计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(11):62−72 [doi: 10.15888/j.cnki.csa.009293]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

Web 应用漏洞报告理解及漏洞复现①

李子东[1], 王微微[1], 尤　枫[1], 杨　羊[2], 赵瑞莲[1]

[1](北京化工大学 信息科学与技术学院, 北京 100029)
[2](浙江理工大学 信息科学与工程学院, 杭州 310018)
通信作者: 赵瑞莲, E-mail: rlzhao@mail.buct.edu.cn

摘　要: 随着 Web 应用的功能日趋复杂, 其安全问题不容乐观, Web 应用安全性测试成为软件测试领域的研究重点之一. 漏洞报告旨在记录 Web 应用安全问题, 辅助 Web 应用测试, 提升其安全性与质量. 然而, 如何自动识别漏洞报告中的关键信息, 复现漏洞, 仍是当前的研究难点. 为此, 本文提出一种自动化的漏洞报告理解和漏洞复现方法, 首先, 依据漏洞报告的特点, 归纳其语法依存模式, 并结合依存句法分析技术, 解析漏洞描述, 提取漏洞触发的关键信息. 其次, 不同于常规自然语言描述, Web 漏洞的攻击负载通常是非法字符串, 大多以代码片段的形式存在, 为此, 本文针对攻击负载, 设计提取规则, 完善漏洞报告中攻击负载的提取. 在此基础上, 考虑漏洞报告与 Web 应用文本描述不同但语义相近, 提出基于语义相似度的漏洞复现脚本自动生成方法, 实现 Web 应用漏洞的自动复现. 为验证本文方法的有效性, 从漏洞收集平台 Exploit-db 的 300 余个 Web 应用项目中收集了 400 份漏洞报告, 归纳出其语法依存模式; 并针对 23 个开源 Web 应用涉及的 26 份真实漏洞报告进行漏洞复现实验, 结果表明本文方法可有效提取漏洞报告的关键信息, 并据此生成可行测试脚本, 复现漏洞, 有效减少人工操作, 提升漏洞复现效率.

关键词: Web 应用测试; Web 应用漏洞报告分析; Web 应用漏洞复现; 自然语言处理; 测试脚本自动生成

引用格式: 李子东,王微微,尤枫,杨羊,赵瑞莲.Web 应用漏洞报告理解及漏洞复现.计算机系统应用,2023,32(11):62–72. http://www.c-s-a.org.cn/1003-3254/9293.html

Report Comprehension and Reproduction of Web Application Vulnerability

LI Zi-Dong[1], WANG Wei-Wei[1], YOU Feng[1], YANG Yang[2], ZHAO Rui-Lian[1]

[1](School of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China)
[2](School of Information Science and Engineering, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: As Web applications become increasingly complex, their security issues happen frequently. Web application security testing has become one of the research priorities in the field of software testing. Vulnerability reports aim to document Web application security issues and assist Web application testing to improve its security and quality. However, how to automatically identify the key information in vulnerability reports and reproduce the vulnerability is still a research challenge. To this end, this study proposes an automatic approach to comprehend vulnerability reports and reproduce the vulnerability. Firstly, based on the characteristics of vulnerability reports, the study summarizes their grammar dependency patterns and combines them with dependency syntactic parsing techniques to parse vulnerability descriptions and extract key information about vulnerability triggers. Secondly, unlike conventional natural language descriptions, the payload of Web vulnerability is usually an illegal string, mostly in the form of a code fragment. For this reason, the study designs extraction rules for the payload solely to improve the extraction of vulnerability reports. On this basis, considering that vulnerability reports and Web application text descriptions are different but semantically similar, the study proposes a

① 基金项目: 国家自然科学基金 (62077003)
收稿时间: 2023-04-10; 修改时间: 2023-05-11; 采用时间: 2023-06-06; csa 在线出版时间: 2023-09-15
CNKI 网络首发时间: 2023-09-18

semantic similarity-based method to achieve the automatic reproduction of Web application vulnerability. To verify the effectiveness of this study, 400 vulnerability reports are collected from more than 300 Web application projects in the vulnerability collection platform Exploit-db, and their grammar dependency patterns are summarized. A total of 26 real vulnerability reports involving 23 open-source Web applications are used for vulnerability reproduction experiments. The results show that the proposed method can effectively extract key information from vulnerability reports and generate feasible test scripts to reproduce vulnerability, reducing manual operations, and improving the efficiency of vulnerability reproduction.

Key words: Web application testing; report analysis of Web application vulnerability; reproduction of Web application vulnerability; natural language processing (NLP); automatic generation of test script

随着网络的快速发展, Web 应用普及到人们日常生活中的方方面面, Web 应用的安全性越发重要. 据国家信息安全漏洞共享平台 (CNVD)[1] 统计, 2022 年 9 月份共收集信息安全漏洞 1 854 个, 其中 Web 应用安全漏洞占比高达 48%. 如何提高 Web 应用的质量和安全性已成为学术界和工业界普遍关注的问题.

漏洞报告是测试过程产生的用来描述漏洞相关信息的文本, 其对漏洞定位与修复至关重要[2,3]. 但由于漏洞报告是由自然语言编写, 本质上是不精确不完整的[4], 如何识别并抽取出涉及漏洞触发的关键信息是目前的研究难点; 同时, Web 应用业务逻辑较为复杂, 涉及的页面及元素种类繁多[5], 导致漏洞复现更加困难. 现有关于漏洞报告的研究大多通过简单的命名实体识别或关系抽取, 分析漏洞报告中的结构化信息, 如漏洞类型、软件名称等, 没有涉及漏洞触发的相关信息, 难以辅助开发人员对 Web 应用漏洞进行有效的修复. 同时, 由于漏洞报告存在高度非结构化的自然语言描述, 简单的正则表达式技术难以解析出漏洞报告涉及的复杂逻辑. 此外, 有研究发现, 现有的漏洞复现过程主要依赖于人工经验, 并且要求漏洞复现人员在该领域有一定的专业知识, 漏洞定位及修复成本较高. 因此, 研究一种 Web 应用漏洞报告的自动理解与漏洞复现技术, 以提高漏洞修复效率, 提升 Web 应用软件的安全性, 具有重要的科学研究意义和实际应用前景.

