分 类 号: TP393　　密　　级:

学　　号: 16207205061

西安科技大学
XI'AN UNIVERSITY OF SCIENCE AND TECHNOLOGY

硕士学位论文

Thesis for Master's Degree

WEB 注入漏洞检测方法的研究与实现

申请人姓名：张隆涛

指 导 教 师：刘涛（校内） 陈晓兵（校外）

类　　别：全日制专业学位硕士

工 程 领 域：电子与通信工程

研 究 方 向：网络安全

2019 年 6 月

西安科技大学

学位论文独创性说明

本人郑重声明：所呈交的学位论文是我个人在导师指导下进行的研究工作及其取得研究成果。尽我所知，除了文中加以标注和致谢的地方外，论文中不包含其他人或集体已经公开发表或撰写过的研究成果，也不包含为获得西安科技大学或其他教育机构的学位或证书所使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

学位论文作者签名：张齐涛　　日期：2019.6.16

学位论文知识产权声明书

本人完全了解学校有关保护知识产权的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属于西安科技大学。学校有权保留并向国家有关部门或机构送交论文的复印件和电子版。本人允许论文被查阅和借阅。学校可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。同时本人保证，毕业后结合学位论文研究课题再撰写的文章一律注明作者单位为西安科技大学。

保密论文待解密后适用本声明。

学位论文作者签名：张齐涛　　指导教师签名：刘[illegible]

2019 年 6 月 16 日

论文题目：WEB 注入漏洞检测方法的研究与实现

工程领域：电子与通信工程

硕 士 生：张隆涛　　(签名)

指导教师：刘涛(校内)　　(签名)

陈晓兵(校外)　　(签名)

摘 要

随着互联网技术的快速发展，网络应用平台逐渐成为信息交互的主要渠道，人们的生活服务呈现出网络化和信息化的特点。网络给人们的生活带来便利的同时，也伴随着各种各样的数据安全问题，保障 Web 应用的安全，及时发现并修复漏洞已变得越来越重要。Web 应用漏洞检测的本质是站在攻击者的角度审视 Web 应用程序的安全性，本文作者在研究和分析网络爬虫技术、漏洞攻击原理的基础上，比对现有漏洞检测工具，针对目前漏洞检测存在的不足进行改进，以提高漏洞检测的准确率及效率。本文的主要工作如下：

针对网站对爬虫的限制，为了解决爬虫在模拟登录时遇到的复杂验证码解析问题，本文采用 Cookie 方式模拟用户登录，通过设置相关请求头，绕过网站的反爬虫机制，突破网站对爬虫的限制，提高爬取效率。针对爬虫爬取到的 URL 存在相似数据的问题，提出一种基于树结构特征的 HASH 去重算法 HDTSF，依据 URL 的结构特征，将 URL 数据以深度为 2 进行分组得到子路径，然后对相同子路径内的 URL 数据进行相似去重处理；同时对 HDTSF 去重算法、编辑距离算法和余弦相似度算法进行对比测试，结果表明 HDTSF 相似去重算法能有效提高爬虫的工作效率。

设计并实现了基于改进爬虫技术的 Web 注入漏洞检测程序 Icrawler-scan，主要包含 URL 爬取模块、SQL 注入漏洞检测模块、XSS 漏洞检测模块和检测结果报告模块；并对 Icrawler-scan 进行了功能测试和性能测试，测试结果表明 Icrawler-scan 程序具有良好的漏洞检测效果。

关 键 词：漏洞检测；URL 相似去重；网络爬虫；HDTSF 算法；网络安全

研究类型：应用研究

Subject : **Implementation and Study on WEB Injection Vulnerability Detection Method**

Specialty : **Electronics and Communication Engineering**

Name : **Zhang Longtao** **(Signature)** Zhang Longtao

Instructor : **Liutao** **(Signature)** Liu Tao

Chen Xiaobing **(Signature)** Chen Xiaobing

ABSTRACT

With the rapid development of Internet technology, network application platform has gradually become the main channel of information interaction, People's life service presents the characteristics of network and informationization .While the network brings convenience to people's life, it is also accompanied by a variety of data security problems. It has become more and more important to ensure the security of Web applications and timely discover and repair vulnerabilities.The essence of WEB Application vulnerability detection is to look at the security of WEB applications from the perspective of attackers. Based on the research and Analysis of Network Crawler technology and Vulnerability attack principle, the author improves the shortcomings of current vulnerability detection in order to improve the accuracy and efficiency of vulnerability detection. The main work of this paper is as follows:

Aiming at the limitation of Web site to crawler, in order to solve the problem of complex verification code analysis encountered by reptiles in simulating login, this paper uses Cookie to simulate user login, by setting the relevant request header, bypassing the anti-crawler mechanism of the website, breaking through the restriction of the website to the crawler, and improving the crawl efficiency. Aiming at the problem that similar data exists in the URL crawled by the crawler, a HASH deduplication algorithm HDTSF based on tree structure feature is proposed, According to the structural characteristics of the URL, the URL data is grouped into subpaths with a depth of 2, and then the URL data within the same subpath is similarly deduplicated. At the same time, the HDTSF de-duplication algorithm, edit distance algorithm and cosine similarity algorithm are compared and tested. The results show that the HDTSF de-duplication algorithm can effectively improve the efficiency of the crawler.

The WEB Injection Vulnerability detection program Icrawler-scan based on improved

crawler technology is designed and implemented, which mainly includes URL crawl module, SQL Injection Vulnerability detection module, XSS vulnerability detection module and detection result report module. And the function test and performance test of Icrawler-scan are carried out, and the test results show that the Icrawler-scan program has good vulnerability detection effect.

Key words : Vulnerability Detection; URL Similarity Deduplication; Web Crawler; HDTSF Algorithm; Network Security

Thesis : Application Research

目 录

1 绪论

本章介绍课题的研究背景与意义和国内外研究现状，概述论文所做的主要工作，及论文的结构安排。

1.1 研究背景与意义

互联网和计算机在知识经济时代扮演着非常重要的角色，在众多的互联网应用产品中，Web 网络应用程序因其易用性、跨平台性和开放性等特点，被广泛应用于各个领域，小到个人博客、网站，大到各种各样的电商平台、数据中心及门户网站[1]。从 CNNIC 发布的中国互联网络发展报告公布的数据来看，中国的网站数量在不断增加，如图 1.1 所示。由图 1.1 可以看到，2013 至 2018 年中国的网站数量呈现逐年增加的趋势，表明人们的生活服务开始逐渐网络化、信息化，网络应用平台成为人们消息交互传递的主要方式。

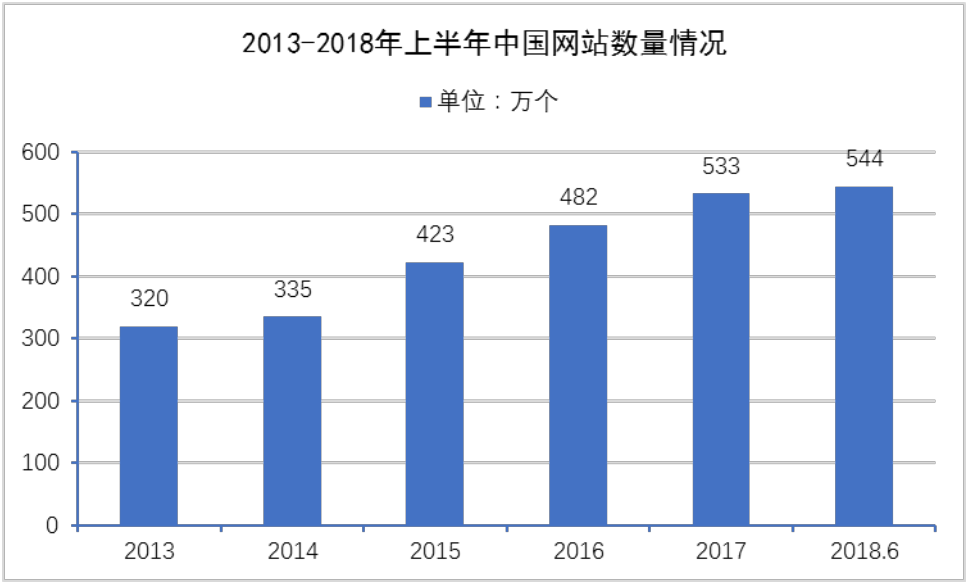

图 1.1 2013-2018 年上半年中国网站数量情况

随着信息在网络应用上的存储、交换和传输，Web 应用的安全问题也日益突显出来。2017 年 3 月，58 同城网站被爆出漏洞，大量的简历数据被泄露，很多的应届毕业生被恶意骚扰诈骗，2018 年 8 月，华住旗下酒店 5 亿酒店开房数据被泄露，开房信息在网络上被售卖，网络威胁真实的发生在我们的身边。攻击者利用 Web 应用程序的漏洞进行各种恶意的网络攻击，严重威胁到用户的信息安全，一旦发生信息泄露，这将给企业和个人带来无法挽回的后果和不可估量的损失。在所有的 Web 应用产品中，还没有哪个应用可以声称自己是绝对安全的，都存在各种各样的缺陷[2]。Web 应用是互联网和企事业单位信息管理的主要模式[3]，而信息安全网络攻击多发生在 Web 应用层而非网络层面上，并

且据 Gartner 最新数据显示，三分之二的 Web 网站相当脆弱，存在各种漏洞，容易受到黑客攻击[4]。漏洞的本质是 Web 应用程序存在缺陷，其中既有设计方法和工程实现上的技术问题，也有质量管理上的问题，更受限于程序开发者认识上的局限性[5]。

OWASP(开放式 Web 应用程序安全项目)每年都会公布 Top 10 漏洞(10 大漏洞安全列表)，这个列表总结了 Web 应用程序最常见、最危险、最可能的十大漏洞，可以帮助互联网科技公司和开发团队有效规范应用程序开发流程和测试流程，从而提高 Web 应用产品的安全性[6]。其中注入漏洞居于榜首，可见其危害性及广泛性[7]。引发注入漏洞攻击的原因是 Web 应用程序开发者在编写代码时，只注重于业务功能的实现，忽略了 Web 应用程序本身的安全性，未能有效的对输入进行正确的设置或验证判断，而且数据库本身提供的安全策略是有限的，使得攻击者可以有目的的对数据库进行攻击，实现自己的目标意图，严重时可以获取超级管理员的权限。

保证 Web 应用程序的安全性，从长远来看，最佳方案是在开发阶段从源头上避免程序产生异常漏洞，而不是在后期建立修复措施。如何有效的评估 Web 应用程序的安全性，以及及时发现 Web 应用中存在的安全漏洞，是保障企业与个人用户信息数据安全、避免遭受网络攻击的关键。因此，为了提高 Web 应用程序的安全性，需要采取主动防御的措施，在攻击发生之前将损失降到最低。漏洞扫描正是这样的一种主动防御技术，通过对 Web 应用程序进行模拟攻击，发现潜在的漏洞，并及时给出修复建议，在一定程度上保证了应用系统的安全性。

在日益成熟的网络技术和 Web 应用攻击手段多样化的情况下，现有的商业级别的 Web 应用程序扫描工具开始慢慢暴露出自身的局限性。因此，在一些中小型的实际开发项目中，大多数中小型企业会参考现有的软件工具，结合自身的功能需求，为项目量身定做一款实用的 Web 漏洞检测工具。这类工具通常具有较强的实用性，可以根据项目的实际需求便捷改进。故基于已有的漏洞扫描检测工具，掌握其设计原理和软件结构，针对存在的潜在问题和不足，提出改进措施，设计并实现符合自身需求的漏洞扫描检测工具，具有一定的现实意义。

1.2 国内外研究现状

《2015 年全球数据泄露调查报告》中表明数据泄露的原因是源于 Web 应用研发技术的缺陷，其中 Web 应用程序的 SQL 注入漏洞首当其冲，这种情况将会持续很长一段时间。SQL 注入漏洞攻击者一般情况下是将目标定位于 Web 应用后台数据库信息[8]。随着 B/S 模式网络应用的普及，SQL 注入攻击开始成为数据库攻击的首要选择，由 SQL 注入攻击引发的信息安全威胁的比例也在不断增加。

越来越多的程序开发者开始意识到 Web 应用程序安全注入漏洞带来的危害，国内外的研究人员对 Web 漏洞检测进行了广泛和深入的研究，并取得许多有价值的学术成果，

提出了许多针对 SQL 注入漏洞和 XSS 漏洞的预防和检测方案，并设计了一些针对 SQL 注入攻击和 XSS 漏洞的检测工具，对 SQL 注入和 XSS 漏洞在互联网中的蔓延起到了有效的遏制作用。

1.2.1 国内研究现状

国内对于漏洞注入攻击的研究起步较晚，针对注入漏洞(以 SQL 注入为主)的防范研究基本体现在以下四个方面：

（1）用户提交信息的数据安全验证机制。主要用于防止用户输入与数据库相关的关键字符或特殊数字，将这些特殊数据（如 or、union 等）正式提交业务逻辑模块进行处理之前对所包含数据的安全性进行验证，从而避免可能由注入攻击带来的各种潜在危害[9]。

（2）屏蔽请求报错消息提示。注入攻击者通常根据系统请求返回的报错页面中所含有的服务器或者数据库相关的关键信息提示来获取一些相关的安全漏洞信息。通过对产生的错误信息进行屏蔽以使攻击者无法获取到所期许的错误提示，从而不能从报错信息提示中获得和数据库相关的有用信息[9]。

（3）预处理语句。对 SQL 语句进行预处理操作，并通过参数绑定进行数据传值，而不是直接对字符串进行拼接，这样可以大大降低由于数据库连接和操作方式不当而触发攻击发生的概率。

（4）敏感信息加密。敏感信息往往是 SQL 注入攻击者的攻击重点(如身份信息、用户账号密码等)，在程序开发阶段通过对敏感信息进行加密处理，防止攻击者直接或间接获取敏感数据。

对于 Web 注入漏洞的异常检测，国内一些学者进行了相关技术研究，并且取得了一定的学术成果。

杨高明在总结了现有的 SQL 注入漏洞检测技术后，改进了 SQL 注入漏洞的检测方法，提出了基于模糊测试、常规检测和基于启发式漏洞检测的检测技术相结合的一种混合式的 SQL 注入漏洞检测方法[10]。

刘磊等人提出了一种形式化的 SQL 注入漏洞测试用例生成模型，通过全局测试规则(GTM)来检测生成的测试用例，并提出一种多阶段检测方法(MPDA)来实现测试用例的动态生成和检测程序控制[11]。

彭赓等人改进了现有的网络爬虫技术，增加了爬虫深度，并改进了 URL 的筛选策略和流程。其设计的漏洞检测工具丰富了检测的手段，降低了检测的漏报率，提高了检测的速度和效率[12]。

秦广赞等人提出一种使用静态分析的方法来检测 SQL 注入攻击，通过对 Web 应用程序中包含的代码文件进行静态分析，分析用户输入的参数在 Web 应用中的传播途径与 SQL 语句的构造过程，形成策略文档。在动态执行过程中，用户输入的内容被策略文档

里面的输入参数所替代，在输入参数进行比较后，判断新形成的 SQL 语句和原来的 SQL 语句在语法结构和语义内容上是否存在差异，进而判断 SQL 注入攻击是否存在[13]。

李鑫提出了一种动静结合的检测方法，通过静态分析方法获取持久存储信息，解决动态分析无法解决的 Web 应用多阶段间逻辑联系问题；通过动态分析方法获取目标元数据，解决静态分析无法正确定位污点信息持久存储位置的问题；最后通过模糊测试技术验证疑似漏洞，降低误报率[14]。

肖泽力提出了基于用户行为分析的 SQL 注入攻击检测方法，通过研究 SQL 注入攻击者行为和正常用户的行为，以及 Web 应用在正常条件下和多种攻击行为下的响应和状态变化，然后提取出能够代表 SQL 注入攻击的行为特征，并将行为特征转化为数值化特征向量，利用聚类分析的方法和算法，训练并构建出能够检测 SQL 注入攻击的模型。同时也提出了基于 DNN 的 SQL 注入攻击检测方法，通过建立 URL-SQL 映射模型，对执行的 SQL 语句进行语句分析和响应结果分析，经过合理的特征选择和特征向量合成，提取出能够标识 SQL 语句自生结构和运行结果的特征，最后运用深度神经网络(DNN)训练并构建出合理的异常检测模型，达到了 SQL 注入异常检测的目的[2]。

翟涵在理解并掌握网络爬虫技术、SQL 注入攻击、XSS 漏洞攻击原理的基础上，通过大量的模拟攻击检测实验，掌握了漏洞的攻击模式，设计了更为全面的漏洞测试样例库，并使用广度优先爬取策略对 Web 页面爬取[15]。

1.2.2 国外研究现状

注入漏洞扫描检测的相关研究最早出现在国外，国外的研究者在入侵检测、安全资源管理、注入攻防、防御黑客和防火墙产品开发等方面都有过深入的研究[16]。漏洞扫描检测的目的是防止恶意网络攻击者对潜在的漏洞进行注入攻击，通过对 Web 应用程序的注入点进行模拟攻击，可以及时发现漏洞，从而为最终漏洞修复解决方案的实施提供参考。国外的研究学者在 SQL 注入检测方法中的研究思路为通过重点分析用户的输入、SQL 语句的动态构建、SQL 语句执行的整个过程，同时研究 Web 应用程序的正常运行状态变化和应用受到攻击时的状态变化，从而达到检测异常入侵的目的。

