,22 Aug 2018	内存泄露	在 array.c 文件中,未释放 result 变量,如果回调函数异常,可能会产生内存泄露,导致系统崩溃或敏感信息泄露
php-src(0a1c...b68)	Sat,15 Oct 2011	整数溢出	ext/soap/php_http.c 文件中的 emalloc 函数存在整数溢出漏洞,有权限执行该脚本的攻击者可以利用此处进行提权操作

4 存在的问题及改进方向

VpatchFinder 仍存在值得改进的地方,本节将讨论这些问题。

第一,由于 patch 页面格式的特殊性,目前系统仅面向 GitHub 平台。GitHub 是全球热门的开源代码托管平台,但其他平台曝出的漏洞数量也在持续增长,未来会考虑将 VpatchFinder 跨平台拓展。

第二,由于其他编程语言训练数据不足,VpatchFinder 只适合检测 C/C++项目。随着数据集不断增加,未来只需根据不同编程语言的特性调整代码统计特征组和漏洞类型特征组,就可以适应不同类别的语言,获得更强大的检测能力。

第三,数据集来源较为单一,从 CVE 收集的数据可能会导致系统偏向于检测特定类型的漏洞。实际中,CVE 并非收录着所有漏洞,它倾向发布一些严重或高危的漏洞,这可能会导致漏报,但也是合乎情理的,因为高危漏洞理应有一定的优先性,未来需要平衡上述情况。

5 结束语

针对开源代码托管平台存在的安全问题,本文设计并实现了基于项目版本差异性的漏洞识别系统—VpatchFinder。首先收集 C/C++项目更新时的 patch 页面数据,并通过多种过滤方法有效改善数据集质量,接着提出 4 个特征组(注释信息特征组、页面统计特征组、代码统计特征组和漏

洞类型特征组),其中包含 11 个首次提出的特征。实验表明,VpatchFinder 具有准确、及时和高效的检测性能,能够为开发人员和其他目标用户提供有效的漏洞检测技术支持。

参考文献:

- [1] ALFADEL M, COSTA D E, SHIHAB E, et al. On the use of dependabot security pull requests[C]//Proceedings of 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR). 2021: 254-265.
- [2] 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.
- [3] SABETTA A, BEZZI M. A practical approach to the automatic classification of security-relevant commits[C]//Proceedings of 2018 IEEE International Conference on Software Maintenance and Evolution. 2018: 579-582.
- [4] KAMIYA T, KUSUMOTO S, INOUE K. CCFinder: a multilingual token-based code clone detection system for large scale source code[J]. IEEE Transactions on Software Engineering, 2002, 28(7): 654-670.
- [5] LI Z, LU S, MYAGMAR S, et al. CP-Miner: finding copy-paste and related bugs in large-scale software code[J]. IEEE Transactions on Software Engineering, 2006, 32(3): 176-192.
- [6] 王雅文,姚欣洪,宫云战,等.一种基于代码静态分析的缓冲区溢出检测算法[J].计算机研究与发展,2012,49(4): 839-845.
WANG Y W, YAO X H, GONG Y Z, et al. A method of buffer overflow detection based on static code analysis[J]. Journal of Computer Research and Development, 2012, 49(4): 839-845.
- [7] 王蕾,李丰,李炼,等.污点分析技术的原理和实践应用[J].软