然而, 自动分析理解 Web 应用漏洞报告并自动复现其漏洞面临诸多挑战. 一是漏洞报告通常由高度非结构化的自然语言描述, 报告书写者可能在不同抽象层次上对漏洞触发步骤进行描述, 描述的方式也不尽相同. 因此, 如何准确理解漏洞报告的自然语言描述, 识别并抽取出与漏洞触发的关键信息是本文研究面临的一个挑战. 二是漏洞报告的自然语言描述使得其表述的信息不够准确, 而现代 Web 涉及的页面元素种类繁杂. 因此, 如何依据漏洞报告中的漏洞触发步骤, 识别每一步在 Web 应用中相应的页面元素, 进而构建完整的 Web 应用漏洞复现脚本, 是本文研究面临的又一挑战.

因此, 本文提出一种自动化的漏洞报告理解与安全漏洞复现方法. 具体而言, 首先, 针对高度非结构化的 Web 应用漏洞报告, 依据漏洞触发描述的特点, 归纳其语法依存模式, 并据此解析漏洞报告, 提取漏洞触发的关键信息; 其次, 设计对应的攻击负载识别方法, 准确提取攻击负载; 基于漏洞触发关键信息, 考虑漏洞报告与 Web 应用文本描述不同但语义相近, 提出基于语义相似度的漏洞复现脚本自动生成方法, 实现 Web 应用漏洞的自动复现. 为验证本文方法的有效性, 选取 23 个开源 Web 应用, 并从漏洞收集平台 Exploit-db[6] 中收集其涉及的 26 份真实漏洞报告, 进行漏洞报告理解与漏洞复现实验. 结果表明本文方法能有效分析漏洞报告并提取其关键信息, 并据此生成可行测试脚本, 复现漏洞, 大大减少人工操作, 提升漏洞复现效率.

1 相关工作

1.1 漏洞报告分析

现代 Web 应用在使用过程中用户会提交大量的漏洞报告, 软件开发/测试人员可利用漏洞报告, 识别与漏洞触发相关的关键信息, 重现软件漏洞, 进而分析漏洞产生的原因, 最终修复漏洞[7,8]. 显然, 漏洞报告分析是上述工作的重中之重.

目前, 已有一些学者在漏洞报告分析方面取得了初步进展, 例如, Feng 等人[9] 从互联网上收集物联网漏洞报告, 将漏洞报告中的信息检索问题建模为 NLP 中

的命名实体识别问题, 利用关键字和正则匹配技术, 识别漏洞报告中与物联网漏洞相关实体, 如设备类型、供应商、产品名称和漏洞类型等, 通过实体之间的依赖关系来分析确定物联网设备是否存在漏洞. Dong 等人[8]给出了一个自动化识别系统, 利用基于深度学习的命名实体识别和关系抽取, 从非结构化漏洞描述文本中, 识别其易受攻击的软件名称和版本. 上述研究只通过简单的命名实体识别或关系抽取, 分析漏洞报告中的结构化信息, 如漏洞类型、供应商、软件名称等, 没有涉及漏洞触发的相关信息, 难以辅助开发人员对 Web 应用漏洞进行有效的修复. 再者, 漏洞报告包含高度非结构化的自然语言描述, 简单的正则表达式技术, 难以解析出漏洞报告涉及的复杂逻辑. 此外, Mu 等人[10] 对实际的安全漏洞报告进行了实证分析, 以分析安全漏洞的可重现性, 研究发现, 现有的漏洞复现过程主要依赖于人工经验, 并且要求漏洞复现人员在该领域有一定的技术背景. 作者指出有必要引入更有效和自动化的方式从报告中收集常见漏洞触发的关键信息.

另一方面, 也有一些学者关注漏洞报告分类和漏洞报告质量, 例如, Kudjo 等人[7] 针对 CVE 和 NVD 的漏洞报告, 结合两种分类算法 TF-IGM 和 TF-IDF, 提高漏洞报告分类的精确度. Guo 等人[11] 针对由于漏洞报告提交者上报的信息不完整, 导致漏洞报告存在一定程度的信息缺失等问题, 通过细化漏洞本身的其他信息, 设计机器学习方法, 预测漏洞报告中缺失的漏洞类型. 但未见以漏洞重现为目标, 从漏洞报告出发, 对 Web 应用漏洞报告进行自动分析理解的相关研究成果发表. 现存研究已表明, 漏洞报告中大多含有漏洞触发相关策略, 若能自动获取漏洞触发步骤, 必将提升漏洞修复效率[12,13].

1.2 基于漏洞报告的漏洞复现

漏洞报告记录了报告提供者遇到的软件问题, 开发和测试人员可通过查看漏洞报告, 了解漏洞出现的相关操作, 分析漏洞原因或复现漏洞, 为漏洞定位和修复提供帮助. 然而, 人工复现漏洞严重依赖于开发人员的个人经验, 而且漏洞复现过程可能耗时较多, 尤其当有大量故障报告且故障报告涉及许多与漏洞触发步骤无关的描述信息时.

为此, 有研究人员提出从漏洞报告自动复现的思想. 例如, Zhao 等人[14] 给出一种从 Android 应用程序的漏洞报告复现崩溃漏洞的方法, 通过自然语言处理技术对漏洞报告文本进行分析, 识别其漏洞复现所需的 GUI 组件和相关信息, 利用此类信息在 Android 应用上进行动态搜索, 生成测试脚本来复现 Android 应用程序的崩溃故障. Li 等人[15] 给出一种从用户评论中复现 Android 应用程序错误的方法, 通过对用户评论进行语法分析, 提取与错误相关的信息词, 利用与错误相关的信息词进行事件序列探索, 生成能重现用户评论涉及错误的事件序列. Feng 等人[16] 提出了一种轻量级的工具 GIFdroid, 针对存在漏洞触发流程视频的 Android 漏洞报告, 采用图像处理技术从视频中提取关键帧, 将它们映射为 GUI 转换图中的状态, 并生成这些状态的执行轨迹以触发错误, 但针对 Web 应用的漏洞报告通常不包含此类文字信息以外的辅助信息.