Ouarda 等人提出一种基于网页相似度分析的方法来检测和防御 SQL 注入攻击，该方法认为 Web 应用程序对合法的 Request 请求返回的响应内容是稳定的，而受到网络入侵攻击后的 Web 应用服务器返回的响应与正常情况是不一样的；该方法是基于客户端向 Web 服务器发送 HTTP 请求，然后分析服务器返回的响应页面信息，通过响应页面的错误信息识别技术和服务器返回的正常页面内容的相似度来判断注入漏洞是否存在[17]。Ouarda 等人将发送的 Request 请求划分为三类：语法无效的 Request 请求、语法有效的 Request 请求、随机的 Request 请求，并将不同请求得到的服务器响应页面分成四类：Ref、Alert、Inval、Val，通过将 Alert、Inval、Val 页面集合和正常页面集合 Ref 进行比较并计

算相似度，通过这种方式来判断 Web 应用是否受到了攻击[17]。

C Shi 等人提出了一种基于自学习的 SQL 注入攻击检测过滤方法，这个方法可以自动地学习合法的 SQL 语句结构从而构建自身的安全检测知识库，在检测漏洞时通过匹配知识库里的 SQL 语句来判断是否存在 SQL 注入漏洞[18]。

1.2.3 URL 去重算法的研究现状

在对 Web 应用站点进行链接爬取的过程中，URL 链接去重也是爬虫工作的主要部分。链接去重可以避免相同或相似的页面被重复的爬取，节约资源，提高爬虫的工作效率，同时也可以提高后续漏洞扫描检测的效率。国内外对于 URL 链接去重也有一定的学术研究。为了解决编辑距离算法在长文本的大规模匹配效率的不足，张衡等人提出了一种提前终止的优化策略，根据距离矩阵中元素内在的联系，归纳出一种递推关系式，可提前判断两个文本是否满足预先设定的相似度阈值[19]。现今大量算法的改进主要是为了满足普通用户对快速搜索的要求，较少考虑对特定主题准确性的需求，徐晨初等人提出一种基于优化爬行路径的聚焦爬虫算法(OPFA)，通过计算主题和页面、页面和页面间语义相似性，得到页面相似性排序及分类结果，形成爬虫优先级并优化爬行路线[20]。张春燕等提出一种基于重复数据删除的新型 URL 过滤匹配机制(DBM)，DBM 在哈希表中缓存重复 URL 的信息，以避免 URL 过滤系统重复扫描重复的 URL[21]。Lim H 等通过分析布隆过滤器的优缺点，有针对性的提出了多维布隆过滤器，它由多个基本的布隆过滤器组成，当所有的布隆过滤器产生误判时，才会产生误判，这样提高了去重的准确度[22]。

国内外学者对去重算法的研究取得了一定成果，但仍然存在一些问题亟待解决，大多数研究学者针对的是相同 URL 链接去重问题，却忽略了相似 URL 链接的去重问题，在漏洞扫描检测中，对相似的 URL 链接进行模拟攻击异常检测，相当于进行了多次重复的请求操作，这是一种资源浪费。因此，如何设计高效的 URL 去重算法是漏洞扫描领域中一个值得考虑的问题。

1.3 主要研究内容

本文的研究内容主要包含以下几个方面：

（1）针对网站对于爬虫的限制，比如登录限制、反爬虫等，本文采用 Cookie 的形式保持会话跟踪，解决登录验证问题；同时，通过添加相关请求头字段，以避过反爬策略。

（2）通过研究分析现有的两种相似度去重算法即编辑距离算法和余弦相似度算法的不足及缺点，根据 URL 的结构特征，提出一种基于树结构特征的 HASH 去重算法(Hash Deduplication algorithm based on Tree Structure Features，HDTSF)，由于 URL 的树结构特性，且相似 URL 主要出现在同一子结构目录下，因此只需要判断同一子结构目录下的

URL 是否相似，这样可以提高 URL 去重的效率及准确度。

（3）设计并实现了基于爬虫技术的 Web 注入漏洞检测程序 Icrawler-scan，将 HDTSF 算法运用到 Icrawler-scan 程序中，进行了程序的可行性对比测试。Icrawler-scan 程序包含四个模块：URL 爬取模块、SQL 注入检测模块、XSS 检测模块和检测结果报告模块；其中 URL 爬取模块是用于获取 URL 链接并对链接进行去重；SQL 注入检测和 XSS 检测模块是对爬虫获取的 URL 进行异常检测，判断是否存在漏洞；检测结果报告模块是在完成漏洞扫描检测后，生成结果报告，以直观地展示 Web 站点的漏洞情况。

（4）漏洞检测程序 Icrawler-scan 对 Web 注入漏洞检测进行了性能优化，并将检测结果与其他漏洞扫描软件的检测结果进行比较，以验证本文实现的 Icrawler-scan 程序的检测效果。

1.4 论文结构安排

论文章节安排如下：

第一章为本文的绪论部分，首先讲述了课题的研究背景及研究意义，然后概述了在 Web 漏洞扫描检测领域国内外的研究现状及 URL 去重算法的研究现状，最后概括了论文的主要研究内容及结构安排。

第二章为本文的理论基础，主要阐述了论文所涉及到的相关理论技术知识，包括 SQL 注入、网络爬虫等。

第三章主要阐述 Web 注入漏洞检测方法的设计和改进，通过分析现有的 URL 去重方法的不足，根据 URL 的结构特征，提出基于树结构特征的 HASH 去重算法(HDTSF)，并对 SQL 漏洞和 XSS 漏洞检测模块的工作流程及原理进行了阐述。

第四章主要阐述 Web 注入漏洞检测方法的实现，包括程序的工作流程，程序的初始化参数设置，URL 爬取模块的实现、注入检测模块的实现和检测结果报告的实现。

第五章是测试与分析，介绍程序测试的实验环境，对提出的 HDTSF 去重算法进行对比测试及可行性测试与分析，对实现的 Icrawler-scan 程序进行功能测试，并将本文所设计实现的漏洞检测程序 Icrawler-scan 与 AIscanner 漏洞检测软件、WebVulScan 漏洞扫描软件从检测效率、准确率等方面进行对比分析。

第六章是总结与展望。对本文研究内容作了归纳总结，分析了不足的地方，并指出未来的研究方向与接下来的工作。

2 关键技术

本章主要讲述 Web 安全相关的技术理论，包括 HTTP 协议、SQL 注入、网络爬虫和 XSS 漏洞，为后续方法实现奠定理论基础。

2.1 HTTP 协议

本文研究的 Web 注入漏洞检测方法本质上是通过向服务器发送 HTTP 请求数据包，然后通过服务器返回的响应数据来判断是否存在漏洞。本节主要概述 HTTP 协议的相关知识。

HTTP 协议全称是超文本传输协议(Hyper Text Transfer Protocol)，是一种用于分布式、协作式和超媒体信息系统的应用层协议[23]。

HTTP 是一个客户端(用户)和服务器(网站)之间请求资源和应答资源的标准(TCP)。通过使用网络爬虫、网页浏览器或者其他网络请求工具，客户端向指定的服务器端口(端口默认为 80)发送一个 HTTP 请求，该客户端被称为用户代理程序(User Agent)。一些数据资源，如超文本标记语言文件和图像，存储在响应服务器上，该服务器被称为源服务器(Origin Server)。在源服务器和用户代理程序之间可能有多个中间层，比如网关、代理服务器或者隧道。

（1）一次完整的 HTTP 请求工作流程如图 2.1 所示。

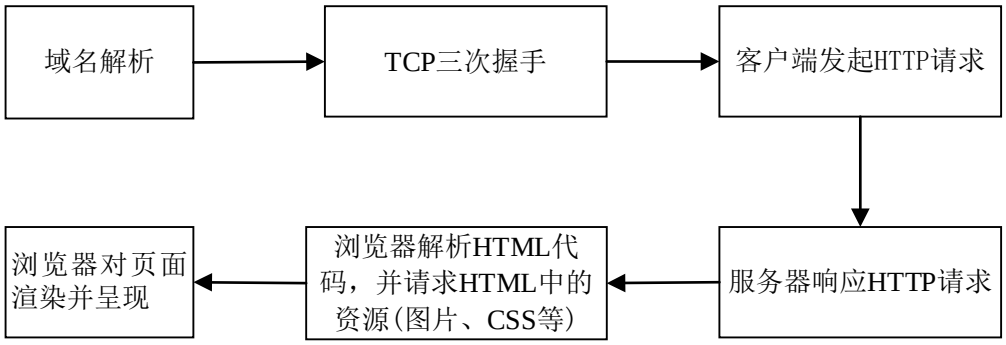

图 2.1 HTTP 的工作流程

① 域名解析：域名系统 DNS 解析输入的网址得到主机 IP 地址，并结合主机相关信息，封装成 HTTP 请求数据包。

② TCP 三次握手：在进行数据请求之前，客户端与服务器通过三次握手建立 TCP 连接。

③ 客户端发起 HTTP 请求：建立 TCP 连接后，客户端发起 HTTP 请求信息给服务

器。请求的数据格式为：资源标识符(URL)、客户端信息、协议版本和请求内容等[24]。

④ 服务器响应 HTTP 请求：服务器接收到来自客户端的请求后，返回相应的响应资源信息。其响应数据格式为：协议版本号、响应状态码及其状态描述、服务器信息和实体信息等，客户端接收到来自服务器的资源信息后，经过渲染最终呈现给用户。

⑤ 关闭 TCP 连接：一般情况下，当服务器返回了响应的资源信息后，就会尝试关闭 TCP 连接，但是如果服务器或浏览器在其头部信息中加入了字段键值如 Connection：keep-alive，则 TCP 连接在数据响应后仍然保持活跃状态。保持连接节省了每一个资源请求建立新的连接所需要的时间，并且节约了网络带宽。

（2）HTTP 请求报文格式如图 2.2 所示。

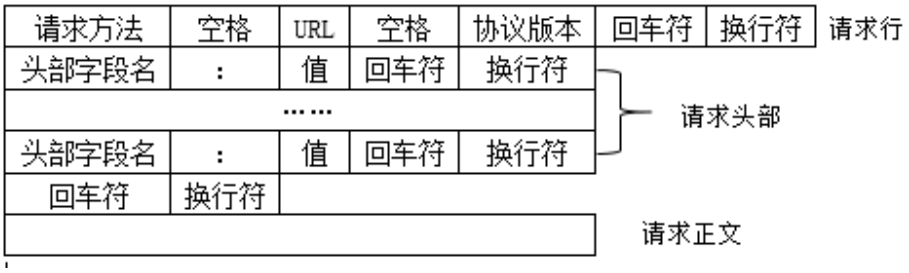

图 2.2 HTTP 请求报文格式

① 请求行：主要声明请求的方法、资源路径、协议版本。

② 请求头部：由键值对组成，每行一对，关键字和值用英文冒号“:”分隔。请求头部是用于通知服务器有关于客户端请求的信息，常见的数据请求头有：User-Agent：产生请求的浏览器类型，爬虫中添加该请求头可以将程序伪装成浏览器以达到欺骗目标服务器的目的。Accept：客户端可识别的资源类型列表。Host：请求的主机名，允许多个域名共用一个 IP 地址，即虚拟主机。

③ 空行：位于请求头之后，请求正文之前，表示请求头结束。

④ 请求正文：请求数据一般不在 GET 方法中使用，而在 POST 方法中使用。POST 方法常见适用场景为数据请求、表单提交，而 GET 方法适用场景为数据查询等操作。与数据请求相关的常见的请求头是 Content-Type、Content-Length。

（3）HTTP 响应报文格式如图 2.3 所示。

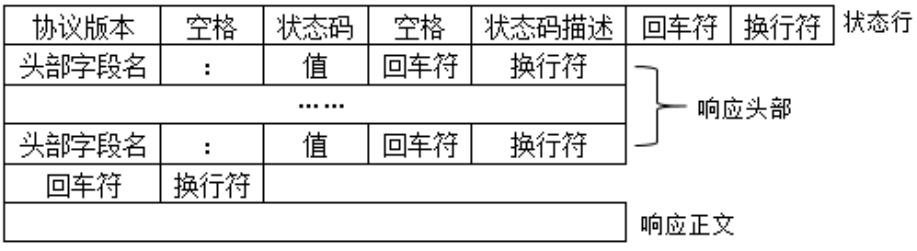

图 2.3 HTTP 响应报文格式

① 状态行：由 HTTP 协议的版本，响应状态码和对状态码的描述组成。状态码主要有 5 类，每一类表示的含义如表 2.1 所示。

表 2.1 状态码含义

类别	含义
1xx	指示信息，表示请求已接收，继续处理
2xx	成功，表示请求已被成功接收、理解、接受
3xx	重定向，表示要完成请求必须进行更进一步的操作
4xx	客户端错误，请求有语法错误或请求无法实现
5xx	服务器端错误，服务器未能实现合法的请求

② 响应头部：主要用来说明服务器或客户端的一些附加信息，由键值对组成。

③ 空行：用于将响应头部和响应正文间隔开，是必须的。

④ 响应正文：主要是服务器返回给客户端的信息，可以是键值对信息，也可以是文本信息。

2.2 SQL 注入

随着技术的发展，B/S 模式开发越来越受关注。一般情况下，Web 应用程序在开发的过程中对数据的增删改查都会使用到数据库，数据一般都存储在数据库中，而网络安全问题本质上就是数据安全问题。与数据库相关的安全问题中，SQL 注入(SQL Injection)是黑客最为广泛使用的数据库攻击手段之一。SQL 注入攻击是发生在应用程序和数据库层的安全漏洞，由于程序开发者的开发水平和安全知识及经验参差不齐，缺乏代码安全意识，只关注业务功能模块的快速开发实现，而忽略了代码的安全性，这种现象在中小型的非安全类的公司尤为普遍。大部分的程序开发人员在编写应用代码时，没有对用户输入数据的合法性进行充分的验证判断，导致应用程序在上线运行的过程容易出现安全问题。恶意攻击者通过某种方法得知系统存在漏洞后，提交一段携带可疑数据的 SQL 代码，由于服务器没有对提交的数据进行安全性验证，被误认为是可执行的 SQL 指令去，获取目标数据，然后攻击者根据程序返回的结果信息，分析并获取系统的敏感信息，比如用户的账号、密码等隐私数据，进而获取系统更高的权限。如图 2.4 所示为 SQL 注入攻击的一个例子。

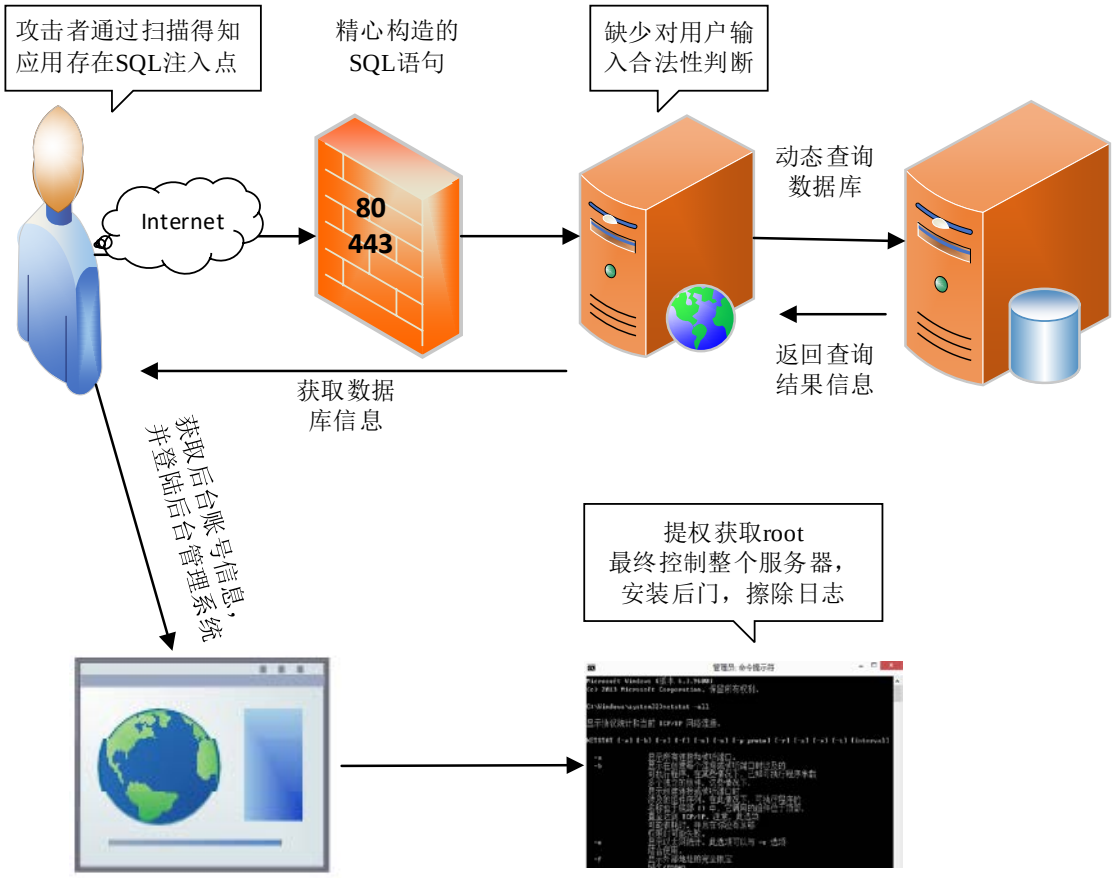

图 2.4 SQL 注入攻击例子

2.2.1 SQL 注入产生原因

SQL 注入攻击通常表现在以下几个方面：

（1）转义字符处理不当

SQL 数据库将单引号字符(‘)解析为代码与数据间的分界线；单引号外面的内容均是需要编译运行的代码，而用单引号引起来的内容为数据[25]。如果将单引号添加到 Web 页面字段或 URL 后面，则可以通过服务器返回的错误信息得到数据库相关信息，使用此方法能快速识别 Web 站点是否会容易受到 SQL 注入攻击。