- 件学报, 2017, 28(4): 860-882.
- WANG L, LI F, LI L, et al. Principle and practice of taint analysis[J]. *Journal of Software*, 2017, 28(4): 860-882.
- [8] YAMAGUCHI F, LOTTMANN M, RIECK K. Generalized vulnerability extrapolation using abstract syntax trees[C]//*Proceedings of the 28th Annual Computer Security Applications Conference*. 2012: 359-368.
- [9] LI J Y, ERNST M D. CBCD: cloned buggy code detector[C]//*Proceedings of 2012 34th International Conference on Software Engineering (ICSE)*. 2012: 310-320.
- [10] LI Z, ZOU D Q, XU S H, et al. VulDeePecker: a deep learning-based system for vulnerability detection[C]//*Proceedings 2018 Network and Distributed System Security Symposium*. 2018.
- [11] TIAN Y, LAWALL J, LO D. Identifying Linux bug fixing patches[C]//*Proceedings of 2012 34th International Conference on Software Engineering (ICSE)*. 2012: 386-396.
- [12] PERL H, DECHAND S, SMITH M, et al. VCCFinder: finding potential vulnerabilities in open-source projects to assist code audits[C]//*Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015: 426-437.
- [13] 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.
- [14] 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.
- [15] WANG X D, SUN K, BATCHELLER A, et al. An empirical study of secret security patch in open source software[M]//*Adaptive Autonomous Secure Cyber Systems*. 2020: 269-289.
- [16] NEUHAUS S, ZIMMERMANN T, HOLLER C, et al. Predicting vulnerable software components[C]//*Proceedings of the 14th ACM Conference on Computer and Communications Security*. 2007: 529-540.
- [17] 郑荣锋, 方勇, 刘亮. 基于动态行为指纹的恶意代码同源性分析[J]. *四川大学学报(自然科学版)*, 2016, 53(4): 793-798.
- ZHENG R F, FANG Y, LIU L. Homology analysis of malicious code based on dynamic-behavior fingerprint[J]. *Journal of Sichuan University (Natural Science Edition)*, 2016, 53(4): 793-798.
- [18] KONG D G, ZHENG Q, CHEN C, et al. ISA: a source code static vulnerability detection system based on data fusion[C]//*Proceedings of the 2nd International ICST Conference on Scalable Information Systems*. 2007: 55.
- [19] SONNEKALB T. Machine-learning supported vulnerability detection in source code[C]//*Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2019: 1180-1183.
- [20] 李元诚, 崔亚奇, 吕俊峰, 等. 开源软件漏洞检测的混合深度学习方法[J]. *计算机工程与应用*, 2019, 55(11): 52-59.
- LI Y C, CUI Y Q, LYU J F, et al. Combined deep learning method for open source software vulnerability detection[J]. *Computer Engineering and Applications*, 2019, 55(11): 52-59.
- [21] JIANG L X, MISHERGHI G, SU Z D, et al. DECKARD: scalable and accurate tree-based detection of code clones[C]//*Proceedings of 29th International Conference on Software Engineering (ICSE'07)*. 2007: 96-105.
- [22] ALON U, ZILBERSTEIN M, LEVY O, et al. code2vec: learning distributed representations of code[J]. *Proceedings of the ACM on Programming Languages*, 2019, 3: 40.
- [23] LIU C, CHEN C, HAN J W, et al. GPLAG: detection of software plagiarism by program dependence graph analysis[C]//*Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2006: 872-881.
- [24] PHAM N H, NGUYEN T T, NGUYEN H A, et al. Detection of recurring software vulnerabilities[C]//*ASE '10: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*. 2010: 447-456.
- [25] 刘凯, 方勇, 张磊, 等. 基于图卷积网络的恶意代码聚类[J]. *四川大学学报(自然科学版)*, 2019, 56(4): 654-660.
- LIU K, FANG Y, ZHANG L, et al. Malware clustering based on graph convolutional networks[J]. *Journal of Sichuan University (Natural Science Edition)*, 2019, 56(4): 654-660.
- [26] SHIN Y, MENEELY A, WILLIAMS L, et al. Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities[J]. *IEEE Transactions on Software Engineering*, 2011, 37(6): 772-787.
- [27] NEIL L, MITTAL S, JOSHI A. Mining threat intelligence about open-source projects and libraries from code repository issues and bug reports[C]//*Proceedings of 2018 IEEE International Conference on Intelligence and Security Informatics*. 2018: 7-12.
- [28] 曹琰, 刘龙, 王禹, 等. 基于函数语义分析的软件补丁比对技术[J]. *网络与信息安全学报*, 2019, 5(5): 56-63.
- CAO Y, LIU L, WANG Y, et al. Software patch comparison technology through semantic analysis on function[J]. *Chinese Journal of Network and Information Security*, 2019, 5(5): 56-63.
- [29] PONTA S E, PLATE H, SABETTA A, et al. A manually-curated dataset of fixes to vulnerabilities of open-source software[C]//*Proceedings of 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 2019: 383-387.
- [30] RAMOS J. Using TF-IDF to determine word relevance in document queries[J]. *Proceedings of the First Instructional Conference on Machine Learning*, 2003: 29-48.
- [31] RISTAD E S, YIANILOS P N. Learning string-edit distance[J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998, 20(5): 522-532.

- [32] 吕维梅, 刘坚. C/C++程序安全漏洞的分类与分析[J]. 计算机工程与应用, 2005, 41(5): 123-125, 228.

LYU W M, LIU J. The classification and analysis on safety holes of C/C++ programs[J]. Computer Engineering and Applications, 2005, 41(5): 123-125, 228.

- [33] BREIMAN L. Random forests[J]. Machine Learning, 2001, 45(1): 5-32.

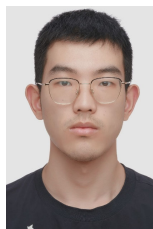


段仁语 (1998-), 男, 重庆人, 主要研究方向为漏洞情报挖掘、计算机视觉和人工智能。

[作者简介]



黄诚 (1987-), 男, 重庆人, 四川大学副教授, 主要研究方向为网络空间安全、攻击检测、威胁溯源、数据挖掘、社交网络、机器学习和自然语言处理。



吴苏晟 (1999-), 男, 浙江杭州人, 主要研究方向为漏洞挖掘和开源代码漏洞库的分析与构建。



孙明旭 (2000-), 男, 黑龙江绥化人, 主要研究方向为数据挖掘、自然语言处理和漏洞情报分析。



陈斌 (1999-), 男, 江西南昌人, 主要研究方向为漏洞挖掘与自然语言处理。