上述研究主要针对 Android 应用, 根据漏洞报告对漏洞进行复现. Android 应用漏洞报告有多种信息可以使用, 漏洞复现相对简单. 而 Web 应用业务逻辑更为复杂, 涉及的页面及元素种类繁多, 漏洞类型更为多样, 漏洞报告的自动分析理解与漏洞复现更加困难, 目前主要靠人工分析, 难以实现 Web 应用程序漏洞的快速复现与定位分析, 因此漏洞修复效率不高.

2 Web 应用漏洞报告理解与漏洞复现

漏洞报告中涉及漏洞触发的部分通常由自然语言书写, 简单的正则表达式技术难以解析出漏洞触发涉及的复杂逻辑. 因此, 需要收集大量的漏洞报告, 总结归纳出漏洞触发信息的书写特征, 再结合自然语言处理技术, 依据其特征识别并提取出漏洞复现所需要的关键信息; 此外, 由于漏洞报告是人工书写的, 漏洞触发步骤中涉及的操作对象常常与实际 Web 应用有所偏差, 例如, Web 应用中的按钮“begin”可能会被人工记录为按钮“start”, 传统的字符串匹配方法无法准确匹配漏洞报告与 Web 应用中对应的元素, 但“begin”和“start”语义相近, 因此, 本文利用语义相似度匹配漏洞报告与 Web 应用的对应元素, 生成漏洞复现脚本, 自动复现 Web 应用漏洞. 下面将详细介绍本文方法.

2.1 整体概述

Web 应用漏洞报告由自然语言书写的, 其中涉及的漏洞触发步骤一般包含 4 部分关键信息: 动作、动作对象、对象类型以及参数值, 其中, 动作信息是触发漏洞需要做出的行为, 动作对象是行为作用的目标, 对象类型是指目标的类型, 参数值是指输入类动作中需

要输入的值. Web 应用漏洞报告理解与漏洞复现的整体框架如图 1 所示. 为获取漏洞触发步骤的关键信息, 本文首先从 400 余份真实的漏洞报告中归纳出此类信息的语法依存模式; 其次, 基于 NLP 的漏洞报告分析模块依据语法依存模式, 结合词法分析和依存语法分析, 提取触发操作的关键信息, 此外, 针对部分操作涉及的攻击负载, 本文设计了对应的提取方法, 当漏洞报告处理完成后, 得到由有序的漏洞触发关键信息构成的漏洞触发步骤序列; 最终, 漏洞复现脚本自动构建模块利用语义相似度, 识别与漏洞报告关联的 Web 应用页面及元素, 构建漏洞复现测试脚本, 实现 Web 应用漏洞的自动复现.

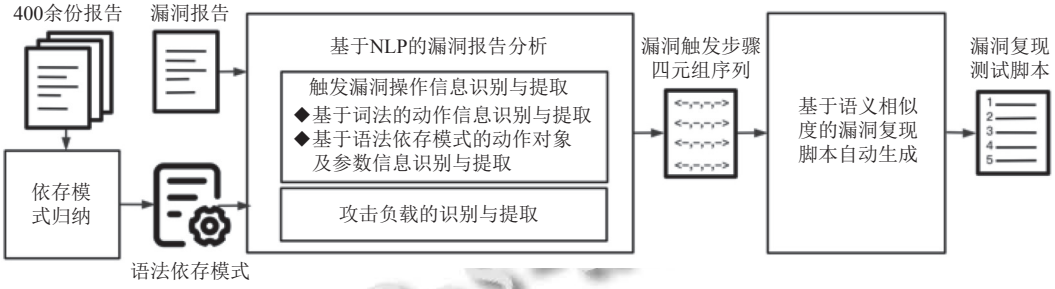

图 1 Web 应用漏洞报告理解及漏洞复现

在漏洞报告分析模块, 动作信息一般是漏洞报告描述语句成分中最重要的根词, 因此, 先根据词法分析提取语句中的动词作为动作信息; 然后, 基于语法依存模式提取动作关联的对象、对象类型及携带的参数值; 此外, 不同于常规自然语言描述, Web 漏洞的攻击负载通常是代码片段, 本文设计对应的攻击负载识别方法, 完善漏洞报告中攻击负载的提取.

2.2 语法依存模式归纳

通过对真实 Web 应用漏洞报告进行分析研究, 我们观察到漏洞报告的漏洞触发描述中, 不同动作类型的句法结构一般是不同的, 而相同的动作类型, 其句法结构一般相同. 例如: 输入类操作的动作对象类型与动作根词的关系通常是介词宾语关系或间接名词关系, 而点击类操作的动作对象类型与动作根词的关系除上述两种关系外还包括非确定依存和状语修饰语关系.

因此, 本文参考了漏洞收集平台 Exploit-db 中获得的 300 余种项目的 400 份漏洞报告, 手工构建了每类动作不同的语法依存模式, 如表 1 所示, 该模式共包含 6 类动作, 除输入类和使用类额外包含输入的参数值外, 每类动作包含两种信息: 动作对象和动作对象类型, 每种信息内包含若干种依存关系, 即句子中具有此类依存关系的词将被认为是对应的信息, 例如文本“Click on 'login' button”, 这是一个点击类型的动作, 且 button 与 click 的依存关系是间接名词、login 与 button 的依存关系是复合表达式, 所以 button 被认为是动作 click 的动作对象类型、login 被认为是动作 click 的动作对象. 在后续方法中, 将基于该模式提取特定的依存关系以获取漏洞触发的关键信息.