（2）错误信息处理不当

错误信息处理不当会给 Web 应用站点带来许多安全方面的问题，最常见的问题是将详细的服务器内部错误信息(如错误的代码，使用的数据库等)展示给攻击者用户，正常情况下，当网站上线运行后，应隐藏错误细节，不应该暴露给攻击者，因为往往攻击者就是通过这些细节得到可疑漏洞的主要线索。如图 2.5 所示为程序片段直接暴露了数据库错误信息。

```
if (!$result) {  //如果有任何的错误
                 //检查错误的信息并且显示错误信息
   die ('<p>Error: '. mysql_error() . '</p>');

              }
```

图 2.5 数据库错误信息处理不当例子

如图 2.5 所示，PHP 中有个内置函数 mysql_error，该函数的作用是返回一个 mysql 操作产生的错误文本信息，然后通过 die 函数来显示错误信息。程序开发者在开发站点的过程中，系统未上线之前，为了便于排查错误，往往会添加错误提示类函数，但是在网站交付上线时却忘记注释掉或擦除掉这些错误信息提示代码，这就给恶意用户提供了漏洞攻击的机会。

（3）不安全的数据库配置

数据库在安装的时候都会为用户配置默认信息。如 SQL Server 数据库使用“sa”作为系统默认的管理员账户，MySQL 数据库使用“root”或“anonymous”作为系统默认的管理员账户，Oracle 数据库创建数据库时通常会创建 SYS、SYSTEM、DBSNMP 和 OUTLN 等账户。一般情况下用户会按照默认方式进行预设配置，如果攻击者通过初始默认的口令来尝试获取数据库的权限，将可能会导致数据库的信息遭到泄露。因此，一般情况下建议程序开发人员在安装数据库时注意下数据库的安全配置，尽量避免使用系统默认配置。

（4）多个提交处理不当

在面对大型的 Web 开发项目时，有些程序开发人员会对输入的数据进行安全性验证，而有些程序开发人员为了急于求成而忽略了数据的安全性验证，以理想化输入来进行代码的逻辑编写。程序开发人员想当然的认为用户会老老实实的遵循他们设计好的逻辑程序流程去操作，因此在处理第一个表单时对数据进行了验证，而后续的则不进行验证，期望用户按照表单的先后顺序进行操作。但实际上，通过 URL 的乱序请求资源时，能够很容易的避开预期设计的逻辑操作流程。

2.2.2 SQL 注入的危害

攻击者在执行 SQL 注入攻击时，可以获取相应的数据库权限，更有甚者可以获得超级管理员权限[26]。一旦攻击者获取权限后，将对网站拥有者和用户造成巨大的危害，可能造成的危害有：

（1）数据库中的数据信息泄露，例如企业和个人用户的隐私数据、账号密码、身份信息等，并且数据库的数据信息可能被随意修改删除。

（2）数据库的表结构信息被攻击者获取，以进一步实现更深层次的攻击，例如使用 select * from sys.tables 可以获取数据库的所有表信息。

（3）获取系统某些权限后，恶意攻击者通过在网页中加入恶意链接、恶意代码等，来达到破坏企业内部系统的目的。

（4）网络攻击者通过使用命令 xp_cmdshell “FORMAT C:”来破坏硬盘数据，最终使系统瘫痪。

（5）企业门户网站被黑客篡改，添加各种垃圾及违法广告，最终导致企业的经济收入蒙受重大损失，并有可能降低人们对企业的信用评价。

2.2.3 SQL 注入的避免方法

SQL 注入攻击是可以避免的，可避免网站产生 SQL 注入攻击的方法有：

（1）在设计应用程序时，数据访问功能全部通过参数化查询来实现，避免使用字符串拼接来组合 SQL 语句。

（2）使用正则表达式对输入的参数进行过滤，以加强对用户输入数据的安全合法性验证。

（3）使用其他更安全的方式连接和操作数据库，例如已修正过 SQL 注入问题的数据库连接组件、ASP.net 的 SqlDataSource 对象等。

（4）在网站交付上线前，将 SQL 错误信息提示代码注释或删除，如类型错误、信息不匹配等信息提示，以防止攻击者利用错误信息进行 SQL 注入攻击。

（5）在应用程序上线发布之前使用安全漏洞扫描检测工具对 Web 应用程序进行检测，以便及时修补存在的漏洞。

2.3 XSS 漏洞

XSS(Cross Site Scripting，跨站脚本，为了和 CSS 层叠样式表区别，所以改为 XSS)是 Web 站点常见的另一种攻击漏洞，它也是注入攻击漏洞的一种，网络用户在浏览携带有恶意代码的网页时就会受到影响[27]，恶意注入代码包括超文本标记语言及其它客户端脚本语言，常见的恶意程序代码是 JavaScript 脚本。当漏洞攻击成功后，恶意攻击者可以获取更高的系统权限(如执行一些操作)、敏感的网页内容、会话信息等。

对于 XSS 漏洞的检测，可以通过构建携带恶意数据的 HTTP 请求来模拟浏览器请求操作，通过判断服务器返回的响应数据中是否包含恶意脚本代码来验证是否存在 XSS 漏洞。

2.3.1 XSS 攻击手段和目的

一般情况下，攻击者会自己搭建一个站点，将收集到的被攻击者获取的相关隐私数据(如 Cookie 或其他敏感信息)以数据库等形式记录下来，常见的 XSS 漏洞攻击手段和目的如下：

（1）盗取用户登录站点后的 Cookie 信息，获取敏感信息，比如账号密码。

（2）通过使用 Iframe、Flash 或 Frame 等方式，以被浏览器信任的用户身份去执行一些操作，如发布敏感微博内容、进行不正当的投票活动、添加或删除好友等。

（3）在用户访问量大的一些页面添加并执行 XSS 脚本代码，用来攻击系统性能较差的中小型网站，达到 DDoS 攻击的效果。

2.3.2 XSS 漏洞的防御

XSS 漏洞产生的本质是由于服务器端对用户提交的数据没有经过严格的安全验证及过滤处理，因此针对 XSS 漏洞的防御原则就是对输入数据进行过滤，对输出数据进行编码。比如对输入的数据的格式、类型进行严格规定，对一些特殊的 HTML 标签(如<script>、<iframe>)进行过滤；在输出数据之前对潜在的威胁字符进行编码、转义(比如< 转成 <)；通过设置 Cookie 的 HttpOnly 属性，这样就可以避免 XSS 利用 JavaScript 的 document.cookie 方法来获取站点的敏感 Cookie 信息。

2.4 网络爬虫

如果把互联网比作一张巨大的蜘蛛网，那么一台计算机上的数据便是蜘蛛网上的一个猎物，而爬虫程序则是一只小蜘蛛，沿着蜘蛛网抓取自己想要的猎物或者数据。网络爬虫(Web Crawler)[28]从一个被称作种子的统一资源地址(URL)列表开始数据的爬取。网络爬虫的框架结构如图 2.6 所示，爬虫将 URL 链接对应的网页下载下来，并存储到网页库中，然后提取出该页面上的所有可供访问的 URL 链接，对于已爬取的 URL 则不再进行重复的资源访问，对于新的 URL 链接，继续进行资源的请求，提取 URL 链接数据，重复进行这一过程，当满足某一条件要求或者没有新的 URL 链接时，网络爬虫便停止工作。

网络爬虫一般包含三种策略，即深度优先策略、广度优先策略和最佳优先策略。深度优先策略就是指从起始网页开始，选择一个 URL 访问，顺着链接不断爬取下去，直到不能深入，然后继续处理另一条链接的路线。广度优先策略是指在爬取过程中，一层一层的进行抓取，仅当当前层爬取完成后，才能进行下一层的搜索爬取。最佳优先策略是指按照网页分析算法，计算待爬取 URL 与目标网页的相似度，并选取评价较好的 URL 进行爬取，是一种局部最优搜索算法[29]。

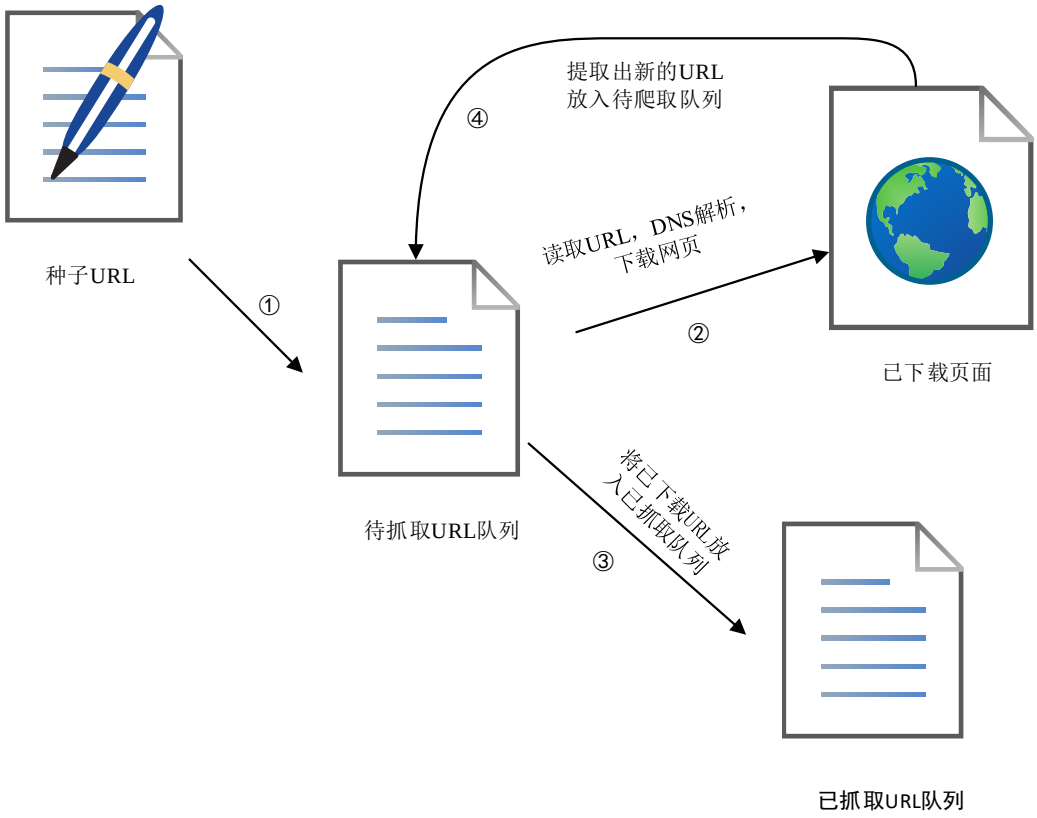

图 2.6 通用网络爬虫框架

2.4.1 网络爬虫的种类

网络爬虫按照系统的组织架构和实现的爬取技术，通常可以分为以下几类：通用网络爬虫(General Purpose Web Crawler)，聚焦网络爬虫(Focused Web Crawler)、增量式网络爬虫(Incremental Web Crawler)、深层网络爬虫(Deep Web Crawler)。网络爬虫实际中通常使用几种技术相互结合实现的[30]。

（1）通用网络爬虫：又称为全网爬虫，顾名思义就是对整个互联网资源进行数据采集，从一个种子 URL 开始搜索并扩充到整个 Web 的数据采集。由于涉及到商业利益，因此技术细节的具体实现一般情况下是保密的。这类爬虫的爬行范围和数量庞大，对速度和存储要求较高，常用于搜索引擎搜索广泛的主题，有较强的应用价值[30]。

（2）聚集网络爬虫：又称为主题网络爬虫，根据与主题的内容相关性来进行有选择性的提取数据。和通用网络爬虫相比较，聚焦爬虫仅仅有针对性的提取与主题相关的内容页面数据，这样便极大地节省了网络资源和硬件资源，聚集网络爬虫的适用场景为某一特定领域的信息需求。

（3）增量式网络爬虫：可以称为更新爬取，它是指爬虫仅对发生变化的页面或者新增加的页面进行爬取，而不是对所有网页进行爬取更新，这在一定程度上保证了爬虫爬

取的页面基本上都是最新的页面。和周期性爬虫相比，增量式爬虫只爬取内容发生变化的页面，有效地减少了数据的下载量。

（4）深层网络爬虫：Web 页面一般分为两种类型，即表层网页和深层网页。表层网页是指 Web 页面主要由静态资源构成，多是不变的网页；而深层网页是指不能直接通过静态链接获取信息，而是需要通过动态网页搜索技术才能获取到的 Web 页面，例如用户注册登录后才能查看的网页便是深层网页。

2.4.2 URL 相似性去重

在实际的网络爬虫过程中，不可避免的会爬取到重复或相似的链接，重复的抓取 URL 是一种资源浪费，同时降低了爬虫效率。因此，有必要对 URL 进行重复或相似性判断，对重复或相似的 URL 不做任何处理，而对于不同的 URL 链接则进行后续的操作。URL 本质上还是字符串，因此可以看作是特殊的字符串。针对字符串去重的方法有很多，比如 Bloom Filter 去重算法、数据库去重、HASH 去重等，相关学者已作了研究，本文不再进行讨论；而针对 URL 相似性去重的研究比较少，因此本文介绍两种字符串相似性去重算法，以引出下一章节提出的 URL 相似度去重算法。

（1）编辑距离算法

编辑距离是指将源字符串 S 转换为目标字符串 T 需要的最少的编辑操作次数。这里的编辑操作过程包括插入、删除一个字符，或者将一个字符替换成另一个字符。一般情况下，编辑距离越小，则说明两个字符串的相似度越大[31]。

设有源字符串 S 和目标字符串 T：$S=s_1s_2s_3...s_m$，$T=t_1t_2t_3...t_n$。则字符串 S 和 T 的(m+1)x(n+1)阶匹配关系矩阵 LD 为：

$$LD_{(m+1)x(n+1)}=\left(d_{ij}\right)\left(0\le i\le m,0\le j\le n\right) \tag{2.1}$$

对于 LD 矩阵中的每一个元素，按照下述公式进行数据填充：

$$d_{ij}=\begin{cases} i & i=0 \\ j & j=0 \\ \min\left(d_{i-1,j-1},d_{i-1,j},d_{i,j-1}\right)+a_{ij} & i,j>0 \end{cases} \tag{2.2}$$

其中：

$$a_{ij}=\begin{cases} 0 & s_i=t_i \\ 1 & s_i\neq ti \end{cases} \quad \left(i=1,2,3,4,5,6.....n\right) \tag{2.3}$$

矩阵 LD 右下角的元素 d_{mn} 表示源字符串 S 和目标字符串 T 之间的编辑距离，记为 Ld。它表示源字符串 S 到目标字符串 T 所需的最少的编辑操作次数[32]。

则基于编辑距离算法计算两个字符串的相似度的公式为[33]：

$$sim=1-\left\{LD/\max\left(m,n\right)\right\} \tag{2.4}$$

其中，Ld 表示两个字符串之间的编辑距离；m 和 n 分别表示 2 个字符串的长度；sim 值越大，则表明 2 个字符串越相似[34]。

（2）余弦相似度算法

余弦相似度算法是用向量空间中两个向量夹角的余弦值作为衡量两个个体间的差异大小。余弦值越接近 1，就表明夹角越接近 0 度，即两个向量越相似[35]。余弦相似度的计算公式如下：

$$\cos\theta=\frac{\sum_{i=1}^{n}\left(x_i\times y_i\right)}{\sqrt{\sum_{i=1}^{n}x_i^{\,2}}\times\sqrt{\sum_{i=1}^{n}y_i^{\,2}}} \tag{2.5}$$

要使用余弦相似度计算 URL 的相似度，首先要对 URL 字符串进行数值特征化，并且还要使其维度相同。判断 URL 相似的流程如下：

① 给定一个目标 URL 和一个 URL 列表。

② 将目标 URL 和 URL 列表中的每一条链接都转化为相同维度的数值化特征向量。

③ 计算目标 URL 和列表 URL 数据集的余弦相似度。

④ 设定一个阈值，大于阈值的则认为相似，否则判定为不相似。

2.5 本章小结

本章主要介绍了与本文研究相关的技术。首先介绍了 HTTP 协议的工作流程及其报文内容；然后介绍了 SQL 注入攻击产生的原因、带来的危害及 SQL 注入攻击的避免方法，并对 XSS 漏洞进行了简单介绍；最后介绍了网络爬虫的爬取策略、爬虫的种类及字符串相似度去重的两种算法。