表 1 不同类型动作的语法依存模式

动作类型	依存关系		
	动作对象	动作对象类型	参数值
跳转	Nmod (名词修饰语) Amod (形容词修饰语)	Obj (宾语) Parataxis (无连词并列) Obl (间接名词)	—
粘贴	Nmod (名词修饰语) Amod (形容词修饰语)	Obj (宾语)	—
输入	Amod (形容词修饰语) Compound (复合表达式) Appos (同位语修饰语)	Pobj (介词的宾语) Obl (间接名词)	Dobj (直接宾语) Obj (宾语)
使用	Amod (形容词修饰语) Compound (复合表达式)	Pobj (介词的宾语) Obl (间接名词)	Dobj (直接宾语) Obj (宾语)
点击	Nmod (名词修饰语) Amod (形容词修饰语) Compound (复合表达式)	Pobj (介词的宾语) Obl (间接名词) Dep (非确定依存) Advmod (状语修饰语)	—
修改	Nmod (名词修饰语) Amod (形容词修饰语)	Obj (宾语)	Obl (间接名词)

2.3 基于 NLP 的漏洞报告分析

漏洞报告包含漏洞名称、时间等结构化信息和描述漏洞触发自然语言描述的非结构化信息. 对于结构化信息可以使用正则匹配的方法轻松获取, 然而, 对于漏洞触发描述的非结构化信息处理起来则更具挑战. 自然语言处理 (NLP) 技术是此类信息抽取常用的手段. 因此本文首先依据漏洞报告的特点, 归纳其语法依存模式, 并结合词法分析和依存句法分析, 解析漏洞描述,

提取漏洞触发的关键信息, 并形式化为漏洞触发序列, 为 Web 应用漏洞复现奠定基础.

2.3.1 漏洞触发步骤的形式化表示

对于漏洞报告中的漏洞触发步骤, 需要提取动作、动作的对象和对象的类型, 若该动作携带输入参数, 则需要记录参数值. 此外, 参数值可能是攻击负载, 其识别及提取详见第 2.3.3 节. 因此, 本文以四元组的形式记录漏洞触发步骤中必要的信息, 即<动作, 对象, 对象类型, 参数值>, 每个漏洞触发步骤对应一个四元组. 漏洞报告包含多个连续的漏洞触发步骤, 可由漏洞触发步骤对应的四元组序列表示, 称为漏洞触发序列.

2.3.2 漏洞触发关键信息识别与提取

为获取漏洞触发序列, 首先需要在漏洞报告中定位到与漏洞触发相关的描述部分, 后续再对该部分文字进行分析与理解. 我们观察到, 对于漏洞报告, 漏洞触发步骤信息一般以 STEP 关键字开头, 所以本文通过该关键字利用正则匹配定位漏洞触发描述的文本信息. 由于报告中对于漏洞触发每一步的描述可能会占据多行, 因此, 本文将两个 STEP 关键字中间的所有文字归结到一个步骤中, 以提取完整的触发步骤文本.

对于漏洞触发步骤, 其文本描述存在不规范问题, 这严重影响自然语言处理的准确性. 最常见的不规范问题为多空格问题, 即漏洞触发步骤描述文本存在多处空格, 而英文文本的标准分词器通常使用空格分割整个句子, 难以准确分析漏洞触发步骤的多空格文本. 例如, 语句“click on save button”, 单词 on 和单词 save 之间存在多个空格, 若直接进行分词操作, 则该语句将会被分成 click, on, 空格, save, button 这 5 个词, 可以看出这产生了明显的错误. 因此, 得到漏洞触发自然语言描述文本后, 需要对其进行预处理, 去除文本中多余的空格.

在得到漏洞触发相关描述并预处理后, 由第 2.2 节可知, 漏洞报告的漏洞触发描述中, 不同动作类型的句法结构一般是不同的, 因此, 本文首先通过词法分析识别当前漏洞触发步骤的动作信息; 在此基础上, 依据其动作种类, 利用归纳的语法依存模式获取漏洞触发步骤的动作对象、对象类型以及参数值. 即本阶段分为两步: 一是基于词法的动作信息识别与提取; 二是基于语法依存模式的动作对象与参数信息识别与提取.

(1) 基于词法的动作信息识别与提取

漏洞触发的动作一般是漏洞报告描述语句成分中最重要的根词, 与漏洞触发描述的动词直接相关, 因此, 本文通过词性分析的方法获取漏洞触发的动作信息. 具体来说, 通过词性分析的方法标注漏洞触发描述中每个词的词性, 并寻找其中的动词作为漏洞触发的动作信息.

需要注意的是, 一个漏洞触发步骤描述可能包含多个动作. 因此, 除了提取句中的根词外, 本文还通过提取所有与根词的依存关系为 advcl (状语从句修饰语)、conj (连词) 或 parataxis (无连词并列) 的词作为动作信息, 然后依据这些词将语句分割为若干区间, 对每一个区间进行后续四元组信息的提取. 该方法可避免一个语句中有相同动词的情况, 例如语句“click on the login button, enter username and password, click on the submit button”, 如不利用动词分割语句, 则会出现第 2 个 click 的相关信息覆盖了第 1 个 click 的相关信息, 影响漏洞触发步骤提取的准确性.

此外, 上述方法提取的动词可能与漏洞复现无关. 例如语句“login as a admin by typing admin to username and admin123 to password”, 其中真正需要作为动作属性的词只有 typing, 而动词 login 是非必要的. 针对此类无关动词, 本文采用如下判定规则, 即当存在动词没有动作对象与之匹配, 则判定其为无关动词并删除.

在动作信息识别完成后, 由于动作对应的动词形式多种多样, 但如表 1 所示, Web 应用的核心动作类型涉及 6 类, 为此, 本文归纳了漏洞报告的动词分类规则, 如表 2 所示, 并据此对漏洞报告动作对应的动词进行分类, 为后续提取四元组其他信息提供基础.

表 2 漏洞报告动词分类

动作类型	包含动词
跳转	go, login, navigate, visit, triggered, using
粘贴	capture
输入	enter, type, add, write, paste, put, insert, fill
使用	use
点击	click
修改	change

(2) 基于语法依存模式的动作对象与参数信息识别与提取

经过动词提取后, 语句被分割成数个区间, 对每个区间进行依存句法分析, 识别并提取四元组中的其他 3 个属性: 动作对象、动作对象类型以及参数信息. 针对不同的动作类型, 本文将利用第 2.2 节归纳的语法依

存模式来提取关键信息以生成对应的漏洞触发四元组.

在提取动作对象时, 漏洞触发描述语句中可能会出现一个动词对应多个动作对象的情况, 因此, 漏洞触发四元组的动作对象与参数均以集合的形式记录. 例如语句 “click on the ‘save’ and ‘submit’ button”, click 这个动作对应了 save 和 submit 两个对象. 具体来说, click 作为语句的根词, button 与 click 的关系是 obl, 依据语法依存模式, 点击动作的动作对象类型应该提取的关系中包含着 obl, 所以, button 作为四元组中的对象类型被提取出来; save 与 button 的关系是 compound, 所以 save 应该被提取为四元组的对象属性; 而 save 与 submit 两个词是 conj 关系, 所以 save 和 submit 都应该作为动作对象加入到四元组中, 最后生成的四元组为 <click, [save, submit], [button], none>.

2.3.3 攻击负载的识别与提取

攻击负载在漏洞报告中的表现形式一般分为两种: 1) 在漏洞触发步骤内部直接给出攻击负载的具体内容; 2) 在步骤内部仅提及输入攻击负载这一动作, 没有给出攻击负载具体的内容, 而在漏洞触发步骤以外的其他位置, 以 payload 为标题给出攻击负载的具体内容. 针对上述两种形式, 本文设计了两种对应的攻击负载提取方法, 具体如下.

(1) 漏洞触发步骤内的攻击负载识别与提取

不同于常规自然语言描述, Web 漏洞的攻击负载通常是符号连接的非法字符串, 大多以代码片段的形式存在. 标准分词会将攻击负载分为多个词, 导致后续自然语言处理的错误, 因此, 本文重新设计了一种自定义分词器, 将攻击负载涉及的所有单词作为一个整体进行看待, 然后, 利用依存关系分析识别并提取攻击负载.

具体而言, 自定义分词器首先利用传统分词方法根据符号或空格将句子分割成多个单词, 然后通过正则匹配的方法识别攻击负载位置, 并把攻击负载涉及的相关词组进行合并, 形成完整的攻击负载. 为精准定位攻击负载, 本文为攻击负载设计了用于提取特殊情况单词的正则表达式, 如表 3 所示.

表 3 用于提取特殊情况单词的正则表达式

匹配情况	范例	正则表达式
``	`admin 1=1`	`[^`|^]+[^`]*[^`|^]+`
""	"admin 1=1"	"[^"|^]+[^"]*[^"|^]+"
''	'admin 1=1'	'[^'|^]+[^']*[^'|^]+'
()	(something)	([^(|^)]+)
{}	{something}	{[^(|^)]+}

经过自定义分词器正确地进行分词后, 攻击负载涉及的词已经合并为一个整体. 再经过依存句法分析后, 若当前漏洞触发步骤四元组中的参数值为词语 “payload”, 则可提取与词“payload”有依存关系为 amod、compound、appos 或 advmod 的词作为攻击负载.

(2) 其他位置的攻击负载识别与提取

在漏洞报告中, 在触发漏洞的描述内可能仅提及要在某一位置输入攻击负载, 而攻击负载涉及的具体内容则位于漏洞报告的其他位置, 以 payload 为标题给出. 这种情况下, 语句中将不存在与单词“payload”有依存关系为 amod、compound、appos 或 advmod 的词. 因此, 本文将以 payload 为关键字, 使用正则匹配的方式全文搜索攻击负载.

除了攻击负载, 首页地址 URL 信息也可能出现类似的情况, 表现形式分为两种: 1) 在漏洞触发描述的第 1 个步骤内直接书写. 2) 在漏洞触发描述第 1 个步骤内仅提及登录动作, 而在漏洞报告的其他位置以 login page 为标题给出 URL. 因此, URL 的识别和提取方法与攻击负载一致.

2.4 基于语义相似度的漏洞复现脚本自动生成

漏洞报告中提取的信息不够精确, 导致使用传统字符串匹配方法无法准确匹配漏洞报告中的操作与页面中的元素, 进而构建完整的 Web 应用漏洞复现可行脚本. 因此, 本文提出一种基于语义相似度的 Web 应用漏洞复现脚本动态探索策略, 从 Web 应用首页开始探索, 探索过程的关键是将当前页面中的元素与正在处理的漏洞触发步骤进行语义相似度匹配, 如果匹配成功则触发匹配的元素并继续处理报告中的下一步, 直到漏洞报告中涉及的所有漏洞触发操作均已成功在应用中匹配到元素并执行完毕则视为复现成功.

具体来说, 如算法 1 所示, 从 Web 应用首页开始 (第 2 行), 对于漏洞报告中提取的漏洞触发步骤的每一步, 首先获取当前页面的 html 源码, 经过 bs4 转换成 BeautifulSoup 对象, 该对象可有效分析 html 标签中的信息 (第 4 行); 然后, 遍历所有页面元素, 通过将漏洞触发四元组中的对象属性与页面元素的 id 或 name 属性进行匹配的方法, 寻找与当前漏洞触发步骤四元组最相关的元素. 但漏洞报告所给的信息不一定是真实的标签属性, 传统的字符串匹配算法很难准确地匹配漏洞报告中的步骤与 Web 页面中的元素, 所以本文引入相似度计算的方法以更有效地匹配二者 (第 8 行).