3 WEB 注入漏洞检测方法设计与改进

本章在研究现有漏洞检测程序的基础上，提出了本文的设计思路及总体框架；在分析爬虫模块中 URL 去重算法局限性的基础上，提出 URL 相似去重算法 HDTSF，并说明了注入检测模块的实现原理。

3.1 WEB 注入漏洞检测方法的改进

Web 站点的漏洞扫描检测原理是利用黑盒测试技术来进行漏洞检测，通过构造特殊的攻击字符串，发送 HTTP 请求数据，然后分析服务器返回的响应数据来判断是否存在 XSS 漏洞和 SQL 漏洞。

3.1.1 研究思路

Web 漏洞扫描程序均是基于爬虫实现的，爬虫的性能直接影响了漏洞扫描程序的工作效率和准确度[36]。通过分析研究现有的扫描软件，发现其爬虫模块存在以下局限性：

（1）没有针对登录限制的站点做模拟登录处理。现有的网站大部分都需要经过登录以获取更多数据信息，即使实现了模拟登录，但无法应对复杂且样式多变的验证码，验证码的解析比较耗时。

（2）没有考虑相似 URL 的去重问题。在站点的同一子目录下，存在相似 URL，它也属于重复数据的一种，重复的发送模拟请求会带来时间的消耗和内存空间的浪费。

针对以上缺陷，本文设计了基于改进爬虫的 Web 注入漏洞扫描检测程序，与现有的爬虫相比增加了以下功能：

（1）爬虫去重算法的改进。在对网站进行扫描爬取的过程中，会不可避免的遇到重复或者相似的 URL 链接，如果不对 URL 进行去重处理，将会增加爬虫的工作量，做些重复的扫描工作，降低了爬虫的工作效率。因此，通过对现有的相似性去重算法的研究与分析，本文提出一种基于树结构特征的 HASH 去重算法 HDTSF，使用算法对相似的 URL 链接进行去重，能有效提高爬虫的工作效率。

（2）采用 Cookie 的方式绕过登录限制。现在大部分 Web 应用程序都要求用户登录后才能进行资源的访问，且验证码方式多样化，若直接采用模拟登录的方式，则还需要对验证码进行识别，这涉及到图像识别与处理技术，增加了工作的难度并降低了爬虫工作效率。因此本文所设计的漏洞检测程序采用 Cookie 方式解决登录限制问题，并设置一些必要的请求头信息(如 User-Agent)使爬虫程序更加健壮。

3.1.2 总体设计

本文设计的注入漏洞扫描检测程序主要分为四个模块：URL 爬取模块、XSS 检测模块、SQL 检测模块及检测结果模块，总体设计功能模块结构如图 3.1 所示。

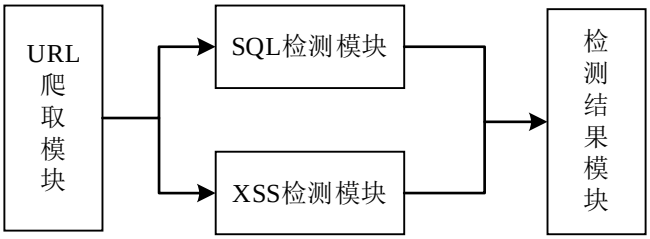

图 3.1 Web 注入漏洞检测方法的总体设计框架图

（1）URL 爬取模块。URL 爬取模块是整个漏洞检测程序的基础，主要为后续的漏洞检测模块提供 URL。通过配置相关参数(如线程数、Cookie 等)，使爬虫的效率得到有效提升。针对提取的链接中存在重复及相似 URL 的情况，采用本文提出的 URL 相似去重算法来对 URL 进行过滤，避免爬虫程序做重复性工作。

（2）SQL 检测模块。在获取 URL 链接后，读取 payload(数据有效载荷)，构建不同的 HTTP 请求，通过对服务器返回的响应信息进行比较来判断是否存在可疑 SQL 漏洞。

（3）XSS 检测模块。在获取 URL 链接后，读取 payload(数据有效载荷)，构建 HTTP 请求，通过判断服务器返回的响应信息中是否存在可疑字符串来判断是否存在可疑 XSS 漏洞。

（4）检测结果模块。检测结果是漏洞扫描检测完成后，将检测结果以文本或其他类型的文件展示，检测结果报告包含目标站点信息、检测时间、漏洞统计等数据。

3.2 URL 爬取模块的设计与改进

设计爬虫程序时本文使用到了 Python 的 Lxml 和 Urllib2 函数库，Lxml 是一种 Python 网页解析器，它支持 HTML 文档和 XML 文档的解析，支持 XPath 解析方式，解析效率非常高，以文档 DOM 树为标准进行结构化解析。Urllib2 是包含一系列和 URL 请求相关的函数库。爬虫程序提取 URL 链接的流程图如图 3.2 所示。

具体操作步骤如下：

① 输入目标种子 URL。

② 构建 HTTP 请求，若站点需要用户登录则先人工登录网站，然后提取其 Cookie 信息，通过设置相关反爬虫参数以绕过反爬虫机制，爬虫开始网页下载并解析提取出 URL 链接，并排除掉非同源 URL。

③ URL 相似度去重，对于重复及相似的 URL 进行过滤，不相似的 URL 则放入待爬取 URL 容器。

④ 判断待爬取 URL 容器是否为空，若为空，则执行步骤⑥，否则执行步骤⑤。

⑤ 随机取出一个 URL，放入已爬取 URL 容器，并执行步骤②。

⑥ 结束，退出爬虫程序。

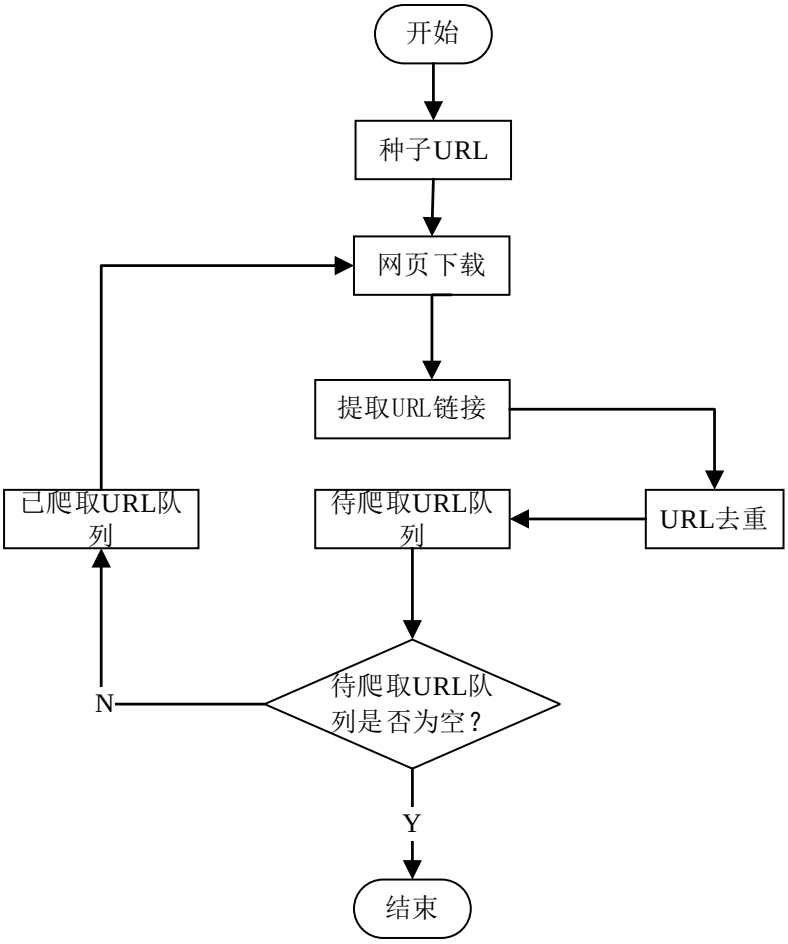

图 3.2 爬虫提取 URL 流程图

URL 爬取模块的关键功能描述如下：

（1）绕过反爬虫

构建一个用户代理(User_Agent)数据集列表，使用 random 模块随机产生一个 User_Agent 数据，然后使用 urllib2 模块的 Request 函数接收 URL 构建 HTTP 请求，并设置随机产生的 User_Agent 请求头参数，来模拟浏览器对网站的访问，避免被当作机器人而禁止访问。User_Agent 用户代理函数部分代码如图 3.3 所示。除了设置 User-Agent，也可以通过设置多个 IP 代理来防止 IP 地址被封，同时设置爬虫请求数据的频率，避免单位时间内请求过多导致爬虫被封。

（2）网页解析与链接提取

使用 Lxml 模块的 html.document_fromstring 函数对爬虫爬取的网页进行结构化解析，

由于存在 URL 链接是相对的，因此可以调用 make_links_absolute 函数对 URL 补充完善得到绝对 URL 地址；iterlinks 函数用于提取出所有的 URL 链接，通过 urlparse 函数解析出 URL 的域名，然后和种子 URL 对应的域名进行比较，若相等，则添加到 URL 队列，否则过滤掉该 URL 链接，这样就排除了非同源 URL。

```
def get_random_agent():
# 可以是User-Agent列表，也可以是代理列表
    ua_list = ["Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/22.0.1207.1 Safari/537.1",
        "Mozilla/5.0 (X11; CrOS i686 2268.111.0) AppleWebKit/536.11 (KHTML, like Gecko) Chrome/20.0.1132.57 Safari/536.11",
        "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.6 (KHTML, like Gecko) Chrome/20.0.1092.0 Safari/536.6",
        "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.6 (KHTML, like Gecko) Chrome/20.0.1090.0 Safari/536.6",
        "Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/19.77.34.5 Safari/537.1",
        "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.9 Safari/536.5",
        "Mozilla/5.0 (Windows NT 6.0) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.36 Safari/536.5",
        "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3",
        "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3",
        "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_0) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536
        "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1062.0 Safari/536.3",
        "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1062.0 Safari/536.3",
        "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3",
        "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3",
        "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3",
        "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.0 Safari/536.3",
        "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/535.24 (KHTML, like Gecko) Chrome/19.0.1055.1 Safari/535.24",
        "Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/535.24 (KHTML, like Gecko) Chrome/19.0.1055.1 Safari/535.24"
```

图 3.3 反爬虫代理函数

（3）需要用户登录才可以进行的操作

由于一些 Web 站点需要用户登录才可以获取到更多信息，且设置有验证码，验证码形式复杂且多样化，如果采用模拟登录的方式来解决登录限制，则需要考虑复杂多变的验证码的解析问题，这样就降低了爬虫的工作效率，因此本文采用 Cookie 的方式解决登录限制问题，可以在构建爬虫请求的时候加入用户的 Cookie 信息，因为 Cookie 信息是用户的一种身份认证标准，通过用户的 Cookie 信息可以模拟用户身份登录 Web 应用程序。

（4）URL 相似度去重

URL 相似度去重模块采用本文提出的 URL 相似度去重算法 HDTSF 进行处理，算法的输入为待处理的 URL，输出为判断的结果。算法描述参考 3.2.2 章节。

3.2.1 现有的 URL 去重算法的局限性

在 2.4.2 中提到的两种字符串相似度去重算法，即编辑距离算法和余弦相似度算法[，这两种算法均存在一定的局限性。

对于编辑距离算法[37]，其设计思想是计算一个字符串变换为另一个字符串所需要的最小编辑操作次数，对于 URL 相似性这一细分领域，如果采用编辑距离算法进行判断，则容易出现编辑距离一样，但实际上 URL 字符串并不相似的情况，如图 3.4 所示。可以看到这三个 URL 的编辑距离是一样的，但是对于 URL 来说，前两个是相似的，第三个则不是，这就造成了误判。

```
https://www.xxx.com/123.html
https://www.xxx.com/456.html
https://www.xxx.com/abc.html
```

图 3.4 相似 URL 数据

对于余弦相似度算法，为了使用余弦相似度算法来计算 URL 的相似度问题，首先需要将 URL 字符串转换为数值化特征，计算数值化特征的方法是将 URL 协议和域名后面的字符串进行 ASCII 码转化，为了计算方便，需要将每一个向量转化为相同维度的特征向量，然后进行余弦相似度计算。这种算法原理简单，容易理解，但数值化特征向量的提取方式简单，特征向量的维度对计算结果影响很大；由于该余弦相似度算法是将 URL 字符对应的 ASCII 值作为向量元素，因此改变少量字母会对相似度产生较大影响，同时特征维度越大，其计算量越大，内存消耗越大。针对上面的分析及存在的不足，本文提出基于树结构的 HASH 去重算法(HDTSF，Hash Deduplication algorithm based on Tree Structure Features)。

3.2.2 HDTSF 去重算法

由上面的分析可知，将编辑距离算法和余弦相似度算法应用于 URL 相似度判定是不可取的，因为 URL 和普通的字符串是有一定的区别的，其中 URL 是有结构的，而普通字符串没有，因此在计算相似度时不可忽略其结构特征[35]。在数据检索中，HASH 算法是性能较好的[38]，而树结构[39]可以很好的体现出 URL 的结构特征，因此本文提出基于树结构特征的 HASH 去重算法(HDTSF)。将 URL 数据集按照树结构划分为不同的子集，由于相似 URL 主要出现在同一子结构目录下，在不同子结构目录下出现的可能性很小，因此只需要计算同一子结构下的 URL 相似度，大大缩小了比较范围。URL 树结构如图 3.5 所示。

由于一个 Web 站点存在无数个 URL 链接，且包含不同的功能模块(如某防火墙产品的系统监控、数据中心、安全防护、风险分析功能模块)，不同的功能模块对应着不同的子目录，属于某一模块的 URL 链接则位于相应的子目录下，某一 Web 站点的 URL 目录结构如图 3.6 所示。这里以深度为 2 作为分类依据是将 URL 链接更好的细分到不同类别中。为了维护和管理的方便，目录的层次建议不超过 3 层[40]。假设以深度为 3 作为分类依据，则目录总深度小于或等于 3 的站点内的所有链接则被划分为一组，这样就意味着要对所有的 URL 链接进行相似度计算比较，增加了时间成本，因此考虑到计算的时间效率，本文以目录深度为 2 作为 URL 链接分组依据。

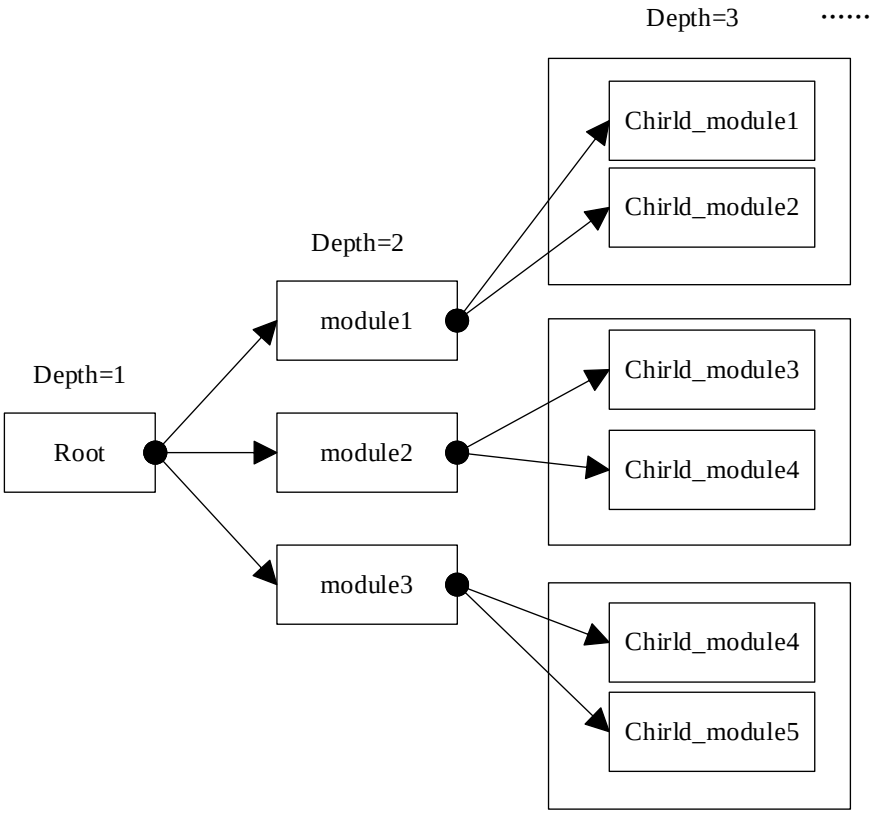

图 3.5 URL 树结构

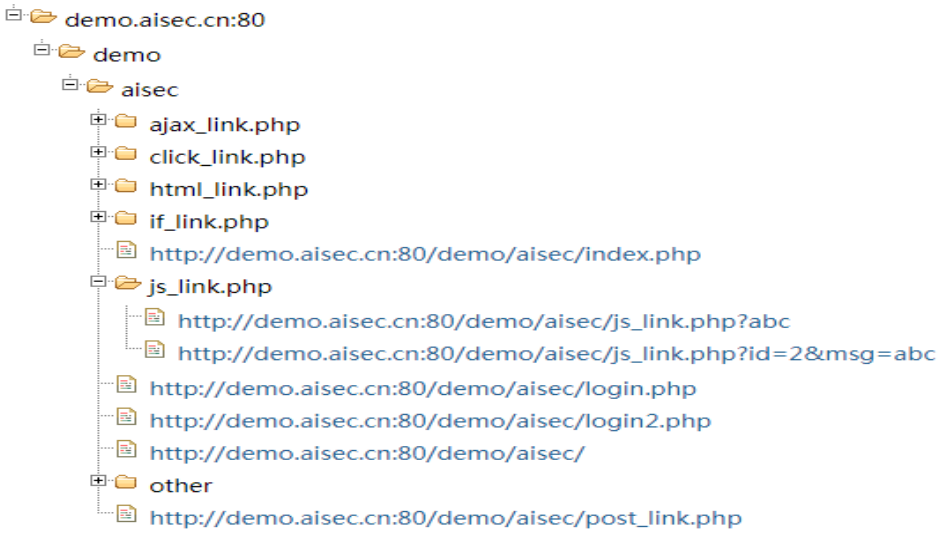

图 3.6 URL 目录结构

基于树结构的 URL 相似性判断工作流程图如图 3.7 所示。首先将爬取到的 URL 数据集进行分类，以深度为 2 作为分类依据进行聚类，对其下属的子 URL 数据集进行相似度计算。

URL 相似度算法描述如下：

① 新建一个空字典 dict，以深度 depth 为 2 进行 URL 数据集聚类，得到分类子路

径 url_prex；

② 计算 URL 子路径下的相对路径 url_path；

③ 计算 URL 子路径下的请求参数名称序列 url_query，由于可能存在参数顺序不一样但名称一样的情况，所以需要对参数进行排序组合处理；

④ 计算 url_path 和 url_query 的 hash 值并求和得到 hash_url；

⑤ 以子路径 url_prex 为键，hash_url 为键对应的值，判断该键值对是否存在字典 dict 中，若不存在，将该键值对添加到字典中，返回 True，否则返回 False。

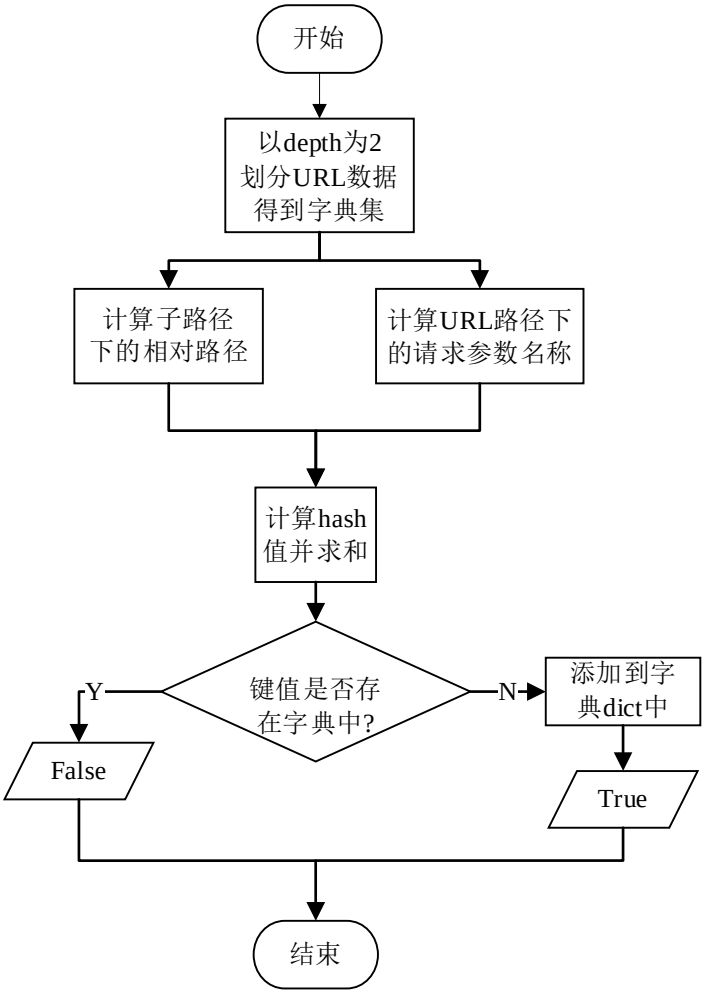

图 3.7 URL 相似性判断流程图

对于 URL=https://root/r1/r2/…?k1=x&&k2=y，定义如下（由于公式中不同值对应的条件不便用数学符号描述，故在下一段落用文字描述）：

$$url=\begin{cases} url\_prex \\ url\_path \\ url\_query \end{cases} \tag{3.1}$$

其中：

$$url\_prex=\begin{cases}root//r1\\rt\end{cases} \tag{3.2}$$

$$url\_path=\begin{cases}r2../..\\""\end{cases} \tag{3.3}$$

$$url\_query=\begin{cases}k1|k2...\\""\end{cases} \tag{3.4}$$

式 3.1 中，将 URL 分为三部分，分别为 url_prex、url_path、url_query，式 3.2、3.3、3.4 为这三部分的计算过程。其中 url_prex 表示 URL 链接的前缀，即表示 URL 所在的分类子路径，若 URL 的深度为 1，则设 url_prex 为固定值 rt，否则通过正则表达式计算出其子路径；url_path 表示子路径下的相对路径，若该 URL 存在三级目录，则其值为除去分类子路径后的路径，否则将其值设为空字符串；url_query 表示 URL 的请求参数序列，若该 URL 存在参数，则其值为请求参数经过排序及组合后的字符串序列，若该 URL 不存在请求参数，则将其值设为空字符串。以 url_prex 为键，url_path 和 url_query 的 hash 值的和为键对应的值，判断该键值对是否存在于字典中，若存在，则表明该键值对对应的 URL 是相似的，将其过滤掉，否则表明该 URL 不相似。

3.3 注入检测模块的设计

注入检测模块的实现方式就是通过将 URL 进行重构，在 URL 中加入带有可疑参数的攻击字符串，发起请求进行模拟攻击测试来判断是否存在漏洞。

3.3.1 SQL 漏洞检测模块

对于 SQL 注入漏洞的攻击检测，分别构建正常和带可疑参数的 HTTP 请求，通过返回的响应信息来判断是否存在漏洞，并将存在可疑漏洞的 URL 输出到漏洞文档。SQL 注入漏洞检测流程如图 3.8 所示，SQL 漏洞检测过程描述如下：

① 从 URL 队列中读取 URL；

② 构建带正常参数 HTTP 请求，并添加相关请求头，如 User-Agent 和 Cookie 等；

③ 构建带可疑参数的 HTTP 请求，并添加相关请求头，如 User-Agent 和 Cookie 等；

④ 对正常请求和带可疑参数请求的服务器返回的响应信息进行比较；

⑤ 若比对结果符合预期情况，则表明该 URL 链接存在 SQL 注入，并保存 SQL 注入漏洞信息，否则表明该 URL 链接不存在 SQL 注入；

⑥ 判断 URL 队列中是否还存在待检测的 URL，若存在待检测的 URL，则执行步骤①，否则结束检测。

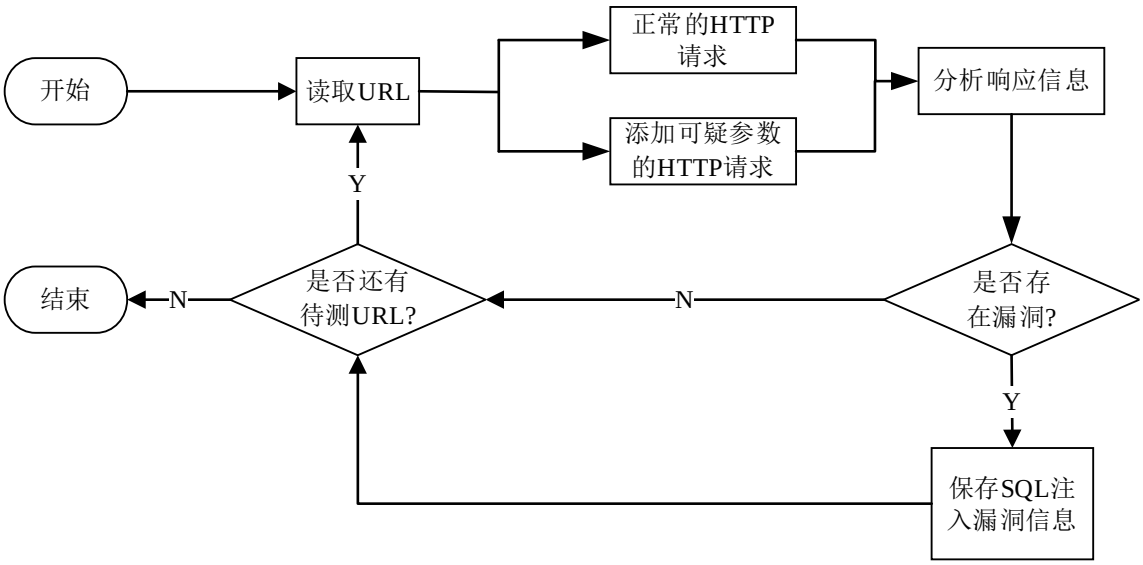

图 3.8 SQL 注入漏洞检测流程图

本文实现的漏洞扫描检测程序目前实现了 SQL 注入漏洞的数值型注入漏洞检测、字符型注入漏洞检测和数据库错误注入漏洞检测。举例说明如下：

（1）数值型注入漏洞检测

数值型注入漏洞的检测原理如下面例子所示，分别构建三个 HTTP 请求：

① http://demo.aisec.cn/demo/aisec/html_link.php?id=2；

② http://demo.aisec.cn/demo/aisec/html_link.php?id=2 and 1=2；

③ http://demo.aisec.cn/demo/aisec/html_link.php?id=2 and 1=1；

计算服务器返回结果的加密摘要。若请求①得到的加密摘要和请求③得到的加密摘要是一致的，而请求②和请求①得到的响应结果加密摘要不一致，则说明该 URL 链接存在数值型 SQL 注入漏洞。

（2）字符型注入漏洞检测

字符型注入漏洞的检测原理如下面例子所示，分别构建三个 HTTP 请求：

① http://demo.aisec.cn/demo/aisec/html_link.php?id=2；

② http://demo.aisec.cn/demo/aisec/html_link.php?id=2' and 0;--；

③ http://demo.aisec.cn/demo/aisec/html_link.php?id=2' and 1;--；

计算服务器返回结果的加密摘要。若请求①得到的加密摘要和请求③得到的加密摘要是一致的，而请求②和请求①得到的响应结果加密摘要不一致，则说明该 URL 链接存在字符型 SQL 注入漏洞。

（3）数据库错误注入漏洞检测

对 URL 的请求参数添加单引号，通过查找服务器返回的响应页面是否包含数据库关键字信息来判断 Web 站点采用的是什么类型的数据库，然后可以针对特定数据库来进行入侵。常见的部分数据库错误信息列举如表 3.1 所示。

表 3.1 数据库对应的错误信息表

数据库	错误信息
	supplied argument is not a valid MySQL
MySQL	Column count doesn\'t match value count at row
	You have an error in your SQL syntax
	MySQL server version for the right syntax to use
	80040e14
MSSQL	Unclosed quotation mark after the character string
	System.Data.SqlClient.SqlException
ORACLE	(PLS|ORA)-[0-9][0-9][0-9][0-9]

3.3.2 XSS 漏洞检测模块

XSS 漏洞检测模块接收传入的 URL 后，随机选取一个 XSS 攻击字符串重新构造新的 URL，并向 Web 服务器发送 HTTP 请求，通过分析返回的响应信息来判断是否存在 XSS 漏洞，若响应信息中能查出 XSS 攻击字符串，则说明该 URL 存在 XSS 漏洞，否则该 URL 不存在漏洞。XSS 漏洞检测流程如图 3.9 所示。

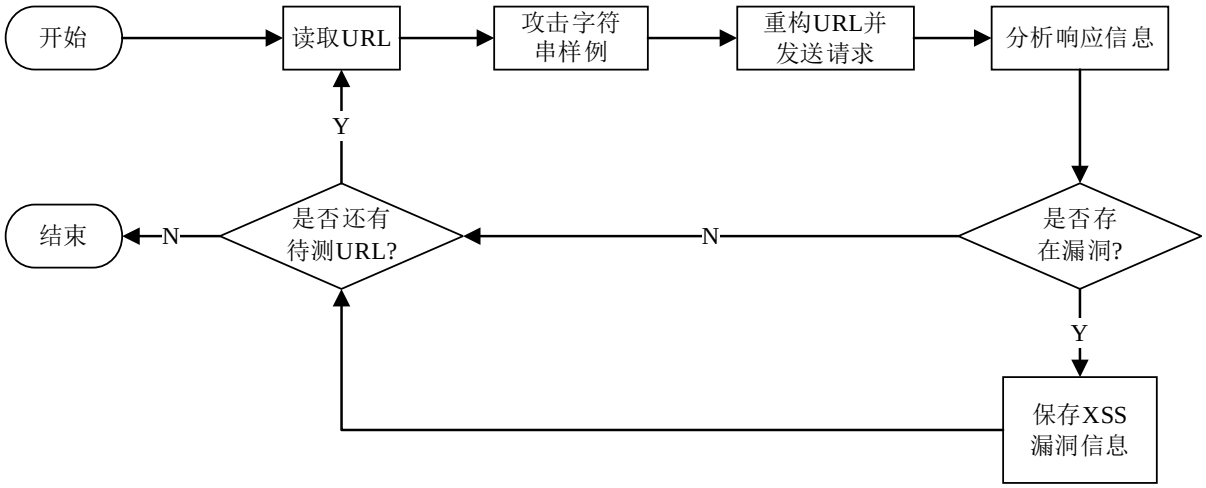

图 3.9 XSS 漏洞检测流程图

XSS 漏洞检测过程描述如下：

① 从 URL 队列中读取 URL；

② 从 XSS 攻击字符串样例集合中随机读取一条字符串数据，重新构建带可疑参数的 HTTP 请求，并添加相关请求头，如 User-Agent 和 Cookie 等，向服务器发送重构的 HTTP 请求；

③ 对服务器返回的响应信息进行分析，然后判断返回的网页代码中是否包含 XSS 攻击字符串，若存在攻击字符串，则表明该链接存在 XSS 漏洞，保存 XSS 漏洞信息，否则该链接不存在 XSS 漏洞；

④ 判断 URL 队列中是否存在待检测的 URL，若存在待检测的 URL，则执行步骤①，否则结束检测。

3.4 本章小结

本章首先提出了 Web 漏洞扫描检测方法的设计思路和功能模块总体结构，然后对现有爬虫的局限性进行了分析，提出爬虫的改进方案，通过分析编辑距离算法和余弦相似度算法的不足及缺点，根据 URL 的树结构特征，提出了基于 URL 树结构的相似性去重算法 HDTSF；最后对 SQL 注入漏洞检测模块和 XSS 漏洞检测模块的工作流程和原理进行了阐述。

4 WEB 注入漏洞检测方法的实现

本章是在第三章 Web 注入漏洞检测方法的设计与改进的基础上，完成漏洞扫描检测方法各个模块的实现。

4.1 漏洞检测流程

本文实现的 Web 漏洞扫描检测程序在开始工作时，需要先配置相关的参数(如域名、线程、Cookie 等)，以便初始化程序。采用多线程的方式对网页进行爬取，提取出 URL 链接然后对 URL 链接进行相似去重操作，调用漏洞检测模块开始漏洞检测，并保存含有漏洞的 URL 信息。漏洞检测程序的工作流程如图 4.1 所示。

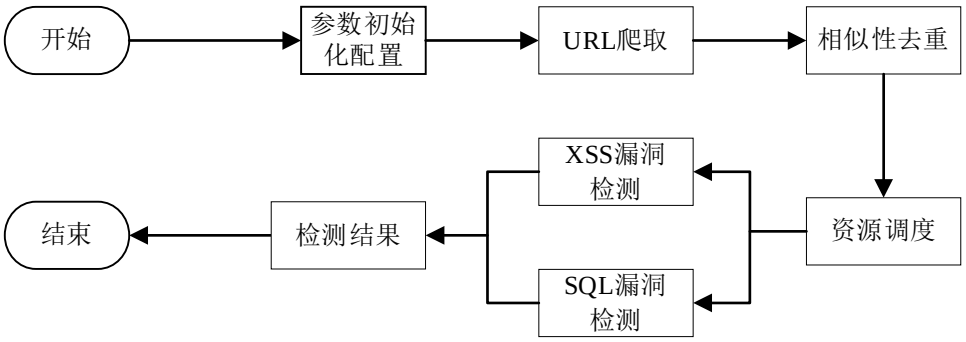

图 4.1 漏洞检测程序的工作流程

如图 4.1 所示，程序开始时，配置相关的参数信息，完成程序的初始化配置，然后程序开始数据的爬取，并对 URL 数据进行相似性去重处理，接着开始资源调度，选择进行 SQL 漏洞检测还是 XSS 漏洞检测，也可以同时检测 SQL 漏洞和 XSS 漏洞，检测完毕后，生成漏洞检测结果报告。