具体来说, 本文使用 Word2Vec 技术[17] 把词转换成词向量进而计算词与词之间的相似程度, 并通过 similarity 函数计算页面元素的 id 或 name 属性与当前漏洞触发步骤的对象属性之间的相似度, 若相似度大于 0.7, 则认为此元素与当前漏洞触发步骤是相关的. 本文将从所有相关页面元素中选择相似度最大的元素作为匹配成功的元素, 生成对应的测试脚本 (第 16 行). 如果出现当前页面内没有与漏洞触发步骤相关元素的情况, 本文判断此时出现了漏洞报告“跳步”的情况. “跳步”是指漏洞报告中描述的漏洞触发过程缺少某个步骤使漏洞触发步骤序列不连续, 即当前漏洞触发步骤与上一步间缺少了某一步, 这种情况会导致当前页面内没有与漏洞触发步骤匹配的元素. 想要解决这种“跳步”问题, 就需要在当前页面内寻找一个元素, 尝试触发它以填补报告中漏洞触发过程缺失的这一步, 使 Web 应用的探索能够继续进行下去. 为此, 本文提出了一种“随机一步探索”策略 (第 21 行). 具体来说, 以任意的顺序与当前页面内每个元素进行交互, 用触发元素后新页面中所有元素与当前正在处理的漏洞触发步骤的相似度得分和来替换该元素的原相似度得分. 当前页面内所有元素均触发后, 根据更新后的相似度得分重新对它们进行排序, 选取得分最高的元素生成脚本并触发, 在新页面中继续处理当前漏洞触发步骤即可. 当所有漏洞触发步骤均处理完毕时, 返回完整的测试脚本集合, 算法结束.

算法 1. 测试脚本自动生成算法

输入: Web 应用首页 URL; 漏洞触发步骤四元组序列 *step*[ ].
输出: 测试脚本数组 *TestScript*[ ].

```
1 TestScript[ ] = null;
2 web.get(URL); /*以 get 方式请求 URL*/
3 for i = 0; i < step.size(); i++ do
4   element[ ] = BeautifulSoup(web.page_source)
5   flag = 0; //成功处理标志
6   rFlag = false; //是否处理过跳步情况
7   for j = 0; j < element.size(); j++ do
8     if match(step[i], element[j]) then
9       rFlag = false;
10      flag = 1;
11      matchedElement = element[i];
12      break;
13    end if
14  end for
15  if flag == 1 then
16    script = transform(matchedElemrnt);
17    TestScript.add(script);
18    execute(script); /*触发对应元素*/
19  else
20  if rFlag == false then //没有处理过跳步
21      rElement = ROSE(); //随机一步探索
22      TestScript.add(transform(rElement));
23      rFlag = true;
24  else //已经处理过跳步了
25      return −1;
26  end if
27  i−−;
28  end if
29 end for
30 return TestScript[ ]
```

3 实验结果与分析

3.1 实验设计

本文提出了一种漏洞报告理解方法, 使用自然语言处理技术提取漏洞报告中与漏洞触发相关的关键信息并形式化为漏洞触发序列, 并基于此序列自动生成测试脚本. 为了验证本文方法整体的有效性, 本文从漏洞报告平台中选取漏洞报告, 然对其行分析与复现, 并提出以下 3 个研究问题.

研究问题 1: 本文提出的攻击负载识别与提取方法是否有效?

研究问题 2: 本文提出的漏洞报告分析方法是否有效?

研究问题 3: 本文的漏洞报告复现方法是否可以为漏洞报告生成完整的执行脚本?

3.2 实验对象与环境

本文选取了漏洞收集平台 Exploit-db 中 300 余种项目的 400 份漏洞报告, 手工构建了语法依存模式, 并在此基础上, 针对 23 个开源 Web 应用对应的 26 份真实漏洞报告进行漏洞报告理解与复现实验. 其中, 漏洞报告理解实验使用 spaCy[18] 提供的 API 构建自定义分词器并结合 Stanza[19] 的其他处理管道进行漏洞报告的自然语言处理工作. 实验中将 26 份漏洞报告分为 3 类: 第 1 类是仅存在页面操作类型的报告, 在这类报告中, 步骤信息只包括对 Web 页面元素的操作, 如点击按钮, 在输入框输入信息等操作; 第 2 类是包含了请求信息的漏洞报告, 这类报告中的步骤信息里会包含发送请求的操作; 第 3 类是含有第三方软件操作的漏洞报告, 例如使用 burp 来拦截请求信息. 表 4 详细列出了漏洞

报告分类、编号与名称, 其中, 漏洞类型分别用 O、P、Q 来表示.

表 4 实验对象及分类

漏洞报告类型	Web应用编号	Web应用名称	项目对应的漏洞报告数
O 仅存在操作漏洞报告	O-1	Loan management system	1
	O-2	Simple online college entrance exam system	1
	O-3	Hospitals patient records management system	3
	O-4	Student quarterly grading system	1
	O-5	Simple student quarterly result/grade system	1
	O-6	Online DJ booking management system	1
	O-7	Rates system	1
	O-8	Online railway reservation system	1
	O-9	Online veterinary appointment system	1
	O-10	Online thesis archiving system	1
	O-11	Online enrollment management system in PHP and PayPal	1
	O-12	Employee and visitor gate pass logging system	1
	O-13	Online project time management system	1
	O-14	Online diagnostic lab management system	1
	O-15	Hospitalss patient records management system	1
	O-16	Gadget works online ordering system	1
	O-17	COVID19 testing management system	2
	O-18	Simple chatbot application	1
	O-19	Online employees work from home attendance system 1.0	1
P 含请求信息漏洞报告	P-1	Young entrepreneur e-negosyo system	1
	P-2	WordPress plugin Wappointment	1
Q 含第三方软件操作漏洞报告	Q-1	Cab management system	1
	Q-2	Online diagnostic lab management system	1

对于漏洞报告理解实验, 以上 3 种类型的漏洞报告均会进行; P、Q 两类实验报告由于其复杂性仍然需要人工处理且报告中包含的部分操作无法自动生成测试脚本, 所以漏洞复现实验只针对仅存在页面操作的 O 类报告进行.

漏洞报告理解与漏洞复现实验的硬件环境均为个人 PC 机器, 相关配置为 MacBook Pro (16-inch, 2019), 2.3 GHz Intel Core i9 处理器, 16 GB 运行内存; 编程语言为 Python.