4.2 参数初始化

本文实现的漏洞检测程序采用命令行模式与用户进行交互，在命令行模式下输入 Python 命令：Python run.py -h 进入扫描软件初始界面，可以查看需要配置的相关参数，配置参数初始化函数代码如图 4.2 所示。

```
def main():
    parser = optparse.OptionParser()
    parser.add_option('-d','--domain',dest = 'domain',help = "Start the domain name")
    parser.add_option('-t','--thread',dest = 'thread_num',default = 10,help = "Numbers of threads")
    parser.add_option('--depth',dest = 'depth',default = 3,help = "Crawling dept")
    parser.add_option('--module',dest = 'module',default = 'all',help = "vulnerability module(sql,xss)")
    parser.add_option('--cookie', dest='cookie', default=None, help="Cookie validation")
    parser.add_option('--log',dest = 'logfile_name',default='log.txt',help="save log file")
    (options, args) = parser.parse_args()
    target = options.domain
```

图 4.2 初始化参数配置函数

参数配置说明如下：

-d：配置漏洞检测工具要扫描的站点信息，即网址信息。

-t：配置线程数，默认值为 10。

--depth：配置爬虫爬取的深度，避免爬虫无限爬取下去。

--module：配置要检测的模块，默认值为 all，表示 SQL 漏洞和 XSS 漏洞均检测。

--cookie：配置 HTTP 请求 Cookie 信息，默认值为空，对于需要登录获取数据的 Web 应用则需要登录后获取 Cookie 信息。

--log：配置保存的日志文件。

当开始扫描某一目标站点时，可配置命令参数如下：

python run.py -d http://testphp.vulnweb.com/ --module sql ，即可开始对目标站点进行漏洞检测。

对于参数值的获取和解析本文主要采用 Python 的 optparse 模块库来实现，定义一个对象 parser = optparse.OptionParser()，调用 add_option 方法来定义所需要的参数信息，调用 parser.parse_args 方法来对输入的参数值进行解析。

4.3 URL 爬取模块的实现

由于一些网页是 JavaScript 代码动态渲染生成的，采用 Python 的 requests，urllib 模块并不能正常获取网页的内容，因此本文采用 Selenium 来解决该问题，它是一种自动化测试工具，支持多种类型的浏览器，在爬虫中主要用来解决 JavaScript 渲染问题。PhantomJS 是基于 webkit 的无界面浏览器，且内存占用较小，和 Selenium 结合使用可以在不打开浏览器的情况下解决 JavaScript 代码渲染问题。PhantomJS 和 Selenium 请求数据的核心代码如图 4.3 所示。

```
#随机用户代理
user_agent=get_random_agent()
#设置请求头
dcap=dict(DesiredCapabilities.PHANTOMJS)
dcap["phantomjs.page.settings.userAgent"]=(user_agent)
#请求头生效
driver=webdriver.PhantomJS(desired_capabilities=dcap)
driver.set_page_load_timeout(30)
driver.get(self.deep_url[1])
html=driver.page_source.encode()
```

图 4.3 配置 PhantomJS 和 Selenium

图 4.3 中，dcap 是配置对象 DesiredCapabilities 调用 PhantomJS 生成的一个字典，用于设置相关请求头，比如用户代理、Cookie、是否加载图片等。driver 对象表示打开带配置请求头信息的 PhantomJS 浏览器。driver 对象的 page_source.encode 方法用于获取网页

数据。

虽然爬取到了 URL 链接数据，但是数据中存在相似的 URL 链接，如果不对这一部分数据进行过滤，则在进行后面的漏洞检测时，相当于进行了重复的漏洞验证检测，增加了漏洞检测的时间消耗，降低了检测效率。3.2 章节通过对现有的两种算法(余弦相似度和编辑距离算法)进行了分析，说明其不足后，设计了 HDTSF 算法，并对算法的原理和计算流程进行了阐述。如图 4.4，图 4.5 所示为 HDTSF 算法的核心代码。

```
def get_url_prex(url):
    pattern = re.compile(r'http[s]?://(.*?)/\w+/?')
    path = pattern.match(url)
    if path:
        url_prex=path.group()
        url_path=url[len(url_prex):]
    else:
        url_prex='root'
        url_path=url
    return url_prex
```

图 4.4 获取 URL 前缀

```
def url_simility_filter(url):
    hash_size = 199999
    #对url进行分组聚类
    url_prex=get_url_prex(url)
    #对url各部分按照一定格式进行拆分，结果形如：ParseResult(scheme='https', netloc='i.cnblogs.com',
    # path='/EditPosts.aspx', params='', query='opt=1', fragment='')
    tmp=urlparse.urlparse(url)
    tmp_query_key= '|'.join(sorted([ i.split('=')[0] for i in tmp.query.split('&') ]))
    tmp_path_key='/'.join(tmp.path.split('/')[2:])
    hash_query_key=hash(hashlib.new("md5", tmp_query_key).hexdigest()) % hash_size
    hash_path_key=hash(hashlib.new("md5", tmp_path_key).hexdigest()) % hash_size
    tmp_hash_url=hash(hashlib.new("md5", str(hash_query_key+hash_path_key)).hexdigest()) % hash_size

    if url_prex not in UNSIMILITY_URL:
        UNSIMILITY_URL[url_prex]=[]
        UNSIMILITY_URL[url_prex].append(tmp_hash_url)
        return url,True
    else:
        if tmp_hash_url not in UNSIMILITY_URL[url_prex]:
            UNSIMILITY_URL[url_prex].append(tmp_hash_url)
            return url,True
        else:
            return url,False
```

图 4.5 URL 相似过滤

图 4.4 中，获取 URL 前缀作为分组依据，前缀相同的则划分为同一组，URL 前缀的获取采用正则表达式实现。图 4.5 中的函数表示具体的过滤算法，分别获取 URL 路径和请求参数数据，因为存在参数名称一样，位置不一样的情况，所有需要对参数进行排序

组合。以 URL 前缀作为字典的键，URL 子路径和请求参数 Hash 后的和作为字典的值。当键和值均不存在字典中时，添加键值对数据到字典中，表明该键值对对应的 URL 数据不是重复的，添加键值对数据到字典中；当键存在字典中，而值不存在字典中，则也表明该键值对对应的 URL 数据不是重复的，添加键值对数据到字典中；反之则说明对应的 URL 数据是重复的。通过返回的结果(True 和 False)来决定是否调用后面的漏洞检测模块。

4.4 注入检测模块的实现

前面讲述了 URL 爬取模块的实现，爬取网页数据后，提取出 URL 数据，采用本文提出的 HDTSF 算法对 URL 进行过滤，用于漏洞检测。构建带有可疑参数的 URL 数据链接，发送 HTTP 请求，通过分析服务器响应信息来判断是否存在可疑漏洞。本节实现了 SQL 漏洞和 XSS 漏洞的检测。

4.4.1 SQL 漏洞检测模块的实现

本文实现了三种类型的 SQL 漏洞检测，分别为：数值型注入漏洞检测、字符型注入漏洞检测和数据库错误注入漏洞检测。

（1）数值型注入漏洞检测

数字型注入的检测是通过在参数后面添加“and 1=1”和“and 1=2”来分别发送 HTTP 请求，分析服务器返回的响应信息判断是否存在数字型注入漏洞。数值型注入漏洞检测的核心代码如图 4.6 所示。

```
def Integer_sqlinj_scan(self):
    try:
        res_md5_1 = md5_encrypt(requests.get(url=self.url,headers=HEADER).text)
        res_md5_2 = md5_encrypt(requests.get(url=self.url+urlencode('and 1=2'),headers=HEADER).text)
        res_md5_3 = md5_encrypt(requests.get(url=self.url+urlencode('and 1=1'),headers=HEADER).text)
    except Exception,e:
        print e
        res_md5_1 = res_md5_2 = res_md5_3 = 0
        pass

    if ( res_md5_1 == res_md5_3 ) and res_md5_1 != res_md5_2:
        return 1
    return 0
```

图 4.6 数值型注入漏洞检测

如图 4.6 所示，重构 URL 链接，发送 HTTP 请求数据，分别计算响应网页的信息摘要，通过对指纹摘要进行比较来判断是否存在数字型注入漏洞。

（2）字符型注入漏洞检测

字符型注入的检测也是构建不同的 HTTP 请求，通过分析服务器返回的响应信息来

判断是否存在字符型注入漏洞。和数值型注入检测不同的是，字符型注入采用的异常字符串是是单引号、双引号、注释符号、SQL 关键字等字符的随机组合，这样做的目的是尽可能产生较多的异常字符串特征，保证了漏洞检测的准确性，降低漏报率。字符型注入漏洞检测的核心代码如图 4.7 所示。

```
def Str_sqlinj_scan(self):
    quotes = ['\'','"','']
    payload_0 = [" and 0;-- ","/**/and/**/0;#","\tand\t0;#","\nand/**/0;#"]
    payload_1 = [" and 1;-- ","/**/and/**/1;#","\tand\t1;#","\nand/**/1;#"]

    for i in quotes:
        for j in range(len(payload_0)):
            p0 = i + payload_0[j]
            p1 = i + payload_1[j]
            try:
                res_md5_1 = md5_encrypt(requests.get(url=self.url,headers=HEADER).text)
                res_md5_2 = md5_encrypt(requests.get(url=self.url+urlencode(p0),headers=HEADER).text)
                res_md5_3 = md5_encrypt(requests.get(url=self.url+urlencode(p1),headers=HEADER).text)
            except Exception,e:
                print e
                res_md5_1 = res_md5_2 = res_md5_3 = 0
                pass
            if ( res_md5_1 == res_md5_3 ) and res_md5_1 != res_md5_2:
                return 1,self.url+urlencode(p0)
    return 0
```

图 4.7 字符型注入漏洞检测

（3）数据库错误注入漏洞检测

错误注入的检测是通过页面返回的错误信息来判断站点使用的是什么类型的数据库，然后针对特定的数据库进行攻击破解。错误注入漏洞检测的函数代码如图 4.8 所示。在得到 HTTP 响应网页后，通过正则表达式来搜索响应网页中是否存在数据库特征字符串，进而判断 Web 站点采用的是什么类型的数据库。数据库错误信息特征是将数据库错误信息和对应的数据库类型以元组的形式表示，获取数据库错误信息特征函数如图4.9所示。

```
def Sql_error_scan(self):
    '''
    This method searches for SQL errors in html's.

    @parameter response: The HTTP response object
    @return: A list of errors found on the page
    '''
    r1 = requests.get(url=self.url,headers=HEADER)
    r2 = requests.get(url=self.url+urlencode('\''),headers=HEADER)

    res = []
    for sql_regex, dbms_type in self.Get_sql_errors():
        match1 = re.search(sql_regex,r1.text)
        match2 = re.search(sql_regex,r2.text)
        if match2 and not match1 :
            msg = 'A SQL error was found in the response supplied by the web application,'
            msg += match2.group(0) + '". The error was found '
            #res.append( (sql_regex, match.group(0), dbms_type) )
            return 1,self.url+urlencode('\'')
    return 0
```

图 4.8 数据库错误注入漏洞检测

```
def Get_sql_errors(self):

    if len(self.sql_errors) != 0:
        return self.sql_errors

    else:
        errors = []

        # ASP / MSSQL
        errors.append( ('System\.Data\.OleDb\.OleDbException', dbms.MSSQL ) )
        errors.append( ('\\[SQL Server\\]', dbms.MSSQL ) )
        errors.append( ('\\[Microsoft\\]\\[ODBC SQL Server Driver\\]', dbms.MSSQL ) )
        errors.append( ('\\[SQLServer JDBC Driver\\]', dbms.MSSQL ) )
        errors.append( ('\\[SqlException', dbms.MSSQL ) )
        errors.append( ('System.Data.SqlClient.SqlException', dbms.MSSQL ) )
        errors.append( ('Unclosed quotation mark after the character string', dbms.MSSQL ) )
        errors.append( ("'80040e14'", dbms.MSSQL ) )
        errors.append( ('mssql_query\\(\\)', dbms.MSSQL ) )
        errors.append( ('odbc_exec\\(\\)', dbms.MSSQL ) )
        errors.append( ('Microsoft OLE DB Provider for ODBC Drivers', dbms.MSSQL ) )
```

图 4.9 获取数据库错误信息队列

4.4.2 XSS 漏洞检测模块的实现

XSS 漏洞的检测是通过构建带有可疑字符串的 HTTP 请求，在得到服务器的响应信息后，判断响应信息中是否含有可疑字符串，若搜索发现存在可疑的攻击代码字符串，则表明该 URL 链接存在 XSS 漏洞。XSS 漏洞检测的函数代码如图 4.10 所示，可疑字符数据集如图 4.11 所示。

```
for test in XSS_PAYLOAD:
    r = requests.get(url=self.url+urlencode(test),headers=HEADER)

    if test in r.text:
        return 1,test
return 0
```

图 4.10 XSS 漏洞检测

```
'<script>alert(1);</script>',
'<script>prompt(1);</script>',
'<script>confirm(1);</script>',
'<scr<script>ipt>alert(1)</scr<script>ipt>',
'<object data="data:text/html;base64,PHNjcmlwdD5hbGVydCgxKTs8L3Njcmlwd4=">',
'<svg/onload=prompt(1);>',
'<marquee/onstart=confirm(1)>/',
'<body onload=prompt(1);>',
'<select autofocus onfocus=alert(1)>',
'<textarea autofocus onfocus=alert(1)>',
'<keygen autofocus onfocus=alert(1)>',
'<video><source onerror="javascript:alert(1)">'
```

图 4.11 XSS 可疑数据集

图 4.10 中，XSS_PAYLOAD 表示 XSS 漏洞特征库，即图 4.11 所示的部分可疑数据集，通过判断请求返回的响应信息中是否含有可疑数据代码来判断是否存在 XSS 漏洞。若可疑数据存在响应的信息中，则表明存在 XSS 漏洞。

4.5 检测结果报告的实现

将检测结果以 HTML 网页或 PDF 文档的形式输出展示，可以使用户直观的看到 Web 应用站点的漏洞信息。检测结果报告是通过模板文件进行数据填充生成的。首先构建 HTML 模板文件，模板文件内包含的关键信息即需要填充的数据有种子 URL 信息、扫描时间、漏洞信息等；HTML 模板文件使用到了 jinja2 模板语言，jinja2 是基于 Python 的一个模板引擎系统。使用 jinja2 提供的特殊语法在需要数据填充的地方编写相应的控制结构代码、变量取值代码等，用于后面的实际真实的数据填充，检测结果报告实现的代码如图 4.12 所示。

```
def render_template(template_filename, context):
    return TEMPLATE_ENVIRONMENT.get_template(template_filename).render(context)
#检测报告生成
def create_index_html(name, rtime):
    domain = name
    fname = name[7:12] + ".html"

    length, vul= read_file("vulfile.txt")
    context = {
        'domain': domain,
        'time': rtime,
        'length': length,
        'vul': vul,

    }
    #
    path_file = os.path.join(PATH, 'Report/') + fname
    with open(path_file, 'w') as f:
        html = render_template('index.html', context)
        f.write(html)
```

图 4.12 检测报告生成代码

图 4.12 所示代码是实现将真实的数据填充到模板文件中，生成检测报告，其中 domain 表示目标站点信息，rtime 表示扫描时间，length 表示漏洞个数，vul 表示漏洞数据信息，生成的检测报告的名字以检测的目标站点信息进行命名。

4.6 本章小结

本章介绍了 Web 注入漏洞检测的工作流程，详细阐述了各个模块的实现过程，包括程序初始化设置、URL 爬取模块、注入检测模块、检测结果报告模块。

5 测试与分析

本章在第三、第四章的基础上实现基于爬虫的 Web 注入漏洞检测程序(Icrawler-scan)的测试与分析；为了验证 Icrawler-scan 的有效性，同时使用 AIScanner 软件和 WebVulScan 软件对相同的目标网站进行检测，并将检测结果和 Icrawler-scan 程序的检测结果进行比对分析，衡量漏洞检测程序 Icrawler-scan 的检测效率和准确率。

5.1 测试环境

系统测试环境配置信息如下：

（1）硬件配置：Intel(R) Core(TM) i5-3230M 主频为 2.60GHz，内存为 8G，硬盘为 1T。

（2）软件配置：64 位的 win8.1 操作系统，使用的编程语言为 Python，开发工具为 PyCharm。

5.2 URL 爬取模块测试及分析

本文实现的爬虫和其他爬虫相比具有以下优势：①针对需要登录的站点，由于登录验证码的复杂性和多样性，验证码的识别会耗费大量时间，会大大降低爬虫效率，本文采用 Cookie 会话方式绕过登录限制，读取 Web 应用登录后的 Cookie 信息作为登录凭证，以爬取到更多的网页数据。②提出基于树结构的 URL 相似性去重算法 HDTSF，在对重复 URL 进行去重的同时，增加相似性 URL 去重，避免不必要的资源浪费，能提高爬虫的工作效率。

本文对提出的 HDTSF 算法和编辑距离算法、余弦相似度算法进行测试对比分析，以证明 HDTSF 算法优于另外两种算法；同时对改进后的爬虫和改进前的爬虫进行测试，验证 HDTSF 算法的可行性。

5.2.1 相似去重算法对比测试

本文对编辑距离算法、余弦相似度算法和本文提出的 HDTSF 算法三种算法进行对比测试，证明本文提出的算法 HDTSF 优于其他两种算法。实验数据来源为网络爬虫爬取到的真实 URL 数据。

（1）HDTSF 算法测试

为了验证 HDTSF 算法确实能够对相似 URL 数据进行去重，本文首先选取少量的 URL 数据进行测试，从爬虫提取的 URL 数据中选取部分数据如图 5.1 所示，使用 HDTSF 算法进行测试后得到的结果如图 5.2 所示。