3.3 实验结果分析

3.3.1 Web 应用攻击负载识别与提取方法的有效性分析

本文对所有漏洞报告进行了分析, 其中应用 O-17 和 P-1 对应的漏洞触发自然语言描述中不包含 URL 或攻击负载信息, 其余报告均包含至少一个此类信息. 对 26 份漏洞报告的 42 个 URL 与攻击负载进行实验, 实验结果如表 5 所示, 未使用本文提出的自定义分词器处理漏洞报告, 平均识别出了 42.9% 的此类信息, 而利用本文提出的攻击负载提取方法可准确识别 66.7% 的 URL 或攻击负载. 由此可见, 本文提出的攻击负载提取方法可以有效识别并提取漏洞报告中的攻击负载, 特别的, 使用本文针对攻击负载的特征构建的自定义分词器处理漏洞报告, 攻击负载识别准确率提升了 55.48%, 证明了本文提出的自定义分词器的有效性.

表 5 URL 与攻击负载识别效果

Web应用编号	报告中URL或攻击负载数量	URL或攻击负载识别准确率 (%)	
		未使用自定义分词器	使用自定义分词器
O-1	2	50	100
O-2	2	50	100
O-3	3	100	100
O-4	2	50	100
O-5	1	100	100
O-6	3	100	100
O-7	2	0	50
O-8	2	0	0
O-9	2	50	50
O-10	3	0	67
O-11	1	0	0
O-12	2	0	0
O-13	1	100	100
O-14	2	0	0
O-15	2	0	100
O-16	1	100	100
O-17	3	33	67
O-18	—	—	—
O-19	2	50	100
P-1	—	—	—
P-2	1	0	0
Q-1	2	100	100
Q-2	3	33	33
合计	42	42.9 (平均值)	66.7 (平均值)

对于使用本文方法后仍未识别出的 URL 和攻击负载, 经过分析, 主要是由于漏洞报告中涉及漏洞触发步骤的描述不规范导致, 例如 Web 应用 O-14 对应的漏洞报告中的语句“Create new user by adding following payload in First Name and Last Name fields.<image src/onerror=prompt(document.cookie)>”, 该语句的攻击负载直接书写在了句子后面, 既不作为前一句话的句子成分, 也没有任何标识将其标识为攻击负载; 又如 Web 应用 Q-2 中的语句“Login as admin to wp-admin portal, Go to Wappointment → Calendar (http://localhost/wordpress/wpadmin/admin.php?page=wappointment_calendar)”, 该语句不规范地出现了符号“→”, 这也对自然语言处理的结果产生了影响.

3.3.2 漏洞报告理解方法的有效性分析

为了回答研究问题 2, 本文对所有漏洞报告进行实验, 并把结果与人工手写结果进行匹配, 以评估漏洞报告理解方法的有效性.

表 6 展示了每个项目的漏洞报告理解结果的准确度. 从表 6 中可以看出, 对于大部分漏洞报告, 本文提出的基于 NLP 的漏洞报告分析方法成功完整理解了 23 份漏洞报告, 漏洞触发步骤识别的平均准确率高达 77%. 由此可见, 本文的漏洞报告分析方法可有效处理漏洞报告, 较为准确识别并提取出漏洞触发的关键信息. 值得注意的是, 部分报告的理解仍然存在错误, 主要原因同样是漏洞触发自然语言描述的不规范, 例如, Web 应用 O-8 对应的漏洞报告中的语句“Navigate to 'Schedule' > go to 'Book' or 'Revervation Form' page using the following URL: http://localhost:8000/orrs/?page=reserve&sid=1”, 该语句使用了不规范的符号“>”, 这对自然语言处理的结果产生了影响.

3.3.3 测试脚本自动生成的有效性分析

本文的测试脚本自动生成方法主要针对纯页面操作 (类型 O) 且正确完整理解的共 11 份漏洞报告进行复现, 实验结果如表 7 所示, 可见, 本文方法成功完整复现了 6 个漏洞报告, 成功率为 55%. 因为现有的漏洞复现过程主要依赖于人工完成, 而本文的测试脚本自动生成方法可以成功生成半数以上的漏洞复现脚本, 所以本文方法可以大大减少人工操作, 有效提升漏洞复现效率.

对漏洞复现失败的情况进行了原因分析, 可归结为以下 3 点原因: 1) 部分漏洞报告的步骤信息不完全, 例如, 报告中步骤信息为“login as admin, click on setting”, 但是登录的用户名和密码并没有在漏洞报告中给出, 而在项目下载页面中给出相应的信息; 2) 漏洞报告里经常出现一个步骤分几行来展示的, 并且分行也并没有规律性, 本文方法以直接拼凑的方法进行整合, 但是在少数情况下这种拼凑方法并不理想, 从而导致漏洞报告的分析出现错误; 3) 网页标签的 id 和 class 命名是人为命名的, 并且漏洞报告提取的信息有时候也不是一个有含义的词, 然而这些词并不会出现在 Word2Vec 的标准词库中, 所以会导致相似度计算的失败, 影响漏洞复现效果.

表 6 漏洞报告理解准确度

Web应用编号	总步骤数量	漏洞报告理解成功率 (%)
O-1	3	100
O-2	3	100
O-3	9	100
O-4	3	100
O-5	1	100
O-6	5	60
O-7	2	50
O-8	3	33
O-9	3	67
O-10	3	67
O-11	3	33
O-12	4	50
O-13	4	100
O-14	4	50
O-15	3	67
O-16	5	100
O-17	5	60
O-18	5	80
O-19	3	100
P-1	3	67
P-2	2	50
Q-1	5	100
Q-2	5	80
合计	86	77

表 7 测试脚本自动生成实验结果

Web应用编号-漏洞报告编号	是否成功复现
O-1-1	T
O-2-1	T
O-3-1	F
O-3-2	F
O-3-3	F
O-4-1	T
O-5-1	T
O-13-1	F
O-16-1	F
O-17-2	T
O-19-1	T

此外, 本文进一步对 “随机一步探索”策略进行有效性验证, 该策略用于解决报告中缺少漏洞触发步骤的问题. 本实验将成功复现的 6 个漏洞报告依次人工去掉一个步骤得到新报告, 再对所有新报告进行复现实验. 结果如表 8 所示.

表 8 “随机一步探索”策略有效性的验证

Web应用-漏洞报告-删除步骤	是否成功复现漏洞	
	本文探索策略	传统探索策略
O-1-1-D1	F	F
O-1-1-D2	T	T
O-1-1-D3	T	F
O-2-1-D1	F	F
O-2-1-D2	T	F
O-2-1-D3	T	F
O-4-1-D1	F	F
O-4-1-D2	T	F
O-4-1-D3	T	F
O-5-1-D1	F	F
O-17-2-D1	F	F
O-17-2-D2	F	F
O-17-2-D3	T	T
O-17-2-D4	T	F
O-19-1-D1	F	F
O-19-1-D2	T	F
O-19-1-D3	T	F
合计 (%)	58.8	11.8

实验结果表明, 针对缺少一个漏洞触发步骤的报告, 使用“随机一步探索” 策略的动态探索对比传统探索策略 (从首页开始探索, 每次在 Web 应用中选择与漏洞触发步骤最相关的元素触发, 直到当前页面内没有与步骤相关的元素时停止探索), 复现成功率提升了 4 倍. 由此可见, 本文提出的“随机一步探索”策略可有效处理报告中缺少漏洞触发步骤的情况. 本文对复现失败的漏洞报告进行分析, 可以发现, 此类报告缺失的步骤一般提供了用户名或密码等关键信息, 此类关键信息的缺失导致了漏洞复现失败.

4 结论

漏洞报告旨在记录用户在 Web 应用使用过程中, 发现的漏洞信息. 开发/测试人员可通过漏洞报告提供的信息复现相应的漏洞. 然而, 由于漏洞报告的复杂性, 目前复现报告中漏洞的工作主要由人力完成, 漏洞复现效率不佳且成本较高. 因此, 本文以提升 Web 应用漏洞复现效率为目标, 提出一种 Web 应用漏洞报告理解方法, 并基于漏洞报告理解的结果指导脚本自动生成以实现漏洞自动复现. 针对漏洞报告, 识别并抽取其漏洞触发的关键信息并形式化为漏洞触发序列; 利用语义相似度将漏洞触发序列与 Web 应用页面元素关联起来, 在此基础上自动生成测试脚本, 实现 Web 应用漏洞的自动复现. 经实验, 本文提出的基于 NLP 的漏洞报告分析方法的识别准确率达 77%, 并据此, 利用本文提出的测试脚本自动生成方法成功复现了 55% 的漏洞, 因此, 利用本文方法处理漏洞报告并复现漏洞将大大减少人工操作, 有效提升漏洞复现效率.

在未来的工作中, 我们计划继续完善漏洞报告关键信息提取的方法, 进一步提高形式化漏洞触发操作序列的准确率, 还将优化测试脚本自动生成的策略, 使其能够处理更复杂的漏洞复现情况, 扩展应用范围.

参考文献

1 国家信息安全漏洞共享平台. https://www.cnvd.org.cn/. (2022-12-30).

2 Bhuiyan FA, Shakya R, Rahman A. Can we use software bug reports to identify vulnerability discovery strategies? Proceedings of the 7th Symposium on Hot Topics in the Science of Security. Lawrence: ACM, 2020. 7.

3 Bhuiyan FA, Sharif MB, Rahman A. Security bug report usage for software vulnerability research: A systematic mapping study. IEEE Access, 2021, 9: 28471–28495. [doi: 10.1109/ACCESS.2021.3058067]

4 Fazzini M, Prammer M, d'Amorim M, *et al*. Automatically translating bug reports into test cases for mobile APPs. Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis. Amsterdam: ACM, 2018. 141–152.

5 Ricca F, Tonella P. Testing processes of Web applications. Annals of Software Engineering, 2002, 14(1): 93–114.

6 Exploit database. https://www.exploit-db.com/. (2022-12-30).

7 Kudjo PK, Chen JF, Zhou MM, *et al*. Improving the accuracy of vulnerability report classification using term frequency-inverse gravity moment. Proceedings of the 19th IEEE International Conference on Software Quality, Reliability and Security (QRS). Sofia: IEEE, 2019. 248–259.

8 Dong Y, Guo WB, Chen YQ, *et al*. Towards the detection of inconsistencies in public security vulnerability reports. Proceedings of the 28th USENIX Conference on Security Symposium. Santa Clara: ACM, 2019. 869–885.

9 Feng X, Liao XJ, Wang XF, *et al*. Understanding and

securing device vulnerabilities through automated bug report analysis. Proceedings of the 28th USENIX Conference on Security Symposium. Santa Clara: ACM, 2019. 887–903.

10 Mu DL, Cuevas A, Yang LM, *et al*. Understanding the reproducibility of crowd-reported security vulnerabilities. Proceedings of the 27th USENIX Conference on Security Symposium. Baltimore: ACM, 2018. 919–936.

11 Guo H, Xing ZC, Li XH. Predicting missing information of vulnerability reports. Companion Proceedings of the 2020 Web Conference. Taipei: ACM, 2020. 81–82.

12 Nakajima A, Watanabe T, Shioji E, *et al*. A pilot study on consumer IoT device vulnerability disclosure and patch release in Japan and the United States. Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security. Auckland: ACM, 2019. 485–492.

13 Huang Z, Lie D, Tan G, *et al*. Using safety properties to generate vulnerability patches. Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP). San Francisco: IEEE, 2019. 539–554.

14 Zhao Y, Yu TT, Su T, *et al*. ReCDroid: Automatically reproducing Android application crashes from bug reports. Proceedings of the 41st IEEE/ACM International Conference on Software Engineering (ICSE). Montreal: IEEE, 2019. 128–139.

15 Li SY, Guo JQ, Fan M, *et al*. Automated bug reproduction from user reviews for Android applications. Proceedings of the 42nd IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). Seoul: IEEE, 2020. 51–60.

16 Feng SD, Chen CY. GIFdroid: Automated replay of visual bug reports for Android APPs. Proceedings of the 44th International Conference on Software Engineering. Pittsburgh: ACM, 2022. 1045–1057.

17 Word2Vec. https://github.com/danielfrg/word2vec. (2022-12-30).

18 spaCy. https://github.com/explosion/spaCy. (2022-12-30).

19 Stanza. https://github.com/stanfordnlp/stanza. (2022-12-30).

(校对责编: 孙君艳)