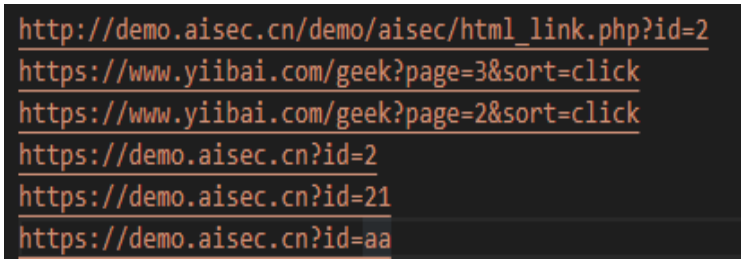

图 5.1 部分 URL 数据

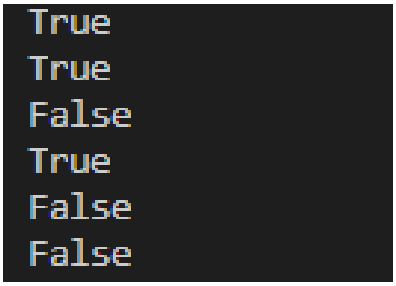

图 5.2 测试结果

由图 5.1 和图 5.2 可知，有三个数据是相似的，第三个 URL 数据和第二个 URL 数据时相似的，第五个、第六个 URL 数据和第四个 URL 数据是相似的，通过观察图 5.1 中的 URL 数据可以发现，测试结果和预期的结果是一致的。

（2）三种算法对比测试

为了对比三种相似去重算法的效果，本文采用混淆矩阵来对测试结果进行分析。混淆矩阵[41]也称为误差矩阵，它是一种评价标准，用 n 行 n 列的矩阵表示，混淆矩阵的每一列表示数据的预测类别，每一列的总数表示数据被预测为该类别的数据的数目[42]；混淆矩阵的每一行表示数据的真实类别，每一行的总数表示属于该类别的数据的数目，每一列中的数据表示真实的数据被预测为该类别的数据数目。混淆矩阵有四个重要概念，分别是 TP、TN、FP、FN；其中 TP 表示真阳性(True Positive)，它的含义是样本的实际类别为正例，其预测结果也是正例；TN 表示真阴性(True Negative)，它的含义是样本的实际类别为为负例，其预测结果也是负例；FP 表示假阳性(False Positive)，它的含义是样本的实际类别为负例，其预测结果为正例；FN 表示假阴性(False Negative)，它的含义是样本的实际类别为正例，其预测结果为负例。则混淆矩阵的图形化描述如图 5.3 所示。

根据混淆矩阵的理论，可以得到数据的评价指标，如数据的分类准确率、错误率等。数据的准确率表示被正确分类的样本数量或者比例，错误率表示被错误分类的样本数量或者比例[43]。那么样本数据的准确率的具体计算方法为：(TP+TN)/(TP+FP+TN+FN)，样本数据的错误率的具体计算方法为：(FP+FN)/(TP+FP+TN+FN)。本文以数据的准确率来衡量相似去重算法的效率。

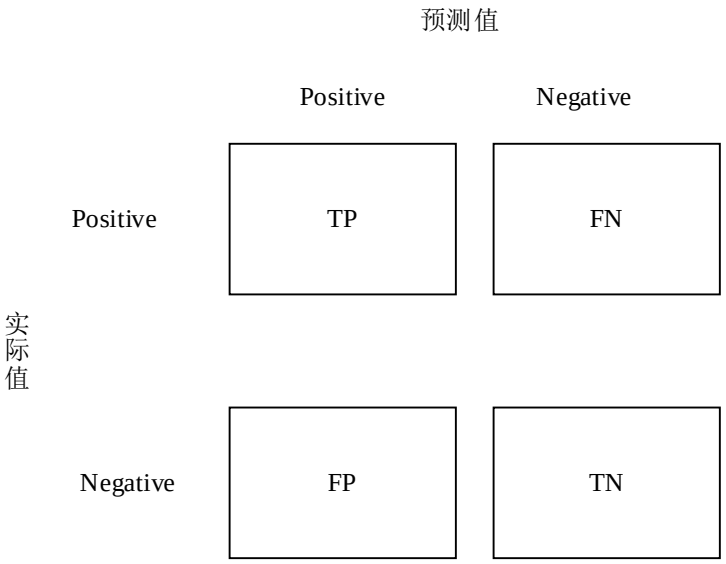

图 5.3 混淆矩阵图形化描述

为了衡量算法的时间效率，随机从爬取的 URL 数据集中选取 1000 条 URL 数据，其中相似 URL 为 200 条，通过实验测试，得到的时间数据如表 5.1 所示。

表 5.1 三种算法准确率对比

算法类别	TP	FP	TN	FN	准确率	时间
编辑距离	450	40	160	350	0.61	0.8s
余弦相似度	500	50	150	300	0.65	1.8s
HDTSF 算法	750	80	120	50	0.87	0.06s

由表 5.1 可知，使用编辑距离算法对 URL 数据进行测试，得到的 TP 数据量为 450，FP 数据量为 40，TN 数据量为 160，FN 数据量为 350，则计算出的测试数据的准确率为 0.61，耗时 0.8s；使用余弦相似度算法对 URL 数据进行测试，得到的 TP 数据量为 500，FP 数据量为 50，TN 数据量为 150,FN 数据量为 300，则计算出的测试数据的准确率为 0.65，耗时 1.8s；本文提出的 HDTSF 算法对 URL 数据进行测试，得到的 TP 数据量为 750，FP 数据量为 80，TN 数据量为 120，FN 数据量为 50，则计算出的测试数据的准确率为 0.87，耗时 0.06s。由得到的数据测试结果可以看出，在对 URL 数据进行相似去重时，本文提出的 HDTSF 算法在准确率上明显优于编辑距离算法和余弦相似度算法，且 HDTSF 算法的时间效率明显优于编辑距离算法和余弦相似度算法，对 URL 相似去重效果良好。

5.2.2 HDTSF 算法可行性分析

为了验证本文设计的爬虫的可行性，使用本文改进的基于 HDTSF 去重算法的爬虫模块和未改进的爬虫模块对站点进行扫描。在相同条件下，通过对比爬取到的 URL 的数

量和花费的时间来验证本文设计的 URL 爬取模块的性能，测试结果如图 5.4 和图 5.5 所示。

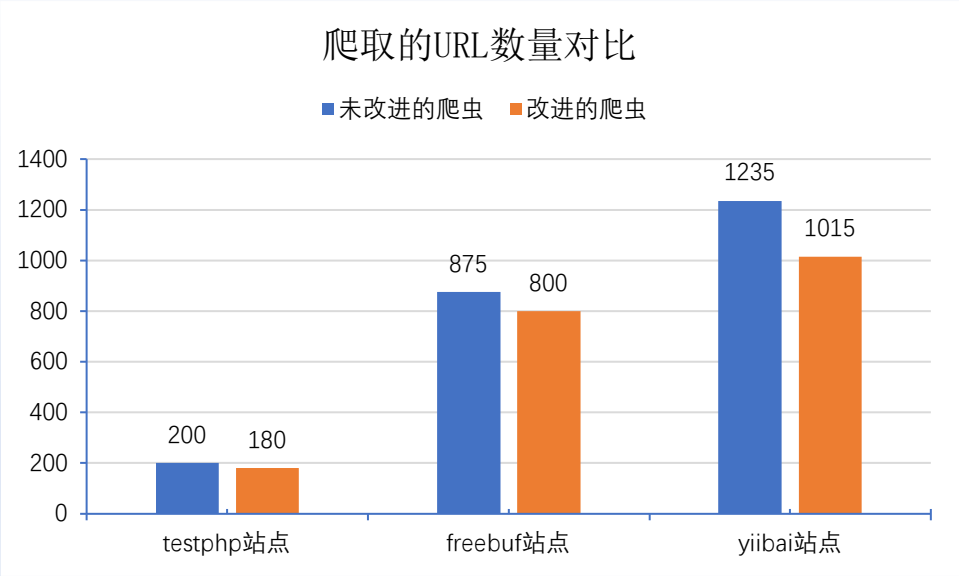

图 5.4 爬取的 URL 数量对比

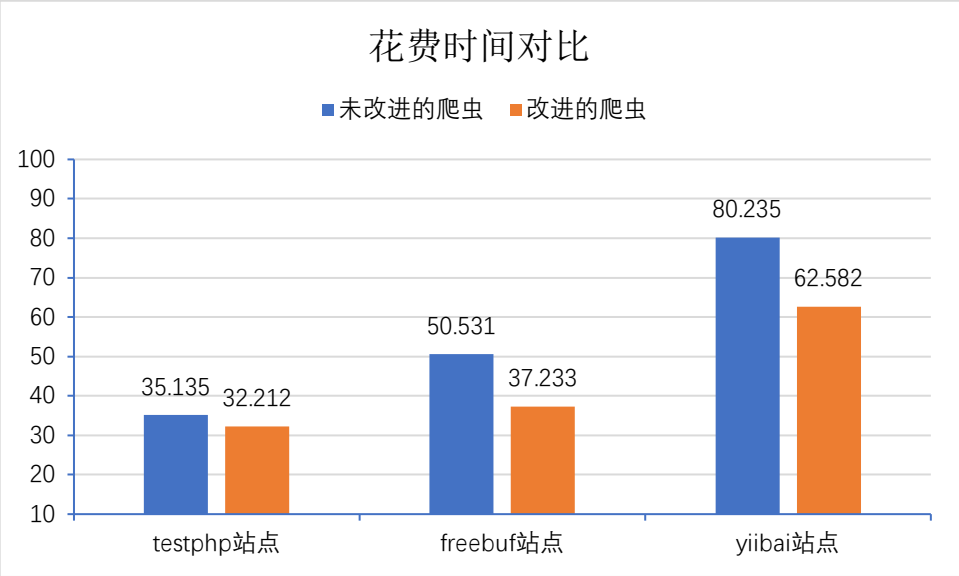

图 5.5 花费的时间对比

实验测试了三个目标站点，分别为 testphp 站点、freebuf 站点和 yiibai 站点，图 5.4 表示爬取的 URL 数量对比，纵坐标单位为条，如 200 条数据，由图 5.4 可以看出，针对三个目标站点，改进后的爬虫爬取的 URL 数量比未改进的爬虫爬取的 URL 数量少，这是因为对 URL 数据进行了过滤，不再对相似的 URL 数据进行请求爬取，由于未改进的爬虫没有对相似 URL 进行过滤，因此爬取的 URL 数量会相对多一些。图 5.5 表示爬虫爬取 URL 数据花费的时间，时间单位为秒，由图 5.5 可以看出，使用 HDTSF 算法去重

前，testphp 站点 URL 提取花费时间为 35.135s，freebuf 站点 URL 提取花费时间为 50.531s，yiibai 站点 URL 提取花费时间为 80.235s，而在使用 HDTSF 算法去重后，testphp 站点 URL 提取花费时间为 32.212s，freebuf 站点 URL 提取花费时间为 37.233s，yiibai 站点 URL 提取花费时间为 62.582s，可以看到改进后的爬虫爬取 URL 花费的时间比改进前的爬虫爬取 URL 花费的时间少。综合上述分析，可以看到本文设计的爬虫去重效果良好，爬取 URL 数据花费的时间较少，具有良好的性能。

5.3 WEB 注入漏洞检测方法的功能测试

功能测试（也叫黑盒测试），不用考虑产品内部结构和程序处理过程，它是站在用户的角度去验证程序的功能，只需要测试产品的各个功能，对产品的各个功能模块进行验证，判断产品是否能达到预期的效果[44]。

本文实现的 Icrawler-scan 程序包括 URL 爬取模块、SQL 漏洞检测模块、XSS 漏洞检测模块和检测结果报告模块，同时为了验证程序的检测结果是否正确，本文采用手动注入的方式构建 HTTP 请求，根据响应的信息判断是否确实存在漏洞，保证了本文所实现的程序的功能的正确性。本文所实现的 Icrawler-scan 程序的初始界面如图 5.6 所示，通过 help 命令可以查看需要配置的参数。

程序扫描过程如图5.7所示，它显示了扫描每一个链接的时间点及扫描的链接信息，对于存在漏洞的链接，会直接显示漏洞类型，并输出到漏洞文档。出于对 Web 站点的隐私安全保护，图中对站点的域名信息作了模糊处理。

在对 Web 站点扫描结束后，程序会自动生成漏洞扫描检测结果报告，本文设计实现的漏洞扫描检测程序目前支持两种格式的结果报告，分别为 HTML 格式和 PDF 格式的报告。如图 5.8 所示为生成的 HTML 格式的报告，报告的名称为 Web 站点域名信息关键字。报告中列出了扫描的目标站点信息、扫描检测花费的时间和漏洞数量，并且详细列出了漏洞的 URL 链接和漏洞类型。

```
E:\Coding_Workpace\python\web_vulScan>python2 run.py -h
Usage: run.py [options]

Options:
  -h, --help            show this help message and exit
  -d DOMAIN, --domain=DOMAIN
                        Start the domain name
  -t THREAD_NUM, --thread=THREAD_NUM
                        Numbers of threads
  --depth=DEPTH         Crawling dept
  --module=MODULE       vulnerability module
  --cookie=COOKIE       Cookie validation
  --log=LOGFILE_NAME    save log file
```

图 5.6 程序初始化配置界面

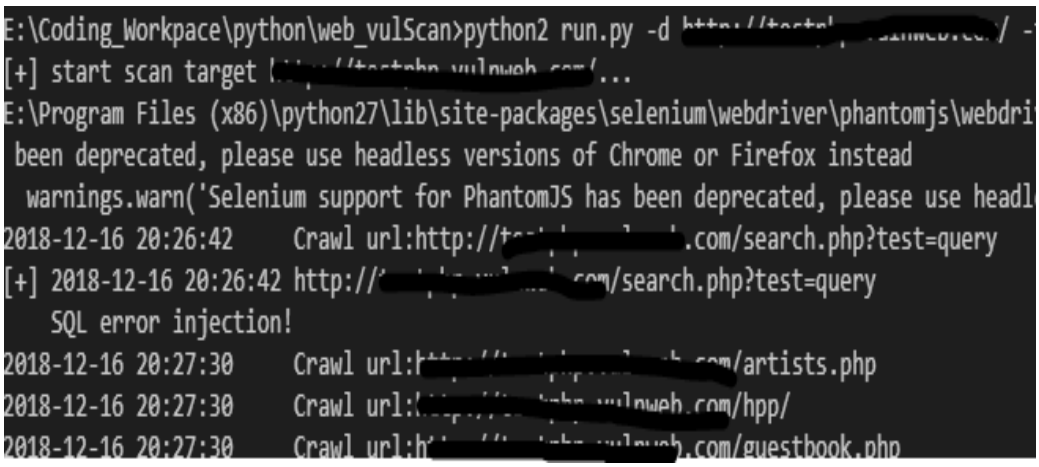

图 5.7 Icrawler-scan 扫描过程

图 5.8 漏洞扫描检测结果报告

当检测出 Web 站点存在漏洞后，对于常见或者公有的漏洞，可以直接向用户推送漏洞补丁或者下载补丁的站点，由用户决定是否安装漏洞补丁；而针对于系统的漏洞，可以向系统开发者发送邮箱说明存在的漏洞问题，由程序开发者进行系统升级服务以修复漏洞，一般情况下，漏洞修复是两种方式并存的。

为了验证 Icrawler-scan 程序检测出的 SQL 漏洞，采用人工注入的方式构建注入参数为' and 0;--的 URL 数据，然后发送 HTTP 请求，服务器返回的响应页面信息如图 5.9 所示。

由图 5.9 可以看出，该站点确实存在 SQL 注入漏洞，且从页面显示的错误信息可以看出该 Web 站点使用的数据库为 MySQL，在得到数据库的类型后，便可以通过暴力破

解或者利用数据库自身默认生成的 information_schema(信息数据库)来猜解表和列的信息，通过该库可以查看 MySQL 下的所有数据库、表权限等信息。

图 5.9 SQL 漏洞验证

为了验证 Icrawler-scan 的 XSS 漏洞检测功能有效性和正确性，选择对某一存在 XSS 漏洞的网站进行检测，经过测试该网站存在 XSS 漏洞，为了验证是否确实存在 XSS 漏洞，本文采用手动注入的方式对网站进行模拟攻击测试，测试结果如图 5.10 所示。

图 5.10 谷歌浏览器手动注入测试 XSS 漏洞

图 5.10 所示页面显示网页无法正常运作是因为谷歌浏览器内核添加了安全过滤功能，其目的是为了抵御 XSS 攻击，通过图 5.10 所示页面可以清楚的看到一个字符串提示信息为 ERR_BLOCKED_BY_XSS_AUDITOR，谷歌浏览器在 XSS 攻击脚本执行之前就将其拦截了。

下面使用火狐浏览器对 XSS 漏洞进行模拟攻击测试，得到的结果如图 5.11 所示，由图 5.11 可以看到，该页面确实存在 XSS 漏洞，XSS 攻击脚本直接执行了并没有被火

狐浏览器过滤拦截。从这一点可以看出谷歌浏览器的安全性能比火狐浏览器安全性能好。

图 5.11 火狐浏览器手动注入测试 XSS 漏洞

通过本文实现的 Icrawler-scan 程序测试及手动注入验证测试可以看出，本文所设计实现的 Icrawler-scan 程序符合预期效果，能够准确检测出 SQL 漏洞和 XSS 漏洞。

5.4 WEB 注入漏洞检测方法的性能测试

为了验证本文的 Icrawler-scan 程序的性能，将使用两款其他扫描软件与 Icrawler-scan 进行对比测试，分别从检出漏洞的数量、检出漏洞的准确率和运行时间效率这三个方面来比较。本文选取三个已知漏洞的目标网站进行对比测试，目标网站及其漏洞信息如表 5.2 所示。在相同测试环境下，将本文的 Icrawler-scan 程序和 AIscanner、WebVulScan 两款漏洞扫描工具进行实验测试对比。AIscanner 是国内某公司开发的一款网络安全漏洞扫描工具[45]，WebVulScan 是一款采用 PHP 开发的 Web 应用程序漏洞扫描工具。

表 5.2 Web 应用站点

Web 站点	SQL 漏洞数量	XSS 漏洞数量
testphp	8	2
aisec	6	2

其中，testphp 应用一共有 10 个漏洞，SQL 漏洞 8 个，XSS 漏洞 2 个；aisec 应用一共有 8 个漏洞，SQL 漏洞 6 个，XSS 漏洞 2 个。在相同测试环境下，本文实现的 Icrawler-scan 和 AIscanner、WebVulScan 扫描软件对表 5.2 所示的两个 Web 站点扫描检测，得到的结果如表 5.3 和表 5.4 所示。

表 5.3 目标站点 testphp 检出漏洞对比表

测试工具	扫描时间	SQL 注入漏洞	XSS 漏洞
AIscanner	700.240s	7	0
WebVulScan	520.494s	4	1
Icrawler-scan	277.941s	5	1

表 5.4 目标站点 aisec 检出漏洞对比表

测试工具	扫描时间	SQL 注入漏洞	XSS 漏洞
AIscanner	180.135s	5	0
WebVulScan	160.395s	5	1
Icrawler-scan	82.805s	6	0

① 运行效率

由表 5.3 和表 5.4 可以看出针对站点 testphp 应用，AIscanner 扫描耗时 700.240s，WebVulScan 扫描耗时 520.494s，本文实现的 Icrawler-scan 程序扫描耗时 277.941s；对于站点 aisec 应用，AIscanner 扫描耗时 180.135s，WebVulScan 扫描耗时 160.395s，本文实现的 Icrawler-scan 程序扫描耗时 82.805s。通过以上对比分析可以看出，针对以上两个目标站点，本文实现的 Icrawler-scan 程序扫描耗时均比 AIscanner 和 WebVulScan 少，改进爬虫后，程序的运行效率得到显著提高。

② 误报率

误报率是衡量漏洞扫描工具的一个重要评价标准，误报是指正确样本被识别为错误样本。为了验证漏洞扫描结果的准确性，本文采用手动注入的方式进行攻击测试。其中，testphp 应用的漏洞个数为 10 个，AIscanner 扫描出 7 个漏洞，有 1 个误报的 SQL 漏洞，误报率为 14.3%；WebVulScan 扫描出 5 个漏洞，有 1 个误报的 SQL 漏洞，误报率为 20%；本文实现的 Icrawler-scan 扫描出 6 个漏洞，没有误报，误报率为 0%。aisec 应用的漏洞个数为 8 个，AIscanner 扫描出 5 个漏洞，没有误报，误报率为 0%；WebVulScan 扫描出 6 个漏洞，有 2 个误报，一个 SQL 误报，一个 XSS 误报，误报率为 33.3%；本文实现的 Icrawler-scan 扫描出 6 个漏洞，没有误报，误报率为 0%。

③ 漏报率

漏报率是衡量漏洞扫描工具的另一个重要评价标准，漏报是指错误样本被识别为正确样本。目标站点 testphp 实际有 10 个漏洞，由扫描结果及手动注入测试结果可知，

AIscanner 有 4 个漏报，漏报率 40%；WebVulScan 有 6 个漏报，漏报率为 60%；Icrawler-scan 有 4 个漏报，漏报率为 40%。目标站点 aisec 实际有 8 个漏洞，由扫描结果及手动注入测试结果可知，AIscanner 有 3 个漏报，漏报率为 37.5%；WebVulScan 有 4 个漏报，漏报率为 50%；Icrawler-scan 有 2 个漏报，漏报率为 25%。

通过上面的测试分析得到的漏洞检测准确率如表 5.5 和表 5.6 所示。

表 5.5 目标站点 testphp 漏洞检测准确率对比表

测试工具	扫描时间	误报率	漏报率
AIscanner	700.240s	14.3%	40%
WebVulScan	520.494s	20%	60%
Icrawler-scan	277.941s	0%	40%

表 5.6 目标站点 aisec 漏洞检测准确率对比表

测试工具	扫描时间	误报率	漏报率
AIscanner	180.135s	0%	37.5%
WebVulScan	160.395s	33.3%	50%
Icrawler-scan	82.805s	0%	25%

综合上述分析可以看出，本文实现的 Icrawler-scan 程序的扫描检测时间均比 AIscanner 和 WebVulScan 的扫描时间短，可以看出本文提出的 HDTSF 算法是有效的；并且 Icrawler-scan 扫描程序能够较为准确地检测出 Web 应用程序中存在的 SQL 漏洞和 XSS 漏洞，但由于漏洞攻击数据集并不完善，因此依然存在一定的漏报率和误报率。

5.5 本章小结

本章首先介绍了实验测试环境，然后对 URL 爬取模块进行了测试和分析，将提出的 HDTSF 去重算法、编辑距离算法和余弦相似度算法进行了实验对比测试，得出了本文提出的 HDTSF 算法在爬虫相似性去重方面优于另外两种算法，然后对改进的爬虫的可行性进行了测试，实验表明改进的爬虫具有较好的性能。最后对本文设计实现的漏洞检测扫描程序 Icrawler-scan 进行了功能测试和性能测试。

6 总结与展望

6.1 总结

随着生活服务网络化和信息化程度的加深，越来越多的企事业单位及个人开始搭建自己的站点，并将其作为与人们交互的一种信息传送及分享渠道。网络给人们带来生活便利的同时，也伴随着各种各样的安全问题，其中注入漏洞是 Web 应用程序比较大的危害之一，国内外研究学者对 Web 应用安全也有一定研究。本文的主要研究工作及创新点总结如下：

（1）概述了本文的研究背景及意义，对国内外研究现状及 URL 去重算法的研究现状进行了分析，确立本文研究内容，然后概述了 Web 应用安全相关的技术理论。

（2）对基于改进爬虫的 Web 注入漏洞检测方法进行设计与改进，分别就爬虫模块和注入检测模块进行研究，针对现有爬虫模块中的 URL 去重算法存在的问题进行分析后，提出基于树结构的 URL 相似性去重算法 HDTSF，经过可行性测试分析和对比测试分析，表明相似性去重算法 HDTSF 具有较好的去重效果。

（3）实现了基于改进爬虫的 WEB 注入漏洞检测系统 Icrawler-scan，并进行了功能测试和性能测试，测试结果表明本文实现的 Icrawler-scan 扫描系统漏洞检测效果良好。

6.2 展望

本文对 WEB 注入漏洞检测方法中涉及到的爬虫技术进行了改进，实现了漏洞扫描检测系统 Icrawler-scan，并对系统进行了测试与分析。但由于个人能力的局限及时间原因，依然存在很多不足之处，后续将进行完善。主要包括：

（1）完善爬虫性能。由于网站开发技术的复杂性，爬虫程序存在一些限制，无法覆盖全部数据，后期会对爬虫程序进行完善，使其更加健壮和高效爬取。

（2）增加其他类型漏洞检测功能。本文实现的 Icrawler-scan 仅针对 SQL 注入漏洞和 XSS 漏洞进行了检测，但是在实际情形中，Web 漏洞类型很多，要想全面检测 Web 站点的安全性，就需要增加其他类型的漏洞验证方法。因此后期可以对其他类型的漏洞进行研究分析，并增加其他类型漏洞的检测模块，以提高系统的漏洞检测能力。

致谢

三年的读研时光转瞬即逝，论文的顺利完成预示着硕士生活已步入尾声。三年的时间不短，承载了太多的喜怒哀乐；三年的时间不长，还没好好把握就悄然而逝。回顾过往，初入研究生院时的彷徨与迷茫，课堂上的专注认真，同学间的欢声笑语，寻找工作时的感慨与焦急，一切都历历在目。白驹过隙，未来可期，即将离开校园的我们又将踏上新的征途，在此感谢所有帮助过我的老师、同学、亲友和同事。

首先要感谢我的导师刘涛老师。在研究生期间，刘老师对我的课堂学业非常关心，并对我的小论文和毕业论文给予了指导，耐心的帮助我修改小论文及毕业论文。刘老师一丝不苟的学术作风，渊博的专业知识和循循善诱的教导风格让我钦佩。在生活中，刘老师也给我一些建设性的意见，帮助我正确认识问题，解决问题。在此，谨向刘老师及所有给予我帮助的老师们表示真挚的感谢和敬意。

感谢实习期间带我工作的陈晓兵经理和陈康康组长，以及其他友好的同事们，在项目开发过程中遇到困难的时候，是他们不厌其烦的给我讲解，耐心地教我优化代码，帮助我克服困难，使我得以顺利的完成工作任务。

感谢我的同学，三年的朝夕相处，互助友爱，欢声笑语，是他们陪我度过了一段美好的时光。

感谢我的家人一直以来给予的支持和鼓励，是他们无时无刻的悉心关怀，让我可以更好地成长。

最后，谨向百忙之中抽出宝贵的时间出席硕士论文答辩和审阅硕士论文的各位专家老师学者致以诚挚的感谢。感谢西安科技大学为我提供了良好的研究生科研环境，衷心希望西安科技大学越办越好，桃李遍天下。

参考文献

[1] Murshed S , Al-Hyari A , Wendel J , et al. Design and Implementation of a 4D Web Application for Analytical Visualization of Smart City Applications[J]. ISPRS International Journal of Geo-Information, 2018, 7(7).

[2] 肖泽力.SQL 注入攻击检测方法研究[D].长春:东北师范大学,2018.

[3] 张玉凤,楼芳,张历.面向软件攻击面的 Web 应用安全评估模型研究[J].计算机工程学,2016,38(01):73-77.

[4] 许子先,卜哲,裴立军.基于纵深防御的 Web 系统安全架构研究[J].信息网络安全,2013(06):90-93.

[5] 仝青,张铮,张为华,邬江兴.拟态防御 Web 服务器设计与实现[J].软件学报,2017,28(04):883-897.

[6] 谭军.OWASP 发布十大 Web 应用安全风险[J].计算机与网络,2017,43(23):52-53.

[7] 孙基男,潘克峰,陈雪峰等.基于符号执行的注入类安全漏洞的分析技术[J].北京大学学报（自然科学版）,2018,54(1):1-13.

[8] Alwan Z S, Younis M F. Detection and Prevention of SQL Injection Attack: A Survey[J].International Journal of Computer Science and Mobile Computing, 2017, 6(8): 5-17.

[9] Joshi P N, Ravishankar N, Raju M B, et al. Encountering SQL Injection in Web Applications[C]//2018 Second International Conference on Computing Methodologies and Communication (ICCMC). IEEE, 2018: 257-261.

[10] 杨高明.SQL 注入的自动化检测技术研究[D].成都:电子科技大学,2015.

[11] Lei L, Jing X, Minglei L, et al. A Dynamic SQL injection vulnerability test case generation model based on the multiple phases detection approach[C]//Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual. IEEE, 2013: 256-261.

[12] 彭赓,范明钰.基于改进网络爬虫技术的 SQL 注入漏洞检测[J].计算机应用研究,2010,27(07):2605-2607.

[13] 秦广赞,郭帆,徐芳,余敏.一种防 SQL 注入的静态分析方法[J].计算机工程与科学,2013,35(02):68-73.

[14] 李鑫,张维纬,郑力新.动静结合的二阶 SQL 注入漏洞检测技术[J].华侨大学学报(自然科学版),2018,39(04):600-605.

[15] 翟涵.基于网络爬虫的 Web 安全扫描工具的设计与实现[D].北京:北京邮电大学,2018.

[16] Alzahrani A, Alqazzaz A, Zhu Y, et al. Web Application Security Tools Analysis[C]//Big

Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2017 IEEE 3rd International Conference on. IEEE, 2017: 237-242.

[17] Ouarda Lounis, Salah Eddine Bouhouita Guermeche, Lalia Saoudi, Salah Eddine Benaicha. A new algorithm for detecting SQL Injection attack in Web application[C]//Science and Information Conference (SAI), 2014: 589-594.

[18] Joshi A, Geetha V. SQL Injection detection using machine learning[C]//Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference on. IEEE, 2014: 1111-1115.

[19] 张衡,陈良育.Levenshtein 算法优化及在题库判重中的应用[J].华东师范大学学报(自然科学版),2018(05):154-163.

[20] 徐晨初,张燕平,刘国涛.一种优化路径的聚焦爬虫爬行策略[J].小型微型计算机系统统,2016,37(08):1721-1724.

[21] Lu Y, Liu Y, Zhang C, et al. A Data-Deduplication-Based Matching Mechanism for URL Filtering[C]//2018 IEEE International Conference on Communications (ICC). IEEE, 2018: 1-6.

[22] Lim H, Lee J, Byun H, et al. Ternary Bloom filter replacing counting Bloom filter[J]. IEEE Communications Letters, 2017, 21(02): 278-281.

[23] 倪平,张治兵,周开波,李莉.网络设备 HTTP 服务安全威胁及其防范措施[J].现代电信科技,2016,46(03):21-28.

[24] 杨海军,施敏,梁汝峰,蔡立志.基于用户行为模型的移动 APP 信息采集方法[J].计算机应用与软件,2018,35(06):158-162.

[25] 汪彩梅,马婷婷.基于手动 SQL 注入攻击及防范设计与实现[J].计算机安全,2013(11):36-41.

[26] 王云,郭外萍,陈承欢.Web 项目中的 SQL 注入问题研究与防范方法[J].计算机工程与设计,2010,31(05):976-978+1016.

[27] 胡康,谈德才,陈华.基于网络技术的白水江梯级水电站调度系统研究[J].中国农村水利水电,2017(01):203-207.

[28] Kausar M A, Dhaka V S, Singh S K. Web crawler: a review[J]. International Journal of Computer Applications, 2013, 63(02).

[29] 李明,刘滨.基于数据驱动的司法公开信息化监管系统[J].河北科技大学学报,2016,37(04):407-415.

[30] 孙歆,戴桦,孔晓昀,赵明明.基于 Scrapy 的工业漏洞爬虫设计[J].网络空间安

全,2017,8(01):66-71.
[31] 姜华,韩安琪,王美佳,王峥,吴雲玲.基于改进编辑距离的字符串相似度求解算法[J].计算机工程,2014,40(01): 222-227.
[32] 李震,何泾沙.基于用户体验的模糊密码算法研究[J].信息网络安全,2016(11):73-78.
[33] 周长红,曾庆田,刘聪,段华,原桂远.基于模型结构与日志行为的流程相似度计算[J].计算机集成制造系统,2018,24(07):1793-180.
[34] 刘静,何运,赖英旭.SDN 架构下的安全审计系统研究与实现[J].北京工业大学学报,2017,43(02):180-191.
[35] 刘文,马慧芳,脱婷,陈海波.融合共现距离和区分度的短文本相似度计算方法[J].计算机工程与科学,2018,40(07):1281-1286.
[36] 胡萍瑞,李石君.基于 URL 模式集的主题爬虫[J].计算机应用研究,2018,35(03):694-699+726.
[37] 药珍妮.基于主题和特征的文本相似度算法研究[J].软件,2016,37(10):123-126.
[38] 郭晓军,孙海霞,张国梁.基于布隆过滤的藏区 Web 站点流量识别[J].计算机工程与设计,2018,39(02):365-369.
[39] 严磊,丁宾,姚志敏,马勇男,郑涛.基于 MD5 去重树的网络爬虫的设计与优化[J].计算机应用与软件,2015,32(02):325-329+333.
[40] 冉丽敏,郑钢锋.电子商务概论[M].成都:西南财经大学出版社,2016.
[41] 徐健锋, 苗夺谦, 张远健. 基于混淆矩阵的多目标优化三支决策模型[J]. 模式识别与人工智能, 2017, 30(09):859-864.
[42] 米爱中,陆瑶.基于聚类与排序修剪的分类器集成方法[J].计算机应用研究,2018,35(07):2034-2037.
[43] 丁春晖.关于大数据网络中数据分类优化识别研究[J].计算机仿真,2018,35(08):307-310+414.
[44] 李吟, 方建勇, 江梦. 面向需求覆盖的 Web 服务自动化测试框架[J]. 计算机科学与探索, 2017(11):51-67.
[45] Hassanshahi B , Jia Y , Yap R H C , et al. Web-to-Application Injection Attacks on Android: Characterization and Detection[C]// European Symposium on Research in Computer Security. Springer International Publishing, 2015.

附录

攻读学位期间的学术成果

[1] 刘涛，张隆涛.Application of Logistic Regression in WEB Vulnerability Scanning[C]//2018 International Conference on Sensor Networks and Signal Processing (SNSP), Xi'an, China, 2018:486-490(EI:20191006584684).